Iterative Batch Reinforcement Learning via Safe Diversified Model-based Policy Search

Amna Najib, Stefan Depeweg, Phillip Swazinna
Siemens AG
{amna.najib, stefan.depeweg, phillip.swazinna}@siemens.com

Abstract: Batch reinforcement learning enables policy learning without direct interaction with the environment during training, relying exclusively on previously collected sets of interactions. This approach is, therefore, well-suited for high-risk and cost-intensive applications, such as industrial control. Learned policies are commonly restricted to act in a similar fashion as observed in the batch. In a real-world scenario, learned policies are deployed in the industrial system, inevitably leading to the collection of new data that can subsequently be added to the existing recording. The process of learning and deployment can thus take place multiple times throughout the lifespan of a system. In this work, we propose to exploit this iterative nature of applying offline reinforcement learning to guide learned policies towards efficient and informative data collection during deployment, leading to continuous improvement of learned policies while remaining within the support of collected data. We present an algorithmic methodology for iterative batch reinforcement learning based on ensemble-based model-based policy search, augmented with safety and, importantly, a diversity criterion.

1 Introduction

The objective of batch (or offline) reinforcement learning (RL) is to extract the best possible behavior out of existing data, called a batch, without any learning during deployment. This implies that learning is more successful if the initial data is diverse or collected through the deployment of an expert agent. In a real setting, the initial batch is prone to limitations (low data coverage, low reward actions, etc.), forming a challenge in learning for real-world applications. This challenge of limited information calls for safety mechanisms, such as regularization, to ensure reliable performance of the agent [1, 2].

In many industrial setups, the application of offline reinforcement learning is not a one-time process but iterative. After an RL agent is trained and deployed on the system, a new set of recordings becomes available. The principal contribution of our work relies on the formulation of iterative batch reinforcement learning (**IBRL**), a novel framework to iteratively refine the initial data batch and improve learned policies after each new batch collection, without dropping performance due to overly adventurous exploration. In every iteration, we seek to improve the data coverage by deploying a set of policies, that we previously trained to be diverse, i.e. that act in a variety of ways to explore, without compromising the rewards too much. The proposed IBRL algorithms adhere to safety constraints by restricting the state or action space of the learned policies depending on the data support. Through experiments in an illustrative 2D environment, as well as on the Industrial Benchmark [3], we demonstrate the improved exploration capability and resulting improved performance of our approach, all while maintaining safety considerations and not underperforming the behavioral. Conceptually, our work is most closely related to [4, 5, 6], however these works do not address the combination of diversity and safety, or focus solely on the iterative process without incorporating an exploration incentive.

Workshop on Safe and Robust Robot Learning for Operation in the Real World (SAFE-ROL) at CoRL 2024.

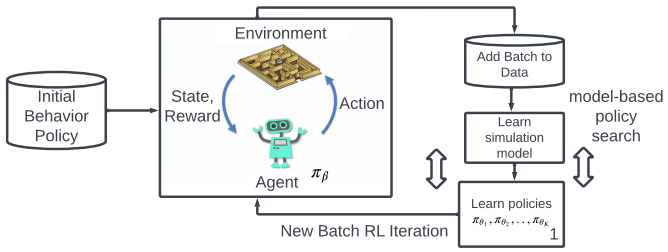


Figure 1: Illustration of iterative model-based policy search.

2 Related Work

Offline RL: Early works in the so-called "batch RL" setting include [7, 8, 9, 10], as well as more recently [11, 12, 13, 14]. While these works focused on reinforcement learning in the batch setting, a key distinction to later proposed offline methods is that they mostly assume the initial batch to be collected under uniform random actions, allowing a relatively well explored environment. While the batch RL setting is certainly closer to the requirements imposed by real-world deployments of RL algorithms than commonly popular online algorithms such as [15, 16, 17], it is still not exactly what industry practitioners need most of the time. Algorithms such as [18, 19, 20, 21, 22, 23] employ various regularization techniques to keep the trained policies in regions of the environment where the models are sufficiently accurate. Most common techniques include behavior regularization [24, 19] and uncertainty-based regularization [25, 26]. In offline RL, model-based RL appears to have an edge over model-free methods, which enjoy better asymptotic performance but have generally been attributed lower data efficiency [27, 23, 28]. Methods such as [14, 1, 29, 30] have thus developed ways to extend action divergence regularizations to the model-based setting. **Offline-to-Online**: Some of the early batch RL works mentioned above as well as [4, 31] introduced the idea of the growing batch setting, a problem in between offline and online learning where one is constrained to only deploy a limited number of times to the real system to collect new data. While the idea is appealing, our approach differs significantly since their almost unconstrained exploration can be an issue due to safety constraints or requirements on the minimal performance. Other works that explored online finetuning of offline pretrained policies without the deployment constraint exhibit similar issues due to their exploration strategies [32, 23, 33, 34, 35]. **Diversity**: Intrinsically motivated agents have been introduced in [36]. Agents have since been found to perform effective exploration using intrinsic reward objectives like curiosity or diversity and have been studied e.g. in [37, 38, 39]. Previous work has also explored the effect of collecting initial diverse data that is further used for offline learning in several downstream tasks [40, 41, 42], as well recently [43, 44].

3 Method

Offline reinforcement learning uses a fixed dataset $\mathcal{D}$ collected by one or more behavior policies π_β to learn a policy π. Once trained, the agent's policy is fixed and no further learning occurs during deployment. To ensure safety, the learned policy is often constrained to avoid significant deviations from the original behavior policies used to generate the data.

One approach for learning a policy is a model-based policy search. This is a two-step process. First, a simulation model is learned from the available data. In the second step, one or a set of policies are optimized using virtual rollouts. For policy search, we seek to minimize the loss function

$$L(\theta) = -\frac{1}{N}\frac{1}{K}\frac{1}{H}\sum_{k=1}^{K}\sum_{s_{k,1}\sim D}\sum_{t=1}^{H}\gamma^t e(s_{k,t}, a_{k,t}) \ , \tag{1}$$

where K is the number of policies, $e(s_{k,t}, a_{k,t})$ the reward received while taking action $a_{k,t}$ at state $s_{k,t}$ and $\theta = \theta_1, \ldots, \theta_K$ the parameters of the policies. We aim to maximize the accumulated reward of trajectories generated by following the K learned policies. Every trajectory is formalized as

$$\begin{aligned} T_k &= \{s_{k,1}, a_{k,1}, s_{k,2}, a_{k,2}..s_{k,H-1}, a_{k,H-1}, s_{k,H}\} \\ &= \{s_{k,i} \mid s_{k,i} = f(s_{k,i-1}, a_{k,i-1}; \eta), a_{k,i-1} = \pi(s_{k,i-1}; \theta_k), 1 < i < H+1, s_{k,1} \sim D\} \end{aligned} \tag{2}$$

where $f(\cdot; \eta)$ denotes the learned transition model.

In **iterative offline reinforcement** learning the offline RL setting is repeated at different times over the lifetime of a system. The collected data in every iteration of deployment is added to the batch and further used to refine the training of the policy. We can apply the method of model-based policy search to the iterative offline setting: every time a new batch of data becomes available, we update the transition model and then redo the policy training and deployment. This will serve as the backbone of our proposed algorithm, which we illustrate in Figure 1. We note that alternative approaches, such as a model-free approach are viable as well. We choose model-based policy search, because it tends to work well in practice, can be easily extended (e.g. via uncertainty modeling), and allows automatic differentiation in continuous state, action, and reward spaces.

Importantly, we argue that two components are highly beneficial for model-based policy search in the iterative batch scenario: **safety** and **diversity**. The former is ubiquitous in all real-world applications and should safeguard against exploiting model inaccuracies due to limited data. The latter should exploit the iterative nature and improve information gain over each iteration. We hypothesize that an ideal algorithm for iterative batch RL is safe but diverse.

3.1 Safety

We formulate three approaches towards realizing safety in policy search. Safety may be used (i) as an additional objective in the loss function, (ii) as a soft constraint, or finally, (iii) it may also be possible to constrain the policy directly as part of its architecture.

Safety as an objective Safety is injected as an explicit objective in the loss function to balance between high rewards and not deviating from the behavior policy. It was used in previous research work including [45, 46, 20]. At first, a behavior policy is learned in a supervised manner from the fixed initial dataset as denoted in Eq. (3). Afterwards, for every state, the behavioral policy action and the new policy actions are predicted and used to define the deviation between the behavior policy and the learned policy. The deviation in the simplest case is a mean-squared error between both actions. If a distribution of actions is learned for both policies, the Kullback-Leibler divergence (KL divergence) could instead be used to assess the deviation. In the MSE case we have:

$$L(\phi) = \sum_{s,a \sim D} \|a - \pi_\beta(s; \phi)\|_2 \ . \tag{3}$$

Then, the loss function including the reward maximization and safety objectives is defined as

$$\begin{aligned} L(\theta) &= \frac{1}{K}\frac{1}{H}\sum_{k=1}^{K}\sum_{s_{k,1}\sim D}\sum_{t=1}^{H}[-\gamma^t \lambda e(s_{k,t}, a_{k,t}) + (1-\lambda)p(a_{k,t}) \ , \\ p(a_{k,t}) &= \|\pi(s_{k,t}; \theta_k) - \pi_\beta(s_{k,t}; \phi)\|_2 \ . \end{aligned} \tag{4}$$

While this is the standard approach for safe offline RL it has one key drawback: Weighing safety against performance (and possibly diversity) in the form of a trade-off (in this case via the parameter λ) appears counter-intuitive for safety-critical applications.

Safety as a soft constraint An alternative approach is to specify a loss term that is flat inside a safe region and then provides a large loss value outside, thereby implementing a differentiable constraint. In contrast to the aforementioned objective-based approach, here the safety term is not weighted against the objective but instead is effectively restricting the allowed solution space of the policy.

We note that in iterative batch RL, the transitions in the batch may be generated by the execution of different policies. To that end, we propose to approximate the behavior policy using a probabilistic approach. We assume that the learnt behavior policy is a Gaussian distribution with mean μ_β and diagonal covariance matrix Σ_β, $\pi_\beta(s;\phi) \sim \mathcal{N}(\mu_\beta,\, \Sigma_\beta)$. Σ_β reflects the aleatoric uncertainty in the model. A prediction has a high aleatoric uncertainty in the case of diverse actions present in the behavior data. As a measure of safety we consider how likely an action is under the behavior policy. We make 3 considerations for the proposed metric: to be normalized between 0 and 1, to be sensitive to low likelihoods in certain action dimensions and lastly to be more permissible in situations where diverse actions were executed. To that end we use the negative unnormalized likelihood and compute the geometric mean over the action space:

$$G(S,A) = -[\prod_d \exp\left(-\frac{1}{2}(a-\mu_\beta(s;\phi))^T \Sigma_\beta(s;\phi)^{-1}(a-\mu_\beta(s;\phi))\right)]^{\frac{1}{d}} \ , \tag{5}$$

where d is the dimensionality of the actions. To enforce safety during training of policy π we can use thresholding:

$$L_S(\theta) = \alpha_s \frac{1}{K}\frac{1}{H}\sum_{k=1}^{K}\sum_{t=1}^{H} \max(G(S,A)+\delta, 0) \quad \text{with} \quad S \sim f(\cdot;\eta), A \sim \pi(S;\theta) \ , \tag{6}$$

where $\delta \in [0,1]$ represents the safety threshold and controls the permissibility of the training. Any deviation of the actions represented by $G(S,A)$ lower than $-\delta$ is not penalized in the learning. To conclude, the loss in Eq. (6) implements a likelihood-based safety zone.

Constrained Policy The direct approach to fulfill safety is arguably to directly constrain the expressiveness of the policy itself. In this scenario, we consider safety with respect to the state space and not relative to a behavior policy. We assume that professionals may have prior knowledge about safety ranges (bounds) of sensors. We start building on this assumption and limit all the states over all virtual rollouts for the different ensembles to lie within the predefined bounds. We inject this bound constraint in the policy specification.

This approach is only applicable if actions a affect a subset of system variables s' in a known linear way. Let a_{lower} and a_{upper} be the lower and upper bounds of actions. Further, let $\pi(s_t;\theta)$ be designed such that only actions in this bound can be computed. Let $B1$ and $B2$ be the lower and upper safety bound of the state. We then compute the valid action range $a_{\text{min}}, a_{\text{max}}$ such that $B1 < s'(t+1) < B2$. We then define the constrained policy such that:

$$\pi_{\text{constr}}(s_t;\theta_k) = a_{\text{min}} + (a_{\text{max}} - a_{\text{min}}) * \frac{\pi_{\text{constr}}(s_t;\theta_k) - a_{\text{lower}}}{a_{\text{upper}} - a_{\text{lower}}} \tag{7}$$

3.2 Diversified Policy Search

We define diversity as the ability to discover different or dissimilar state regions. This translates back to having a high entropy on the trajectory samples used for training and deployment. Given an ensemble of K policies, we draw K trajectory samples $T_1, \ldots, T_K$, where T_k results from the interaction between the learned transition model $f(s,a;\eta)$, reward model $f(s,a;\omega)$ and policy π_{θ_i} under the same starting state s_1. Let $D(\theta,\eta,\omega)$ denote the distances between all trajectories. The diversity-based exploration enforces the maximization of $D(\theta,\eta,\omega)$, by adding the diversity loss as an intrinsic motivation for exploration.

$$L_d(\eta,\omega,\theta) = -D(T_1, T_2, .., T_K) = -D(\theta,\omega,\eta)$$

Diversity measures We wish for the diversity-based objective to be differentiable w.r.t θ_k [47, 48]. Different distance metrics between trajectories were used in different communities [49, 50, 51]. The simplest distance metric is the pairwise distance between trajectories, also called lock-step Euclidean

distance (LSED). LSED measures the spatial discrepancy between time-corresponding states over the rollout of different K ensemble policies. The distance between trajectory T_i and T_j is calculated as

$$D(T_i, T_j) = \frac{1}{H} \sum_{t=1}^{H} \|s_{i,t} - s_{j,t}\|_2$$

The pairwise distance between all trajectories induced by the different K policies is

$$D = \frac{1}{K!} \sum_{k=1}^{K} \sum_{k'=1, k' \neq k}^{K} D(T_k, T_{k'}) = \frac{1}{H} \frac{1}{K!} \sum_{k=1}^{K} \sum_{k'=1, k' \neq k}^{K} \sum_{t=1}^{H} \|s_{k,t} - s_{k',t}\|_2 \tag{8}$$

LSED diversity can suffer from outlier behavior. Due to the fact that diversity and reward maximization potentially act as conflicting objectives, LSED diversity can incentivize all policies but one to focus on cost minimization and learns one outlier policy to bring the diversity to a high value, by that fulfilling the tradeoff to balance these competing objectives. Using the L1 norm instead of the L2 norm reduces the outlier behavior to an extent.

An alternative approach is to instead use the minimum pairwise distance between trajectories (MinLSED) to mitigate the outlier behavior of LSED. The MinLSED diversity represents the minimum mean distance over the horizon of the pairwise trajectories and is formalized as

$$D = \frac{1}{H} \min_{k' \neq k, k \in K} D(T_k, T_{k'}) \ . \tag{9}$$

In this work we will utilize the MinLSED diversity specified in Eq. (9).

3.3 Algorithm

We will show here one instantiation of a possible loss function using a soft-constrained policy and minLSD diversity.

$$\begin{aligned} L\theta) = &-\frac{1}{NKH} \sum_{k=1}^{K} \sum_{s_{k,1} \sim \mathcal{D}} \sum_{t=1}^{H} \gamma^t e(s_{k,t}, a_{k,t}) \\ &+ \alpha_s \frac{1}{KH} \sum_{k=1}^{K} \sum_{t=1}^{H} \max(G(s_{k,t}, \pi(s_{k,t}; \theta_k)) + \delta, 0) \\ &- \alpha_d \frac{1}{H} \min_{k' \neq k, k \in K} D(T_k, T_{k'}) \end{aligned} \tag{10}$$

where we note that in Eq. (10) by default we reduce the scope of the diversity term to exclude one (the first) policy. The reasoning is to have always one policy available that is only influenced by the reward and the safety. All policies are used for data generation, while only the first is used for evaluating the performance.

The aforementioned instantiation of the training algorithm for one batch iteration is shown in Algorithm 1.

Algorithm 1 Safe Diversified model-based policy search (Soft Constraint)

Input: Data $\mathcal{D}$, behavior policy $\pi_\beta(\mathbf{s}; \phi)$
Train transition model $f(s_t, a_t; \eta)$ on $\mathcal{D}$
Train reward function $f(\mathbf{s}, a; \omega)$ on $\mathcal{D}$
while not converged **do**
  $s_{k,1} \sim \mathcal{D}$
  for k=1,...,K **do**
    for t=2,...,H **do**
      $\mu_\beta(s_{k,t})$,
      $\Sigma_\beta(s_{k,t}) = \pi_\beta(s_{k,t}; \phi)$
      $a_{k,t} = \pi(s_{k,t}; \theta_k)$
      $s_{k,t+1} = f(s_{k,t}, a_{k,t}; \eta)$
      $r_{k,t} = f(s_{k,t}, a_{k,t}; \omega)$
      $R_k += \gamma^t r_{k,t}$
      $s_{k,t} = s_{k,t+1}$
    end for
  end for
  Compute Eq. (9) and Eq. (6)
  Train $\pi(.; \theta_1) \ldots, \pi(.; \theta_k)$ on Eq. (10)
end while

This process is repeated whenever a new batch arrives after execution of the previous trained set of policies, with the batch appended to the existing data set, following the schema illustrated in Figure 1.

4 Experiments

We investigate the advantage of the proposed iterative batch reinforcement learning framework in improving learned policies. We seek to investigate the supplementary advantage of incorporating diversity in the process. Finally, we investigate how different forms of safety impact the diversity objective. We evaluate our methods on a 2D grid environment and the industrial benchmark [3].

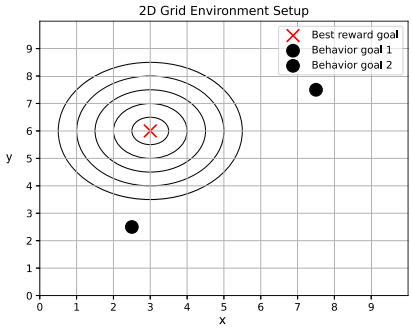


Figure 2: 2D grid environment: The behavior policy guides agent towards the nearest behavior goal. The best reward goal represents the state with the highest reward. Reward decreases according to Gaussian distribution represented by circle lines.

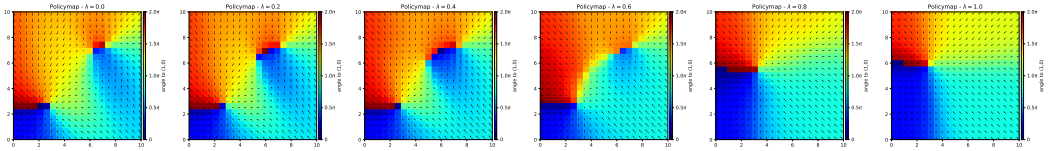


Figure 3: Policy maps for different λ values, colors illustrate action directions in every cell of the grid. For $\lambda = 0.0$, the policy imitates behavior policy by navigating to the closest behavior goal. With increasing λ values, the policy moves slowly towards the reward goal.

4.1 2D Grid Environment: Single Iteration

The 2D grid environment is a simplistic benchmark illustrating common navigation tasks. The state (x, y) represents the position of an agent in space. The agent is rewarded according to its position, following a Gaussian distribution with mean $\mu = (3, 6)^T$ and diagonal covariance matrix Σ with standard deviation vector $(1.5, 1.5)^T$. Concretely, the reward of an agent at state s_t is the likelihood of the reward Gaussian distribution: $r(s_{k,t}) = \frac{1}{(2\pi)^{\frac{3}{2}}|\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(s_{k,t}-\mu)^T\Sigma^{-1}(s_{k,t}-\mu)}$. See Figure 2 for a visualization of the reward distribution. We gather the initial data by deploying a behavior policy that navigates to one of the behavior goals denoted in Figure 2 and fixed to the positions $(2.5, 2.5)^T$ and $(7.5, 7.5)^T$. The agent navigates to the goal closest to its current position. Additionally, the behavior policy is augmented with 10% uniform random actions.

For reference, we first train a single policy without diversity for different values of λ for one single iteration. Figure 3 represents the policy maps of actions taken by policies learned with increasing λ values in the 2D grid environment. With $\lambda = 0.0$, the objective is reduced to mimicking the behavior policy. In the case of $\lambda = 0.4$, the policy predominantly adheres to the behavior policy with slight adjustments in the direction of certain actions. Nevertheless, convergence towards both behavior poles is still observable. This validates our further assumption of fixing λ at 0.4.

Figure 4 compares the results of a policy training without and with diversity. Here, we used the "safety as an objective" approach. We can see that diversity leads to a more heterogeneous set of policies. Additionally, we also observe that the safety term still has a significant impact on the policy. This result shows that diversity and safety can be combined to obtain safe, but different policies.

4.2 Industrial Benchmark: Iterative Batch RL

The Industrial Benchmark serves as a reinforcement learning simulator specifically designed to replicate challenges commonly encountered in industrial settings, such as high dimensionality,

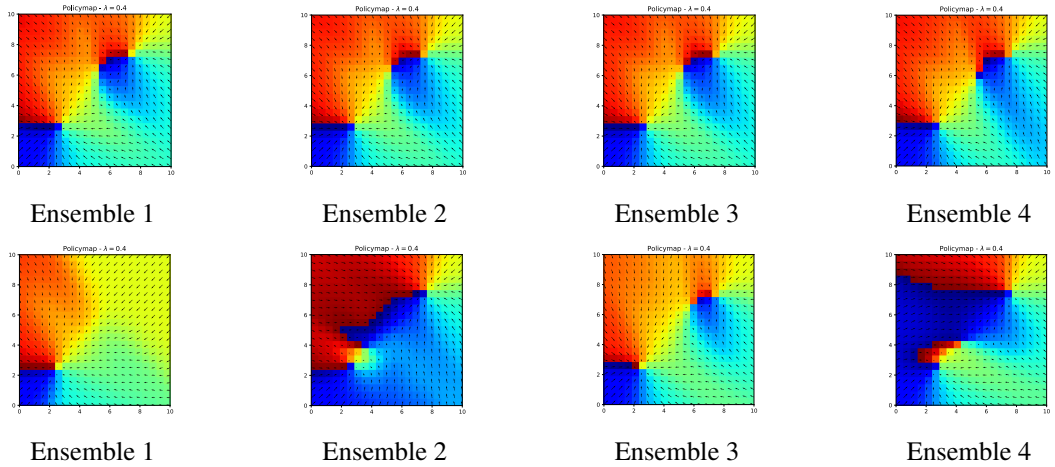


Figure 4: Policy maps of the ensemble policies using IBRL with safety as objective and parameter $\lambda = 0.4$. Colors represent the directions of the actions, where red corresponds to 2π and blue corresponds to 0.Top row: without diversity $\alpha_d = 0.0$. Bottom row: with diversity $\alpha_d = 0.15$.

partial observability, sparse rewards, and the consideration of Pareto optimality [13, 45, 1]. Every observation $s_t = (p, v_t, g_t, h_t, f_t, c_t)$ consists of the variables (setpoint, velocity, gain, shift, fatigue, consumption). Velocity, gain, and shift are bounded within [0, 100].

In order to see how the proposed algorithm from Section 3.3 performs in an iterative batch scenario and how different definitions of safety interact with the diversity term, we perform the following two experiments: First, we use a *random bounded dataset* collected by uniform sampling from the action space such that the states are restricted to lie within the safety bound [30, 70]. The samples are collected by generating five rollouts of horizon 200 each. In the supplementary material we show the velocity, gain, and shift observations of the random bounded batch. As a safety mechanism, we use a constrained policy with the same bounds during training.
For the second experiment, we use a simple policy (referred to as medium) to generate the initial batch, which tries to navigate to a fixed point in the state space, as also done in [45, 13]. The batch is collected by randomly sampling a starting state in the bound [0, 100] and subsequently following:

$$\pi_{\beta_{medium}}(\mathbf{s_t}) = \begin{cases} 50 - v_t \\ 50 - g_t \\ 50 - h_t \end{cases} \tag{11}$$

Additionally, the policy is augmented with 33% randomness. The velocity, gain, and shift observations of the medium policy batch are illustrated in the Appendix. The medium policy can be thought of as an example of a human technician who instructs the observable states to remain close to a fixed small region. The small region is, in this case, an ϵ-surrouding of state [50 (velocity), 50 (gain), 50 (shift)], where ϵ reflects the additional randomness introduced in the behavior policy. In this experiment we use the soft-constrain safety mechanism and thus the objective given by Eq. (10) for policy training.

In both experiments, we perform multiple iterations of batch RL. We start the iterative process by learning a dynamics model based on the available batch. A new batch is generated by executing each trained policy over 200 time steps. To account for partial observability, we incorporate fifteen past observations to construct the utilized state. The learned transition and reward models are subsequently used to learn an ensemble of ten policies through an ensemble model-based policy search. We rollout the policies for a horizon of 100 with a discount factor of 1. Simulation models, policy, and reward functions are two-layer MLPs with 50 hidden units each. We repeat each experiment three times and report average results. For diversity, we use the MinLSED diversity criterion from Eq. (9).

Experiment 1: Constrained policy
We summarize the main finding of this experiment in Table 1a. We observe that diversity accelerates

Cost (-Reward) over Iterations			
Policy	$\alpha_d = 0.0$	$\alpha_d = 0.15$	Initial data
Iteration0	x	x	216.5
Iteration1	203.5 ± 2.6	204.3 ± 2.4	x
Iteration2	194.0 ± 4.5	190.6 ± 1.0	x
Iteration3	189.2 ± 2.0	186.5 ± 1.5	x
Iteration4	188.9 ± 4.8	182.7 ± 1.2	x

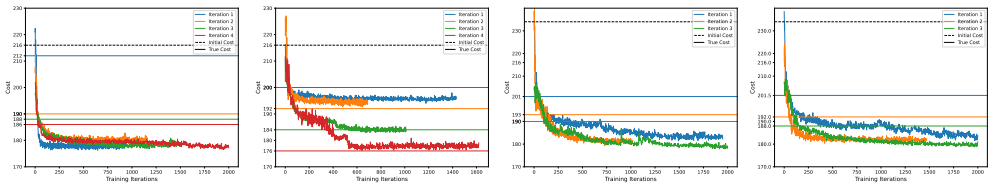
(a) Constrained Policy.

Cost (-Reward) over Iterations			
Policy	$\alpha_d = 0.0$	$\alpha_d = 0.15$	Initial data
Iteration0	x	x	234.0
Iteration1	199.8 ± 3.4	198.7 ± 3.5	x
Iteration2	197.4 ± 3.8	197.0 ± 1.6	x
Iteration3	196.8 ± 15.2	194.0 ± 1.4	x

(b) Soft Constraint.

Table 1: Costs over iterations of IBRL with ($\alpha_d = 0.15$) and without diversity ($\alpha_d = 0.0$) with standard error over 6 repetitions.

the loss reduction over increasing iterations. While the cost is also decreasing without a diversity loss (left column) it does so much slower and converges to a higher cost value. Because the safety constraint is integrated in the policy specification directly, this experiment shows that diversity enables better exploration in the iterative batch scenario. We also note that adding diversity leads to more stable results which can be seen from the lower variation over the experimental repetitions. Figure 5 shows the policy costs for virtual rollouts (curves) as well as the true costs after evaluation (straight lines). In addition to the aforementioned improvement over the batch iterations, we can also see that the predicted costs are much more realistic compared to the true cost when utilizing diversity. We theorize that this is due to the improved quality of the simulation model because it is trained on more diverse trajectory data. By that, model bias decreases.



(a) $\alpha_d = 0.0$ (b) $\alpha_d = 0.15$ (a) $\alpha_d = 0.0$ (b) $\alpha_d = 0.15$

Figure 5: Constrained Policy. Figure 6: Soft Constrain.

Figure 7: Cost loss across iterations of IBRL with soft constraint for different iterations. The straight lines depict the true costs in deployment per batch iteration, while curves the costs based on the learned transition model over training. Dashed line corresponds to the cost of initial batch.

Experiment 2: Safety as soft constraint

We summarize the main finding of this experiment in Table 1b. We observe that diversity accelerates the loss reduction over increasing iterations, however, the difference is more modest compared to the previous experiment. Still, we notice that the diversified policies lead to a much more robust procedure: the variation over the experiment repetitions is significantly lower than in the no-diversity scenario. The results in Figure 6 show that there is no significant difference in model bias in both scenarios. We believe the reason for this is that our proposed safety mechanism Eq. (6) implicitly encourages diversity. In the case of diverse policies, the unnormalized likelihood will expand, so the impact of the safety criterion will naturally decrease over the batch iterations.

5 Conclusion

In this paper, we presented an algorithm for iterative batch reinforcement learning based on model-based policy search. We extended the aforementioned method via a safety mechanism using two distinct approaches and incorporated diversity to exploit the iterative nature of the problem. Our experiments demonstrate the effectiveness of using an iterative process in an offline reinforcement learning setting to enhance policy learning. Moreover, incorporating diversity provides targeted improvements to the policies with each iteration compared to setups without diversity.

References

[1] P. Swazinna, S. Udluft, and T. Runkler. Overcoming model bias for robust offline deep reinforcement learning. *Engineering Applications of Artificial Intelligence*, 104:104366, 2021.

[2] T. Yu, A. Kumar, R. Rafailov, A. Rajeswaran, S. Levine, and C. Finn. Combo: Conservative offline model-based policy optimization. *Advances in neural information processing systems*, 34:28954–28967, 2021.

[3] D. Hein, S. Depeweg, M. Tokic, S. Udluft, A. Hentschel, T. A. Runkler, and V. Sterzing. A benchmark environment motivated by industrial control problems. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8. IEEE, 2017.

[4] T. Matsushima, H. Furuta, Y. Matsuo, O. Nachum, and S. Gu. Deployment-efficient reinforcement learning via model-based offline optimization. *arXiv preprint arXiv:2006.03647*, 2020.

[5] X. Hu, Y. Ma, C. Xiao, Y. Zheng, and Z. Meng. In-sample policy iteration for offline reinforcement learning. *arXiv preprint arXiv:2306.05726*, 2023.

[6] L. Zhang, L. Tedesco, P. Rajak, Y. Zemmouri, and H. Brunzell. Active learning for iterative offline reinforcement learning. In *NeurIPS 2023 Workshop on Adaptive Experimental Design and Active Learning in the Real World*, 2023. URL https://www.amazon.science/publications/active-learning-for-iterative-offline-reinforcement-learning.

[7] D. Ernst, M. Glavic, P. Geurts, and L. Wehenkel. Approximate value iteration in the reinforcement learning context. application to electrical power system control. *International Journal of Emerging Electric Power Systems*, 3(1), 2005.

[8] M. Riedmiller. Neural fitted Q iteration–first experiences with a data efficient neural reinforcement learning method. In *European Conference on Machine Learning*, pages 317–328. Springer, 2005.

[9] M. Riedmiller, T. Gabel, R. Hafner, and S. Lange. Reinforcement learning for robot soccer. *Autonomous Robots*, 27(1):55–73, 2009.

[10] S. Lange, T. Gabel, and M. Riedmiller. Batch reinforcement learning. In *Reinforcement Learning*, pages 45–73. Springer, 2012.

[11] D. Hein, A. Hentschel, T. A. Runkler, and S. Udluft. Reinforcement learning with particle swarm optimization policy (PSO-P) in continuous state and action spaces. *International Journal of Swarm Intelligence Research (IJSIR)*, 7(3):23–42, 2016.

[12] S. Depeweg, J. M. Hernández-Lobato, F. Doshi-Velez, and S. Udluft. Learning and policy search in stochastic dynamical systems with bayesian neural networks. *International Conference on Learning Representations*, 2017.

[13] D. Hein, S. Udluft, and T. A. Runkler. Interpretable policies for reinforcement learning by genetic programming. *Engineering Applications of Artificial Intelligence*, 76:158–169, 2018.

[14] S. Depeweg, J.-M. Hernandez-Lobato, F. Doshi-Velez, and S. Udluft. Decomposition of uncertainty in bayesian deep learning for efficient and risk-sensitive learning. In *International Conference on Machine Learning*, pages 1184–1193. PMLR, 2018.

[15] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller. Deterministic policy gradient algorithms. In *International conference on machine learning*, pages 387–395. PMLR, 2014.

[16] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[17] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

[18] S. Fujimoto, D. Meger, and D. Precup. Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*, pages 2052–2062. PMLR, 2019.

[19] A. Kumar, J. Fu, M. Soh, G. Tucker, and S. Levine. Stabilizing off-policy Q-learning via bootstrapping error reduction. In *Advances in Neural Information Processing Systems*, pages 11761–11771, 2019.

[20] Y. Wu, G. Tucker, and O. Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

[21] A. Kumar, A. Zhou, G. Tucker, and S. Levine. Conservative Q-learning for offline reinforcement learning. *arXiv preprint arXiv:2006.04779*, 2020.

[22] S. Fujimoto and S. S. Gu. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021.

[23] I. Kostrikov, A. Nair, and S. Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

[24] N. Y. Siegel, J. T. Springenberg, F. Berkenkamp, A. Abdolmaleki, M. Neunert, T. Lampe, R. Hafner, N. Heess, and M. Riedmiller. Keep doing what worked: Behavioral modelling priors for offline reinforcement learning. *arXiv preprint arXiv:2002.08396*, 2020.

[25] Y. Jin, Z. Yang, and Z. Wang. Is pessimism provably efficient for offline rl? In *International Conference on Machine Learning*, pages 5084–5096. PMLR, 2021.

[26] M. Yin and Y.-X. Wang. Towards instance-optimal offline reinforcement learning with pessimism. *Advances in neural information processing systems*, 34:4065–4078, 2021.

[27] M. Deisenroth and C. E. Rasmussen. PILCO: A model-based and data-efficient approach to policy search. In *28th International Conference on Machine Learning (ICML-11)*, pages 465–472, 2011.

[28] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7559–7566. IEEE, 2018.

[29] T. Yu, G. Thomas, L. Yu, S. Ermon, J. Y. Zou, S. Levine, C. Finn, and T. Ma. MOPO: Model-based offline policy optimization. In *Advances in Neural Information Processing Systems*, volume 33, pages 14129–14142, 2020.

[30] R. Kidambi, A. Rajeswaran, P. Netrapalli, and T. Joachims. MOReL: Model-based offline reinforcement learning. *arXiv preprint arXiv:2005.05951*, 2020.

[31] J. Li, X. Hu, H. Xu, J. Liu, X. Zhan, and Y.-Q. Zhang. Proto: Iterative policy regularized offline-to-online reinforcement learning. *arXiv preprint arXiv:2305.15669*, 2023.

[32] M. S. Mark, A. Ghadirzadeh, X. Chen, and C. Finn. Fine-tuning offline policies with optimistic action selection. In *Deep Reinforcement Learning Workshop NeurIPS 2022*, 2022. URL https://openreview.net/forum?id=ELmiPlCOSw.

[33] A. Nair, A. Gupta, M. Dalal, and S. Levine. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

[34] S. Lee, Y. Seo, K. Lee, P. Abbeel, and J. Shin. Offline-to-online reinforcement learning via balanced replay and pessimistic q-ensemble. In *Conference on Robot Learning*, pages 1702–1712. PMLR, 2022.

[35] X. Hu, Y. Ma, C. Xiao, Y. Zheng, and J. Hao. Iteratively refined behavior regularization for offline reinforcement learning. 2023.

[36] J. Schmidhuber. A possibility for implementing curiosity and boredom in model-building neural controllers. In *Proc. of the international conference on simulation of adaptive behavior: From animals to animats*, pages 222–227, 1991.

[37] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pages 2778–2787. PMLR, 2017.

[38] B. Eysenbach, A. Gupta, J. Ibarz, and S. Levine. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.

[39] Z.-W. Hong, T.-Y. Shann, S.-Y. Su, Y.-H. Chang, T.-J. Fu, and C.-Y. Lee. Diversity-driven exploration strategy for deep reinforcement learning. *Advances in neural information processing systems*, 31, 2018.

[40] D. Yarats, D. Brandfonbrener, H. Liu, M. Laskin, P. Abbeel, A. Lazaric, and L. Pinto. Don't change the algorithm, change the data: Exploratory data for offline reinforcement learning. *arXiv preprint arXiv:2201.13425*, 2022.

[41] N. Lambert, M. Wulfmeier, W. Whitney, A. Byravan, M. Bloesch, V. Dasagi, T. Hertweck, and M. Riedmiller. The challenges of exploration for offline reinforcement learning. *arXiv preprint arXiv:2201.11861*, 2022.

[42] Z.-W. Hong, T.-Y. Shann, S.-Y. Su, Y.-H. Chang, T.-J. Fu, and C.-Y. Lee. Diversity-driven exploration strategy for deep reinforcement learning. *Advances in neural information processing systems*, 31, 2018.

[43] J. Parker-Holder, A. Pacchiano, K. M. Choromanski, and S. J. Roberts. Effective diversity in population based reinforcement learning. *Advances in Neural Information Processing Systems*, 33:18050–18062, 2020.

[44] S. Kumar, A. Kumar, S. Levine, and C. Finn. One solution is not all you need: Few-shot extrapolation via structured maxent rl. *Advances in Neural Information Processing Systems*, 33: 8198–8210, 2020.

[45] P. Swazinna, S. Udluft, and T. Runkler. User-interactive offline reinforcement learning. *International Conference on Learning Representations*, 2023.

[46] D. Brandfonbrener, W. Whitney, R. Ranganath, and J. Bruna. Offline rl without off-policy evaluation. *Advances in neural information processing systems*, 34:4933–4946, 2021.

[47] N. K. Sinha and M. P. Griscik. A stochastic approximation method. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-1(4):338–344, 1971. doi:10.1109/TSMC.1971.4308316.

[48] L. Bottou, F. E. Curtis, and J. Nocedal. Optimization methods for large-scale machine learning. *SIAM review*, 60(2):223–311, 2018.

[49] K. Mangalam, H. Girase, S. Agarwal, K.-H. Lee, E. Adeli, J. Malik, and A. Gaidon. It is not the journey but the destination: Endpoint conditioned trajectory prediction. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pages 759–776. Springer, 2020.

[50] C. Choi, J. H. Choi, J. Li, and S. Malla. Shared cross-modal trajectory prediction for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 244–253, 2021.

[51] H. Zhao, J. Gao, T. Lan, C. Sun, B. Sapp, B. Varadarajan, Y. Shen, Y. Shen, Y. Chai, C. Schmid, et al. Tnt: Target-driven trajectory prediction. In *Conference on Robot Learning*, pages 895–904. PMLR, 2021.

A Dataset visualization for industrial benchmark

A.1 Random bounded dataset

Data collected by uniform sampling from the action space such that the states are restricted to lie within the safety bound [30, 70]. The samples are collected by generating five rollouts of horizon 200 each.

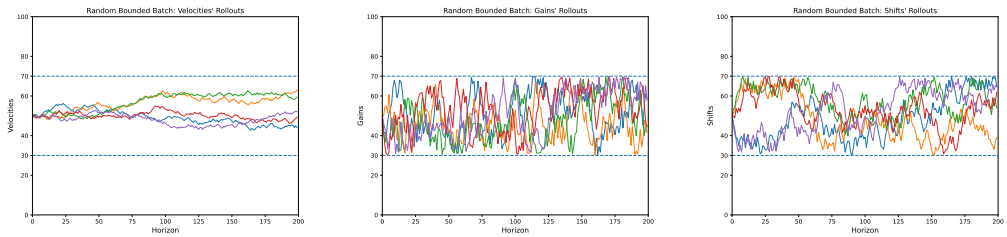


Figure 8: Bounded dataset for the industrial benchmark. Shown are trajectories of Velocity (left), Gain (middle) and Shift (right).

A.2 Medium Policy with Randomness

Wwe use a simple policy (referred to as medium) to generate the initial batch, which tries to navigate to a fixed point in the state space, as also done in [45, 13]. The batch is collected by randomly sampling a starting state in the bound [0, 100] and subsequently following:

$$\pi_{\beta_{medium}}(\mathbf{s_t}) = \begin{cases} 50 - v_t \\ 50 - g_t \\ 50 - h_t \end{cases} \tag{12}$$

Additionally, the policy is augmented with 33% randomness.

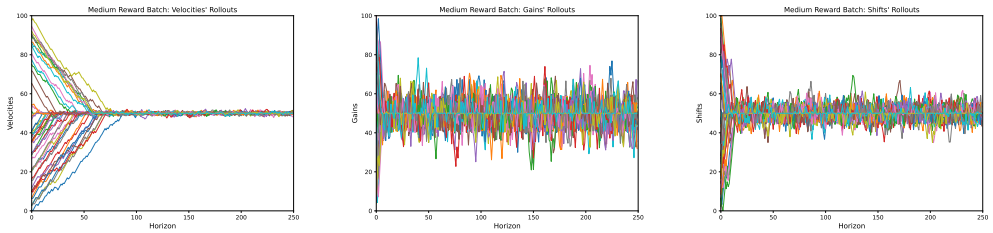


Figure 9: Velocity, gain and shift rollouts generated by following the medium behavior policy starting from a random state.