Proceedings of Machine Learning Research vol xxx:1–10, *Equal contribution. Author ordering determined randomly through a coin flip

Model-Predictive Control via Cross-Entropy and Gradient-Based Optimization

Homanga Bharadhwaj*, Kevin (Cheng) Xie*, and Florian Shkurti
Department of Computer Science, University of Toronto
University of Toronto Robotics Institute
Vector Institute

Abstract

Recent works in high-dimensional model-predictive control and model-based reinforcement learning with learned dynamics and reward models have resorted to population-based optimization methods, such as the Cross-Entropy Method (CEM), for planning a sequence of actions. To decide on an action to take, CEM conducts a search for the action sequence with the highest return according to the dynamics model and reward. Action sequences are typically randomly sampled from an unconditional Gaussian distribution and evaluated on the environment. This distribution is iteratively updated towards action sequences with higher returns. However, this planning method can be very inefficient, especially for high-dimensional action spaces. An alternative line of approaches optimize action sequences directly via gradient descent, but are prone to local optima. We propose a method to solve this planning problem by interleaving CEM and gradient descent steps in optimizing the action sequence. Our experiments show faster convergence of the proposed hybrid approach, even for high-dimensional action spaces, and less getting stuck at local minima. Code accompanying the paper is available in this[1] repo.

1. Introduction

High-dimensional, nonlinear Model-Predictive Control (MPC) and Model-based Reinforcement Learning (MBRL) have seen significant progress over the last years, the task being to first learn a dynamics and a reward model of the environment and then plan using the learned models. While a number of recent approaches Hafner et al. (2018); Sharma et al. (2019); Chua et al. (2018); Wang and Ba (2019) have developed efficient techniques for learning these models in MBRL, fewer papers Amos and Yarats (2019); Srinivas et al. (2018) have investigated the planning problem. Instead, many state-of-the-art MBRL approaches perform planning either using the Cross-Entropy Method (CEM) Rubinstein (1997); Chua et al. (2018); Kobilarov (2012), or via Model-Predictive Path Integral (MPPI) Williams et al. (2016). Both these approaches are population-based search heuristics that sample random actions, execute them under the currently learned model, obtain the sum of rewards, and update the sampling distribution to increase the probability of higher reward action sequences. In MPC, the first action of the sequence is executed in the environment, the remaining planned actions are typically discarded, and the search procedure repeats.

1. https://github.com/homangab/gradcem

Many current MBRL approaches do not leverage gradients through the model, which are cheaply available, and resort to inefficient optimization, particularly in high dimensions, whereas gradient-based planning converges faster.

In this paper we combine the two methods, to take advantage of the convergence speed of gradient-based planning and the broader search, multi-extremum optimization performed by CEM. Gradient based optimization is one of the main approaches for a number of high-dimensional non-convex optimization problems in machine learning, yet it has not been widely adopted in planning problems due to the issue of vanishing or exploding gradients. We investigate situations where gradient-based planning fails, due to the sensitivity of shooting methods and imperfect models, and provide a simple way to mitigate it: we interleave CEM steps with gradient-based optimization, so that the latter can inform the update of the sampling distribution in the former.

2. Preliminaries

In this section we discuss the CEM method, and some of the key issues in gradient-based planning.

2.1. The Cross-Entropy Method for Planning

In model-based reinforcement learning and model predictive control, a model of the environment and reward is learned from real transitions in the environment. To select an action, MPC searches for an optimal action sequence under the learned model and executes the first action of that sequence, discarding the remaining actions. Typically this search is repeated after every step in the real environment, to account for any prediction errors by the model and to get feedback from the environment. In many works this planning step is done using the so-called 'Cross-Entropy Method' (CEM) Chua et al. (2018); Hafner et al. (2018); Wang and Ba (2019). CEM samples action sequences from a time-evolving distribution, usually a diagonal Gaussian $a_{t:t+H} \sim \mathcal{N}(\mu_{t:t+H}, \sigma^2_{t:t+H}\mathbb{I})$. These open-loop action sequences are simulated using the learned dynamics model to obtain approximate resulting state sequences and rewards. By repeatedly sampling random action trajectories, evaluating them under the model, and re-fitting the sampling distribution to the best K trajectories, a new Gaussian distribution $\mu_{t:t+H}, \sigma^2_{t:t+H}$ of actions for the current time-step is obtained.

Sampling random action sequences in this manner and evaluating the sum of rewards from them is very costly in practice because it does not leverage any implicit structure in the planning problem, and does not take advantage of the fact that gradients through the model can in fact be used to direct the search procedure, instead of naively sampling random action sequences.

2.2. Gradient-based Planning

Gradient-based methods for planning typically correspond to backpropagating derivatives of a cumulative loss (or reward) function with respect to actions for updating the sequence of actions iteratively through gradient descent. In Henaff et al. (2017), the gradients of the cumulative reward with respect to actions are computed by differentiating through the learned reward and forward dynamics models. In Srinivas et al. (2018), gradients of the inner loss of the Gradient-Descent Planner (GDP) with respect to actions are computed in the latent space, by differentiating through a learned latent forward dynamics model. The ultimate aim is to update actions through an iterative gradient descent approach:

$$\bar{a}_{0:H}^{(k)} := \bar{a}_{0:H}^{(k-1)} + \alpha \nabla_{\bar{a}} \bar{R}(\bar{a}_{0:H}^{(k-1)}, \bar{s}_{0:H}^{(k-1)}), \qquad k = 1, 2, ..., K$$

Here, H denotes the time-horizon of the episode, a denotes action and s denotes state. One of the most important drawbacks of gradient descent for non-convex optimization is that the optimization procedure is only guaranteed to converge to a local optima, not the global optima. In MPC for MBRL, these planners may converge to sub-optimal plans. In addition, for a long horizon H, there is the exploding and vanishing gradients problem which must be taken care of during optimization. An important point to note is that when the action dimension increases, CEM becomes highly inefficient and requires significantly more optimization epochs due to a blow-up of the search space , whereas there is only a slight increase (one gradient dimension) in computational burden for gradient descent.

3. Approach

Our approach is based on the motivation that in MPC, model gradients should be effectively used for conducting a more informed search during the planning phase. In the subsequent subsections, we describe a simple technique for doing this in practice.

3.1. CEM+Gradient Descent

Since gradient descent is prone to getting stuck at local optima and in practice requires sufficiently different random initializations to alleviate this, we consider a very simple idea - interleave CEM steps with gradient descent on the samples to locally refine each plan. This method incorporates gradients through the model, thereby yielding more refined action sequences that can be used to update the CEM sampling distribution faster. Instead of resampling all plans, we choose to keep the top K plans from the previous iteration to continue optimizing them via gradient descent.

Let f_ϕ denote the learned dynamics model, r_ψ the learned reward model, a_h the action at time-step h, s_h the state of the environment at time-step h, and H the planning horizon. Let $\mathcal{N}(\mu_{0:H}^{(t)}, \Sigma_{0:H}^{(t)})$ denote the CEM sampling distribution from which action sequences are sampled in the t^{th} CEM iteration. Here our notation for $\mathcal{N}$, refers to H independent multivariate Gaussian distributions. We arbitrarily set the parameters $(\mu_{0:H}^{(0)} = 0, \Sigma_{0:H}^{(0)} = I)$ of this distribution initially. At the beginning of each CEM iteration, the planner first samples multiple (G) random action sequences:

$$\{(a_0^{(t)}, ..., a_H^{(t)})_g\}_{g=1}^{G} \sim \mathcal{N}(\mu_{0:H}^{(t)}, \Sigma_{0:H}^{(t)})$$

We next evaluate the cumulative reward obtained from each of these action sequences, under the current learned dynamics model f_ϕ and the current reward model r_ψ:

$$R_g^{(t)} = \sum_{h=1}^{H} r_\psi(s_h^{(t)}) \qquad s_h^{(t)} = f_\phi(s_{h-1}^{(t)}, a_{h-1}^{(t)}), \qquad \forall g = 1, .., G$$

Here, t indexes the CEM iterations. Now, treating these initial sampled plans as initialization of the gradient-descent procedure, we perform J steps of gradient descent on all of the sequences. In all of our experiments to ensure fair comparison to CEM, we set $J = 1$.

$$(a_0^{(t)}, ..., a_H^{(t)})_g^{j+1} \longleftarrow (a_0^{(t)}, ..., a_H^{(t)})_g^{j} - \beta \nabla_{a_{0:H}^{(t)}} R_g^{(t)}, \qquad \forall g = 1, .., G, j = 1, ..., J$$

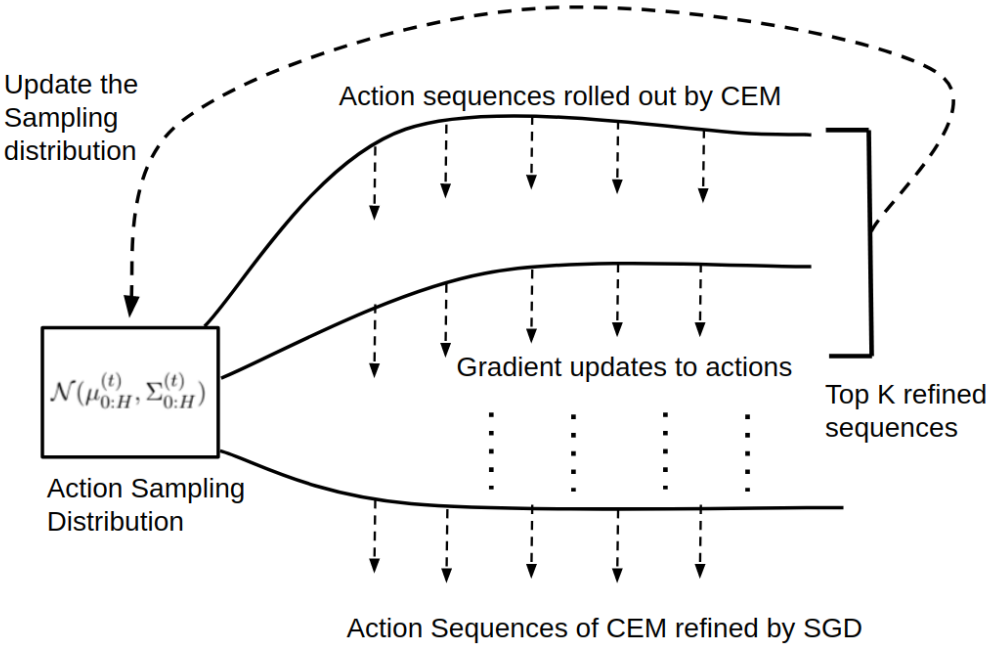


Figure 1: Schematic of the proposed approach. Initial sequences of actions sampled from the CEM sampling distribution are refined by a few gradient descent updates, denoted by downward arrows. Then the action sequences are evaluated under the current model. The top K action sequences (i.e. the top K ones with maximum sum of discounted rewards) are used to refine the CEM sampling distribution from which the actions are sampled. The sampling distribution is typically assumed to be a Gaussian, and is so for our paper as well.

Then we update the parameters of our proposal (sampling) distribution $\mathcal{N}(\mu_{0:H}^{(t+1)}, \Sigma_{0:H}^{(t+1)})$ to match the top K updated action sequences:

$$\mu_{0:H}^{(t+1)} \longleftarrow \text{Mean}(\{(a_0^{(t)}, ..., a_H^{(t)})_k\}_{k=1}^{K})$$

$$\Sigma_{0:H}^{(t+1)} \longleftarrow \text{Variance}(\{(a_0^{(t)}, ..., a_H^{(t)})_k\}_{k=1}^{K})$$

Finally, we replace the bottom $G - K$ action sequences, with samples from the updated proposal distribution. After T iterations of this, the remaining action sequence with the highest reward is returned. Our approach is summarized in Algorithm 1.

4. Experiments

Through the experiments we aim to demonstrate the benefits and pitfalls of CEM and gradient descent, and demonstrate the efficacy of the proposed approach. The gradient-based planner baseline, which is hereafter referred to as `Grad`, samples G initial samples and seperately performs T gradient steps on them. To better demonstrate our claims, we created a toy environment, the details of which are described in the next sub-section. Code for the experiments is available in this repo https://github.com/homangab/gradcem.

4.1. Details of the toy environment

Algorithm 1 Grad+CEM Algorithm (The proposed approach)

Initialize environment transitions data $\mathcal{D} \leftarrow \{\}$
for *trial m=1 to M* **do**
  Train dynamics model $s_{1:H} = f(a_{0:H}, s_0)$, reward model $r(s_{1:H})$ with $\mathcal{D}$
  for *environment step l=1 to L* **do**
    Sample G initial plans $\{{a_{0:H}}^{(0)}\}_{g=1}^{G}$ from $\mathcal{N}(\mu_{0:H}^{(0)} = \mathbf{0}, \Sigma_{0:H}^{(0)} = I)$
    Sample a random environment state s_0
    for *CEM iteration t=1 to T* **do**
      for *gradient descent step j=1 to J* **do**
        $\tau^{(t)} = f(\{{a_{0:H}}^{(t)}\}_g, s_0)$ // $\tau^{(t)}$ is a state sequence
        Calculate model returns for each plan $R_g^{(t)} = r(\tau^{(t)})$
        Update $\{{a_{0:H}}^{(t)}\}_{g=1}^{G}$ by maximizing total model returns via Gradient Descent
      end
      Sort $\{{a_{0:H}}^{(t)}\}_{g=1}^{G}$ based on total model returns on step J
      Update $(\mu_{0:H}^{(t)}, \Sigma_{0:H}^{(t)})$ to match the top K action sequences
      Replace bottom $G - K$ action sequences with samples from $\mathcal{N}(\mu_{0:H}^{(t)}, \Sigma_{0:H}^{(t)})$
    end
    Execute first action from the highest model return action sequence
    Record real transition in $\mathcal{D}$
  end
end

To consider the planning problem in isolation, we created a toy environment in which we have access to ground truth gradients through the dynamics model. The agent controls a mass in an N dimensional space by applying forces at each time step. Fig. 2 shows a 2D projection of the environment. The task is to move towards high reward regions of the state-space (red region) from the blue region. The black lines and dots show 2D projections of multiple rolled out trajectories starting from the origin. The fluorescent green region denotes an obstacle with soft contact. The soft contact is modeled as a repulsive spring force at every time step that increases proportionally to the penetration depth of the agent into the obstacle. The "hardness" of the contact can be tuned by the spring constant. The larger the spring constant is, the stronger is the repulsion force.

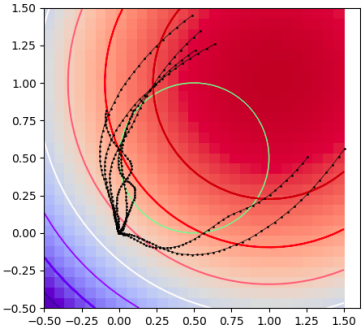
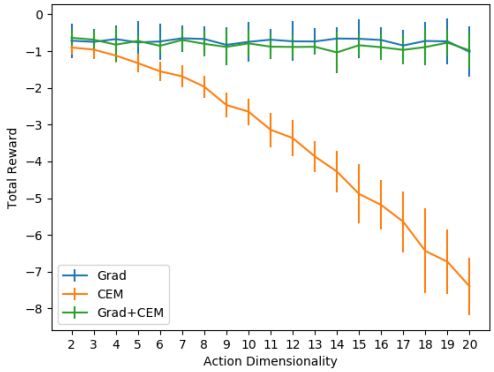
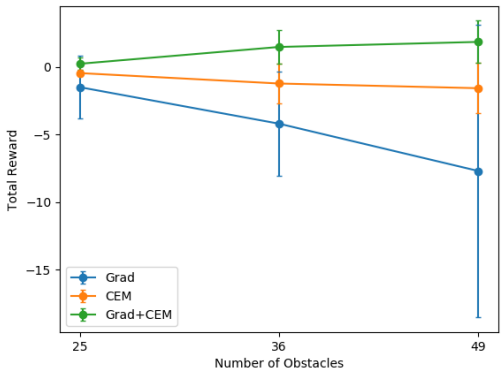


Figure 2: Illustrative diagram of the toy environment. The black paths are 2D projections of the path of a point mass. Red denotes high reward and blue denotes low reward regions. The green circle is an obstacle with soft contact.

For all the toy environment results, all the methods used $T = 10$ number of iterations. To make a fair comparison we set the number of inner gradient steps per iteration $J = 1$ for Grad+CEM. All methods used $G = 20$ sampled plans at each iteration. CEM and Grad+CEM both select the top $K = 4$ plans at each iteration.



(a) Performance of different planners for single obstacle, soft contact case as the environment dimensionality (both states and actions) is increased.



(b) Multi-object, hard contact (spring constant 10 times larger) 2D case as the number of obstacles is increased.

Figure 3: Total reward obtained by CEM vs Grad vs Grad+CEM planners on the toy environment. The total reward is averaged over 50 runs for each data point and error bars denote the standard deviation of those runs. The error bars correspond to one SD for optimization with fifty random seeds. Higher is better.

4.2. Results in high dimensions

In the toy environment shown in Fig. 2, we hypothesize that increasing the dimensions of the action space is likely to deteriorate the performance of a vanilla CEM planner but not of a gradient descent based planner. Fig. 3a shows a comparative analysis of total reward collected in the environment as the number of action dimensions are increased.

It is evident that there is a significant drop in the performance of the CEM based planner with increasing action dimensionality. For optimization CEM utilizes $O(1)$ information per rollout (just the aggregate reward), while Gradient based Planning makes use of $O(D)$ information through the vector of gradients. Hence, the information used scales linearly with D and renders the gradient-based optimization tractable.

4.3. When gradient based optimization fails

Fig. 4 shows an experimental scenario that involves multiple obstacles with non-smooth contact (the spring constant is set 10 times higher). Here it is evident that the purely gradient based approach does not succeed and gets stuck in some local optima. The main reason for this is that the non-smooth contact results in discontinuous gradients (e.g. consider the edge of a table. There is a sudden jump in the magnitude of the gradients when moving from one edge to the other) which make learning difficult. To alleviate this, we show in Fig. 3b that interleaving CEM and gradient-descent update steps helps learn better plans. Note that we decrease the size of obstacles as we increase their number in order to pack them into the same space and that is why it is possible for Grad+CEM to do better as the number of obstacles increases.

4.4. MPC in Model-Based RL (MuJoCo based environments)

In this section, we consider the complete MBRL problem of learning dynamics+reward models and using the learned models to do planning. In particular, we consider the SOTA Planet Hafner et al.

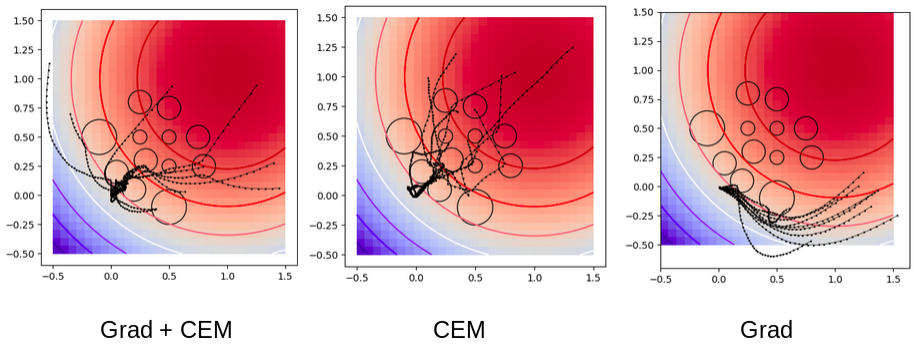
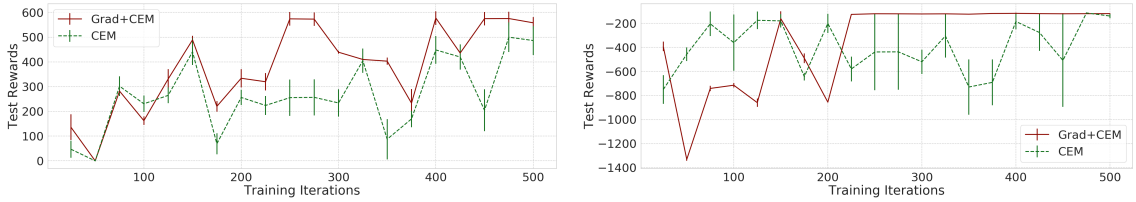


Figure 4: Illustration of trajectories of different algorithms in the multiple obstacles scenario



(a) OpenAI Gym Half-Cheetah Brockman et al. (2016)

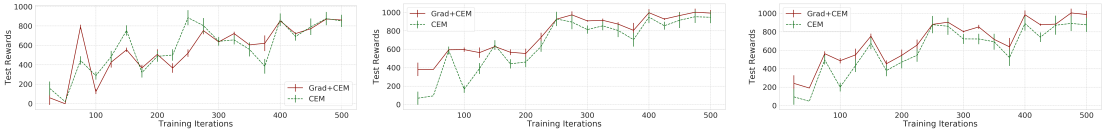
(b) OpenAI Gym Pendulum Brockman et al. (2016)

Figure 5: Variation of rewards at test time during the course of training. OpenAI Gym (a) Pendulum and (b) Half-Cheetah environments. CEM is the default Planet Hafner et al. (2018) algorithm that plans through CEM. Grad+CEM is the version of Planet that plans using the proposed Grad+CEM scheme. The error bars correspond to training and evaluation with three random seeds. Higher is better.

(2018) model and replace the CEM based planning module with the proposed Grad+CEM approach. Fig. 5 shows that for two different OpenAI Gym Brockman et al. (2016) environments, Pendulum and Half-Cheetah, while the default CEM based planning scheme struggles to converge in terms of the test rewards, incorporating model gradients for planning ensures a quick and reliable convergence. So, from the experiments we conclude that the *Grad+CEM* scheme helps in converging to higher rewards faster, with fewer optimization iterations. For Fig. 5b and Fig. 5a, for a pairwise t-test between the two variants CEM and Grad+CEM, we respectively obtain t-values 2.47564 and 3.15453, and p-values respectively 0.019186 and 0.004152. Both results are significant at $p < 0.05$. In the Pendulum environment, the pendulum starts at a random position, and the goal is to swing it up so that it stays upright. In the Half-Cheetah environment, the agent gets rewarded for moving a fast as possible and maintaining proper gait (not toppling over). In both these environments, the input to the policy are high dimensional rendered images, which make the tasks challenging.

5. Related Works

Our paper is broadly based on the theme of model-based reinforcement learning (MBRL) Chua et al. (2018); Hafner et al. (2018), where the idea is to learn a dynamics model of the world and plan using the learned dynamics model. For high dimensional inputs like images, the dynamics are typically learned in a learnt latent space Hafner et al. (2018). Most current MBRL approaches use some version of the so-called 'Cross-Entropy Method' (CEM) for doing a random population based search of plans given the current model Wang and Ba (2019); Hafner et al. (2018). Some papers Sharma et al. (2019)



(a) DM Control Suite Cartpole (b) DM Control Suite WalkerWalk (c) DM Control Suite Reacher

Figure 6: Variation of rewards at test time during the course of training. DeepMind Control Suite Tassa et al. (2018) (a) Cartpole-SwingUp, (b) Walker-Walk, and (c) Reacher environments. CEM is the default Planet Hafner et al. (2018) algorithm that plans through CEM. Grad+CEM is the version of Planet that plans using the proposed Grad+CEM scheme. The error bars correspond to training and evaluation with three random seeds. Higher is better.

use other population based search approaches like Model-Predictive Path Integral (MPPI) Williams et al. (2016) for planning. A recent paper Okada and Taniguchi (2019) discusses how in the control as inference framework, the two approaches, CEM, and MPPI are very similar, and differ only with respect to the reward function $r(\tau)$. Both these random shooting approaches are very costly and take a long time to converge because they involve sampling random action sequences and evaluating them under the current model to determine the high performing sequences. Although Wang and Ba (2019) introduces the idea of performing the CEM search in the parameter space of a distilled policy, it still is very costly and requires a lot of samples for convergence.

Gradient-descent based optimization methods have been successful in a wide range of machine learning domains Finn et al. (2017), but for planning, there are very few papers that have been able to successfully perform gradient-descent based planning. Universal Planning Networks (UPNs) Srinivas et al. (2018) optimizes action sequences in a latent space such that the optimized sequence matches expert demonstrations of actions. So the approach requires high quality expert data, and is based on imitation learning, not end-to-end reinforcement learning. SGD for model predictive control is also done in Henaff et al. (2017) but without a diverse initialization it can lead to local optima. Hence the approach is limited to simple grid worlds, and cannot scale to more challenging robotic tasks.

In the context of model-free reinforcement learning, Pourchot and Sigaud (2018) also introduce the idea of interleaving CEM and policy gradient steps in optimizing in the parameter space of policies. We show how interleaving CEM and gradient descent steps can be used as an effective planner for model predictive control in the context of model based reinforcement learning.

Direct collocation approaches for control, address some of the ill-conditioning of shooting methods, and avoid backpropagating the model through time, by parameterizing the state and action sequences and optimizing both jointly. In this setting, Subbarao and Shippey (2009) propose initializing the collocation optimization from a solution found by a genetic algorithm similar to CEM. However, they do not interleave the two optimizations and the collocation method requires parameterizing state trajectories with analytic functions such as splines.

6. Limitations and Future Works

One of the main directions for future works is to investigate the implications of model-bias in the planning scheme. In MBRL, one of the primary issues leading to a suboptimal plan is that the planner exploits model bias of an imperfectly learned model Wang et al. (2019). So, for better planning, we also need to develop better strategies for learning the dynamics model itself. Some papers Chua et al. (2018); Kurutach et al. (2018) aim to learn a better model by maintaining an ensemble of neural network models. This helps model epistemic uncertainty, but an ensemble of networks for the

dynamics is difficult to scale to image-based environments without introducing a huge computational burden during training.

Another effective direction for tackling model-bias is by learning dynamics models conditioned on some latent variables, instead of trying to learn a global dynamics model. A recent paper, DADS Sharma et al. (2019) does this by conditioning the dynamics model on latent 'skills' and the main idea is to learn smaller behavior-specific dynamics models instead of trying to learn a global dynamics model. The latent 'skills' are basically an abstraction for the low-level action sequences that get executed in the environment. However, DADS does not leverage the latent abstractions for planning, it uses them only for learning the dynamics model. One potential extension of our approach would be to use such latent variable models for planning as well, by backpropagating gradients wrt the latent variables through the model, in order to update the low-level actions.

7. Conclusion

In this paper we investigate model predictive control in the context of model based reinforcement learning. We show that the ubiquitous approach of CEM struggles as the dimensionality of the environment increases, which is an important issue as we scale these methods to real world control problems. Gradient-descent based planning is conveniently applicable in continuous control problems, especially since the learned dynamics models are typically parameterized with differentiable functions. We show that these gradient planners can scale better to higher dimensional problems owing to the gradient signal also scaling with action dimensionality. However, we demonstrate that in environments with discontinuous dynamics and many local optima, pure gradient descent can fail compared to CEM. Finally, we propose a simple method that interleaves CEM and gradient descent updates which is both able to scale to higher dimensions and performs robustly on the discontinuous dynamics setting.

Acknowledgement

We thank David Duvenaud and Animesh Garg for helpful discussions with early versions of this work, and Vector Institute, Toronto for computing infrastructure and support.

References

Brandon Amos and Denis Yarats. The differentiable cross-entropy method. *arXiv preprint arXiv:1909.12830*, 2019.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*, pages 4754–4765, 2018.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR. org, 2017.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. *arXiv preprint arXiv:1811.04551*, 2018.

Mikael Henaff, William F Whitney, and Yann LeCun. Model-based planning with discrete and continuous actions. *arXiv preprint arXiv:1705.07177*, 2017.

Marin Kobilarov. Cross-entropy motion planning. *The International Journal of Robotics Research*, 31(7):855–871, 2012.

Thanard Kurutach, Ignasi Clavera, Yan Duan, Aviv Tamar, and Pieter Abbeel. Model-ensemble trust-region policy optimization. *arXiv preprint arXiv:1802.10592*, 2018.

Masashi Okada and Tadahiro Taniguchi. Variational inference mpc for bayesian model-based reinforcement learning. *arXiv preprint arXiv:1907.04202*, 2019.

Aloïs Pourchot and Olivier Sigaud. Cem-rl: Combining evolutionary and gradient-based methods for policy search. *arXiv preprint arXiv:1810.01222*, 2018.

Reuven Y Rubinstein. Optimization of computer simulation models with rare events. *European Journal of Operational Research*, 99(1):89–112, 1997.

Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-aware unsupervised discovery of skills. *arXiv preprint arXiv:1907.01657*, 2019.

Aravind Srinivas, Allan Jabri, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Universal planning networks: Learning generalizable representations for visuomotor control. In *International Conference on Machine Learning*, pages 4739–4748, 2018.

Kamesh Subbarao and Brandon M Shippey. Hybrid genetic algorithm collocation method for trajectory optimization. *Journal of Guidance, Control, and Dynamics*, 32(4):1396–1403, 2009.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

Tingwu Wang and Jimmy Ba. Exploring model-based planning with policy networks. *arXiv preprint arXiv:1906.08649*, 2019.

Tingwu Wang, Xuchan Bao, Ignasi Clavera, Jerrick Hoang, Yeming Wen, Eric Langlois, Shunshi Zhang, Guodong Zhang, Pieter Abbeel, and Jimmy Ba. Benchmarking model-based reinforcement learning. *arXiv preprint arXiv:1907.02057*, 2019.

Grady Williams, Paul Drews, Brian Goldfain, James M Rehg, and Evangelos A Theodorou. Aggressive driving with model predictive path integral control. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1433–1440. IEEE, 2016.