Model-Based Planning with Energy-Based Models

Yilun Du
MIT CSAIL
yilundu@mit.edu

Toru Lin
MIT CSAIL
torulk@mit.edu

Igor Mordatch
Google Brain
imordatch@google.com

Abstract:
Model-based planning holds great promise for improving both sample efficiency and generalization in reinforcement learning (RL). We show that energy-based models (EBMs) are a promising class of models to use for model-based planning. EBMs naturally support inference of intermediate states given start and goal state distributions. We provide an online algorithm to train EBMs while interacting with the environment, and show that EBMs allow for significantly better online learning than corresponding feed-forward networks. We further show that EBMs support maximum entropy state inference and are able to generate diverse state space plans. We show that inference purely in state space - without planning actions - allows for better generalization to previously unseen obstacles in the environment and prevents the planner from exploiting the dynamics model by applying uncharacteristic action sequences. Finally, we show that online EBM training naturally leads to intentionally planned state exploration which performs significantly better than random exploration.

1 Introduction

Recent advances in reinforcement learning have primarily relied on model-free approaches, and have shown strong performance across a wide range of domains [1, 2, 3, 4, 5]. However, several issues persist in model-free approaches: they have poor sample efficiency [6], and are unable to adapt to new tasks or domains [7]. A key reason for these problems is that in model-free approaches, an agent's policy and its knowledge of environmental dynamics are entangled.

Model-based approaches, on the other hand, separate learning of dynamics model from learning of policies. This means that the dynamics model learned in one domain can then be extracted and applied to other domains and tasks [8], suggesting the better generalizability of model-based methods. We therefore adopt such an approach in this paper, combining planning with learned dynamics models to accomplish a wide variety of tasks.

Most approaches towards planning with dynamics models consider models whose next states are conditioned on both actions and current states. However, such approaches limit models to environments in which the same set of actions are used. Furthermore, planning in action space can lead to the planner exploiting the dynamics models by applying uncharacteristic action sequences that take the agent outside the competence region of the model [9]. In contrast, learning a dynamics model that directly predicts next-state dynamics both generalizes to different action spaces and is less likely to explore risky states. However, directly predicting next state dynamics is difficult, as probability distributions over the next states are difficult to model due to their multi-modal nature. We find the energy-based models (EBMs) are naturally able to capture multi-modal distributions and exhibit maximum entropy sampling of next states that construct diverse plans to reach the goals.

For such approaches to work in an real-world robot learning settings, models must be learned in an online fashion, where dynamics can change abruptly and be heavily correlated. As was also observed in [10] for generative modeling tasks, we find that EBMs are particularly suited to learning in online settings. They significantly outperform corresponding feed-forward networks benchmarks and exhibit robustness to heavily correlated experience.

Furthermore, for online model learning to perform well, it must be able to effectively explore the surrounding environment. We find that training EBMs naturally leads to good exploration that significantly outperforms random exploration.

Our overall contributions in this work are three-fold. First, we propose a framework for online learning with EBMs for planning and show that they perform well under online model learning.

3rd Conference on Robot Learning (CoRL 2019), Osaka, Japan.

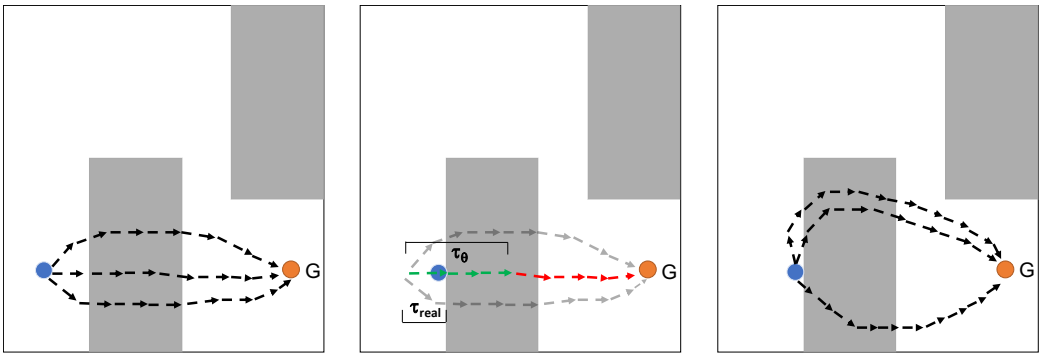

Figure 1: Overview of online training procedure with a EBM where grey areas represent inadmissible regions. Plans from the current observation to goal state is inferred from the EBM (left). A particular plan is chosen and executed until a planned state deviates significantly from an actual state (middle). The EBM is then trained on all real transitions τ_{real} and all planned transitions before the deviation (in green) τ_θ, while transitions afterwards (red) are ignored. A new plan is then generated from the new location of the agent (right)

Second, we show EBMs support the control as inference framework and allow maximum entropy reinforcement learning on next states as opposed to actions, allowing diverse state based planning that shows better generalization than planning conditioned on both states and actions. Finally, we show that online planning with EBMs naturally gives rise to exploration.

2 Related Work

A number of model classes have been explored for model-based planning in robotics and artificial intelligence literature, from feed-forward, recurrent [11], temporal segment [12] and Bayesian [13] neural networks, to locally-linear models [14, 15] and Gaussian processes [16]. By contrast, energy-based models have been an under-explored for model-based planning and it is our aim to showcase the favorable properties this model class exhibits. While work of [17] explored energy-based models of policies for model-free reinforcement learning, we instead use them to model environment dynamics in a model-based setting.

Energy-based models have seen success in other applications, such as natural image modeling [10, 18]. These works noted EBMs for favorable compositionality and online learning properties, which we take advantage of in this work. A number of methods have been used for inference and sampling in EBMs (or equivalently, planning), from Gibbs Sampling [19], to Langevin Dynamics [10], and learned samplers [20]. We have not focused on the choice of sampler/planner in this work, and found method of [21] to work well. Other planning methods, such as those based on direct collocation methods [22, 23] can potentially be used instead.

A common approach to achieve exploration behavior in reinforcement learning has been to use explicit rewards, known as intrinsic motivation [24, 25]. Examples include rewarding empowerment, information gain about model of the dynamics [26], or state space coverage [27, 28]. Maximum entropy models are another approach to induce exploratory behavior [29] and what we rely on this our work as well. We show that contrastive training of EBMs is particularly conducive to exploration.

3 Model-based Planning with Energy-Based Models

In this section, we describe the overall approach towards model-based planning with EBMs. We reformulate planning as inference over a graphical model defined by a composition of EBMs. Inference over the graphical model can be seen as maximum entropy reinforcement learning (see [9] for a review). We first give a background overview of relevant terminology and EBMs, then introduce the graphical model formulation and finally discuss online training of EBMs.

3.1 Energy-Based Models and Terminology

Consider a standard Markov Decision Process (MDP) represented by the tuple $\langle S, A, T, R \rangle$, where S is the set of all possible state configurations, A is the set of actions available to the agent, T is the transition distribution, and R is the reward function. Under this setup, define a state transition pair (s_t, s_{t+1}) between the states at timesteps t and $t+1$. Define $E_\theta(s_t, s_{t+1}) \in \mathbb{R}$ as the energy function, which we parameterize with a deep neural network. We interpret the energy function as unnormalized probability distribution over state transition by defining the distribution as $p_\theta(s_t, s_{t+1}) \propto e^{-E_\theta(s_t, s_{t+1})}$.

To sample from the defined probability distribution, we use Model Predictive Path Integral (MPPI) algorithm [21], which is shown to converge to the full posterior distribution in [30]. The mathematical formulation is shown below, where $x := (s_t, s_{t+1})$:

$$\tilde{x}^k = \sum_i w_i x_i^k, \quad x_i^k \sim N(x^{k-1}, \sigma), \quad w_i = \left(\frac{e^{-E_\theta(x_i^k)}}{\sum_j e^{-E_\theta(x_i^j)}} \right) \tag{1}$$

Other valid inference algorithms that sample from the posterior are also applicable to EBMs, such as Langevin Dynamics [10] or Hamiltonian Monte Carlo (HMC).

We note that this form of sampling makes it easy to add additional constraints to a probability distribution by simply adding the constraint as an energy. To train an EBM, we follow the methodology defined in [10]. We train models by minimizing

$$\mathbb{E}_{x^+ \sim p_D} E_\theta(x^+) - \mathbb{E}_{x^- \sim p_\theta} E_\theta(x^-). \tag{2}$$

Intuitively, doing so decreases the energies of observed transitions (i.e. more likely transitions), and decreases the energies of transitions sampled from the model's distribution (i.e. less likely transitions).

3.2 Planning with Energy-Based Models

In the previous subsection, we described a way to learn state transition models $p_\theta(s_t, s_{t+1})$. We now discuss how to use models to do inference over trajectories. Given a learned model $p_\theta(s_t, s_{t+1})$, we can model the likelihood of a trajectory τ under the model as a product of factors

$$p_\theta(\tau) = p_\theta(s_1, s_2, \ldots, s_T) = \prod_{t=1}^{T-1} p_\theta(s_t, s_{t+1}) \tag{3}$$

$$\propto \exp(-\sum_{t=1}^{T} E(s_t, s_{t+1})) \tag{4}$$

We can likewise do inference across this product using MPPI. Note that we directly sample states rather than actions in our formulation. We generate temporally smooth trajectory perturbations following approach of [31] (where the last two rows of the finite difference matrix A are removed to allow end states of trajectories to be unconstrained).

Given a particular fixed goal state g and start state s_1, we can do inference over intermediate states by sampling from the probability distribution among state transitions between s_2, ..., s_T

$$p_\theta(s_2, \ldots, s_T | s_1, g) \propto \exp(-\sum_{t=1}^{T-1} E(s_t, s_{t+1}) - E(s_T, g)) \tag{5}$$

to get a plan. Alternatively, instead of using a fixed goal state g, we can represent the goal state with a Gaussian distribution around it, $P(g)$, and similarly perform inference over

$$p_\theta(s_2, \ldots, s_T | s_1, g) \propto \exp(-\sum_{t=1}^{T-1} E(s_t, s_{t+1}) - (s_T - g)^2) \tag{6}$$

to get a plan. We found that inference over both distributions worked well using MPPI. Throughout our experiment, we follow Equation 6 to specify a Gaussian distribution around all goal states used in our experiments, to accommodate additional constraints more flexibly and refer to the resulting state distribution as $p_\theta(\tau | s_1, G)$. Actions are not inferred in this sampling process, but are fed into a ground truth inverse dynamics model that given a pair of candidate states outputs an action, but show in Section 4.2 that using a learned model also works well.

We can also represent the goal state as the trajectory that has the highest conditional probability of reaching an optimal reward, where the event of reaching an optimal reward is defined as O_t and the probability $P(O_t | s_t)$ is defined as $e^{R(s_t)}$ for a reward function $R(s)$. Inference can then be done on

$$p_\theta(\tau | O_{1:T}) \propto \exp(R(s_1) - \sum_{t=1}^{T-1} (E(s_t, s_{t+1}) - R(s_{t+1})))$$

which Levine [9] interpret as maximum entropy reinforcement learning on the model. However, while the form proposed in [9] considers maximum entropy over actions given a state, we consider maximum entropy of the next state given the current state.

3.3 Online Learning with Energy-Based Models

The previous sections described inference done by EBMs pre-trained with generated data. We now turn to the question of how to generate the training data in an on-going manner to simultaneously learn the EBM as the robot operates in the environment. We discuss online training methods of EBMs – i.e. how to effectively obtain data on state transitions and learn an energy function given a MDP environment represented as a tuple $\langle S, A, T, R \rangle$ and a Goal G.

In this setup, we first generate a T-step trajectory τ_θ from the model $p_\theta(\tau, s_1, G)$ and use an inverse dynamics model to compute the corresponding actions a_t at each time step. We then execute each action a_i in the real environment to generate another T-step trajectory τ_{real}, stopping prematurely if the real observations deviate significantly from model predictions. After that, we train an EBM to increase the energy of each attempted transition in τ_θ (imagined transitions) while decreasing energy of real transition in τ_{real} (real transitions). We note that perfect planning will have no effect on the model training since $\tau_\theta = \tau_{\text{real}}$. For stability, we maintain a replay buffer of past experiences and simulated trajectories. Figure 1 provides an overview of the process.

Intuitively, our training procedure allows our model to learn a good likelihood distribution over states that the model has observed. However, since we terminate model's planning after significant deviation between real observations and the enacted plan, the model is not trained to minimize the probability of transitions among faraway states. These transitions are thus free to vary over time throughout training, which eventually provides incentive for the model to explore the whole state space. In our experiments section, we illustrate this effect and show that EBMs lead to good exploration.

For completeness, we include pseudo-code for online training of EBMs, where $\Omega(\cdot)$ denotes a collation operator that converts a trajectory to pairs of state transitions.

Algorithm 1 Online training of an EBM

Input: goal state G, step size λ, number of steps K, number of plan steps T, inverse dynamics model ID, replay buffers $\mathcal{B}_{pos}$, $\mathcal{B}_{neg}$
$\mathcal{B}_{pos} \leftarrow \varnothing$
$\mathcal{B}_{neg} \leftarrow \varnothing$
for environment timestep i **do**
  ▷ *Initialize trajectory as a smooth trajectory at start state*
  $\tilde{\tau}_i^0 = s_0$
  ▷ *Generate sample from $p_\theta(\tau | s_i, G)$ via MPPI*
  for sample step $k = 1$ to K **do**

$$\tilde{\tau}^k = \sum_i w_i \tilde{\tau}_i^k, \quad \tilde{\tau}_i^k \sim N(\tilde{\tau}_i^{k-1}, \Sigma), \quad w_i = \left(\frac{e^{(\sum_{i=0}^{T-1} E_\theta(s_i^k, s_{i+1}^k)) + (s_T^k - G)^2}}{\sum_j e^{(\sum_{i=0}^{T-1} E_\theta(s_i^j, s_{i+1}^j)) + (s_T^j - G)^2}} \right)$$

  end for
  $a \leftarrow ID(\tilde{\tau}^K)$
  $\tau_i^+ \sim$ simulate environment with actions a
  $\mathbf{x}^+ = \Omega(\tau^+) \cup$ sample$(\mathcal{B}_{pos})$
  $\mathbf{x}^- = \Omega(\tilde{\tau}^k) \cup$ sample$(\mathcal{B}_{neg})$
  ▷ *Optimize objective $\mathcal{L}_2 + \mathcal{L}_{ML}$ wrt. θ:*
  $\Delta\theta \leftarrow \nabla_\theta \frac{1}{N} \sum_i E_\theta(\mathbf{x}_i^+)^2 + E_\theta(\mathbf{x}_i^-)^2 + E_\theta(\mathbf{x}_i^+) - E_\theta(\mathbf{x}_i^-)$
  Update θ based on $\Delta\theta$ using Adam optimizer
  $\mathcal{B}_{pos} \leftarrow \mathcal{B}_{pos} \cup \mathbf{x}^+$
  $\mathcal{B}_{neg} \leftarrow \mathcal{B}_{neg} \cup \mathbf{x}^-$
end for

4 Experiments

We perform empirical studies to answer the following questions: Firstly, can EBMs be applied to model learning in an online setting? Secondly, can EBMs be successfully used for maximum entropy inference and what advantages does that carry? Lastly, what exploration behavior and properties do EBMs exhibit?

4.1 Setup

We perform experiments on four different environments listed below, with corresponding visualizations in Figure 2:

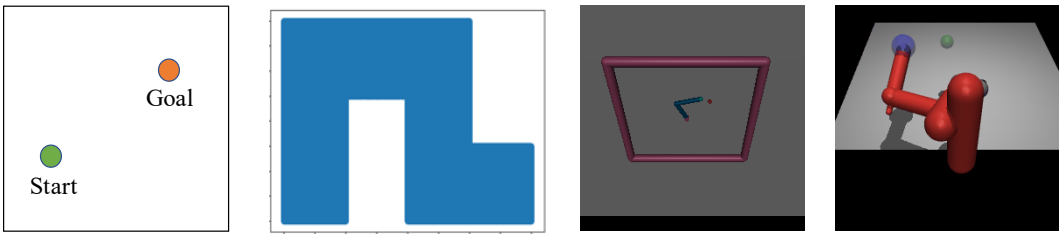

Figure 2: Illustrations of the 4 evaluated environments. Both particle and maze environments have 2 degree of freedom for x, y movement. The Reacher environment has 2 degrees of freedom corresponding to torques to two motors. The Sawyer Arm environment has 7 degrees of freedoms corresponding to torques.

1. **Particle**: An environment in which a particle is spawned at a start position and must navigate to a goal position; each position is represented by an (x, y) tuple. The observation is the current position of particle, and there are two degrees of freedom that correspond to x-displacement and y-displacement. Reward at each timestep corresponds to negative distance from current position to goal position. Agents are able to move in 0.05 uniform ball around their current location, with size of the map being 2 by 2.

2. **Maze**: Same setup as the particle environment, but certain areas contain walls that prevent movement of particle.

3. **Reacher**: The Reacher environment in [32]. The system consists of two degrees of freedom for angles of joints. The observation is the current rotations and angular velocities of joints. Reward at each timestep corresponds to negative distance from current rotations of joints to target rotations of joints.

4. **Sawyer Arm**: A simulation of the Sawyer Arm in Mujoco [33]. The system is second order and contains 7 degrees of freedom. The observation is the position and velocities of each of the joints, as well as the current end-effector finger position. Reward at each timestep is the negative distance between current end-effector finger and target end-effector finger position. Target end-effector position is either fixed or randomized.

For each task, we compare our model's performance with a learned deterministic feedforward network (Action FF) that predicts the next state from the current state/action (with the same architecture as an EBM). We generate plans by sampling over states using MPPI, with score calculated from the L2 distance between final and goal state. On the Sawyer Arm task, we further compare our performance with a model-free baseline PPO [34], using the implementation provided in [35].

We investigate difference in performance of models that have been trained using two different methods, where sources and availability of data are varied. In the case where data is available in advance, models are trained on 100,000 action-state transitions pre-generated from random sampling in each environment. In the case where only online data is available, models are trained by samples generated from interacting with an environment from start state; the training algorithm is outlined by Algorithm 1, with replay buffers used on both models.

4.2 Online Model Learning

Table 4 shows the performance of an EBM compared to action FF on the Particle, Maze and Reacher tasks. First, we compare both methods given a large pre-generated dataset of random interactions; we find that EBM performs slightly better than Action FF. However, when we compare both methods under an online setting, we find that EBM performs **significantly** better than Action FF. For example, an EBM only experiences a drop of 15.24 in score when switched to the online setting, compared to the score drop of 844.56 experienced by an Action FF model on the online setting.

Model	Pretrained (random)	Pretrained (directed)	Pretrained (directed + sequential)	Online (Fixed)	Online (Variable)
EBM	-9569	-4438	-5114	-3782	-3907
Action FF	-10326	-5041	-12838	-9360	-11942

Table 1: Comparison of performance on the Sawyer Arm environment between Action FF and EBM. In the pretraining setting, we compare models trained using random transitions, directed transitions from an EBM, and directed transitions with correlated data. In the online setting, models are trained on 50,000 simulations of the environment. We find that EBMs perform well online.

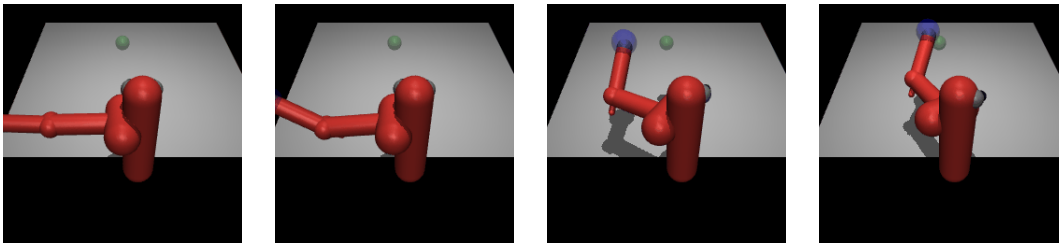

Figure 5: Qualitative image showing EBM successfully navigating finger end effector to goal position.

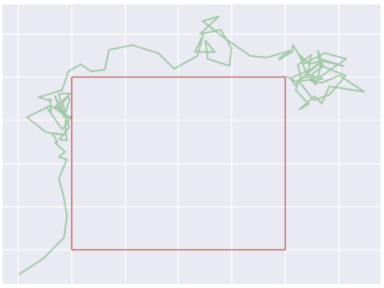

Figure 3: Navigation path with a central obstacle the model was not trained with.

Data	Model	Particle	Maze	Reacher
Pretrained	EBM	**-5.14**	-72.07	**-19.38**
	Action FF	-6.11	**-65.06**	-25.54
Online	EBM	**-20.38**	**-162.97**	**-29.87**
	Action FF	-850.67	-949.99	-42.37

Figure 4: Performance on Particle, Maze and Reacher environments where dynamics models are either pretrained on random transitions or learned via online interaction with the environment. Action FF: Action Feed-Forward Network.

Table 1 shows the performance of EBM compared to action FF on the Sawyer arm scenario. We find that in such a setting, using a large pre-generated dataset of random interactions led to insufficient state coverage. To mitigate this, we construct a directed dataset of 100,000 frames from an EBM trained on Sawyer Arm task. With directed data, we find that a pretrained EBM performs slightly better than Action FF, obtaining scores of -4438 and -5041 respectively. In the online training scenario (with either fixed or varied goals), however, we find that EBM performs **significantly** better (with a score of -3782) than Action FF (with a score of -9360). We show images of execution in Figure 5.

On this task, the model free algorithm PPO obtains performance of -9300 with the same amount of experience, and requires 250,000 to 500,000 frames (5 - 10 times more than used in online training of the models) to achieve comparable performance to online training of an EBM. With 50 - 100 times more experience, PPO is able to obtain better scores of -1000 (note that since PPO does not have acceleration priors we do, it is allowed to reach the goal faster thus producing a higher reward - our method moves slower, but both methods successfully reach the goal). Furthermore, PPO is not able to operate in an online manner and does not exhibit zero-shot generalization results of our model, both of which are important in real-world robot learning regimes.

Table 1 also considers another scenario in which goals are varied across the table. In this setting we find that EBMs still perform better (with a score of -4547 while Action FF obtains a score of (-11942). Furthermore, we can actually apply a model trained on a fixed goal, and generalize to variable goals, and still obtain a score of -4547.

We also ablate dependence on ground truth inverse dynamics. Using recursive least squares [15] to infer inverse dynamics also leads to good performance of -4694. We find that our learned state distributions is not significantly impacted by inverse dynamics inference, as long as action inference does not suffer from mode collapse (which can occur from neural networks based approaches).

To ablate the effect of exploration and the ability of EBMs to learn models online, in Table 1 we train both Action FF and EBM models on the directed dataset, but with batches sequentially sampled from the dataset, with each datapoint repeated 100 times without shuffling to mimic the correlated experiences seen during online training. Under this setting, the performance of EBM drops slightly

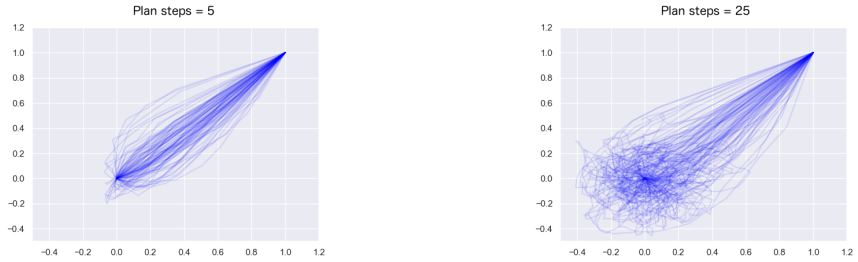

Figure 6: Effects of varying number of planning steps to reach a goal state. As the number of steps of planning increases, there is a larger envelope of explored states.

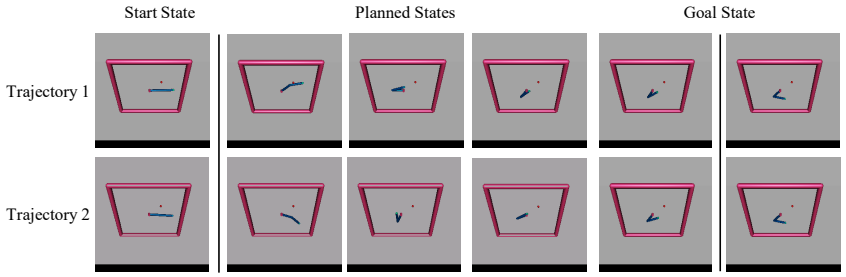

Figure 7: Illustrations of two different planned trajectories from start to goal in the Reacher environment.

by 676, but performance of Action FF drops catastrophically by 7797 and the model fails to train. When training on static data-set, we do not use a replay buffer of past transitions.

Our results indicate that EBMs are able to learn well online, which is an important and necessary characteristic for models to learn in the real world.

4.3 Maximum Entropy Inference

While maximum entropy reinforcement learning has focused on maximizing entropy of actions given a state, sampling from an EBM corresponds to directly maximizing entropy over the next state. In Figure 6, we find EBM sampling is capable of generating diverse plans that go from a given start state to goal state. In Figure 6, we show that given a fixed start state and end goal, increased number of planned steps leads to a larger envelope of possible trajectories. The same diagram also shows that our method is able to sample across a wide range of trajectories that are different from each other. In Figure 7, we show that in the Reacher environment, we are able to make valid plans with both clockwise and counter-clockwise given a start and goal state.

We illustrate the power of diverse plans by comparing the generalization performance between planning conditioned on only state space (EBM planning) and planning conditioned on both state and action space (action-conditional planning). In the particle environment, at test time we add a large obstacle not seen during training as the particle attempts to navigate from start state to goal state, as shown in Figure 3; an EBM is able generalize better obtained a reward of -61.94 while an Action FF obtains a reward of -81.24.

4.4 Exploration

We show that an EBMs model naturally incentive exploration. In Figure 8 we compare the exploration behavior of an EBM without a goal and a random action agent in the Maze environment. In the time it takes a random policy to explore a hallway of a maze, an EBM is able to explore the entirety of the maze. Similarly, we consider 3D occupancy of the finger end-effector in the Sawyer arm; we define 3D occupancy by partitioning space into 3D voxels and measuring the number of voxels that a finger ends up in. We empirically to be found that the maximum system occupancy was 116. We find in Figure 10 that EBMs reaches maximum system occupancy significantly faster compared to random exploration across 4 different seeds. Without a goal, an EBM is able to navigate the arm freely, compared to a random policy that struggles and takes over 100 more times the number of environmental transitions (i.e. over 200,000) to reach maximum occupancy.

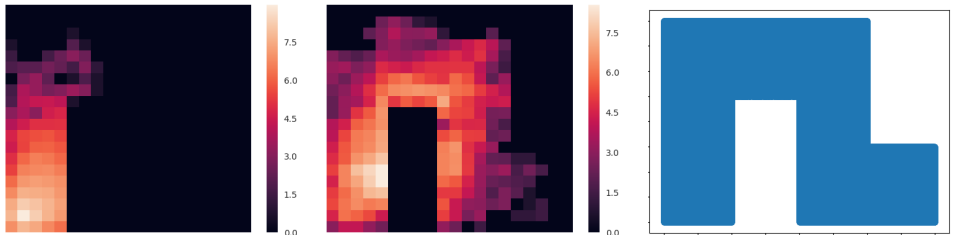

Figure 8: Illustration of exploration in a maze under random actions (left) as opposed to following an EBM (middle). Areas in blue in the maze environment (right) are admissible, while areas in white are not.

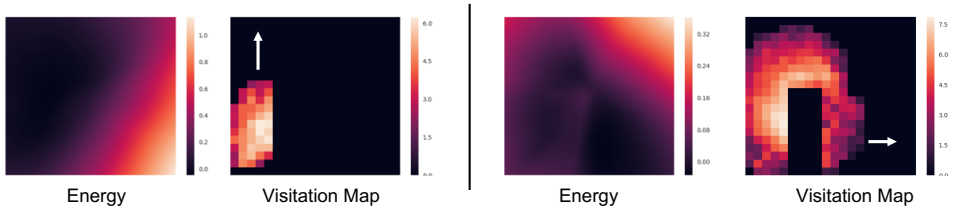

Figure 9: Illustration of energy values of states (computed by taking the energy of a transition centered at the location) and corresponding visitation maps. While an EBM learns a probabilistic model of transitions in areas already explored, energies of unexplored regions fluctuate throughout training, leading to a natural exploration incentive. Early on in training (left), the EBM puts low energy on the upper corner, incentivizing agent exploration towards the top corner. Later on in training (right), an EBM puts low energy on the right lower corner, incentivizing agent exploration towards the bottom corner.

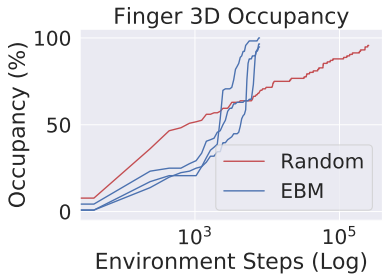

Figure 10: Comparison of 3D spatial occupation of the finger end-effector of the Sawyer Arm, using random exploration versus using an EBM without a goal on a **log** scale across 4 different seeds. An EBM allows more directed exploration and explores more states. For the random policy to reach maximum occupancy, more than 200,000 transitions are required.

We reason that the exploration behavior in EBMs comes from the fact that they learn local dynamics of the world only in the regions that have been explored. This allows the EBMs to assign arbitrary energies to transitions among unexplored states. Values of these energies vary over the course of training, and lead the EBMs to generate plans to reach different unseen states until more of the environment is explored. We illustrate this result in Figure 9, where we show that an EBM puts low energy in a swath of states that are unexplored but reachable in two stages of training, incentivizing exploration of those states while maintaining correct energies for states that have already been explored.

EBMs learn local dynamics models since they are trained on real data transitions and transitions from planning; a plan is followed until it deviates significantly from real transitions. Thus both sets of transitions are consisted of states that are locally close to states an EBM has learned. In contrast, traditional likelihood models for modeling trajectory lower the likelihood of all unseen trajectories, including at unseen states; as a result, planning using such models is unable to explore as adequately.

5 Discussion

We have presented some preliminary results on using EBMs for planning. We show that EBMs are a promising class of models for formulating planning as inference. We further show that EBMs behave well under online model learning, and are able to naturally incentivize exploration. We hope to inspire further investigation towards using EBMs for model-based planning.

References

[1] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the game of go with deep neural networks and tree search. *Nat.*, 529(7587):484–489, 2016.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. In *NIPS Workshop*, 2013.

[3] M. Kempka, M. Wydmuch, G. Runc, J. Toczek, and W. Jaśkowski. Vizdoom: A doom-based ai research platform for visual reinforcement learning. In *Computational Intelligence and Games (CIG), 2016 IEEE Conference on*, pages 1–8. IEEE, 2016.

[4] S. Zhang, M. Petrov, P. Jacob, H. Pondé, B. Chan, F. Wolski, S. Sidor, R. Józefowicz, P. Dębiak, D. Farhi, G. Brockman, J. Raiman, J. Tang, C. Dennison, P. Christiano, S. Hashme, L. Schiavo, I. Sutskever, E. Sigler, J. Schneider, J. Schulman, C. Hesse, J. Clark, Q. Fischer, D. Yoon, C. Berner, S. Gray, A. Radford, and D. Luan. Openai five, 2018.

[5] M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, et al. Learning dexterous in-hand manipulation. *arXiv preprint arXiv:1808.00177*, 2018.

[6] R. S. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *ICML*, 1990.

[7] A. Nichol, V. Pfau, C. Hesse, O. Klimov, and J. Schulman. Gotta learn fast: A new benchmark for generalization in rl. *arXiv preprint arXiv:1804.03720*, 2018.

[8] Y. Du and K. Narasimhan. Task-agnostic dynamics priors for deep reinforcement learning. *arXiv preprint arXiv:1905.04819*, 2019.

[9] S. Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv:1805.00909*, 2018.

[10] Y. Du and I. Mordatch. Implicit generation and generalization in energy-based models. *arXiv preprint arXiv:1903.08689*, 2019.

[11] 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 *ICRA*, 2018.

[12] N. Mishra, P. Abbeel, and I. Mordatch. Prediction and control with temporal segment models. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2459–2468. JMLR. org, 2017.

[13] Y. Gal, R. McAllister, and C. E. Rasmussen. Improving pilco with bayesian neural network dynamics models.

[14] M. C. Yip and D. B. Camarillo. Model-less feedback control of continuum manipulators in constrained environments. *IEEE Transactions on Robotics*, 30(4):880–889, 2014.

[15] I. Mordatch, N. Mishra, C. Eppner, and P. Abbeel. Combining model-based policy search with online model learning for control of physical humanoids. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 242–248. IEEE, 2016.

[16] M. Deisenroth and C. E. Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pages 465–472, 2011.

[17] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine. Reinforcement learning with deep energy-based policies. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1352–1361. JMLR. org, 2017.

[18] B. Dai, Z. Liu, H. Dai, N. He, A. Gretton, L. Song, and D. Schuurmans. Exponential family estimation via adversarial dynamics embedding. *arXiv preprint arXiv:1904.12083*, 2019.

[19] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural Comput.*, 18(7):1527–1554, 2006.

[20] T. Kim and Y. Bengio. Deep directed generative models with energy-based probability estimation. *arXiv preprint arXiv:1606.03439*, 2016.

[21] G. Williams, A. Aldrich, and E. A. Theodorou. Model predictive path integral control: From theory to parallel computation. *Journal of Guidance, Control, and Dynamics*, 40(2):344–357, 2017.

[22] I. Mordatch, E. Todorov, and Z. Popović. Discovery of complex behaviors through contact-invariant optimization. *ACM Transactions on Graphics (TOG)*, 31(4):43, 2012.

[23] T. Erez and E. Todorov. Trajectory optimization for domains with contacts using inverse dynamics. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4914–4919. IEEE, 2012.

[24] P.-Y. Oudeyer and F. Kaplan. What is intrinsic motivation? a typology of computational approaches. *Frontiers in neurorobotics*, 1:6, 2009.

[25] J. Schmidhuber. Formal theory of creativity, fun, and intrinsic motivation (1990–2010). *IEEE Transactions on Autonomous Mental Development*, 2(3):230–247, 2010.

[26] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervised prediction. In *ICML*, 2017.

[27] R. Houthooft, X. Chen, Y. Duan, J. Schulman, F. De Turck, and P. Abbeel. Vime: Variational information maximizing exploration. In *Advances in Neural Information Processing Systems*, pages 1109–1117, 2016.

[28] H. Tang, R. Houthooft, D. Foote, A. Stooke, O. X. Chen, Y. Duan, J. Schulman, F. DeTurck, and P. Abbeel. # exploration: A study of count-based exploration for deep reinforcement learning. In *Advances in neural information processing systems*, pages 2753–2762, 2017.

[29] 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.

[30] G. Williams, N. Wagener, B. Goldfain, P. Drews, J. M. Rehg, B. Boots, and E. A. Theodorou. Information theoretic mpc for model-based reinforcement learning. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1714–1721. IEEE, 2017.

[31] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal. Stomp: Stochastic trajectory optimization for motion planning. In *2011 IEEE international conference on robotics and automation*, pages 4569–4574. IEEE, 2011.

[32] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym. *arXiv:1606.01540*, 2016.

[33] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *IROS*, pages 5026–5033. IEEE, 2012.

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

[35] P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, Y. Wu, and P. Zhokhov. Openai baselines. `https://github.com/openai/baselines`, 2017.