Guiding Search in Continuous State-action Spaces by Learning an Action Sampler from Off-target Search Experience

Beomjoon Kim, Leslie Pack Kaelbling and Tomás Lozano-Pérez
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
{beomjoon,lpk,tlp}@mit.edu

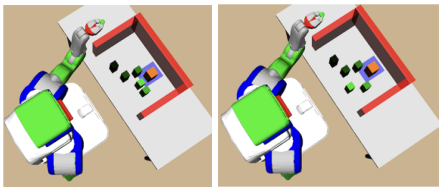
Abstract

In robotics, it is essential to be able to plan efficiently in high-dimensional continuous state-action spaces for long horizons. For such complex planning problems, unguided uniform sampling of actions until a path to a goal is found is hopelessly inefficient, and gradient-based approaches often fall short when the optimization manifold of a given problem is not smooth. In this paper, we present an approach that guides search in continuous spaces for generic planners by learning an action sampler from past search experience. We use a Generative Adversarial Network (GAN) to represent an action sampler, and address an important issue: search experience consists of a relatively large number of actions that are not on a solution path and a relatively small number of actions that actually are on a solution path. We introduce a new technique, based on an importance-ratio estimation method, for using samples from a non-target distribution to make GAN learning more data-efficient. We provide theoretical guarantees and empirical evaluation in three challenging continuous robot planning problems to illustrate the effectiveness of our algorithm.

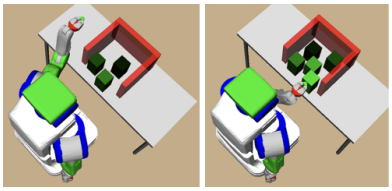
Introduction

Planning efficiently for a long horizon in domains with continuous state and action spaces is a crucial yet challenging problem for a robot. For the classical path-planning problem of getting from an initial state to a goal state, random sampling strategies or gradient-based approaches often work reasonably well (Kuffner and LaValle 2000; Zucker et al. 2013).

In a variety of important planning problems, however, these approaches fall short. Consider the problem in Figure 1a, where the robot has to find a collision-free inverse kinematics solution to reach the orange object by reconfiguring the green objects. An unguided uniform sampling approach performs poorly since the state-space is extremely high-dimensional, consisting of the combined configuration spaces of the robot and many objects. A gradient computation is also difficult, since the robot has to make choices that are both discrete, such as which object to move, and continuous, such as where to place the chosen object, making the problem's optimization manifold non-smooth. This type of hybrid search problem is difficult to solve efficiently.



(a) An example of the reconfiguration task. The movable obstacles are colored green, and the target object is colored with orange.



(b) An example of the bin packing task. The color of objects indicate the order of placement, and the darker the earlier.

Figure 1: Examples of reconfiguration and bin packing. For both tasks, the robot can only grasp objects from the side.

Several task and motion planning (TAMP) algorithms were devised recently to more effectively deal with such complex hybrid search problems (Kaelbling and Lozano-Pérez 2011; Garrett, Lozano-Pérez, and Kaelbling 2014; Toussaint 2015; Vega-Brown and Roy 2016). In sample-based TAMP algorithms, the usual approach is to sample from a continuous action space some predefined number of actions in each state, and then treat the problem as a discrete search problem. This discrete search problem is then solved using a classical graph-search method or a generic discrete PDDL-based planning algorithm. The feasibility of an action, such as picking an object, is verified using a standard motion planner such as rapidly-exploring random tree (RRT) and inverse kinematics.

In such approaches, the on-line computational burden imposed by a large continuous search space heavily depends on the quality of the action sampler: if most of the sampled actions will not lead to a goal, then a planner will spend most of its time sampling actions, expanding nodes, and rejecting them. On the other hand, if the action sampler has positive

probabilities only on actions that will lead to a goal state, then search would be much more efficient.

Based on this observation, this paper presents an approach for learning an action sampler that maps a state and a description of a problem instance to a probability distribution over actions from search experience. Unlike typical learning approaches for planning for continuous action space that estimate a policy whose predicted action gets executed directly, an estimated action sampler is more robust to error since it has a planner to fall back on.

We use a generative adversarial network (GAN), a recent advance in generative model learning, to learn this action sampler (Goodfellow et al. 2014). Unlike other methods, a GAN only requires a forward pass through a neural network to generate a sample, and does not require any prior distributional assumptions.

The main challenge in learning an action sampler from search experience data is that, in a successful episode of search, there is a large number of state and action pairs that are considered but are not known to be on a trajectory to the goal in the episode, which we call *off-target* samples, and only a relatively small number of samples that are known to be on a trajectory to the goal, which we call *on-target* samples. While we could just use the small number of on-target samples, learning would be much more effective if we could find a way to use the abundant off-target samples as well.

In light of this, we propose a principled approach that can learn an action sampler from both on- and off-target samples. To do this, we estimate the importance ratio between the on-target and off-target distributions using both types of samples. We then extend GAN to introduce a new generative model learning algorithm, called generative adversarial network with direct importance estimation (GANDI), that uses the estimated importance-ratio to learn from not only on-target samples, but also from off-target samples. While this algorithm is domain independent, we will demonstrate its effectiveness in learning a target action sampler. We theoretically analyze how the importance-ratio estimation and the difference between target and non-target distributions affect the quality of the resulting approximate distribution.

We evaluate GANDI in three different problems that may occur in warehousing applications. We use GANDI with two sample-based search methods: heuristic-forward search and a breadth-first-search. However, we note that it can be used with any sampled-based planners for hybrid systems. We show our algorithm outperforms a standard uniform sampler and a standard GAN in terms of planning and data efficiency.

Related work

Our work touches upon four different topics: task and motion planning (TAMP), learning to guide planning, importance-ratio estimation, and generative model learning. We provide descriptions of approaches in these areas that are closest to our problem in terms of motivation and technique.

An effective sample-based algorithm for the classical path planning problem in continuous search space is RRT. Although uniform sampling plays a role in this algorithm, it is effective mainly because, in the presence of an appropriate metric, expanding the search node nearest to the randomly sampled point creates a bias, known as the Voronoi bias, that guides the search to unexplored regions of the configuration space. In more complex problems, such as those requiring manipulation of movable obstacles, manually designing appropriate search bias is much more difficult. Recent sample-based TAMP algorithms all employ uniform sampling strategy without any bias, and this causes inefficiencies since a node that is unlikely to get to a goal state gets added and expanded often (Hauser and Ng-Thow-Hing 2011; Kaelbling and Lozano-Pérez 2011; Vega-Brown and Roy 2016; Garrett, Kaelbling, and Lozano-Pérez 2017). One way to view our work is an attempt to resolve this inefficiency by learning an effective action sampling bias.

In learning to guide search, there is a large body of work that attempts to learn a policy or a value function off-line, and then use this information during an on-line search to make planning efficient. These methods are usually applied to discrete-action problems. A recent prominent example is AlphaGo (Silver et al. 2016). In that paper, Silver et. al train a supervised policy off-line, using imitation learning and train a value function using self-play; they then guide Monte Carlo Tree Search (MCTS) in an on-line phase using these functions. For learning to guide search in continuous-space planning problems, Kim et. al (Kim, Kaelbling, and Lozano-Pérez 2017) describe an approach for predicting constraints on the solution space rather than a value function or an action itself. The intuition is that a constraint is much more easily transferable across problem instances than a specific action in complex planning problems. We share this intuition, and we may view the learned action distribution as biasing, if not quite constraining, the search space of a planning problem to promising regions.

Two recent advancements in generative-model learning, GANs (Goodfellow et al. 2014) and Variational Auto Encoders (VAEs) (Kingma and Welling 2014), are appealing choices for learning an action sampler because an inference is simply a feed-forward pass through a network. GANs are especially appealing, because for generic action spaces, we do not have any metric information available. VAEs, on the other hand, require a metric in an action space in order to compute the distance between a decoded sample and a true sample. Both of these methods require training samples to be drawn from a target distribution that one wishes to learn. Unfortunately, in our case we have limited access to samples from the target distribution, and this brings us to the importance-ratio estimation problem.

There is a long history of work that uses importance sampling to approximate desired statistics for a target distribution p using samples from another distribution q, for example, in probabilistic graphical models (Koller and Friedman 2009) and reinforcement learning problems (Precup, Sutton, and Dasgupta 2001; Sutton and Barto 1998). In these cases, we have a surrogate distribution q that is cheaper to sample than the target distribution p. Our work shares the same motivation as these problems, in the sense that in search experience data, samples that are on successful trajectories are expensive to obtain, while other samples are relatively cheaper and more abundant.

Recently, importance-ratio estimation has been studied for the problem of covariate shift adaptation, which closely resembles our setting. Covariate shift refers to the situation where we have samples from a training distribution that is different from the target distribution. Usually, an importance-ratio estimation method (Kanamori, Hido, and Sugiyama 2009; Sugiyama et al. 2008; Huang et al. 2007) is employed to re-weight the samples from the training distribution, so that it matches the target distribution in expectation for supervised learning. We list some prominent approaches here. In kernel-based methods for estimating the importance (Huang et al. 2007), authors try to match the mean of samples of q and p in a feature space, by re-weighting the samples from q. In the direct estimation approach, Sugiyama et al. (Sugiyama et al. 2008) try to minimize the KL divergence between the distributions q and p by re-weighting q. In another approach, Kanamori et al. (Kanamori, Hido, and Sugiyama 2009) directly minimize the least squares objective between the approximate importance-ratios and the target importance-ratios. All these formulations yield convex optimization problems, where the decision variables are parameters of a linear function that computes the importance weight for a given random variable. Our algorithm extends the direct estimation approach using a deep neural network, and then applies it for learning an action sampler using both off-and-on target samples.

Background

Planning in continuous-spaces using sampling

A deterministic continuous state-action space planning problem is a tuple $[\mathcal{S}, \mathcal{A}, s_0, S_g, T]$ where $\mathcal{S}$ and $\mathcal{A}$ are continuous state and action spaces, $T : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ is a transition function that maps a state and an action to a next state, $s_0 \in \mathcal{S}$ is the initial state and $S_g \subset \mathcal{S}$ is a goal set.

A planning problem instance consists of a tuple (s_0, ω, G), where $\omega \in \Omega$ represents parameters of a problem instance. While the state changes according to T when an action is executed, the parameters represent aspects of the problem that do not change within the given instance. For example, a state might represent poses of objects to be manipulated by a robot, while ω might represent their shapes.

Given a continuous state-action space planning problem, a typical approach is to perform a sample-based search, in which the planner repeatedly pops a search node off of a search queue, samples a fixed number of possible actions from an action sampler, possibly conditioned on ω and s, and pushes the successors generated by those actions back onto the search queue. Importantly, the original node must also be pushed back into the queue so that it is available for sampling additional actions in case none of the ones in this batch leads to a goal. This process is repeated until a path to a goal state in G is found.

In order to be applicable to a broad set of problem instances, most of the planning algorithms use a uniform action sampler to sample actions which may be inefficient. We denote this default action sampler as $q_{a|s}$, which defines a distribution over actions given a state[1].

[1]Technically, this should be conditioned on ω as well, but we subsume it as a part of a state

Problem formulation for learning an action sampler

Ideally, our objective would be to learn an action sampler that assigns zero probability to actions that are not on an optimal trajectory to S_g, and non-zero probability to actions on an optimal trajectory. Such a distribution would yield an optimal path to a goal without any exploration.

Unfortunately, in sufficiently difficult problems, optimal planners are not available, therefore it is impossible to determine whether an action was on an optimal path from the search-experience data. Moreover, there may not even be enough information available to learn to discriminate actions on satisficing paths with complete accuracy. Thus, we consider an alternative objective: learn a distribution that assigns low probabilities to actions that do not reduce the distance to a goal state.

Specifically, there are largely two types of such inefficient actions. First, consider an example shown in Figure 1b. Here, the robot is asked to pack five objects into the relatively tight bin. It cannot move an object once it is placed, so if the robot places a first few objects in the front of the bin like in Figure 1b (left), then it will be in a dead-end state, in which it is impossible to reduce the distance to a goal state. A different kind of an inefficient action example is shown in Figure 1a, where the robot has to reconfigure poses of the green objects to find a collision-free path to reach the orange object with its left arm. When the robot moves the light green object as shown in Figure 1a (left) to (right), this action does not constitute a progress towards making room for reaching the target object; albeit it is not a dead-end, this is a wasted effort.

Based on these observations, our objective is to learn a target distribution, which we denote $p_{a|s}$, that assigns low probabilities to such inefficient actions. We denote the m search experience episodes for training data, collected using a domain-independent action sampler $q_{a|s}$, as $\left\{\left[(s_i, a_p^{(i)})_{i=1}^{n_p}, (s_i, a_q^{(i)})_{i=1}^{n_q}, (\omega^{(j)}, s_0^{(j)}, S_g^{(j)})\right]\right\}_{j=1}^{m}$ where $a_p^{(i)}$ is an action on the trajectory from $s_0^{(j)}$ to $s \in S_g^{(j)}$, $a_q^{(i)}$ is an action in the search tree, but not the solution trajectory, and s_i is the state in which $a_p^{(i)}$ or $a_q^{(i)}$ was executed. n_p and n_q denote the number of state and action pairs that are on the trajectory to the goal and the rest, respectively.

Fortunately, the distribution of (s, a) pairs on successful trajectories in experience data have the following properties: they have zero probability assigned to dead-end states and actions, such as Figure 1b (left), since dead-ends cannot occur on a path to the goal. They also have low probability assigned to cycling actions like in Figure 1a, because most actions, though not necessarily all, are making progress. However, we cannot say anything about (s, a) pairs not on a successful trajectory: they may turn out to be a dead-end or a cycle, but they may lead to a goal if searched further enough. Therefore, we will call the $a_p^{(i)}$ values *on-target* samples, and the $a_q^{(i)}$ as *off-target* samples. Our algorithm, GANDI, uses

both of these two sources of data to learn a target distribution.

Generative Adversarial Networks

The objective of a GAN is to obtain a generator G that transforms a random variable z, sampled usually from a Gaussian or a uniform distribution, to a random variable of interest which in our case is an action a. A GAN learns p_G, an approximation of the target distribution, which in our case is $p_{a|s}$. For the purpose of the description of GAN, we will treat the distributions as unconditional; they can be directly extended to condition on s by viewing the transformation as mapping from (s, z) to a.

We denote the on-target samples $\mathbf{A}_p := \{a_p^{(i)}\}_{i=1}^{n_p}$, and the samples generated by G as $\{a_g^{(i)}\}_{i=1}^{n_g}$. To learn the generator G, a GAN alternates between optimizing a function D, which tries to output 1 on on-target samples and output 0 on samples generated by G, and optimizing G, which tries to generate samples that make D to output 1. This leads to the following objectives:

$$\begin{aligned}
&\min_D \mathbb{E}_{a_p \sim p_{a|s}}\big[\log D(a_p)\big] + \mathbb{E}_{a_g \sim p_G}\big[\log(1 - D(a_g))\big] \\
&\approx \min_D \sum_{i=1}^{n_p} \log D(a_p^{(i)}) + \sum_{i=1}^{n_g} \log(1 - D(a_g^{(i)})) \qquad (1) \\
&\max_G \mathbb{E}_{a_g^{(i)} \sim p_G}\big[\log D(a_g^{(i)})\big] \\
&\approx \arg\max_G \sum_{i=1}^{n_g} \log D(a_g^{(i)}) \qquad (2)
\end{aligned}$$

where $n_p = n_g$ to force D to assume the classes are equally likely. After training, given a sample of $z \sim p_z$, the neural network $G(z)$ outputs a with probability $p_G(a)$.

The first term in objective (1) is with respect to the target distribution $p_{a|s}$ which we only have very small number of data points from. We wish to use $q_{a|s}$ instead, since we have much more data from this distribution. This motivates the importance ratio esitmation problem between these two distributions, which we describe next.

Direct importance ratio estimation

We now describe an approach to estimating importance-ratio between the target and uniform distribution, $w(a; s) = p_{a|s}(a|s)/q_{a|s}(a|s)$ using a deep neural network (DNN). If we had these ratios, then we could use $\mathbf{A}_q$ to augment $\mathbf{A}_p$ used for training the generative model p_G to approximate $p_{a|s}$. We will use the least squares approach of Kanamori et al. (Kanamori, Hido, and Sugiyama 2009) because it integrates easily into DNN-based methods. In this approach, we approximate w with $\widehat{w}$ using the objective function

$$J(\widehat{w}) = \int_a (\widehat{w}(a) - w(a))^2 q(a) da\,.$$

In practice, we optimize its sample approximation version, $\hat{J}(\widehat{w})$, which yields

$$\hat{w} = \arg\min_{\widehat{w}} \sum_{i=1}^{n_q} \widehat{w}^2(a_q^{(i)}) - 2\sum_{i=1}^{n_p} \widehat{w}(a_p^{(i)}),\ \text{s.t}\ \widehat{w}(a) \geq 0 \qquad (3)$$

where the constraint is enforced to keep the importance-ratios positive. Intuitively, $\widehat{w}$ attempts to assign high values to $a_p^{(i)}$ and low values to $a_q^{(i)}$, to the degree allowed by its hypothesis class.

The method was originally proposed to be used with a linear architecture, in which $\widehat{w}(a) = \theta^T \phi(a)$; this implies there is a unique global optimum as a function of θ, but requires a hand-designed feature representation $\phi(\cdot)$. For robot planning problems, however, manually designing features is difficult, while special types of DNN architectures, such as convolutional neural networks, may effectively learn a good representation. In light of this, we represent $\widehat{w}$ with a DNN. The downside of this strategy is the loss of convexity with respect to the free parameters θ, but we have found that the flexibility of representation offsets this problem.

Generative Adversarial Network with Direct Importance Estimation

In this section, we introduce our algorithm, GANDI, which can take advantage of off-target samples from $q_{a|s}$ using importance-ratios. We first describe how to formulate the objective for training GANs with importance weights. For the purpose of exposition, we begin by assuming we are given $w(a; s)$, the true importance-ratio between $q_{a|s}$ and $p_{a|s}$, for all a_q distributed according to $q_{a|s}$, and we only have samples from the off-target distribution, $\mathbf{A}_q$, and none from the target distribution. Using importance weights $w(a)$, the objective for the discriminator, equation (1), becomes

$$\begin{aligned}
&\min_D \mathbb{E}_{a_q \sim q_{a|s}}\big[w(a_q)\log D(a_q)\big] \\
&\qquad + \mathbb{E}_{a_g \sim p_G}\big[\log(1 - D(a_g))\big] \\
&\approx \min_D \sum_{i=1}^{n_q} w(a_q^{(i)}) \log D(a_q^{(i)}) + \sum_{i=1}^{n_g} \log(1 - D(a_g^{(i)}))
\end{aligned} \qquad (4)$$

Notice that this objective is now with respect to $q_{a|s}$ instead of $p_{a|s}$. In trying to solve the equation (4), it is critical to have balanced training set sizes n_p and n_g in order to prevent the class imbalance problem for D. In the importance weighted version of the GAN shown in equation 4, however, the sum of the weights $c = \sum_{i=1}^{n_q} w(a_q^{(i)})$, serves as an *effective sample size* for the data $\mathbf{A}_q$. To achieve a balanced objective, we might then select n_g to be equal to c.

Taking this approach would require adjusting the GAN objective function and modifying mini-batch gradient descent algorithm to use uneven mini-batch sizes in every batch in training, which may lead to unstable gradient estimation.

Instead, we develop a method for bootstrapping $\mathbf{A}_q$ that allows us to use existing mini-batch gradient descent without

modification. Specifically, instead of multiplying each off-target sample by its importance weight, we bootstrap (i.e. resample the data from $\mathbf{A}_q$ with replacement), with a probability $p_w(a)$, where $p_w(a) = \frac{w(a)}{\sum_{i=1}^{n_q} w(a_q^{(i)})}$. This method allows us to generate a dataset $\widehat{\mathbf{A}}$ in which the expected number of instances of each element of $\mathbf{A}$ is proportional to its importance weight. Moreover, since we bootstrap, the amount of training data remains the same, and D now sees a balanced number of samples effectively drawn from $p(a) = w(a)q(a)$ and p_G.

One can show that p_w is actually proportional to p.

Proposition 1 *For $a \in \mathbf{A}_q$,*

$$p_w(a) = k \cdot p(a) \text{ where } k = \frac{1}{\sum_{i=1}^{n_q} w(a_q^{(i)})} .$$

This means that samples drawn from the importance-reweighted data $\mathbf{A}$ can play the role of $\mathbf{A}_p$ in the GAN objective.

We now describe some practical details for approximating $w(a)$ with $\widehat{w}(a)$, whose architecture is a DNN. Equation 3 can be solved by a mini-batch gradient-descent algorithm implemented using any readily available NN software package. The non-negativity constraint can also be straightforwardly incorporated by simply using the rectified linear activation function at the output layer[2].

Now, with estimated importance weights and bootstrapped samples, the objective for D shown in equation 4 becomes

$$\hat{D} = \arg\min_D \sum_{i=1}^{n_q} D(a_w^{(i)}) + \sum_{i=1}^{n_g} \log(1 - D(a_g^{(i)})) \quad (5)$$

where $a_w^{(i)}$ denotes a bootstrapped sample from $\mathbf{A}_q$, and $n_q = n_g$. This involves just using $\mathbf{A}_q$, but in practice, we also use $\mathbf{A}_p$, by simply augmenting the dataset $\mathbf{A}_q$ to construct $\mathbf{A} := \mathbf{A}_q \cup \mathbf{A}_p$, and then applying $\widehat{w}(\cdot)$ to $\mathbf{A}$ for bootstrapping, yielding final dataset $\widehat{\mathbf{A}}$. Algorithm 1 contains the code for GANDI.

We illustrate the result of the bootstrapping with a simple example, shown in Figure 2, where we have a Gaussian mixture model for both on-and off-target distributions p and q, where p is a mixture of two Gaussians centered at $(1, 1)$ and $(3, 1)$, and q is a mixture of three Gaussians at $(1, 1), (3, 1)$, and $(2, 2)$ with larger variances than those of p.

In Figure 2, the left-most shows the true distributions of on- and off-target distributions, p and q denoted with blue and red, respectively. The second plot shows our data points from each of these distributions, $\mathbf{A}_p$ and $\mathbf{A}_q$. The next plot shows the estimated importance weights, $\widehat{w}$, using objective (3), where darker color indicates higher $\widehat{w}(a)$ value. We can see that $\widehat{w}$ is almost zero in regions where $\mathbf{A}_p$ and $\mathbf{A}_q$ do not overlap, especially around (2,2). The next plot shows our bootstrapped samples, $\widehat{\mathbf{A}}$, sampled from our bootstrap probability distribution p_w in green, and $\mathbf{A}_p$ again in red. We can see that it reflects the values of $\widehat{w}$. Lastly, the right-most plot shows the result of training GANDI using $\widehat{\mathbf{A}}$, from which we can see it is quite similar to p.

[2]In practice, this often lead gradients to shrink to 0 due to saturation. Although this can be avoided with a careful initialization method, we found that it is effective to just use linear activation functions, and then just set $w(a) = 0$ if $w(a) < 0$.

Algorithm 1 GANDI($\mathbf{A}_p, \mathbf{A}_q$)

$\widehat{w} \leftarrow \text{EstimateImportanceWeights}(\mathbf{A}_p, \mathbf{A}_q)$ // obj. (3)
$p_w(a) := \frac{\widehat{w}(a)}{\sum_{i=1}^{n_q} \widehat{w}(a_q^{(i)}) + \sum_{i=1}^{n_p} \widehat{w}(a_p^{(i)})}$ // bootstrap prob dist
$\mathbf{A} \leftarrow \mathbf{A}_p \cup \mathbf{A}_q$
$\widehat{\mathbf{A}} \leftarrow \text{Bootstrap}(\mathbf{A}, p_w)$ // sample $\mathbf{A} \sim p_w$ with replacement
$G \leftarrow \text{TrainGAN}(\widehat{\mathbf{A}})$ // objs (1) and (2) with $\widehat{\mathbf{A}}$ as data
return G

Theoretical analysis

In this section, we analyze how error in importance estimation affects the performance of p_G in approximating p. The basic result on GANs, shown in the limit of infinite data, representational and computational capacity, is that p_G converges to p (Goodfellow et al. 2014), although subsequent papers have presented more subtle form of analysis (Arjovsky and Bottou 2017).

Now, under the same assumptions, we consider the effect of using importance weighted off-target data. If w is exact, then $p(a) = w(a)q(a)$ and the GAN objective is unchanged. If, however, we use an estimation of importance weighting function $\widehat{w}$, then the objective of $\hat{D}$, the importance-weight corrected discriminator, differs from D and they achieve different solutions.

We wish to analyze the effect of importance estimation error on KL and reverse-KL divergence between p and p_G. First, define $\rho = \sup_{a \in \mathcal{A}_p} q(a)/p(a)$, where $\mathcal{A}_p$ is the support of p. We can see that $\rho >= 1$, with equality occurring when $p(a) = q(a)$ for all a.

For the KL divergence, we have the following theorem.

Theorem 1 *If $w(a) \geq \epsilon \; \forall a \in \mathcal{A}_q$, $\epsilon \geq 0$, and $J(\widehat{w}) \leq \epsilon^2$, then*

$$\text{KL}(p||p_G) \leq \log\left(\frac{1}{1 - \epsilon\rho}\right) .$$

Note that $0 \leq \epsilon\rho \leq 1$ due to the condition $w(a) \geq \epsilon$. For reverse KL we have:

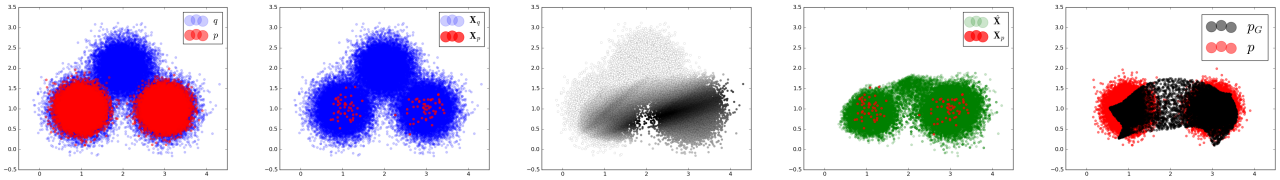
Theorem 2 *If $J(\widehat{w}) \leq \epsilon^2$, $KL(p_G||p) \leq (1 + \epsilon)\log(1 + \epsilon\rho)$.*

The proofs are included in the supplementary material.

These theorems imply three things: (1) If $w = \widehat{w}$, then $\epsilon = 0$, and both divergences go to 0, regardless of ρ; (2) If $p = q$, then the error in importance weight estimation is the only source of error in modeling p with p_G. This error can be arbitrarily large, as ϵ becomes large; and (3) If $p \neq q$ then $\rho > 1$, and it contributes to the error in modeling p with p_G.

Experiments

We validate GANDI on three different robot planning tasks that involve continuous state and action spaces and finite



(a) (b) (c) (d) (e)

Figure 2: (a) target and off target distributions, (b) training data points $a_p^{(i)}$ and $a_q^{(i)}$, (c) importance weight estimation, (d) bootstrapping result and (e) learned distribution p_G and target distribution p.

depths. These experiments have two purposes: first, to verify the hypothesis that learning an action sampler improves planning efficiency relative to a standard uniform sampler and second, a GANDI is more data efficient than a standard GAN that is only trained with on-target samples.

We have three tasks that might occur in a warehouse. The first is a bin-packing task, where a robot has to plan to pack different numbers of objects of different sizes into a relatively tight bin. The second task is planning to stow eight objects into crowded bins, where there already are obstacles. The final task is a reconfiguration task, where a robot needs to reconfigure five randomly placed moveable objects in order to make room to reach a target object.

These are highly intricate problems that require long-term geometric reasoning about object placements and robot configurations; for each object placement, we require that there is a collision-free path and an inverse kinematics (IK) solution to place the object. For the placement of an object, we verify an existence of the IK solution by solving for IK solutions for a set of predefined grasps, and then check for the existence of a collision-free path from the initial state to the IK solution using a linear path[3].

In the first two tasks, the robot is not allowed to move the objects once they are placed, which leads to a large volume of dead-end states that cause wasted computational effort for a planner with a uniform action sampler. In the third task, we have no dead-end states, but a planner could potentially waste computational effort in trying no-progress actions. For all tasks, the robot is only allowed to grasp objects from the side; this is to simulate a common scenario in a warehouse environment, with objects in a place covered on top, such as a shelf. For the first two experiments, we use heuristic search with a heuristic that estimates the cost-to-go to be the number of objects remaining to be placed, since we cannot move objects once they are placed. For the last experiment we use breadth-first-search with no heuristic. For all cases, the number of action samples per node, k, is 4.

In each task, we compare three different action sampler in terms of success rate within a given time limit: one that uniformly samples an action from the action space, a standard GAN trained only with on-target samples, and GANDI, which is trained with both on and off-target samples. We use the same architecture for both the standard GAN and GANDI, and perform 100 repetitions to obtain the 95% confidence intervals for all the plots. The architecture description for the DNNs is in the appendix.

A crucial drawback of generative adversarial networks is that they lack an evaluation metric; thus it is difficult to know when to stop training. We deal with this by testing weights from all epochs on 10 trials, and then picking the weights with the best performance, with which we performed 100 additional repetitions to compute the success rates.

[3]A more sophisticated motion planners, such as RRT, can be used instead.

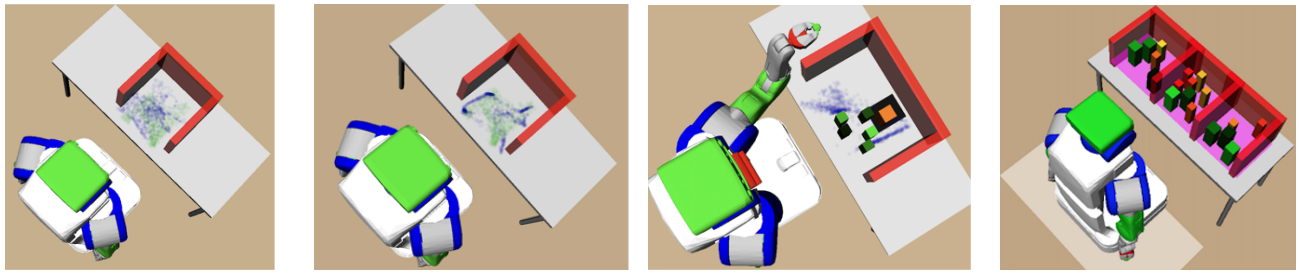
Bin packing problem

In this task, a robot has to move 5, 6, 7 or 8 objects into a region of size 0.3m by 1m. The number of objects is chosen uniform randomly. The size of each object is uniformly random between 0.05m to 0.11m, depending on how many objects the robot has to pack. A problem instance is defined by the number of objects and the size of each object, $\omega = [n_{obj}, O_{size}]$. A state is defined by the object placements. For a given problem instance, all objects have the same size. An example of a solved problem instance with $n_{obj} = 5$ and $O_{size} = 0.11$m is given in Figure 1b (right).

The action space consists of the two dimensional (x,y) locations of objects inside the bin, and a uniform action sampler uniformly samples these values from this region. The robot base is fixed. The planning depth varies from 5 to 8, depending on how many objects need to be placed. This means that plans consist of 10 to 16 decision variables.

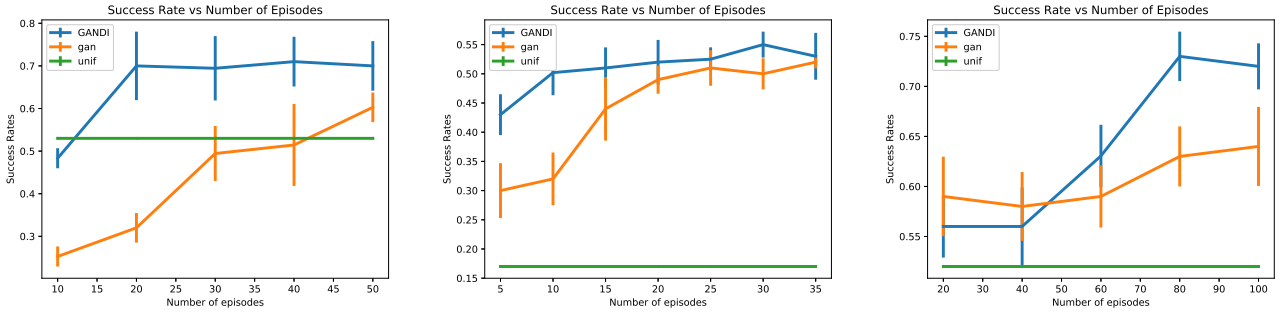
Figure 4a plots, for each method, the success rate when given 5.0 seconds to solve a problem instance. We can see the data efficiency of GANDI: with 20 training episodes, it outperforms the uniform sampler, while a standard GAN requires 50 training episodes to do so. The uniform sampler can only solve about 50% of the problem instances within this time limit, while GANDI can solve more than 70%.

We also compare the action samplers trained using GAN and GANDI when the same number of training data are given. Figures 3a and 3b show 1000 samples from GAN and GANDI for packing 5 objects. While GANDI learns to avoid the front-middle locations, GAN is still close to a uniform action sampler, and has a lot of samples in this region which lead to dead-end states. GANDI focuses its samples on the corners at the back or the front so that it has spaces for all 5 objects.



(a) (b) (c) (d)

Figure 3: Figures (a) and (b) respectively show actions sampled from action samplers trained with GAN and GANDI from the bin packing domain when 20 episodes of training data are used. Green indicates the on-target samples, and blue indicates the learned samplers. Figure (c) shows an action distribution for the reconfiguration domain when given 35 training episodes. Figure (d) is a stow domain problem instance.



(a) (b) (c)

Figure 4: Plots of success rate vs. number of training episodes for the (a) bin packing, (b) stow, and (c) reconfiguration domains

Stowing objects into crowded bins

In this task, a robot has to stow 8 objects into three different bins, where there already are 10 obstacles. A bin is of size 0.4m by 0.3m, an obstacle is of size 0.05m by 0.05m, and the objects to be placed down are all of size 0.07m by 0.07m. A problem instance is defined by the (x,y) locations of 10 obstacles, each of which is randomly distributed in one of the bins. Figure 3d shows an instance of a solved stow problem.

The action space for this problem consists of (x,y) locations of an object to be placed, and the robot's (x,y) base pose. This makes a 4 dimensional continuous action-space. The planning depth is always 8, for placing 8 objects. Thus plans consist of of 36 continuous decision variables. Again, there is a large volume of dead-end actions, similarly to the previous problem: putting objects down without consideration of poses of later objects can potentially block collision-free paths for placing them.

Figure 4b compares the success rates of the algorithms with a 30-seconds time limit for planning. For the uniform sampler, we sample first an object placement pose, and then sample a base pose that can reach the object at its new location without colliding with other objects. Unlike the previous task, learning-based approaches significantly outperform the uniform sampling approach for this task. This is because there is a small volume of action space that will lead to a goal, and a large volume of search space. Again, we can observe the data efficiency of GANDI compared to GAN. When the number of training data points is small, it outperforms it.

Reconfiguration planning in a tight space

In this task, a robot has to reconfigure movable obstacles out of the way in order to find a collision-free IK solution for its left-arm to reach the target object. There are five movable obstacles in this problem, each with size 0.05m by 0.05m, and the target object of size 0.07m by 0.07m, and the reconfiguration must happen within a bin, which is of size 0.7m by 0.4m. A problem instance is defined by (x,y) locations of the movable obstacles and the target object. The movable obstacles are randomly distributed within the bin; the target object location is distributed along the back of the bin. Figure 3c shows an example of a rearrangement problem instance at its initial state, with the black region indicating the distribution of target object locations.

An action specification consists of one discrete value and two continuous values: what object to move and the (x,y) placement pose of the object being moved. There is no fixed depth. For both the uniform random sampler and the learned sampler, we uniformly at random choose an object to move. The robot base is fixed, and the robot is not allowed to use its right arm.

Figure 4c compares the success rates of the algorithms with a 10-seconds time limit for planning. In this problem, the learning-based approaches outperform the uniform sampler even with a small number of training data points. The relationship between GANDI and GAN is similar to the previous experiment, except that GANDI and GAN are within the each other's confidence interval when a small number of training points are used. Eventually, GANDI comes to clearly outperform GAN.

We would like to know if GANDI's distribution indeed assigns low probabilities to no-progress actions. In Figure 3c, we show GANDI's distribution of object placements after training on 35 episodes. The left top corner of the bin is empty because there are no collision-free IK solutions for that region[4]. As the figure shows, there are no placement samples in front of the target object, but only on the sides that would contribute to clearing space for the robot's left arm to reach the target object.

Conclusion

We presented GANDI, a generative-model learning algorithm that uses both on-target and cheaper off-target samples for data efficiency using importance-ratio estimation. We provided guarantees on how the error in importance-ratio estimation affects the performance of the learned model. In three different robot task and motion planning problems that require a long horizon geometric reasoning over continuous state and action spaces, we showed that GANDI can effectively accelerate the search of sample-based planners with a relatively small amount of search experience.

Acknowledgement

We thank Caelan R. Garrett for helpful comments. We gratefully acknowledge support from NSF grants 1420316, 1523767 and 1723381, from AFOSR FA9550-17-1-0165, from ONR grant N00014-14-1-0486, and from ARO grant W911NF1410433. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of our sponsors.

References

Arjovsky, M., and Bottou, L. 2017. Towards principled methods for training generative adversarial networks. In *International Conference on Learning Representations*.

Garrett, C. R.; Kaelbling, L. P.; and Lozano-Pérez, T. 2017. Sample-based methods for factored task and motion planning. In *Robotics: Science and Systems*.

Garrett, C. R.; Lozano-Pérez, T.; and Kaelbling, L. P. 2014. FFRob: An efficient heuristic for task and motion planning. In *International Workshop on the Algorithmic Foundations of Robotics*.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems*.

[4]The robot's left arm will collide with the bin

Hauser, K., and Ng-Thow-Hing, V. 2011. Randomized multi-modal motion planning for a humanoid robot manipulation task. *International Journal on Robotics Research*.

Huang, J.; Smola, A.; Gretton, A.; Borgwardt, K.; and Schölkopf, B. 2007. Correcting sample selection bias by unlabeled data. In *Advances in Neural Information Processing Systems*.

Kaelbling, L. P., and Lozano-Pérez, T. 2011. Hierarchical task and motion planning in the now. In *IEEE Conference on Robotics and Automation*.

Kanamori, T.; Hido, S.; and Sugiyama, M. 2009. A least-squares approach to direct importance estimation. *Journal of Machine Learning Research* 10.

Kim, B.; Kaelbling, L. P.; and Lozano-Pérez, T. 2017. Learning to guide task and motion planning using score-space representation. In *International Conference on Robotics and Automation*.

Kingma, D. P., and Welling, M. 2014. Auto-encoding variational bayes. In *International Conference on Learning Representations*.

Koller, D., and Friedman, N. 2009. *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press.

Kuffner, J., and LaValle, S. 2000. RRT-connect: An efficient approach to single-query path planning. In *International Conference on Robotics and Automation*.

Precup, D.; Sutton, R.; and Dasgupta, S. 2001. Off-policy temporal-difference learning with function approximation. In *International Conference on Machine Learning*.

Silver, D.; Huang, A.; Maddison, C.; Guez, A.; Sifre, L.; van den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; Dieleman, S.; Grewe, D.; Nham, J.; Kalchbrenner, N.; Sutskever, I.; Lillicrap, T.; Leach, M.; Kavukcuoglu, K.; Graepel, T.; and Hassabis, D. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature*.

Sugiyama, M.; Nakajima, S.; H. Kashima, P. B.; and Kawananbe, M. 2008. Direct importance estimation with model selection and its application to covariate shift adaptation. In *Advances in Neural Information Processing Systems*.

Sutton, R., and Barto, A. G. 1998. *Reinforcement Learning: An Introduction*. The MIT Press.

Toussaint, M. 2015. Logic-geometric programming: An optimization-based approach to combined task and motion planning. In *International Joint Conference on Artificial Intelligence*.

Vega-Brown, W., and Roy, N. 2016. Asymptotically optimal planning under piecewise-analytic constraints. In *Workshop on the Algorithmic Foundations of Robotics*.

Zucker, M.; Ratliff, N.; Dragan, A.; Pivtoraiko, M.; Klingensmith, M.; Dellin, C.; Bagnell, J. A.; and Srinivasa, S. 2013. CHOMP: Covariant hamiltonian optimization for motion planning. *International Journal on Robotics Research*.