Learning Sampling Distributions for Robot Motion Planning

Adrian Brandemuehl
ETH Zurich
adrianbr@ethz.ch

Jose L. Vazquez
ETH Zurich
vjose@ethz.ch

Abstract: Today, sampling based motion planners (SMBP's) suffer from the use of uninformative sampling distributions that rely on heuristics designed by roboticists to be effective. Now, with recent work in deep generative models, complex and high-dimensional sampling distributions can be learned in an unsupervised fashion from data generated by the target distribution. We demonstrate the marriage of the two concepts: using a generative model as a sampling distribution for an SBMP. We implement an open source pipeline for training and inference based on standard open source motion planning tools that is capable of learning from previous plans to improve the results on new planning problems. Furthermore we extended the approach to incorporated learning distributions from unstructured conditional data.

Keywords: Robots, Sampling Based Planning, Unsupervised Learning

1 Introduction

Sampling based motion planning has been a successful algorithmic technique in solving motion planning problems in high dimensional spaces. However, SBMP uses uninformative or hand tuned priors that yield sub-optimal behavior in the exploration process of incremental SBMP. Several techniques have been used to overcome this limitation, mainly falling under the umbrella of informative sampling and learned sampling. The former tackles the trajectory refinement process, meaning that once a solution to the SMBP problem is found, informative sampling takes care of exploiting the space structure to find lower cost solutions. Learned sampling tackles the problem of finding a solution in fewer sampling steps and with lower cost but it lacks generalization due to being trained with hand crafted, artificial representations of the workspace. Ichter et al. [1] proposes a general way to learn sampling distributions with a variant of a Variational Autoencoder (VAE) that conditions on the information of the planning problem, however this method suffers from the aforementioned problem, e.g. hand crafted obstacle representations.

Statement of contributions: In this project we extend the work done in Ichter et al. [1] by learning a planning distribution from unstructured data: distance fields. Additionally we provide an open source implementation of the proposed neural network with a C++ deployment pipeline that allows direct use of the learned sampler using OMPL [2].

2 Related Work

Learning sampling distributions can be formulated in different ways. Qureshi and Yip [3] uses a neural network to sequentially sample the next states for a SMBP. Unfortunately, this yields high inference times which limits real-time usage. Zhang et al. [4] proposes a reinforcement learning based rejection sampling approach that learns to reject unimportant samples generated by a uniform distribution, thus, more efficient exploration of the state space.

This project is mainly based on the work by Ichter et al. [1], which proposes a Conditional Variational Autoendoder (CVAE) to directly map the problem information to a distribution to sample states from. Several extensions to [1] have been proposed, for example Kumar et al. [5] proposed an algorithm to efficiently identify narrow passages from the workspace for training a CVAE. All of these methods used a structured representation of the workspace to condition the CVAE, which renders the end sampler unusable in environments where obstacles can not be simply represented by boxes in space. Therefore, we propose to use a distance field as a representation to learn to avoid obstacles.

3 Learning Sampling Distributions

In machine learning, generative models are a class of algorithms that are used to learn distributions of high dimensional data. In general they can be classified into 4 main types: Autoregressive Models, Generative Adverserial Networks (GAN's) [6], Variational Autoencoders (VAE's) [7] and Normalizing Flows [8]. Each class of model aims to learn a distribution that captures the important statistical properties of the training data. Using these learned distributions, new meaningful data samples can be generated from the trained networks. Variational Autoencoders are able to learn a map from a lower dimensional latent distribution to an explicit representation of the data distribution, which makes them suitable to generate samples for high dimensional spaces.

VAE's aim to model the training data by maximizing the Evidence Lower Bound (ELBO) loss. In essence, the VAE is an Autoencoder with a regularization term that aims to force the latent variable to take the distribution of a lower dimensional isotropic Gaussian. This regularization enables decoding Gaussian latent variable samples into samples from the distribution of the training data. Conditional Variational Autoencoders (CVAE's) [9] are an extension of the VAE that learns the distribution $p(x|z, C)$, i.e. the distribution of samples conditioned on a separate variable C. Generating samples from the network then requires C, a condition variable, and z, a latent space sample.

The work of Ichter et al. [1] uses a vanilla CVAE to generate samples, however reproducing the work exactly does not yield good results in the variance of samples drawn. With enough tuning, the results of [1] might be reproducible, but we chose to modify the network and loss to improve the variance of the generated samples. Specifically, we modify the output of the CVAE network to produce two results: the mean, μ_x, and standard deviation, σ_x, of a Gaussian distribution.

The output distribution of a normal CVAE is assumed to have a fixed, diagonal covariance matrix, i.e. $p(x|C, z) = \mathcal{N}(\mu(z), I)$. We instead leave the variance terms to be learned by the decoder as σ_x. Therefore the CVAE reconstruction loss is modified to be the negative log likelihood of a normal distibution with a learned variance. i.e. $p(x|z, C) = \mathcal{N}(\mu(z), \sigma(z))$. A diagram of the CVAE architecture can be seen in 1. As is typical in VAE models a multiplicative weight, β, is introduced to the KL divergence term. β controls the tradeoff between the KL loss and the reconstruction loss. The final loss can be seen in Equation 1.

$$\mathcal{L}_{\text{recon}} = E_q[-\log p(x|z, C)] - \beta D_{KL}(q(z|x, C)||p(z|C)) \tag{1}$$

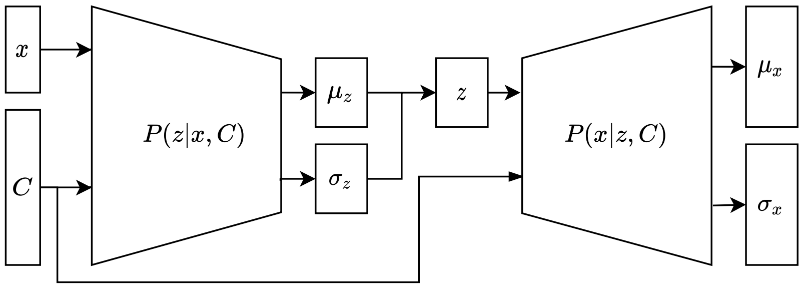


Figure 1: CVAE architecture with mean and variance of x on the decoder.

3.1 Unstructured Environments with Distance Fields

So far, obstacles have been represented in an exact structure. Given a fixed number of obstacles, the exact parameters of the obstacles are used as a condition for training and inference. This is inflexible as it requires the number of obstacles to be set a priori and does not easily allow for variable obstacle shapes. In order to overcome this limitation of [1] we propose to use distance fields, a common robotics map representation, as a condition variable. Distance fields are a dense representation of the environment, where each grid cell contains the distance between it and the nearest occupied cell. Distance fields can represent any number of obstacles in a space with a tunable grid resolution. Unfortunately, distance fields can be quite large depending on their resolution and the size of the environment they represent. Given our problem setup, our condition variable is already significantly larger in dimension than the samples to reconstruct. Using a distance field increases that difference by an order of magnitude, making the problem much harder for a the network to learn. To mitigate this problem, we use a convolutional encoder to reduce the dimension of the distance field to an informative enough minimal representation. We found that using the weights from a convolutional auto-encoder that was trained for map reconstruction was not ideal for use as a condition in our CVAE which leads us to believe that a latent space trained for reconstruction is not informative enough for learning planning related tasks. Jointly training the distance field encoder and CVAE yielded the best results for our problem and does not come with significant training time or convergence overhead.

3.2 Implementation details

The final neural network architecture consisted of a convolutional encoder that generates a conditional vector. Each convolutional block consisted of increasingly deeper channel dimensions and downsampling using a stride of 2, additionally it has a ReLU activation function. After the convolutional blocks, the output features get flattened and fed into a fully connected layer that outputs a conditional variable. The encoder of the CVAE uses only fully connected layers and its input is the concatenation of the encoded map, the start/goal states and the sample from the training data. The decoder gets the concatenation of a sample from the latent distribution, the embedding from the map encoder and the start/goal states. In Fig. 2, we provide a detailed description of the learned sampler.

Training data consisted of pairs of condition data and samples drawn from the optimal path found by an optimal SMBP (RRT* [10]). For generating the training data, we used OMPL to generate planning problems with fixed sized square obstacles. Obstacles were placed on a grid, and start and goal states were sampled from free space within the grid. Gradient descent by grad student was employed to tune the hyper-parameters. The main hyper-parameters to tune are the batch size and β term of the KL loss. The decoder was deployed in OMPL and can generate samples for any SBMP in the library. GPU acceleration enables rapid inference for generating large numbers of samples.

4 Experimental Results

We implemented two experimental scenarios to both train and test our models on. Both represent the environment as a set of square obstacles uniformly sampled from a grid. The first example is a 2 dimensional geometric motion planning problem. The second setup is a non-holonomic kinematic car model. The models seemed to begin stable training of the KL loss early on training but the overall loss takes a lot of time to reach a plateau, as shown in Fig. 3.

We evaluate the usefulness of our learned sampler using two metrics: planning success rate and ratio between cost of the solution found for a uniform sampler versus the learned sampler. We evaluated this on two types of robots: a 2 state geometric point robot and a 3 state kinematic robot.

It is immediately obvious in Figure 4 and 5 that the learned sampler had little to no effect on the result of planning. This is a somewhat intuitive result because the 2 dimensional geometric problem and even the 3 state non-holonomic problem can easily be solved by a SBMP using a uniform distribution. Difficult problems such as multi jointed arms are where the results should shine.

These results are similar to those of [1] and allow some confidence that the distance field obstacle representation can be used successfully. We can visually examine the results of the sampler by sampling many points in latent space and passing them through the encoder. Shown in Figure 6, blue points represent the mean of the decoded distribution and red points are sampled from the full generated distribution.

One problem that we've observed and is discussed in [11] is that even though the training data cannot come close to obstacles, the sampling distributions have a high probability of sampling points that are very close to obstacles. This is a problem because most collision checkers include an increased radius around each state to maintain a safety margin.

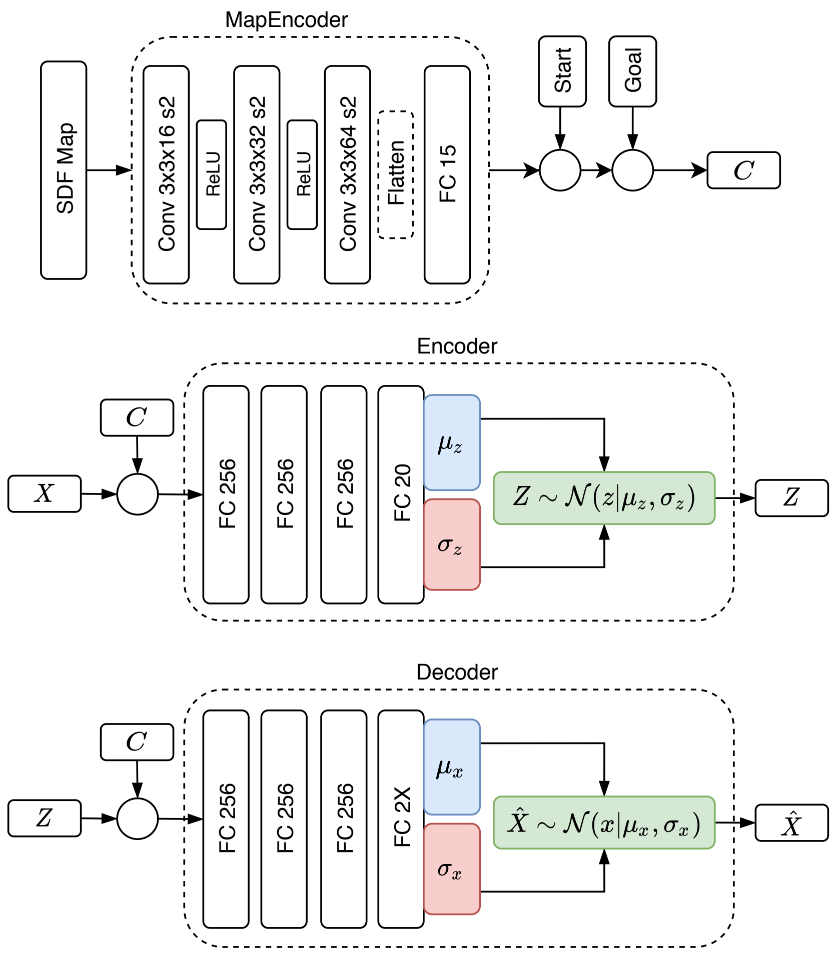


Figure 2: Complete architecture of the learned sampler with a distance field encoder

4.1 Latent Space Interpolation

We found that in nominal cases the latent space holds useful structure that maintains spacing in the sample space. To see this, we linearly interpolate points in latent space between the encoded versions of the problem start and goal states. The interpolation has some failure cases such as Figure 7b. When the problem is ambiguous, i.e. there are multiple equal cost solutions, the network is not able to represent the multimodal nature of the desired sampling distribution and latent space interpolation fails.

5 Conclusion

We developed a pipeline for learning generative models for state sampling for use in SBMP's. Though the results did not improve on the baseline, we demonstrated that an unstructured representation of the obstacles in the workspace could be successfully used to learn to avoid obstacles in sampling. Based on the results of other similar works, we are confident that the method presented will scale up to larger, more complex problems with high dimensions, self collisions and full dynamics. Furthermore, learning these sampling distributions from unstructured data allows this implementation to scale to real robotics environments where the main source of workspace knowledge is signed distance fields.

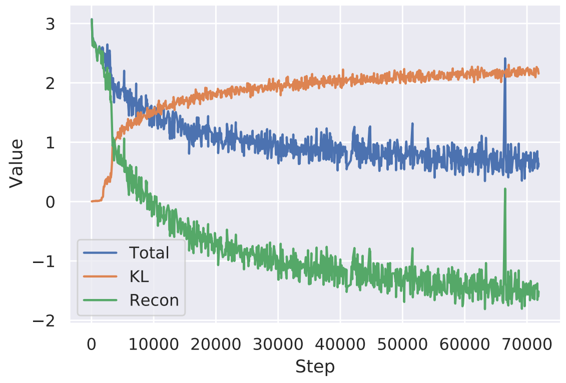
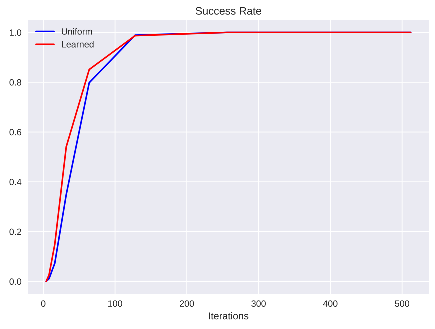
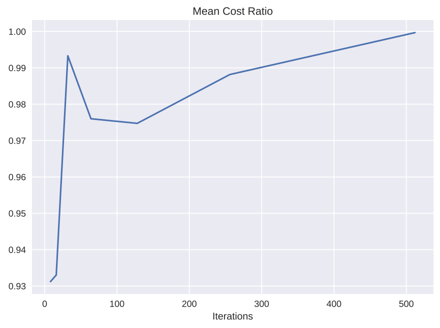


Figure 3: Reconstruction vs KL tradeoff in the geometric 2 state problem.



(a) Success Rate

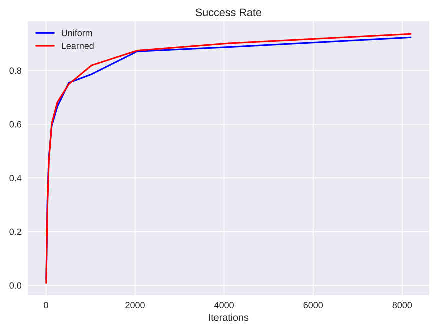


(b) Cost Ratio

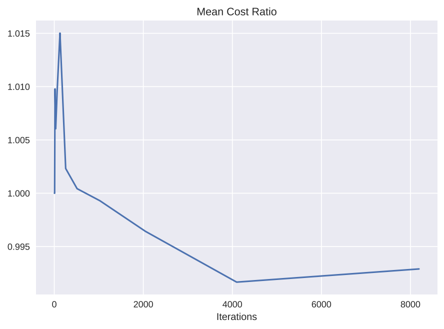
Figure 4: Geometric Planning Result

5.1 Future Work

Many directions can be taken to improve the methods presented. First, the SBMP's that are used have no way to pass information about the state of the planning graph that has been built to the learned sampler. This could greatly accelerate finding solutions to problems by reducing the wastage of samples that are far from the current graph. For example, in an RRT, sampling near the goal is not useful in the beginning. This could also be tackled by a Recurrent Neural Network (RNN) that has a notion of time. In SBMP's, the order of samples can be important, which could be learned by the recurrent nature of RNN's. In addition, reducing the number of samples that are drawn from obstacle regions would greatly increase the time efficiency of the problem. Especially when the optimal path goes close to obstacles, many generated samples are within obstacles. Adding a loss that penalizes colliding samples would help to move the distributions away from occupied spaces. Finally, proving the results on more complex robots would validate the results in a way that cannot be done on smaller and easier problems.

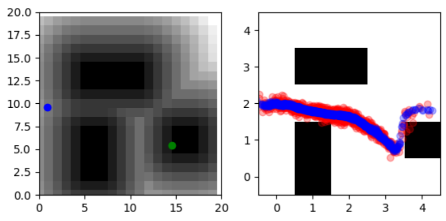


(a) Success Rate

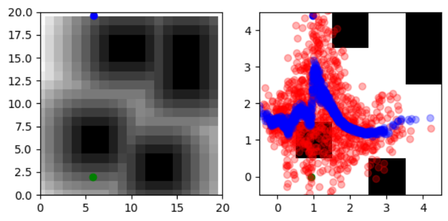


(b) Cost Ratio

Figure 5: Kinematic Car Result

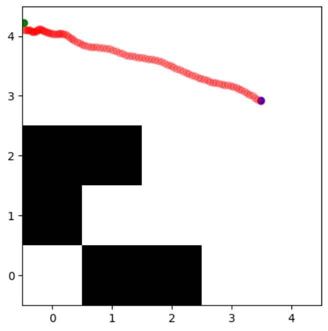


(a) Nominal Case Generated Distribution

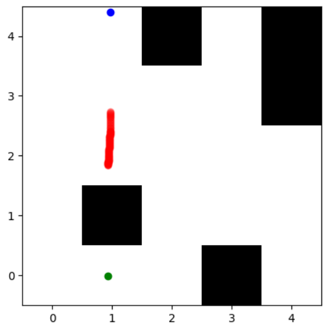


(b) Ambiguous Case Generated Distribution

Figure 6: Distributions generated by learned model, on the right we can see that the model struggles with ambiguous planning scenarios.



(a) Successful Interpolation



(b) Failed Interpolation

Figure 7: Interpolation between start and goal states.

References

[1] B. Ichter, J. Harrison, and M. Pavone. Learning sampling distributions for robot motion planning, 2017.

[2] I. A. Şucan, M. Moll, and L. E. Kavraki. The Open Motion Planning Library. *IEEE Robotics & Automation Magazine*, 19(4):72–82, December 2012. doi:10.1109/MRA.2012.2205651. https://ompl.kavrakilab.org.

[3] A. H. Qureshi and M. C. Yip. Deeply Informed Neural Sampling for Robot Motion Planning. sep 2018. URL http://arxiv.org/abs/1809.10252.

[4] C. Zhang, J. Huh, and D. D. Lee. Learning Implicit Sampling Distributions for Motion Planning. jun 2018. URL http://arxiv.org/abs/1806.01968.

[5] R. Kumar, A. Mandalika, S. Choudhury, and S. S. Srinivasa. LEGO: Leveraging Experience in Roadmap Generation for Sampling-Based Planning. jul 2019. URL http://arxiv.org/abs/1907.09574.

[6] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks, 2014.

[7] C. Doersch. Tutorial on variational autoencoders, 2016.

[8] I. Kobyzev, S. Prince, and M. Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, page 1–1, 2020. ISSN 1939-3539. doi:10.1109/tpami.2020.2992934. URL http://dx.doi.org/10.1109/TPAMI.2020.2992934.

[9] A. Pagnoni, K. Liu, and S. Li. Conditional variational autoencoder for neural machine translation, 2018.

[10] S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning, 2011.

[11] R. Kumar, A. Mandalika, S. Choudhury, and S. S. Srinivasa. Lego: Leveraging experience in roadmap generation for sampling-based planning, 2019.