A dataset bias problem for learning-RRT, with two potential solutions

T.M.Moerland, W.J.Wolfslag, M. Bharatheesha

Delft University of Technology

INTRODUCTION

We consider Rapidly-exploring Random Trees [1], the most popular sampling-based planning algorithm, and its application to kinodynamic planning. RRT builds a graph structure with the states of the system as nodes and the trajectories of the system between two states as edges. The algorithm selects a node to expand by randomly sampling a point in state space and then expand from the *nearest* node in the current graph. Expension is done by means of a local planner that aims to reach the randomly-sampled point. These steps happen online, so computation time is crucial. Unfortunately, in state-space, distance computation and (local) planning are computationally expensive [1].

A promising approach, which we call Learning-RRT, was proposed recently [2, 3] and has already shown promising initial results [4]. Learning-RRT involves an offline machine learning phase that learns the distance and steering function in the RRT. An optimal control algorithm provides a database of optimal trajectories, which is the input for a supervised learning algorithm. This algorithm learns to approximate the functions the RRT requires. Supervised learning provides two benefits: 1) generalization over state-space and 2) fast online predictions. Thereby, the computational burden of trajectory optimization does not have to be repeated for new situations, and is shifted offline.

Note that the final trajectory found by the RRT-algorithm is in general not optimal, even though the trajectories in the database are optimal. The database therefore is not build up of optimal trajectories to improve the cost to go of the RRT-trajectory. Rather, optimality improves learning performance by providing a meaningful cost-to go metric and trajectories that tend to have a similar shape if they connect similar points in state space.

DATASET BIAS PROBLEM

In our recent paper [5], we identified a key problem with the database created by optimal control algorithms: their trajectories are local optima. This causes a bias that interferes with learning performance. The problem is illustrated for an artificial dataset in Figure 1 (top), where in the middle input region we have the global optimum at the bottom, but there are local optima above it. A standard function approximator (with squared loss) for a given point in input space (independent variable) predicts the expectation of the dependent variable. This results in the conditional mean of the datapoints, as shown by the green line in Figure 1 (top). Note how the predicted function deteriorates in the middle segment, where it predicts the average instead of the bottom (optimal) cost.

In the context of learning RRTs, this is problematic for predicting the cost function and especially harmful for predicting the control parameters. Averaging over two locally optimal control inputs by no means guarantees that we end up anywhere close to the target. Below we discuss two potential solutions to this problem: a data-cleaning algorithm, and the use of a generative model.

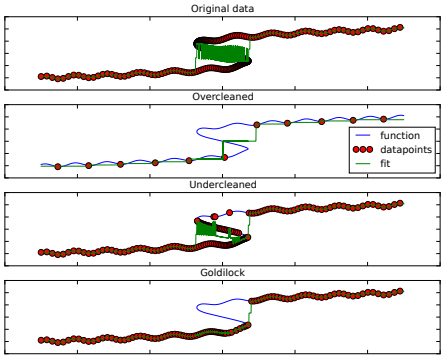

Figure 1: The data-bias problem and the effect of the d parameter in the data cleaning algorithm. The top figure shows an imaginary dataset, which has a problem with bias in the middle of its domain. The fitted function is a poor approximation of the least cost part of the datapoints. The second figure shows a cleaned dataset where the value for d was chosen too large: the bias is gone, but there is not enough resolution left to accurately fit the function. In the third function d is chosen to small: not all bias is removed. Finally, the bottom figure shows a cleaned dataset with a proper choice for d: the bias is removed, and enough resolution remains.

SOLUTION 1: DATA CLEANING

Our dataset contains of points with four attributes: an initial state, a final state, parameters that describe the input signal and a cost to go. The bias occurs due to some initial-final state combination occurring multiple times, but with a different input signal and cost. In those cases, we are interested in retaining the point that is the lower bound of the cost for that initial-final state combination. We prefer to remove points in high-density regions, as in low-density regions there is little to throw away, and we may only hope that our data are accurate. So we implicitly remove from high density regions by first uniformly sampling a point from our dataset. We then search for its nearest neighbour in the dataset based on a Euclidean distance. If this neighbour is within a distance d from our sampled point, we remove the node of the two with the highest cost. Otherwise, we retain both points. This process is repeated until no points are removed any longer, after which we return the cleaned dataset.

The effect of the parameter d on the cleaning is shown in Figure 1. We were able to empirically find an adequate setting for d for the pendulum swing-up problem, see [5]. However, we also noted that performance of the algorithm was sensitive to this parameter. We are therefore looking for ways to make the cleaning algorithm more sophisticated, i.e., capable of more accurately finding points that should be removed.

SOLUTION 2: GENERATIVE MODEL

There is an alternative to a cleaning algorithm. As we are not typically looking for the optimal motion, but a reasonable and feasible one, it is likely fine if our model returns locally optima. That is, intuitively, we are satisfied as long as the model returns a cost and steering signal that are consistent, and get the system from the specified initial state to the final state. One way to obtain such a model is the use of generative models[6]. Specifically, we use a conditional variational auto-encoder[7], for this purpose. Specifically, we train a decoder network to estimate the probability $P(\text{steering signal}, \text{cost} | \text{initial state}, \text{final state}, \text{latent space})$, where the latent space is used to determine which cost-steering input combination is returned by the model.

Following [8], this decoder is trained together with an encoder, such that the latent-space can be sampled according to some pre-specified prior at test-time. By sampling from this prior, and subsequently from the learned distribution, we obtain a consistent cost-steering signal combination.

This approach is currently being implemented. Two important steps are still to be taken: 1) testing the ability of the model to capture the underlying probability function, and 2) studying the effects of using a generative model (i.e., returning locally optimal trajectories) on the convergence of the RRT algorithm.

REFERENCES

[1] Steven M. LaValle and James J. Kuffner-Jr. Randomized Kinodynamic Planning. *The International Journal of Robotics Research*, 20:378–400, 2001.

[2] Mukunda Bharatheesha, Wouter Caarls, Wouter Wolfslag, and Martijn Wisse. Distance metric approximation for state-space RRTs using supervised learning. In *"IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2014.

[3] Luigi Palmieri and Kai O. Arras. Distance Metric Learning for RRT-based Motion Planning for Wheeled Mobile Robots. In *IROS Workshop on Machine Learning in Planning and Control of Robot Motion* , 2014.

[4] Ros Allen and Marco Pavone. A Real-Time Framework for Kinodynamic Planning with Application to Quadrotor Obstacle Avoidance. In *AIAA Guidance, Navigation, and Control Conference. San Diego, California, USA*, pages 5021–5028, 2016.

[5] Wouter Jan Wolfslag, Mukunda Bharatheesha, Thomas Mohamed Moerland, and Martijn Wisse. RRT-CoLearn: towards kinodynamic planning without numerical trajectory optimization. In *UNDER REVIEW IEEE Int. Conf. on Robotics and Automation*, 2018.

[6] Ian Goodfellow. NIPS 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016.

[7] Thomas M Moerland, Joost Broekens, and Catholijn M Jonker. Learning Multimodal Transition Dynamics for Model-Based Reinforcement Learning. *arXiv preprint arXiv:1705.00470*, 2017.

[8] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.