Behaviour Cloning of Cartpole Swing-up Policy with Model-Predictive Uncertainty Regularization

Kuo-Hao Zeng Pengcheng Chen Mengying Leng Xiaojuan Wang
{khzeng, pengcc, lengmy19, xiaojwan}@uw.edu

1 Introduction

In this project, we aim at exploring the effectiveness of model-predictive uncertainty regularization idea for tackling the well-known compounding errors issue in policy behaviour cloning. The motivation is that we cannot directly learn a policy for real robots in the real world, because it is slow, expensive, and unsafe. Thus, a lot of works [2, 10, 11] instead collect control trajectories via expert policy or human demonstration and learn the policy by behaviour cloning (BC) [1, 12]. However, BC is notorious for its compounding errors issue [8], which often causes the learned policy fails on the out-of-domain testing data/environments due to covariate shift.

Henaff *et al.* [5] proposed an uncertainty regularization approach to alleviate the compounding errors issue. The idea is to first learn a dynamic model (transition model) of the agent (environment) by Bayesian Neural Network [4, 9], then, at each time step, utilize the model to predict N possible next states. Further, the learning framework estimates the uncertainty among N possible future states and exploits this measurement to regularize the policy learning. We adapt their proposed idea to our studied task by several modifications, which are introduced in the following paragraph. The introduction video to this project is available at `https://drive.google.com/open?id=1WZJ4ulTXD3X9bEGl0UXfBG9y3CgtK0Ts`.

Contribution - In this project, we adopt the idea of uncertainty regularization to learn a swing-up policy via BC without interacting with the simulator. We make several modifications to adapt the learning framework for our focused task. (i.) since our policy learning entirely relies on BC, our policy network does not need to interact with the environment during the training phase. Therefore, we remove the simulator from our learning framework, except for the data collection process. (ii.) We use state observation instead of image observation to ease the learning of dynamic model. In this case, we are able to focus on the effectiveness of uncertainty regularization approach. (iii.) We slightly modify the learning framework by changing the policy cost to behaviour cloning objective to fit our problem setting. (iiii.) To make the focused task simple, we do not adopt the z-dropout technique proposed by original authors, we rather directly utilize the simplest dropout technique [6] to perform Bayesian Neural Network. In the end, we demonstrate that the policy learned with uncertainty regularization slightly outperform the policy learned without it. The source code for this project are available at `https://github.com/KuoHaoZeng/cartpole_model_based_control`.

2 Method

Behaviour cloning - The input to our policy network M is the state $s_t = [\Delta\theta_t, \Delta p_t, \theta_t, p_t]$, which encodes the angular velocity of the pendulum, velocity of cart, angle of the pendulum, and position of cart at time t. The output from the policy network is a force u_t, which the policy would like to perform on the cart at time t. We utilize the swing-up policy provided in *HW1* as our expert policy to learn our policy network via BC. In practice, we let the expert policy interact with simulator for H time-steps and collect the state $S \in \{s_t\}_{t=0}^{H-1}$ and force $U^* \in \{u_t^*\}_{t=0}^{H-1}$ as our training data. During the training phase, we follow the simplest BC objective to learn our policy network by minimizing the mean absolute loss: $L_1(U, U^*)$, where $U = M(S)$. During the testing stage, we do the rollout for H steps by letting the learned policy network interact with simulator and collect its output force $U \in \{u_t\}_{t=0}^{H-1}$. Then, we evaluate the performance by measuring the least square distance between U and U^*: $L_2(U, U^*)$. Overall, we try 6 different backbone architectures, including 3-layers fully-connected layer (FC), GRU [3], LSTM [7], dropout FC, dropout GRU, and dropout LSTM.

Dynamics model - The input to our dynamics model D is the augmented state $\tilde{s}_t = [\Delta\theta_t, \Delta p_t, sin(\theta_t), cos(\theta_t), p_t, u_t]$ and the model returns a transition state Δs_t at time t. Here, we also use the provided swing-up policy to collect data (S and U^*) and learn the model by minimizing the mean absolute loss: $L_1(\Delta S, \Delta S^*)$, where $\Delta S = D(\tilde{S})$ and $\Delta S^* = \{s_{t+1} - s_t\}_{t=0}^{H-2}$. During the testing phase, we also employ the least squares distance to measure the discrepancy between ΔS and ΔS^*. Overall, we try 3 different backbone architectures, including dropout FC, dropout GRU, and dropout LSTM. However, we found that dropout LSTM performs best (according to our mid-term report), therefore we only report the results with dropout LSTM as dynamic model backbone.

Model-predictive uncertainty regularized behaviour cloning - Fig. 1 shows the learning framework for the uncertainty regularized BC. As presented in the figure, we utilize the learned dynamics model to predict the possible transition state Δs_{t+1} and compute a variance over the $N = 10$ predicted transition states $\{\Delta s_{t+1}^i\}_{i=0}^9$ from 10 dynamics models $\{D^i\}_{i=0}^9$ with different dropout masks. The variance is calculated by $tr\Big[Cov[\{\Delta s_{t+1}^i\}_{i=0}^9]\Big]$, where tr denotes the trace operation and Cov denotes covariance matrix. The variance measures how different the sub-dynamics models' output are, which implies how many discrepant outcomes the force u_t may produce if it is applied to the environment. It measures how much uncertainty the learned dynamics model thinks according to the input state s_t and force u_t. Thus, the uncertainty cost encourages the policy network to produce forces which, when interacts with the dynamics model, will produce predictions which the dynamics model is confident about. Finally, to combine with BC, we times this model uncertainty cost by λ and plus it with the BC cost ($L_1(u_t, u_t^*)$) to obtain our overall learn objective.

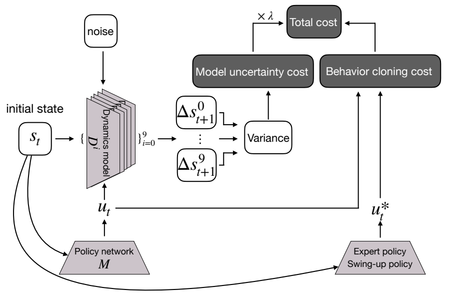

Figure 1: The learning framework.

3 Experiments and conclusion

In this project, we use dropout LSTM as the backbone for dynamics model pretrainig. We train the policy network from scratch with the pretrained dynamics model 5 times with different random seeds and report the average results over them. The hidden size of all model are set by 64. We use batch size of 50, i.,e., 50 trajectories, and each trajectory includes $H = 50$ time steps. We use Adam optimizer with $\beta_1 = 0.9, \beta_2 = 0.99$, and starting learning rate 0.01. The learning rate is decayed by an order at the 60-th and 80-th iteration. The dropout is applied before the last output layer with dropout rate of 0.05. We train the model by 100 iterations, and use the checkpoint with lowest training loss for the final evaluation. We report the results for different $\lambda = \{0, 0.01, 0.1, 0.15\}$, where the learning degenerates to the pure BC as $\lambda = 0$. The performance is evaluated by measuring the least squares distance in the trajectory between U and U^*: $L_2(U, U^*)$. To perform a robust evaluation, we collect 10 rollouts with the noise-enabled simulator and report the average distance.

The results are in Tab. 1. We can find our model achieves lowest error with $\lambda = 0.01$, which shows the uncertainty regularized model performs sligtly better than pure BC. It is also observed that fully connected backbones performs much worse than recurrent models. It is reasonable because the model utilizing memory is able to access more information and adjust its behaviour accordingly. The videos[1] illustrate the model-predictive uncertainty regularized behaviour cloning is able to learn the swing-up behaviour well. However, based on the qualitative results, it is hard to distinguish the difference between the policy learned with uncertainty regularization and the policy learned by pure BC.

Table 1: $L_2(U, U^*)$ w.r.t. different regularization weights λ and policy network backbones. The names start with "d-" means dropout is applied.

	d-fc	d-gru	d-lstm	fc	gru	lstm	Avg.
$\lambda = 0.0$	0.657	0.566	0.539	0.649	0.537	0.534	**0.580**
$\lambda = 0.01$	0.641	0.564	0.527	0.629	0.543	0.516	**0.570**
$\lambda = 0.1$	0.640	0.551	0.554	0.631	0.540	0.527	**0.574**
$\lambda = 0.15$	0.642	0.555	0.539	0.646	0.550	0.510	**0.574**

In conclusion, we demonstrate that the policy learned with uncertainty regularization slightly outperform the policy learned by pure BC. However, some interesting directions may be worthy to explore in the future, such as using the GP to measure the uncertainty instead of a learned dynamics model.

[1] `https://drive.google.com/open?id=1vRpvi3G-4KpnD6k95rBvm104NjcYg501`

References

[1] Charles W Anderson, Bruce A Draper, and David A Peterson. Behavioral cloning of student pilots with modular neural networks. In *ICML*, 2000.

[2] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[3] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NeurIPS Workshop*, 2014.

[4] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *ICML*, 2016.

[5] Mikael Henaff, Alfredo Canziani, and Yann LeCun. Model-predictive policy learning with uncertainty regularization for driving in dense traffic. In *ICLR*, 2019.

[6] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. In *arXiv preprint arXiv:1207.0580*, 2012.

[7] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. In *Neural computation*, 1997.

[8] Mike Laskey. Imitation learning's compounding errors. In *https://people.eecs.berkeley.edu/ laskeymd/research.html*.

[9] Radford M Neal. *Bayesian learning for neural networks*. Springer Science & Business Media, 2012.

[10] Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *NeurIPS*, 1989.

[11] Rouhollah Rahmatizadeh, Pooya Abolghasemi, and Ladislau Bölöni. Learning manipulation trajectories using recurrent neural networks. *ar Xiv preprint arXiv: 1603.03833*, 2016.

[12] Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. 2018.