Immitation Learning for Cart-Pole Systems

Final Project Report for DATA 1030, Fall 2020 at Brown University

Supervised by Prof. Andras Zsom

https://github.com/maitreyakv/imitation-learning.git

Maitreya Venkataswamy

I. INTRODUCTION

A. Motivation for Imitation Learning

Imitation learning is an application of supervised machine learning in robot control that learns a control policy by mimicking another sophisticated controller called the *expert*[1]. The *expert* has the ability to take a robot in any scenario and compute what the best control inputs to the robot are to take it to a desired position and pose. By generating a set of possible scenarios for the robot and querying the *expert* for the optimal control in each scenario, a data-set is built called the *demonstrations*. The features are the variables that describe the state of the robot (the state variables), and the target variables are the control inputs that bring the robot towards the goal. This data-set is used to train a regression model that replaces the *expert* by "predicting" what the optimal control is for the robot. One motivating example for why this might be valuable is autonomous vehicles. The expert in a car might be a human who creates the *demonstrations* to train a model to replace the human as the driver.

B. Description of the Cart-Pole System

The "cart-pole" system, also known as the "inverted pendulum" system, is a popular under-actuated[1] system for studying control because it is low-dimensional, yet still challenging to control [2]. The system consists of a cart on a track with a pole pinned at one end to the cart. The pole rotates about the pivot point around an axis perpendicular to the track and the vertical direction. The system has one input $\mathbf{u}$ in the form of a continuously-variable force on the cart that can act in either direction along the track. The *maneuver* of interest, shown in Fig. 1, is to swing the pole from a hanging position up to a standing position while bringing the cart to rest at a position along the track. The state variables of the system are the cart position z, the angle of the pole with the vertical direction θ, the cart velocity $\dot{z}$, and the pole angular velocity $\dot{\theta}$.

C. Use of Nonlinear Control Algorithms as the Expert

The *expert* is a control algorithm called Differential Dynamic Programming (DDP). This method is an iterative algorithm that provides optimal control to bring a nonlinear system from an initial state to a target state by initializing a random control sequence and then improving it by performing a series of small updates through forward and backwards passes. The algorithm requires a user to specify what is meant by "optimal control" through a cost function that balances minimizing control cost and proximity of the final state to the target state. This cost function is the method by which the user tunes the algorithm to favor aggressive or conservative control policies. When tuned properly the algorithm provides exceptional control inputs for many different nonlinear systems. However, it can be very computationally expensive, motivating replacing it with a properly trained machine learning model can provide large computational benefits.

[1]The number of controls is less than the number of state variables.

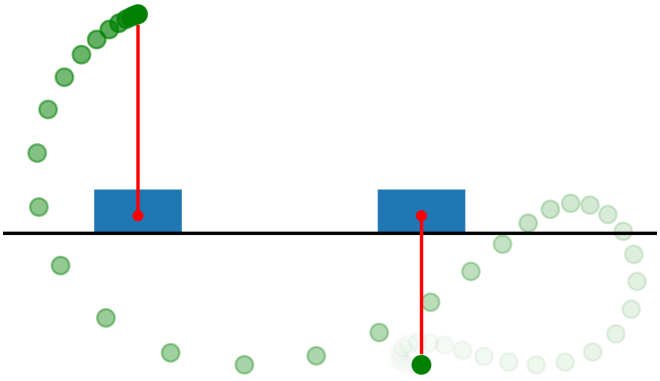

Fig. 1. Example trajectory of the swing-up maneuver.

D. Creation of Demonstrations

The algorithm is used to generate controls for 500 simulations with initial cart positions sampled uniformly on the interval $[-2, 2]$. Each initial position is created with the cart and pole at rest with the pole hanging downwards with an angle sampled uniformly on the interval $[-\pi/16, \pi/16]$. For each simulation, the algorithm returns 60 data-points since each trajectory of $2\,\mathrm{s}$ is discretized with a frequency of $30\,\mathrm{Hz}$. These 30,000 points are collected as the *demonstrations*, with the four state variables z, θ, $\dot{z}$, and $\dot{\theta}$ as the four features, and the control force on the cart u as the target variable.

II. EXPLORATORY DATA ANALYSIS

Fig. 2 shows that there is some separation of the data-points into positive and negative control based on the position of the cart, suggesting that the position of the cart is a decent predictor of the control force. When coupled with the cart velocity, it becomes a stronger predictor, visible as the structure shown in Fig. 3. From the color of the points in the figure, one can see that these two features together are a good predictor of the control force.

From Fig. 4 a similar result is present with respect to the pole angle and its time-derivative. The spiral structure as well as the correlation of the control force with the position along

TABLE I

SUMMARY STATISTICS OF THE FEATURES AND THE TARGET VARIABLE.

Feat.	Mean	Std. Dev.	Min.	25%	50%	75%	Max.
z [m]	0.13	0.70	-1.99	-0.17	0.04	0.46	1.99
θ [rad]	-0.50	1.41	-3.13	-1.41	-0.19	0.40	1.78
$\dot{z}$ [m/s]	-0.00	0.72	-2.02	-0.40	-0.02	0.48	1.79
$\dot{\theta}$ [rad/s]	-1.52	4.65	-11.37	-3.71	-1.27	1.12	8.27
u [N]	0.00	2.15	-5.72	-1.40	-0.13	1.46	5.86

the spiral, indicate that the pole angle and its time-derivative are also good predictors of the control force.

In reality, all four features have a complex relationship with the target variable, with no single feature being a good predictor by itself. Fig. 3 and Fig. 4 are only projections of some four-dimensional scatter-plot with structure into two dimensions, suggesting that if a machine learning model is able to properly consider the non-linear combinations of all the features, then it has a good chance of predicting the control. This is a symptom of the complex and non-linear nature of the dynamics of the system, as well as the fact that it is under-actuated. This complex relationship means that the regression technique used to learn the relationship will need to be a non-linear technique, or some feature engineering will have to be performed to ”linearize” the problem.

III. MACHINE LEARNING METHODOLOGY

A. Data Pre-processing

Within each simulation, the data-points are a time-series, which suggests that a time-series splitting technique needs to be used. However, the problem of interest is not predicting future target variables given past features. Therefore there is no reason to validate the model on points in the time-series that occur after the training points, since previous states have no bearing on the current control decision. The only consideration is that for every simulation in the training set, the entire simulation must be in the training set so that the model can learn the control policy for every stage in the *maneuver*. Points in the validation set should not be part of trajectories in the training set, since it is important that the model's ability to predict the control for every part

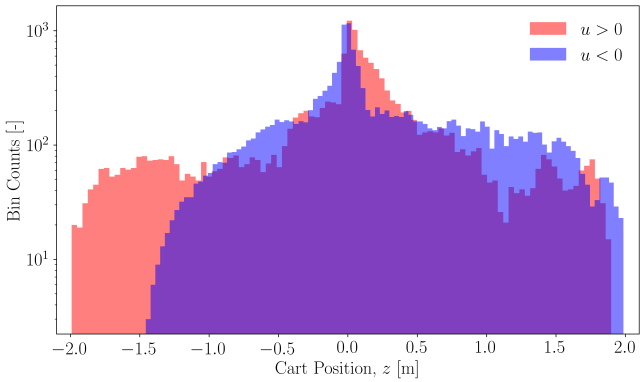

Fig. 2. Histograms of cart position for both positive and negative control forces.

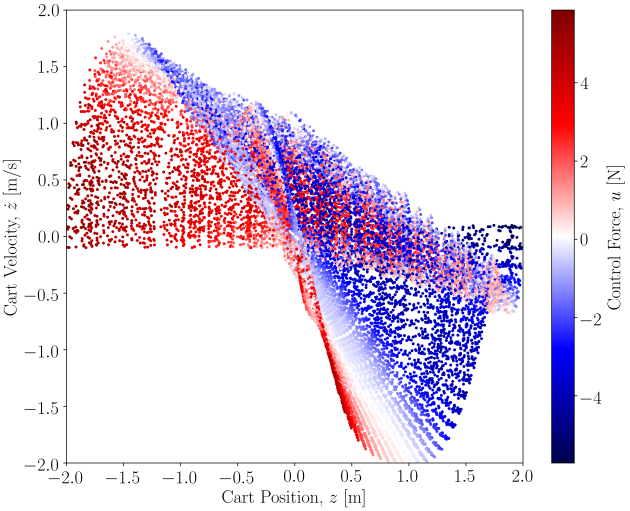

Fig. 3. Scatterplot of cart position and magnitude of cart velocity, colored by the magnitude of the control force.

of a maneuver is evaluated. Therefore, the splitting strategy used is a grouped K-Fold splitting strategy, where a group is defined to be a single simulation.

For some of the supervised machine learning models used in this study, the model benefits from the training data having a vanishing mean and a unit variance. For these models, the features of the *demonstrations* are shifted by their respective means, and normalized by their variances.

B. Model Training

Four models are trained using the *demonstrations*: a linear model with L_2 regularization and polynomial feature engineering [3], a random-forest [3], an adaptive-boosted tree (AdaBoost) [3], and a gradient-boosted tree (XGBoost) [4]. The accuracy of the predictions of each model is measured using the mean arctangent absolute percentage

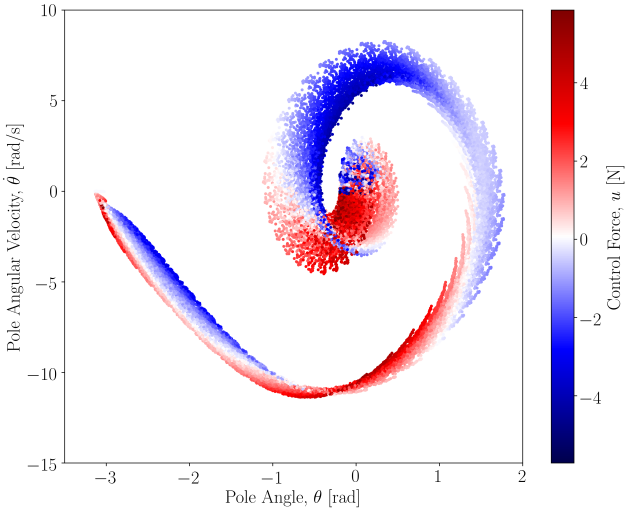

Fig. 4. Scatterplot of pole angle and magnitude of pole angular velocity, colored by the magnitude of the control force.

TABLE II

HYPER-PARAMETER SEARCH SPACES FOR THE MODELS.

Model	Hyper-param.	Search Space	Optimal Value
Poly. Ridge	poly. degree	$[1, 6]$	6
	L_2 reg. strength	$[10^{-5}, 10^{5}]$	1.04×10^{-4}
Random Forest	tree depth	$[1, 50]$	44
	sample size	$(0, 1]$	0.951
AdaBoost	tree depth	$[1, 50]$	32
XGBoost	tree depth	$[1, 50]$	25
	L_1 reg. strength	$[10^{-8}, 10^{8}]$	2.27×10^{-5}
	L_2 reg. strength	$[10^{-8}, 10^{8}]$	0.208

error (MAAPE) [5] in Eq. 1, since it is a scale-invariant regression metric like mean absolute percentage error (MAPE) without the singularity when the actual target value is zero.

$$\text{MAAPE} = \sum_{n=1}^{N} \arctan\left(\left|\frac{\widehat{y}_n - y_n}{y_n}\right|\right) \tag{1}$$

The scale invariance of the metric is important because the magnitudes of the control forces are very different between the initial part of the swing-up maneuver and the stabilization phase once the pole is upright. Without the scale invariance, the errors of predictions of stabilizing control forces would be insignificant compared to those of the swing-up portion. Table. II summarizes the hyper-parameters and the search space for each model.

The hyper-parameters for each model were tuned using Bayesian optimization [6], with a cross-validation strategy utilizing the grouped K-Fold splitting mentioned earlier. The Bayesian optimization algorithm was found to be more efficient than an exhaustive grid-search optimization, and doesn't require an explicit discretization of the search space[2]. After the optimal hyper-parameters were found for each model, the accuracy of the optimal models was estimated by splitting the demonstrations into ten sets of training and test subsets using the grouped K-Fold splitting technique, in order to quantify the uncertainty in the model accuracy due to the randomized nature of some of the model training algorithms. Each subset pair was used to train and evaluate the models, and the resulting test scores are shown in Fig. 5.

IV. RESULTS

A. Model Accuracy and Interpretation

All of the tuned models have test scores that are well below the baseline MAAPE score[3] of 0.7986. In fact, the worst observed test score of every model is more than 50 standard deviations below the baseline score. The adaptive-boosted tree model performed the best, so it is selected for evaluation on the simulated system as a replacement for DDP.

To interpret the model, the Shapely values [7] are computed for each point in the demonstrations. Fig. 6 shows the distributions of the absolute values of the Shapely values

[2]The endpoints of the intervals defining the search spaces still needs to be defined.

[3]This baseline score corresponds to a constant prediction of zero (the average of the target variables in the demonstrations) for the control force.

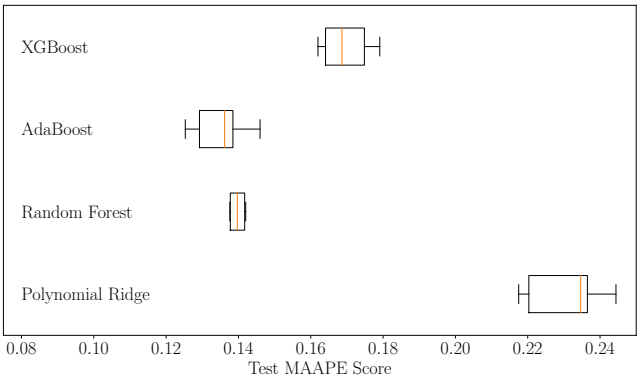

Fig. 5. Test scores of the four tuned models.

for each feature for the purpose of examining global feature importance. While the bulk of the distributions indicate that all four features are important, the upper end of the range of the distribution of Shapely values for the pole angular velocity $\dot{\theta}$ suggest that this feature can sometimes be the dominating factor in determining the control force. This supports the idea that the swing-up maneuver is achieved by leveraging the angular momentum of the pole as the mechanism to bring it to the vertical position, since the pole angle is not directly controllable. This concept is further supported by the fact that the pole angle θ is often the second most important feature, according to Fig. 6. The other two features, the cart position z and velocity $\dot{z}$ are slightly less important, which reflects the fact that the iLQR/DDP algorithm, of which the models are trying to mimic, was tuned in a way that prioritizes swinging the pole to the upwards position first, and then positioning the cart at the desired location along the track.

B. Model Evaluation

The adaptive-boosted tree model is implemented in the simulated cart-pole system to evaluate its ability to replace the current expert algorithm. Fig. 7 shows the pole angle θ histories for 500 simulation with the same time discretization of 30 Hz that was used in the demonstrations. All of the

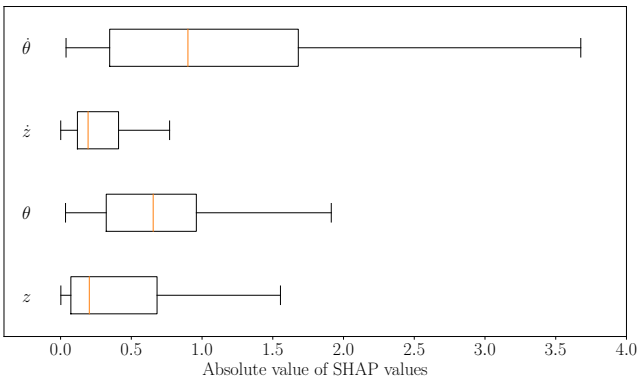

Fig. 6. Boxplots of absolute values of Shapely values for the adaptive-boosted model to inspect global feature importance.

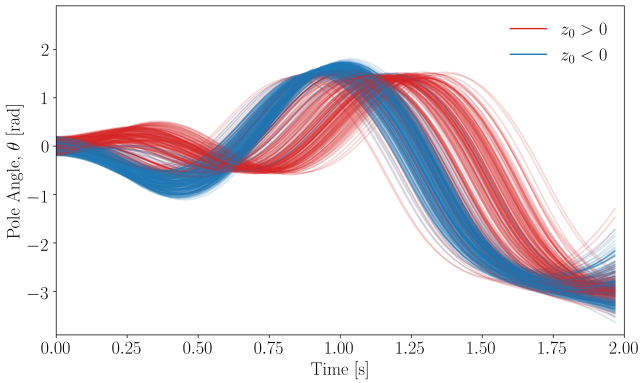

Fig. 7. Histories of the pole angle θ for the 500 simulations using the adaptive-boosted tree model as the controller.

simulations managed to successfully bring the pole to the upright position, but few simulations managed to stabilize the pole in that position. This suggests that the model successfully learned how to perform the bulk of the swing-up maneuver, but it didn't learn how to stabilize the pole once it was in the upwards position. This seems to contradict the good test scores achieved during the model training, but the reality is that extremely high levels of accuracy are required for the fine control portion of the maneuver. The consequence of this phenomenon is that the model score alone cannot be used to evaluate the model's success in imitating the expert; full simulations or tests need to be conducted to perform such evaluations.

V. OUTLOOK

While the trained model was mostly successful in performing the swing-up portion of the maneuver, it failed to learn the fine control required to stabilize the pole in the upright position. A possible way to improve the performance in this regard would be to collect more demonstrations of the pole being stabilized in order to train the current model, or a separate model, on how to stabilize the pole. Another technique which could work is to hand over the control to a linear controller like a LQR controller after the pole has been swung upwards. These simpler controllers have been shows to be very good at stabilization, due to the fact that the model dynamics are approximately linear when the pole is close to the upright position.

REFERENCES

[1] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, *Robot Programming by Demonstration*, pp. 1371–1394. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.

[2] A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuronlike adaptive elements that can solve difficult learning control problems," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-13, no. 5, pp. 834–846, 1983.

[3] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[4] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, (New York, NY, USA), pp. 785–794, ACM, 2016.

[5] S. Kim and H. Kim, "A new metric of absolute percentage error for intermittent demand forecasts," *International Journal of Forecasting*, vol. 32, no. 3, pp. 669 – 679, 2016.

[6] T. Head, M. Kumar, H. Nahrstaedt, G. Louppe, and I. Shcherbatyi, "scikit-optimize/scikit-optimize," Sept. 2020.

[7] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems 30* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), pp. 4765–4774, Curran Associates, Inc., 2017.