6.884 HW2

Gabriel Margolis

March 2020

1 Problem 1: Learning Inverse Model

1.1 Source Code

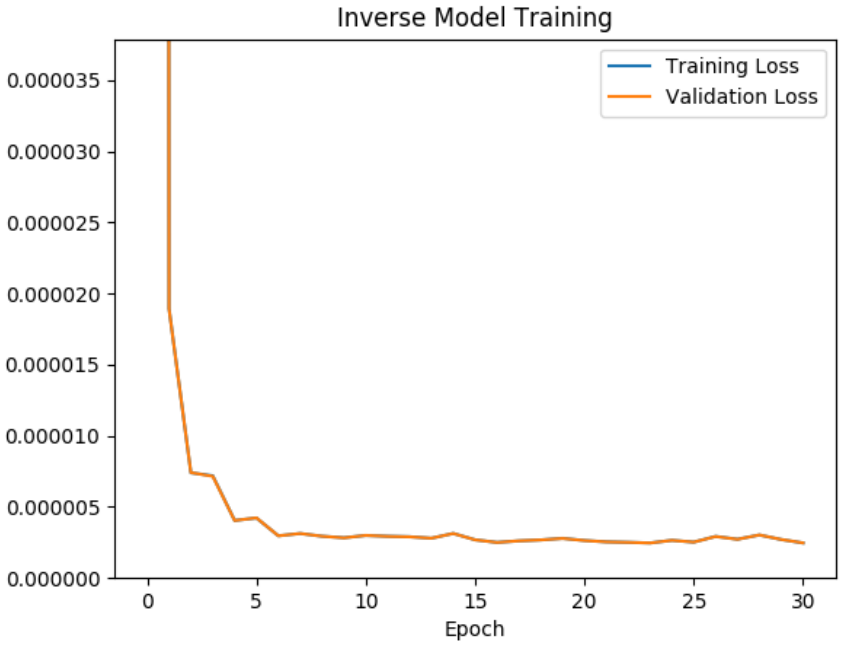
The code for all aspects of this lab including PPO implementation, training, and evaluation is available here: https://github.com/gmargo11/csl-hw2. The relevant files for the inverse model are 'inverse_model.py' and 'push_env.py'.

1.2 Model Description

The inverse model takes as input a four-dimensional vector consisting of the initial and final object coordinates (x_i, y_i, x_f, y_f). The model is trained to output the corresponding end effector action (ex_i, ey_i, ex_f, ey_f).

Our inverse model architecture, implemented in PyTorch, consists of two 30-unit hidden layers with ReLu activation functions. The inverse model is trained with an MSE loss function using the Adadelta optimizer; these are standard choices for this type of regression problem.

1.3 Inverse Model Training Plot



The model was trained for 30 epochs; it seems to have reached a loss minimum around epoch 6. Final training loss (MSE): $2.464E-6$; Final test loss (MSE): $2.467E-6$

1.4 Inverse Model Push Video

Video of a push planned by the inverse model: Ground truth push — Inverse model push

The distance error between final pose and goal pose for the push in the video was 0.0185 m. The average distance error between final pose and goal pose across 10 random seeds was 0.0268 m.

2 Problem 2: Learning Forward Model

2.1 Source Code

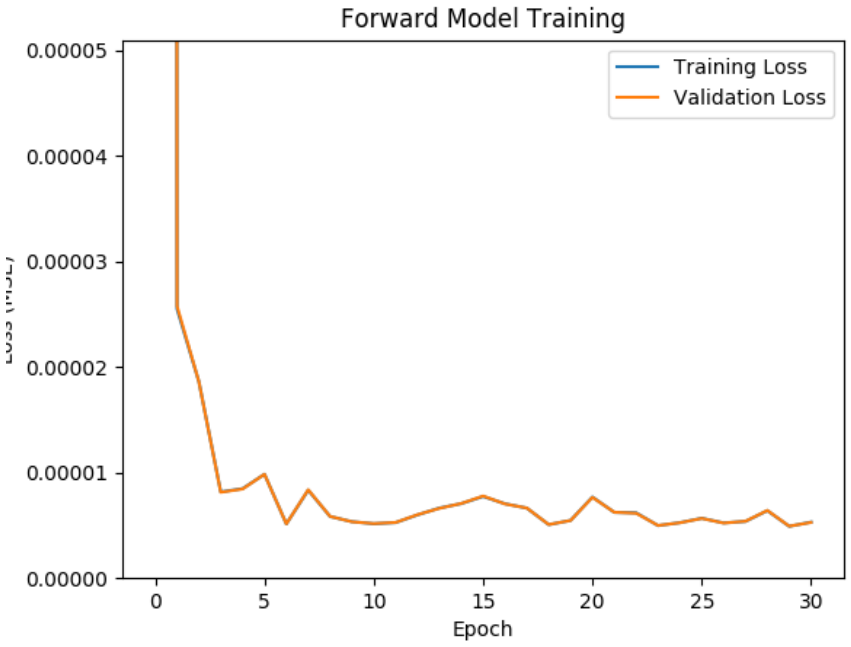
The code for all aspects of this lab including PPO implementation, training, and evaluation is available here: https://github.com/gmargo11/csl-hw2. The relevant files for the forward model are 'forward_model.py' and 'push_env.py'.

2.2 Model Description

The forward model takes as input a six-dimensional vector consisting of the initial object coordinates and end effector action $(x_i, y_i, ex_i, ey_i, ex_f, ey_f)$. The model is trained to output the corresponding final object coordinates (x_f, y_f).

Our forward model architecture, implemented in PyTorch, consists of two 30-unit hidden layers with ReLu activation functions. The forward model is trained with an MSE loss function using the Adadelta optimizer; these are standard choices for this type of regression problem.

2.3 Forward Model Training Plot



The model was trained for 30 epochs; it seems to have reached a loss minimum around epoch . Final training loss (MSE): $5.266e-6$; Final test loss (MSE): $5.267e-6$

2.4 Forward Model Push Video

Video of a push planned by the forward model with CEM: Ground truth push — Forward model push

The distance error between final pose and goal pose for the push in the video was 0.0263 m. The average distance error between final pose and goal pose across 10 random seeds was 0.0257 m.

3 Problem 3: Extrapolating with Learned Model

3.1 Inverse Model Two-Step Push Video

Videos: Inverse truth two-push — Inverse model two-push

The distance error between final pose and goal pose for the push in the video was 0.0397 m. The average distance error between final pose and goal pose across 10 random seeds was 0.0336 m.

3.2 Forward Model Two-Step Push

Videos: Forward truth two-push — Forward model two-push

The distance error between final pose and goal pose for the push in the video was 0.0151 m. The average distance error between final pose and goal pose across 10 random seeds was 0.0245 m.

3.3 Analysis

The inverse model receives inputs it has never seen before and extrapolates reasonably, but with some lost accuracy. The forward model, on the other hand, is not really asked to generalize at all on the first push; it is just used by the CEM action selector to pick an in-distribution action for the first push that moves the state as close to the true state as possible.

As we can see in the video, the result is that the inverse model prefers to achieve the goal in one extra-long (out-of-distribution) push, and then intentionally whiffs the next push if the first is successful. On the other hand, since CEM only tries in-distribution actions, the forward model tends to take two normal-sized pushes to achieve the goal (we can observe this in the example video).

In some sense, the inverse model demonstrates more interesting generalization by not restricting the actions it can take. If the contest were to achieve the goal in as few pushes as possible, we would say that the inverse model wins. But, we also observe that the forward model achieved better final accuracy through its strategy - this demonstrates that there can be some benefit to restricting the action space.