MIDS W251 Spring 2020

Section: Monday 6:30pm Suhas Gupta

HW11 – OpenAI Gym Lunar Landing (Reinforcement Learning)

Model Training and hyperparameter search

The following hyperparameter combinations were tried during training the lunar landing with reinforcement learning algorithms from OpenAI Gym.

Model	Hidden Layers & Sizes	Activation Function	Optimizer	Learning Rate	Batch Size	Epoch
Base Model	[32],[32]	ReLU	Adam	1E-3	32	10
Model 2	[64],[64]	Tanh	Adam	1E-5	32	10
Model 3	[32],[32]	Tanh	Adam	1E-5	16	10
Model 4	[128],[64]	ReLU	SGD	1E-5	64	20

Table 1: Model hyperparameter search

Hyperparameter Discussion:

1. **Optimizer:** The base model uses a 4-layer neural network with a training batch size of 32 and Adam optimizer. The Adam optimizer is known empirically to converge more quickly to local minima compared to SGD optimizer. This can be seen from the training times of the first three models vs. the model 4 that use the Adam and SDG optimizers respectively. SGD converges slowly but to flatter minima compared to Adam and this is evident from the training reward evolution curves (Figure 1).

2. **Activation function:** Tanh is a scaled sigmoid function that is nonlinear and is bounded by (-1,1). It has steeper gradient than sigmoid and also suffers from vanishing gradient problem. We see some of these attributes from the reward curves in figure 1. For model 2 & model 3, reward curves show spikes that don't seem to have a consistent trend with the training steps. ReLU is alos a nonlinear activation function with a key property that the output is 0 for negative input. However, ReLU is not bounded on the upper end unlike tanh and is thus susceptible to exploding gradients. ReLU gives us the benefit of a lighter network because of the 0 output for negative input property.

3. **Hidden layer & sizes:** These are the internal layers of a neural network that define the complexity of the model and determine how much the model "fits" the data. Each hidden layer in the network fits to a feature/set of features of the training data set. A very deep and wide model might fit the training data very well but not the test data ("overfitting") while the other can be true for a light and small neural network ("underfitting"). The best model ("model 4") in table 1 has fairly wide size of the 2 hidden layers used. This gives enough complexity to the network to enable the best landing model for our reinforcement learning game.

4. **Learning rate**: Learning rate is the step size by which the gradients are scaled during each training step during the gradient descent to find the minima. Too slow a learning rate can cause the network to training very slowly while too high a learning rate might cause the network to diverge or oscillate around the minima. From Table 1, the learning rate changes do no seem to have too much impact on the accuracy and speed of the algorithm. This is probably due to the fact that our network is very small.

5. **Batch size**: Batch size impacts the loss function and gradient during one forward pass of the training step. A larger batch size can help the network train faster. However, this is hard to point out from our table because we are modifying other parameters of the network that do not allow a 1:1 comparison for this parameter.

6. **Epochs:** Training epochs is the total number of retraining runs that are performed on the network. Each epoch consists of forward and backward passes on the full training data set. Number of epochs directly

impacts the accuracy of the model. We can see that a larger number of training epochs for model 4 does have a significant impact on the landing accuracy of lunar lander model.

Model performance comparison

Model	Total successful landing during prediction	Training Times (sec)	Prediction Times (sec)
Base Model	32	2720.52	135.55
Model 2	37	2615.07	135.29
Model 3	33	2504.46	157.33
Model 4	**48**	2885.4	152.3

Table 2: Model performance comparison for successful landing and training times

Model reward evolution during training

Figure 1 shows how the rewards vary with the number of steps for each of our models. Model 4 has consistent performance across all the training steps while model 2 is worse in the beginning and then gets better as more training steps are run.

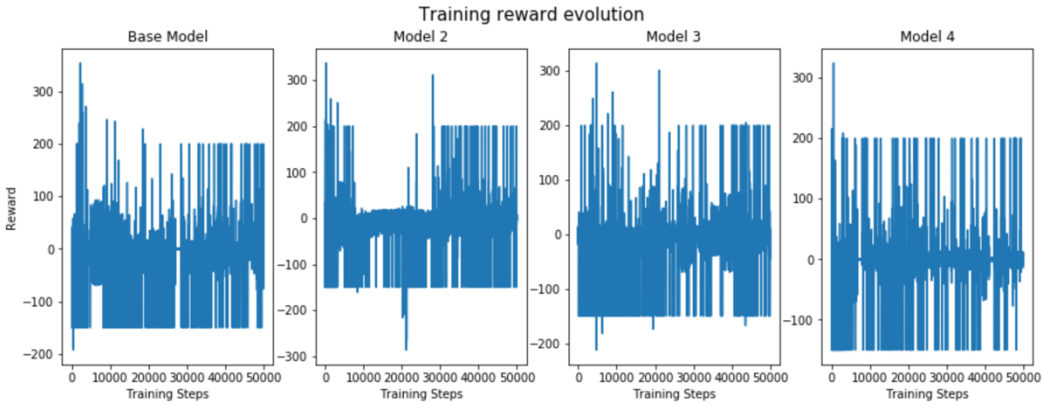

Figure1: Model reward evolution during training

Model rewards during prediction

The high rewards in the prediction indicate the instances when the landing is successful. The figure shows that model 4 is our best model (also see Table 2).

Figure2: Model rewards during prediction showing successful landing

Public Bucket with landing videos of best model:
https://s3.us.cloud-object-storage.appdomain.cloud/w251-suhasgupta-hw11/