Principles of Robot Autonomy II (CS237B): Homework 3

[Chetan Reddy] [Narayanaswamy]

03/07/25

Problem 1:

Part (i)

Weights and Biased Initialized to zero:

- If all the weights are initialized to zero, all the activations will be identical irrespective of the input and will undergo identical updates.
- This will inhibit the learning ability of the network. The second issue is vanishing gradient.

Part (ii)

Xavier Initialization

- It is a weight initialization technique that aims to maintain the variance of activations and gradients across layers, preventing them from becoming too large (exploding) or too small (vanishing) during training.
- The weight initialization is as follows:

$$W \sim \mathcal{U}\left(-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}\right)$$

  where $\mathcal{U}$ is a uniform distribution and n is the number of neurons in the previous layer.
- The paper also talks about a variant of this initialization called Normalized Initialization.
- The best results are obtained when using Xavier initialisation with tanh or softsign activation functions.

Part (iii)

Adam Optimizer

- Adam is an Optimizer incorporates the benefits of both momentum and RMSProp, leading to more efficient optimization. At every time step, it keeps track of the first moment and second moment along with the gradient.
- Adam adjusts the learning rate for each parameter, which helps accelerate learning and improves convergence.
- One drawback is that Adam has multiple hyperparameters (α, β_1, β_2 and ϵ) to update the weights, moments and 2nd order moments.
- A second drawback is that Adam can be slower because of the heavy computer per update step.

Part (iv)

Training for 500 epochs at once vs Training Twice with 250 Epochs each

- Under the assumption that an epoch is divided into fixed sequence of batches (which are recorded as mentioned), **SGD will result in the same network in case (1) and (2)** because an update step includes only the gradient which depends only on a batch and the present parameter value and does not depend on the history of the updates.
- However, when using Adam, the history of the updates matter since it keeps track of the first and second moments (as described in the previous part). Therefore, when the training restarts after 250 epochs, the first and second moments will be set to zero and the updates will be different from the 251st epoch. Therefore, **Adam results in different models for (1) and (2)**.
- However, if we assume that minibatches are sampled within the batches during the update steps, SGD will result in different models in (1) and (2) because of the random sampling.

Problem 2:

Part (i)

When we have a neural network (with weights θ_g) separately trained for each goal g and assuming an MSE loss function L between $h_\theta(o, g)$ and action a, the optimization problem would be as follows for a given goal g:

$$\theta_g^* = argmin_\theta \sum_{i=1}^{N} (w_1(h_\theta(o^{(i)}, g)_1 - a_1^{(i)})^2 + w_2(h_\theta(o^{(i)}, g)_2 - a_2^{(i)})^2)$$

The terms w_1 and w_2 are the weights given to each actions dimensions (steering and throttle). If $w_1 > w_2$ is high, we are penalizing errors in steering more that that of throttle.

Part (ii)

Implemented in Code

Part (iii)

Training Procedure

- A 3 layered neural network was implemented with hidden layer sizes of 32,64 and 32.
- The learning rate was set to the default value for all the 3 goals while the number of epochs was modified for each.
- In the left and right goals, the model was trained for 1000 epochs and the loss curves were logged in wandb.
- In the straight goal, the model was trained for 1500 epochs with a slightly different loss function (described below)
- Adam Optimizer was used for all the goals
- The default learning rate of 0.005 was used.
- The best results were obtained with Xavier Initialization and tanh activation function.
- The Average Epoch Loss Curves are shown below in Figure 1

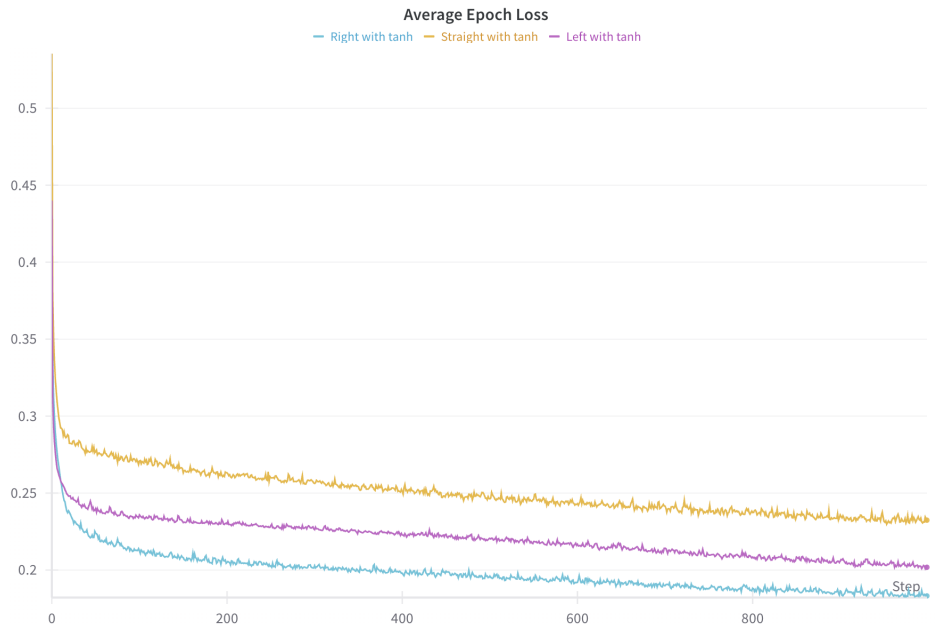

Figure 1: Loss Curves for the Three Goals with tanh

Part (iv)

Success Rates for the three goals:

- Left: 0.94
- Straight: 0.72
- Right: 0.86

Part (v)

The weighted MSE loss as decribed in Part (i) was used. The weights were identical ($w_1 = 0.5, w_2 = 0.5$) for left and right. For the straight case, $w_1 = 0.75,\ w_2 = 0.25$ were used. The mathematical expression is given below.

$$L = \sum_{i=1}^{N} (w_1 * (h_\theta(o^{(i)}, g)_1 - a_1^{(i)})^2 + w_2 * (h_\theta(o^{(i)}, g)_2 - a_2^{(i)})^2)$$

If the data is perfectly fit, the loss would be zero. Therefore, it is **bounded below with zero.**

Part (vi)

Neural Network with Distribution Parameters as the Output

- The neural network will output 6 values, the first two represent the mean $\mu_{steering}, \mu_{throttle}$ and the last 4 values are the entries of the covariance matrix.
- To ensure that the last 4 values belong to a positive definite matrix, an additional step was added within the neural network that takes the 4 values into a matrix A, finds the matrix AA^T, adds a term ϵI to make ensure it is positive definite (not just positive semi definite)

Part (vii)

Negative Mean Log-Likelihood Loss

- If the data is perfectly overfit, the probability densities would approach infinity and thus the loss (negative log likelihood) would **approach** $-\infty$.
- NLL **is unbounded below**

Part (viii)

Implemented in Code

Part (ix)

Training Procedure

- A very small neural network was used with hidden layer sizes 8 and 4.
- Xavier Initialisation was used with tanh activation function.
- As suggested, the model had to undergo multiple training iterations using the –restore flag and also reducing the learning rate.
- Sometimes, the starting value of the loss would be very high and the loss would oscillate around a very high value, these runs were discarded and the training was restarted.
- For the left and straight goal, the model was retrained 2 times with 1000 epochs each and lr = 0.005 and 0.002.
- For the right goal, the model was retrained 3 times with 1000 epochs each and lr = [0.005, 0.002,0.001]

Part (ix)

The success rates (Intersection):

- Left: 1.0
- Straight: 1.0
- Right: 1.0

Problem 3

Part (i)

As the ego vehicle approaches the intersection, it is unsure about the direction and ends up taking a random action and often crashes.

Part (ii)

Implemented in Code

Part (iii)

The agent responds to the keyboard input pretty accurately, sometimes, it is too late to respond and crashes but commits to the goal direction correctly most of the times. The success rates:

- Left: 0.87
- Straight: 0.90
- Right: 0.81

Part (iv)

Training Procedure

- Two shared hidden layers of sizes 32 and 16 were used and each of the branches included two layers of sizes 16 and 8.
- Xavier Initialisation with tanh activation gave the best results.
- The branch is conditionally chosen based on the goal.
- A learning rate of 0.002 was used and the model was retrained twice with 1000 epochs each.

Problem 4: Intent Inference and Shared Autonomy

Part (i)

The probability of the goal given the action and observation can be computed using the bayes theorem. Moreover, it is mentioned as uniform prior.

$$Pr(g \mid o, a) = \frac{f(a \mid o, g)P(g \mid o)}{f(a \mid o)}$$

Pr represents the probability while f represent the probability density. Since, the prior is uniform, we can write the following:

$$Pr(g \mid a, o) \propto f(a \mid o, g)$$

The parameters of the distribution are obtained from the neural network trained earlier. The probability density values can be found and normalized to find the intent probabilities.

Part (ii)

Implemented in Code

Part (iii)

The predictions are mostly right (10 out of 15). The reasons why it fails:

- Pressing left and right can make the intent ambigious
- Too little reaction time is causing the car to crash at times
- The highest accuracy and confidence was for straight when left/right keys weren't pressed

Part (iv)

Implemented in Code (The model had to be retrained around 6 times with slightly decreasing learning rates)

Part (v)

It makes sense to apply a threshold on the computed a_R because we want the robot to only provide a helping hand and not take over completely. If the values of a_R are very high, it might not be intuitive for the human to control the vehicle anymore.

Part (vi)

Implemented in Code

Part (vii

The effect of the robot action is clearly felt. It was interestingly making it worse. This is because I am not sure about the robot's intent and it becomes a game theory problem when I am trying to learn what the robot is trying to do and tend to compensate for that. The sense of independence and control is lost.