CS 237B: Principles of Robot Autonomy II
Problem Set X

Name:
SUID:

Problem 1

(i) Initializing all weights as zero may result in finding local optima, as opposed to using randomized weight selection. Conversely, the results may be more reproducible if using zero weights.

(ii) Xavier initialization attempts to keep the variance across the weights of each layer to be the same. The purpose is to prevent what is termed "exploding or vanishing gradients". Exploding gradients can be understood as when weights are initialized too large, the output of each layer increases exponentially. This leads to poor accuracy of the network, as the target oscillates around a minima (the cost oscillating around the minimum value). Similarly, when the weights are initialized to small, the network may converge too quickly, as the cost gets exponentially lower ("vanishing").

(iii) Adam is a a gradient descent optimization algorithm for stochastic objective function, thus it iteratively used a gradient-based step to find optimal values of parameters for minimizing an objective function. Based on the original paper, Adam achieves faster results in finding optimal parameters for linear/logistic regression, neural networks and convolutional neural networks, as compared to other optimization methods such as SGDNesterov (an accelerated form of stochastic gradient descent) and AdaGrad. The advantages of Adam include working on stochastic (non-stationary) objective and sparse gradients (by using an adaptive learning rate per-parameter). A potential disadvantage has been proposed of Adam being poor at generalizing, compared to SGD for example. [TODO add reference]

Problem 2

(i) Using the associative property of matrix multiplication (for rearranging the torque m

(ii) See code.

(iii) The training procedure for the left and right goals were the same. As directed in the code comments, each training step consisted of a forward pass followed by a backpropagation of the gradients of the weights. The neural network consisted of three hidden layers (1024, 512, 64 units, respectively) each with ReLU activation functions. Xavier initialization was used for each layer. The output layer was a densely connected layer matching the output size (2). The loss function used is discussed in the latter questions. By experimentation, it was found that 200 epochs achieved optimization of the training loss, where the loss oscillated around a minimum with any increase in the number of epochs. The learning rate used was the default for Adam optimizer: 0.001. For the straight goal, the same NN was used; the only change was to the number of epochs and the learning rate. For the straight goal, the network was trained on 150 epochs and a learning rate of 0.002. These were decided on through looking at minimization the training loss over each epoch, as mentioned previously.

(iv) The table below shows the success of the NNs on the test set:

Goal	Test Accuracy
left	0.99
right	0.60
straight	0.32

(v) The loss used was the Mean Squared Error (MSE). It was found through experimentation that this produced the best minimization, in comparison to the Euclidean norm for example. A weighting was chosen for the error, where a 4:1 ratio was used to penalize the steering error. This weighting had a considerable impact on the training loss optimization.

TODO: Is it bounded? TODO: Write out equation?

(vi) It is possible to ensure that the covariance matrix is PSD by allowing the network to learning a matrix, A, such that AA^T is the covariance. Thus the output of the network must be interpreted as A and the creation of the covariance matrix can be performed inside the loss function, ensuring it is PSD. Note, the matrix A could be represented as a column of 4 entries, which is then reshaped into a $2x2$ matrix. So in total the network would have 6 outputs, the 2 mean variables and 4 variables representing A.

(vii)

Problem 3

(i) The problem with this policy is that the agent may exploit a particular trajectory/goal that will maximize the reward but limit itself to that trajectory/goal only. For example, it was observed when training with the goal of "all", the agent continuously veered left in intersection. It performed the turn correctly (achieving an accuracy of 1.0 without visualization), however it did not ever attempt to continue to the straight or right goals, so as a user trying to guide the agent it was not possible to control the trajectory. Thus, it found a local optima to increase it's reward and exploited this to achieve a higher accuracy.

(ii) See code.

(iii) The performance of the tests with respect to left, straight and right goals is listed below. Note that when the '–visualize' argument was used, it was indeed possible to direct the agent to a particular goal based on keyboard input of a command.

After the first set of training with the above, the following was achieved:

Goal	Test Accuracy
left	0.55
right	0.41
straight	0.44

Running for another 50 epochs, restoring the same weights achieved:

Goal	Test Accuracy
left	0.94
right	0.52
straight	0.86

(iv) The network was composed of several modules which were all fully connected layers. The first module was for processing the observations. This consisted of two densely connected layers, with 24 and 8 units respectively. Three modules, all consisting of fully connected 8 units were then included for each command: left, straight and right. Based on the command input, the outputs from the observation module was passed into one of these three modules. These modules were all connected to a layer with

2 units, the output layer. In all cases, the the layers were initialized with Xavier initialization and used ReLU activation functions. The loss function used was the mean squared error (MSE), where the error was calculated between the true actions and the two actions output by the network. The learning rate used was 0.002 and the number of epochs were 100. The restore flag was not used in the training.