Deep Learning / Reinforcement Learning

Plotting and Visualization Handout

(adapted from CS 294-112 at UC Berkeley)

1. General Best Practices

Plotting and visualization are an important component of designing, debugging, prototyping, and evaluating your deep reinforcement learning algorithms. The following tips will help you structure your code in a way that makes it easy to produce good plots:

- Log results to an external file (e.g., CSV or PKL) rather than producing the final plot directly. This way, you can run the learning process once and then experiment with different ways to plot the results. Log more than you think is strictly necessary, since you never know what information will be most useful for understanding what happened. Keep an eye on file size, but generally log: average reward or loss at each iteration, sampled trajectories (for visualization), and useful secondary metrics (e.g., Bellman error, gradient magnitudes).
- Use a separate script to load logs and plot results. If you run the algorithm multiple times with different hyperparameters or seeds, or compare different algorithms, load all data together and plot on the same figure with an automatically generated legend and color scheme that distinguishes methods.
- Run multiple seeds: Deep RL methods, especially model-free ones, exhibit variability between runs. Initially, plot all runs together with the average performance highlighted (thicker line or different color). For many methods, summarize results into mean and standard deviation plots. However, distributions may not be normal, so plotting all runs initially gives better insight into variability.

2. Example Code

In Python, matplotlib and seaborn are useful for plotting. Here's an example for plotting with shaded regions to indicate standard deviation:

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Dummy function to generate arbitrary data
```

```
def get_data():
    base_cond = [[18, 20, 19, 18, 13, 4, 1],
                 [20, 17, 12, 9, 3, 0, 0],
                 [20, 20, 20, 12, 5, 3, 0]]
    cond1 = [[18, 19, 18, 19, 20, 15, 14],
             [19, 20, 18, 16, 20, 15, 9],
             [19, 20, 20, 20, 17, 10, 0],
             [20, 20, 20, 20, 7, 9, 1]]
    cond2 = [[20, 20, 20, 20, 19, 17, 4],
             [20, 20, 20, 20, 20, 19, 7],
             [19, 20, 20, 19, 19, 15, 2]]
    cond3 = [[20, 20, 20, 20, 19, 17, 12],
             [18, 20, 19, 18, 13, 4, 1],
             [20, 19, 18, 17, 13, 2, 0],
             [19, 18, 20, 20, 15, 6, 0]]
    return base_cond, cond1, cond2, cond3

# Load data
results = get_data()
fig = plt.figure()

# Plot iterations 0...6
xdata = np.array([0, 1, 2, 3, 4, 5, 6]) / 5.

# Plot each line
sns.tsplot(time=xdata, data=results[0], color='r', linestyle='-')
sns.tsplot(time=xdata, data=results[1], color='g', linestyle='--')
sns.tsplot(time=xdata, data=results[2], color='b', linestyle=':')
sns.tsplot(time=xdata, data=results[3], color='k', linestyle='-.')

plt.ylabel("Success Rate", fontsize=25)
plt.xlabel("Iteration Number", fontsize=25, labelpad=4)
plt.title("Awesome Robot Performance", fontsize=30)
plt.legend(loc='bottom left')
plt.show()
```