Scott Burns
Machine Learning Engineer - Project 4
Train a Smartcab to Drive

Introduction

This report outlines findings when implementing a simple reinforcement learning agent - a "smartcab", which navigates a grid of streets with other vehicles, attempting to reach a destination several blocks away.

Implement a Basic Driving Agent

*Observe what you see with the agent's behavior as it takes random actions. Does the **smartcab** eventually make it to the destination? Are there any other interesting observations to note?*

With actions assigned randomly to the smartcab agent, it moves around the grid in a haphazard way, accumulating many penalties for traffic violations. Only by luck, in some rare simulations does the cab actually make it to the destination.

Inform the Driving Agent

*What states have you identified that are appropriate for modeling the **smartcab** and environment? Why do you believe each of these states to be appropriate for this problem?*

After trying several different combinations for inputs to describe state for the learning agent, I decided to include the following elements:

- Traffic light value
- Car present ('oncoming')
- Waypoint

I chose to use the first two elements (traffic light value and 'oncoming'), as they are critical for avoiding traffic rule violations, and reinforce agent decisions that avoid these penalties. Including waypoint as an input provides feedback for the cab to help it 'understand' the route toward the assigned destination.

Initially I included a longer list of variables (including time, number of steps in specific direction and others), but discovered that the cab found it difficult to learn from a much wider set of states in the q-table, as it was less likely to encounter a new state that could be informed by similar previous state, with multiple actions and their corresponding rewards.

*How many states in total exist for the **smartcab** in this environment? Does this number seem reasonable given that the goal of Q-Learning is to learn and make informed decisions about each state? Why or why not?*

Scott Burns
Machine Learning Engineer - Project 4
Train a Smartcab to Drive

There are 384 potential states for the smartcab in an environment defined by my chosen inputs. This follows as 1) the product of two valid traffic light states, 2) four states each for: 'oncoming', 'left', 'right' and 3) three states for waypoint - or 2*(4^3)*3.

The inputs I've chosen seems to strike a reasonable balance between being sufficiently detailed to offer insight on the best potential action at a given state, while avoiding excessive specificity, that could hinder a cab from generalizing from previous experience.

Implement a Q-Learning Driving Agent

What changes do you notice in the agent's behavior when compared to the basic driving agent when random actions were always taken? Why is this behavior occurring?

After implementing Q-Learning, I noticed that my agent improved slightly in avoiding penalties for traffic violations and reaching the destination.

The improved navigation behavior occurs because the agent recognizes patterns in the rewards earned given actions for a particular state. As a result, when presented with states where there is a risk of incurring penalties for traffic violations, the agent may 'choose' an action that under similar state features may have avoided a penalty previously. For my early test runs, I may have also set the exploration rate too low, which could have caused the cab to continue selecting low reward actions from similar states without trying new, random choices to learn their rewards.

Improve the Q-Learning Driving Agent

Report the different values for the parameters tuned in your basic implementation of Q-Learning. For which set of parameters does the agent perform best? How well does the final driving agent perform?

To optimize my Q-Learning algorithm I added 4 key features:

1. Strict randomization in the case that the best known q-table rewards are below a constant threshold
2. Adjusting new information added to the q-table with a 'learning rate' (alpha) - a factor between 0 and 1, which is multiplied times the 'learned value' quantity for a given action
3. Addition of action randomization according to a constant 'exploration rate', like the epsilon proposed in later Reinforcement Learning Core Lessons
4. Addition of epsilon decay (per earlier reviewer suggestions), in which the exploration rate decreases over time. However, I quickly noticed that learning performance was extremely sensitive to small variations in the

decay factor, which can be seen in the charts of Appendix A, with a factor of 0.99 (or decay rate of 0.01 per time unit). After several trials with epsilon decay adjustments, I decided to leave the decay factor at 1

With these updates, for trial sets (or simulations) of 100 trials, my cab was able to fairly consistently achieve average trial scores of over 20.

To better understand how variations to parameters affected model performance, I added a 'tester' (smartcab/tester.py) file to generate multiple plots of level-log regression curves over time (each agent move). These curves were fit to each smartcab's scores paths over multiple trials, through which the agent retains, and learns from a common q-table. Each plot of trial sets showed performance for specific parameter values, faceted in a grid.

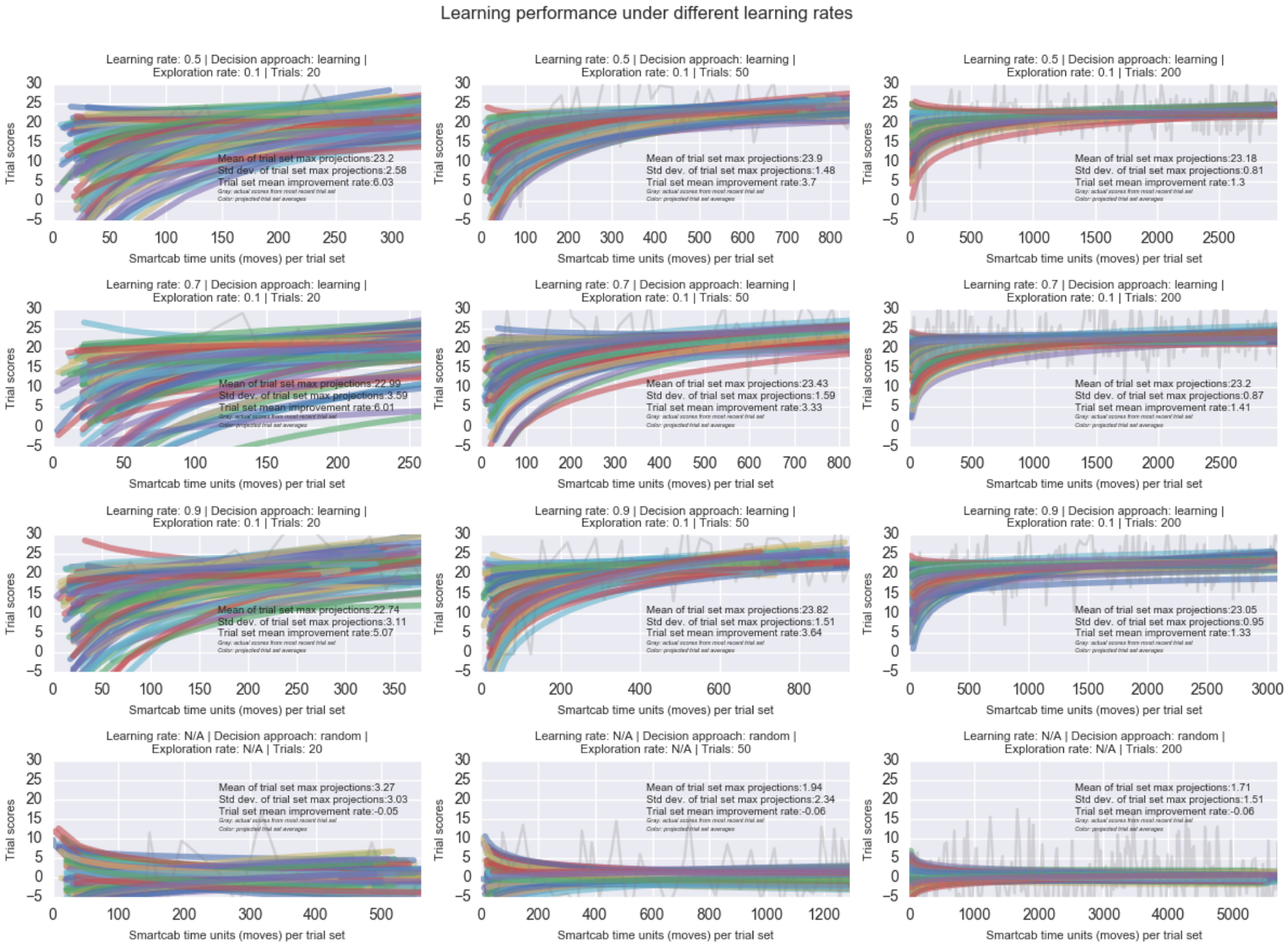
For each set of parameters, I generated three summary metrics to highlight how they affected smartcab navigation performance. These included the

- **Trial set mean improvement rate** – the coefficient of the fitted regression curve - *score = improvement rate * log(time) + constant* – an indicator of how rapidly the smart cab was 'learning' to navigate
- **Mean of trial set max projections** – the average maximum scores per trial reached over the fitted curves for multiple trial sets,
- **Standard deviation of trial set max projections** – the standard deviation of maximum scores per trial reached over the fitted curves for multiple trial sets

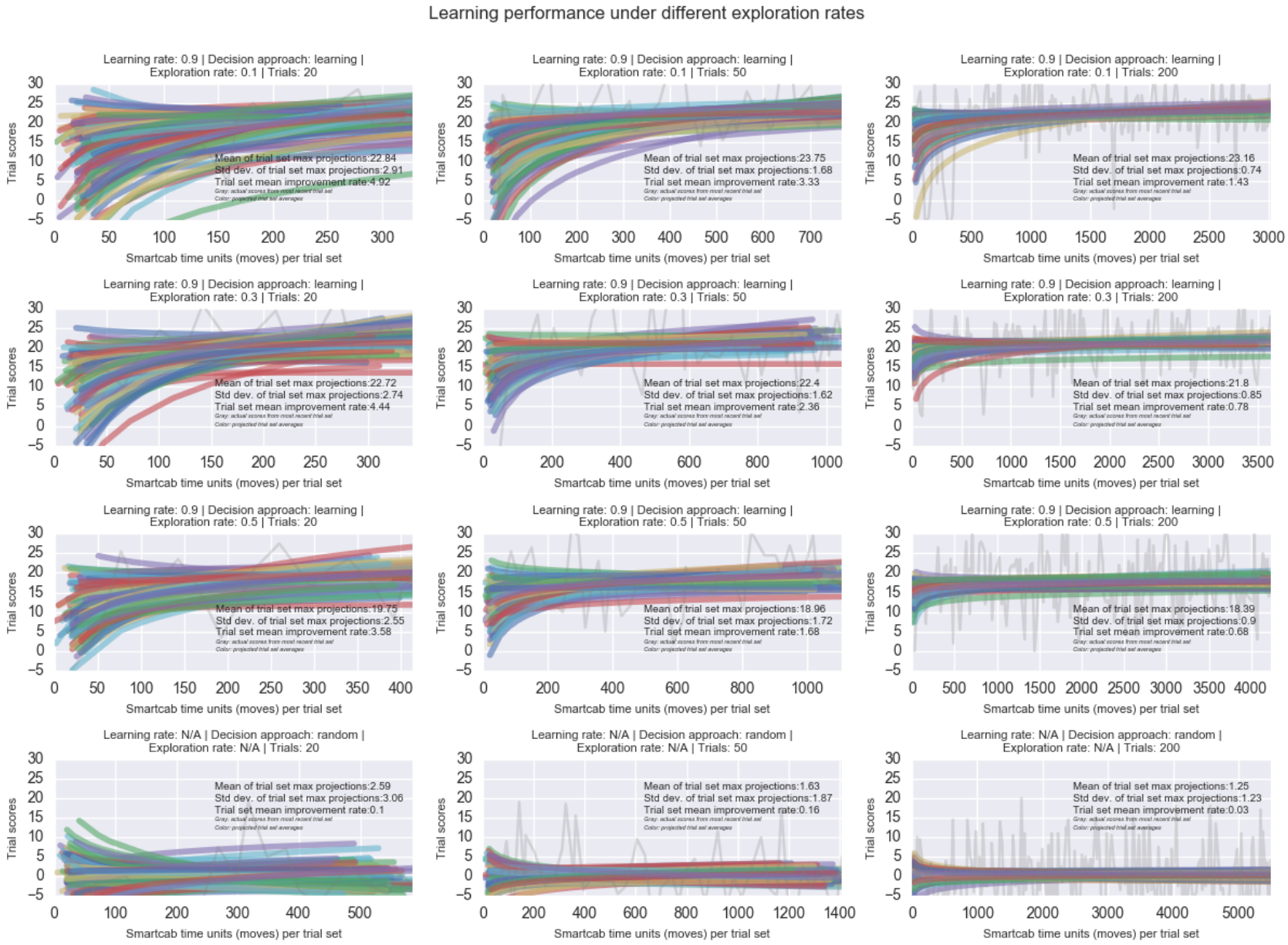
Two faceted charts below show the impact of learning rate and exploration rate on cab navigation performance, where the epsilon delay factor is 1 (a nonfactor). As noted above, found agent performance to be highly sensitive to small changes in this parameter, with even a value of 0.99 remarkably increasing the variance of our fitted learning curves. Example plots tested with an epsilon decay rate can be found in Appendix A.

The impacts are shown for three values of trials per trial set – 20, 50 and 200 – over 100 total trial sets in each plot.

Scott Burns
Machine Learning Engineer - Project 4
Train a Smartcab to Drive



Scott Burns
Machine Learning Engineer - Project 4
Train a Smartcab to Drive



Based on the charts, I see evidence that

1. A higher learning rate of 0.9 doesn't seem to help the smartcab substantially outperform its scores with lower learning rates of 0.7 and 0.5 in terms of maximum projected trial score value, score improvement rate or variance of projected trial set values
2. An agent with a lower exploration rate of around 0.1 seems to substantially outperform agents with higher exploration rates of 0.3 and 0.5, in terms of maximum projected trial store values, while also showing less variance in these values
3. Learning algorithms help agents outperform smartcabs choosing actions at random (as we would hope), and this outperformance becomes more obvious over trial sets with more trials, as the agent has more time to learn

Does your agent get close to finding an optimal policy, i.e. reach the destination in the minimum possible time, and not incur any penalties? How would you describe an optimal policy for this problem?

In the sample trial sets shown in the charts above, average agent trial scores over trial sets of 200, when using learning rates between 0.5 and 0.9, and exploration rates around 0.1, typically reach above 23. I suspect that this is close to, but not quite, optimal. The plotted background line graphs for the last trial set for given parameters suggest that individual trial score values can be greater than 23, so there may still be room for improvement. However, the maximum score also depends on the randomly generated initial distance to the destination, which makes it difficult to determine what consistently optimal scoring would look like.

An optimal policy would describe the correct action for every state possible given the smartcab's position on the grid relative to the destination (whether guided by waypoints or an overall grid view), the traffic light state at the smartcab's intersection, and the state of any other vehicles on the grid.

To further explore sources of suboptimal behavior from my agent, I logged outcomes in later trials (the last 5 of a trial set), for which I collected the current relevant action and q-table reward set, as well as all current state attributes. Based on existing q-table values, it seems that the cab was typically incurring penalties as a result of continuing random 'exploration' rather than in pursuit of a waypoint in violation of traffic rules. Most of the actions in logged suboptimal reward cases did not correspond to maximum available payoffs in the q-table, and were likely generated randomly.

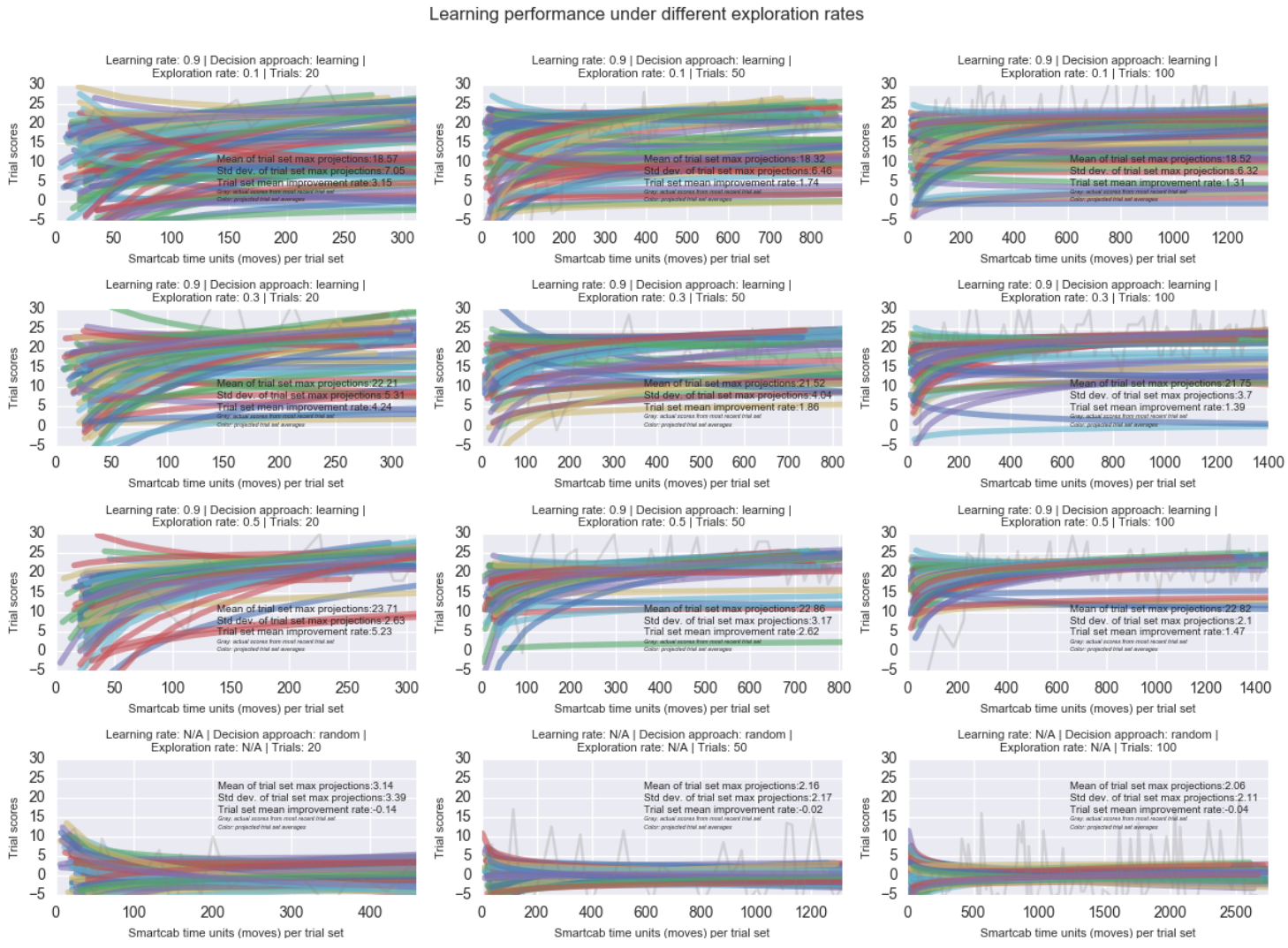
These errors late in a trial set further endorse the benefits of a decaying epsilon, which could prevent exploration where the q-table is already fairly well established.

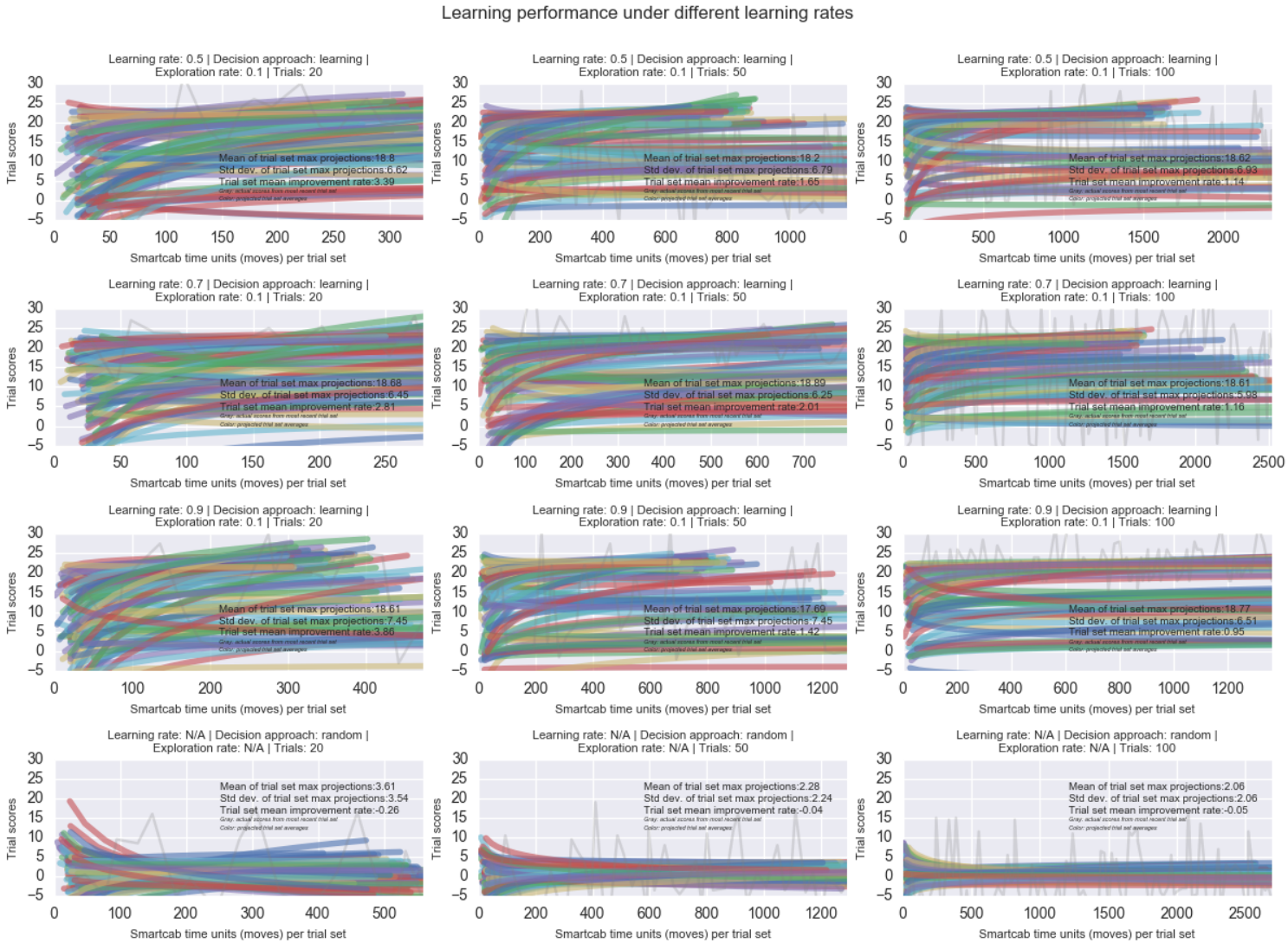
References

http://www.pysnap.com/reinforcement-learning-in-python/
http://stackoverflow.com/questions/8023306/get-key-by-value-in-dictionary
http://stackoverflow.com/questions/29885408/how-plot-ployfit-n-log-n-with-python
http://stackoverflow.com/questions/8248467/matplotlib-tight-layout-doesnt-take-into-account-figure-suptitle
http://matplotlib.org/users/text_props.html
http://stackoverflow.com/questions/14770735/changing-figure-size-with-subplots
https://classroom.udacity.com/nanodegrees/nd009/parts/0091345409/modules/540405889375460
http://mnemstudio.org/path-finding-q-learning-tutorial.htm

Scott Burns
Machine Learning Engineer - Project 4
Train a Smartcab to Drive

Appendix A: Learning Rate Faceted Charts with Epsilon Decay Factor of 0.99




Learning performance under different learning rates
Learning rate: 0.5 | Decision approach: learning | Exploration rate: 0.1 | Trials: 20
Mean of trial set max projections:18.8
Std dev. of trial set max projections:6.62
Trial set mean improvement rate:3.39
Learning rate: 0.5 | Decision approach: learning | Exploration rate: 0.1 | Trials: 50
Mean of trial set max projections:18.2
Std dev. of trial set max projections:6.79
Trial set mean improvement rate:1.65
Learning rate: 0.5 | Decision approach: learning | Exploration rate: 0.1 | Trials: 100
Mean of trial set max projections:18.62
Std dev. of trial set max projections:6.93
Trial set mean improvement rate:1.14
Learning rate: 0.7 | Decision approach: learning | Exploration rate: 0.1 | Trials: 20
Mean of trial set max projections:18.68
Std dev. of trial set max projections:6.45
Trial set mean improvement rate:2.81
Learning rate: 0.7 | Decision approach: learning | Exploration rate: 0.1 | Trials: 50
Mean of trial set max projections:18.89
Std dev. of trial set max projections:6.25
Trial set mean improvement rate:2.01
Learning rate: 0.7 | Decision approach: learning | Exploration rate: 0.1 | Trials: 100
Mean of trial set max projections:18.61
Std dev. of trial set max projections:5.98
Trial set mean improvement rate:1.16
Learning rate: 0.9 | Decision approach: learning | Exploration rate: 0.1 | Trials: 20
Mean of trial set max projections:18.61
Std dev. of trial set max projections:7.45
Trial set mean improvement rate:3.86
Learning rate: 0.9 | Decision approach: learning | Exploration rate: 0.1 | Trials: 50
Mean of trial set max projections:17.69
Std dev. of trial set max projections:7.45
Trial set mean improvement rate:1.42
Learning rate: 0.9 | Decision approach: learning | Exploration rate: 0.1 | Trials: 100
Mean of trial set max projections:18.77
Std dev. of trial set max projections:6.51
Trial set mean improvement rate:0.95
Learning rate: N/A | Decision approach: random | Exploration rate: N/A | Trials: 20
Mean of trial set max projections:3.61
Std dev. of trial set max projections:3.54
Trial set mean improvement rate:-0.26
Learning rate: N/A | Decision approach: random | Exploration rate: N/A | Trials: 50
Mean of trial set max projections:2.28
Std dev. of trial set max projections:2.24
Trial set mean improvement rate:-0.04
Learning rate: N/A | Decision approach: random | Exploration rate: N/A | Trials: 100
Mean of trial set max projections:2.06
Std dev. of trial set max projections:2.06
Trial set mean improvement rate:-0.05
Trial scores
Smartcab time units (moves) per trial set