HW1

Brody Kendall

September 2022

Problem 1

a

$H(P) = -\Sigma_{x,y}[P(x,y)\log P(x,y)]$
$= -\Sigma_{x,y}[P(x,y)\log(P(x)P(y|x))]$
$= -\Sigma_{x,y}[P(x,y)\log P(x) + P(x,y)\log P(y|x)]$
$= -\Sigma_{x,y}[P(x,y)\log P(x)] - \Sigma_{x,y}[P(x,y)P(y|x)]$
$= -\Sigma_{x}[P(x)\log P(x)] + H(y|x)$
$= H(x) + H(y|x)$

b

The total entropy of a bivariate distribution (x, y) can be split into the sum of the entropy of one variable (x) and the conditional entropy of the other variable given the first $(y|x)$.

c

By Jensen's Inequality using the concavity of log,
$H(X,Y) - H(X) - H(Y)$
$= -\Sigma_{x,y}[P(x,y)\log P(x,y)] + \Sigma_{x}[P(x)\log P(x)] + \Sigma_{y}[P(y)\log P(y)]$
$= \Sigma_{x,y}[P(x,y)(-\log P(x,y) + \log P(x) + \log P(y))]$
$= \Sigma_{x,y}[P(x,y)\log \frac{P(x)P(y)}{P(x,y)}] \leq \log \Sigma_{x,y}[P(x,y)\frac{P(x)P(y)}{P(x,y)}] \leq \log 1 = 0$

Therefore, $H(X,Y) \leq H(X) + H(Y)$, so $H(X|Y) = H(X,Y) - H(Y)$
$\leq H(X) + H(Y) - H(Y) \leq H(X)$

Generally, this means that conditioning (in this case, on Y) will only reduce or maintain the entropy (in this case, of X).

d

Note that $H(f(y)|y) = -\Sigma_y P(y)\log P(f(y)|y) = -\Sigma_y 0 = 0$, so $H(y)$
$= H(y) + H(f(y)|y) = H(y, f(y)) = H(y,z) = H(z) + H(y|z)$. Now, since $H(y|z) \geq 0$, $H(z) \leq H(y)$

Problem 2

a

$Q(bird)\log(\frac{Q(bird)}{P(bird)})+Q(cat)\log(\frac{Q(cat)}{P(cat)})+Q(dog)\log(\frac{Q(dog)}{P(dog)})+Q(capybara)\log(\frac{Q(capybara)}{P(capybara)})$
$= 1\log(\frac{1}{0.4})+0+0+0$
≈ 0.916

b

$Q(bird)\log(\frac{Q(bird)}{P(bird)})+Q(cat)\log(\frac{Q(cat)}{P(cat)})+Q(dog)\log(\frac{Q(dog)}{P(dog)})+Q(capybara)\log(\frac{Q(capybara)}{P(capybara)})$
$= 1\log(\frac{1}{0.95})+0+0+0$
≈ 0.051

This is significantly smaller than the loss from part (a), which makes sense since the output probabilities are closer to the ground truth.

c

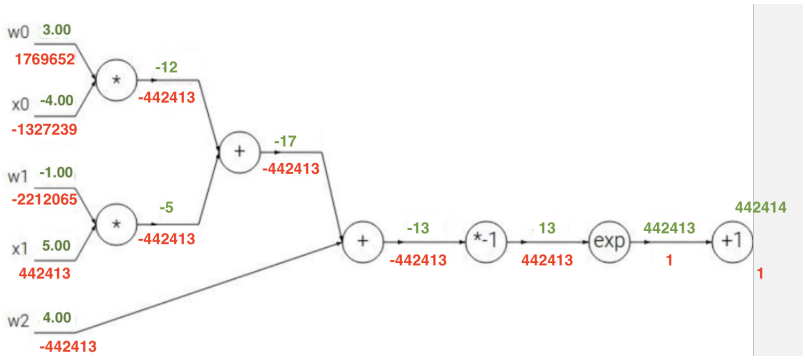
$Q(bird)\log(\frac{Q(bird)}{P(bird)})+Q(cat)\log(\frac{Q(cat)}{P(cat)})+Q(dog)\log(\frac{Q(dog)}{P(dog)})+Q(capybara)\log(\frac{Q(capybara)}{P(capybara)})$
$= 0.5\log(\frac{0.5}{0.4})+0.375\log(\frac{0.375}{0.2})+0+0.125\log(\frac{0.125}{0.3})$
$\approx 0.112+0.236-0.109$
$= 0.239$

d

$\min_\theta D_{KL}(Q, P_\theta)$
$= \min_\theta \Sigma_s[\pi_E(s)\log\pi_E(s)-\pi_E(s)\log\pi_\theta(s)]$
$= \Sigma_s\pi_E(s)\log\pi_E(s)+\min_\theta[-\Sigma_s\pi_E(s)\log\pi_\theta(s)]$
$= \Sigma_s\pi_E(s)\log\pi_E(s)+\min_\theta[H(\pi_E,\pi_\theta)]$

Since $\Sigma_s\pi_E(s)\log\pi_E(s)$ does not depend on θ, this proves that minimizing the behavioral cloning loss is equivalent to minimizing the KL divergence.

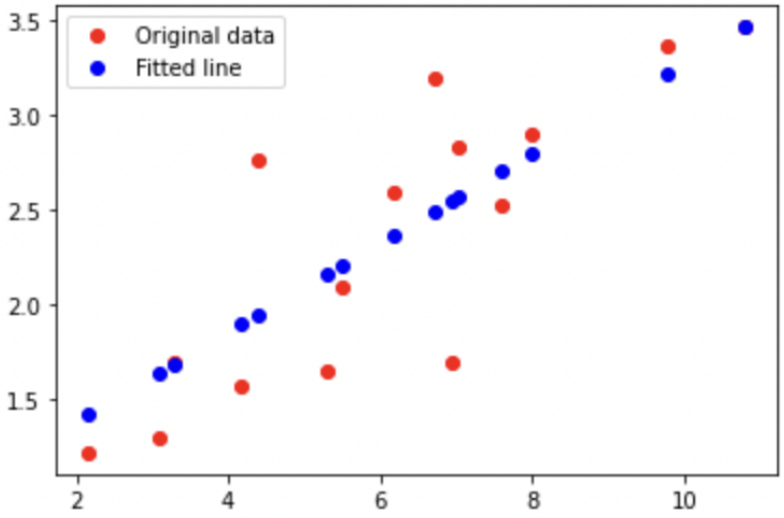
Problem 3



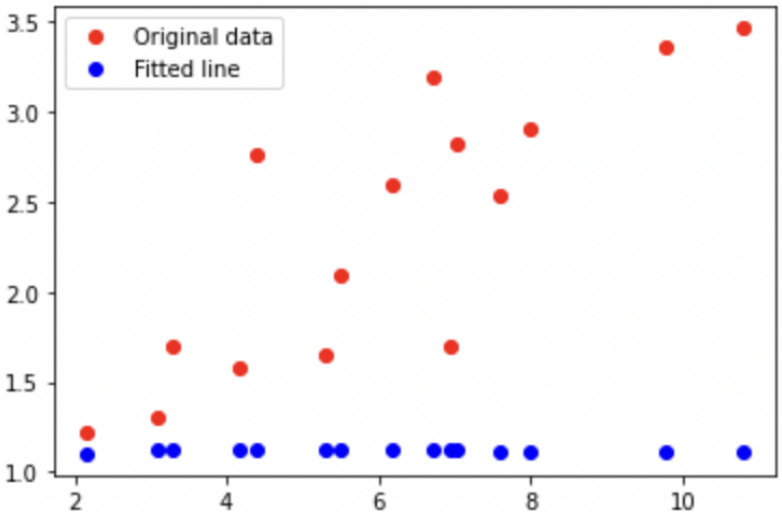
Problem 4

1

The training loss of the neural network model is 0.1724, which is smaller than that of the linear model (0.3219).

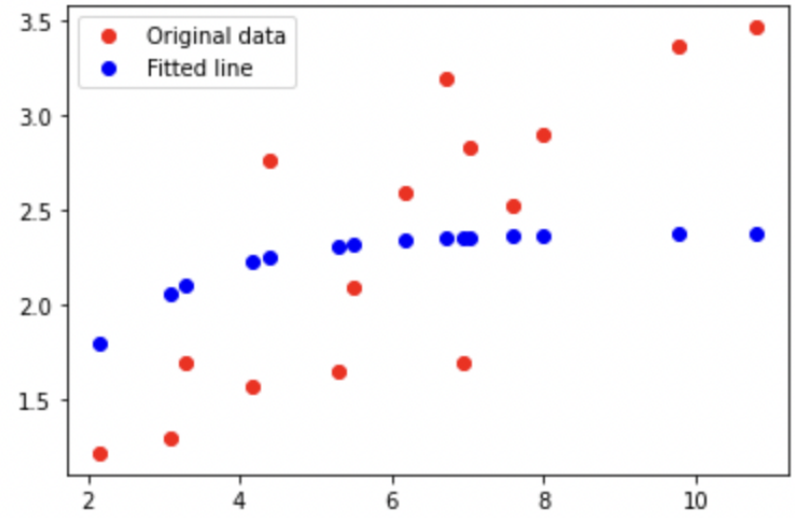


2



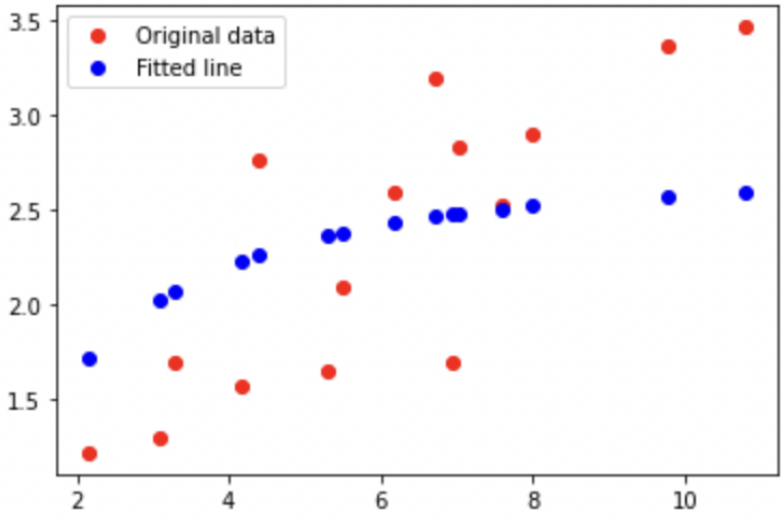
It is clear that this fitted line is worse than the one that used the RelU activation function. This is likely due to the fact that the range of Tanh is limited to -1 to 1, while the range of RelU is 0 to infinity.

3



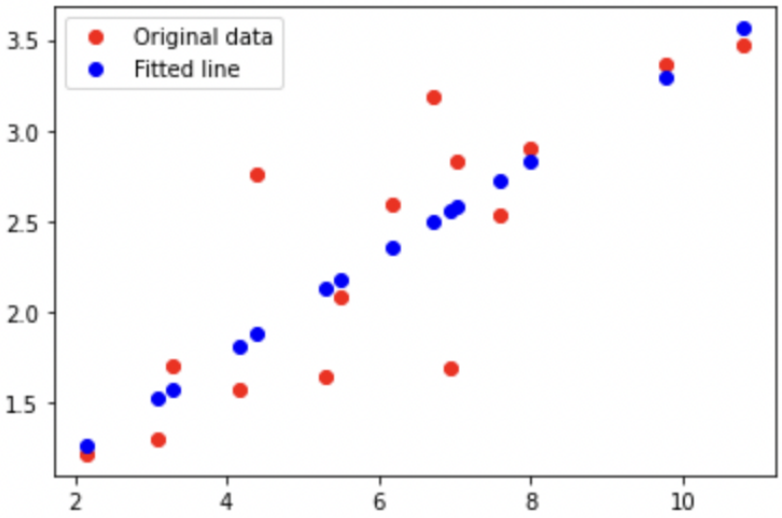
As seen in the plot, increasing the number of hidden units clearly improves the results, achieving a loss of 0.4109.

4



Changing the learning rate again improves the results, achieving a loss of 0.3362.

5

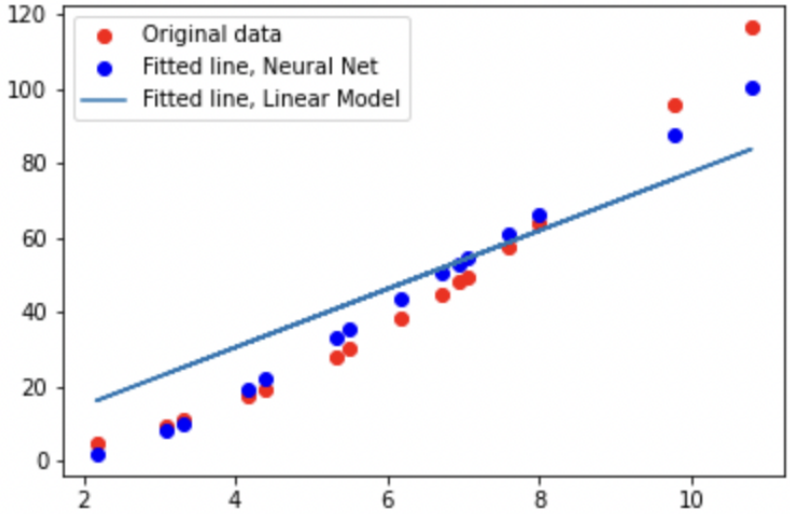


Adding momentum to the optimizer clearly improves the results, achieving a loss of 0.1688. It also converges more quickly, taking around 50-60 iterations to converge compared to more than 200 for normal SGD without momentum.

6

Over time, the loss converges to around 0.168. One option would be to decide to stop training when the loss no longer decreases by some threshold increment.

7



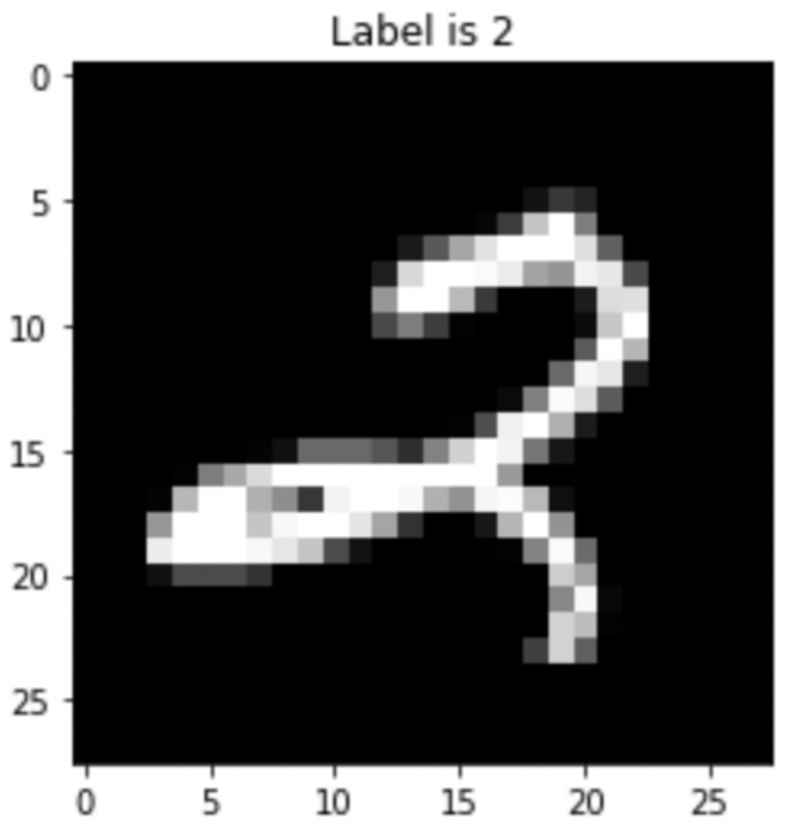
The neural net's fitted line is clearly better than that of the linear model.

8

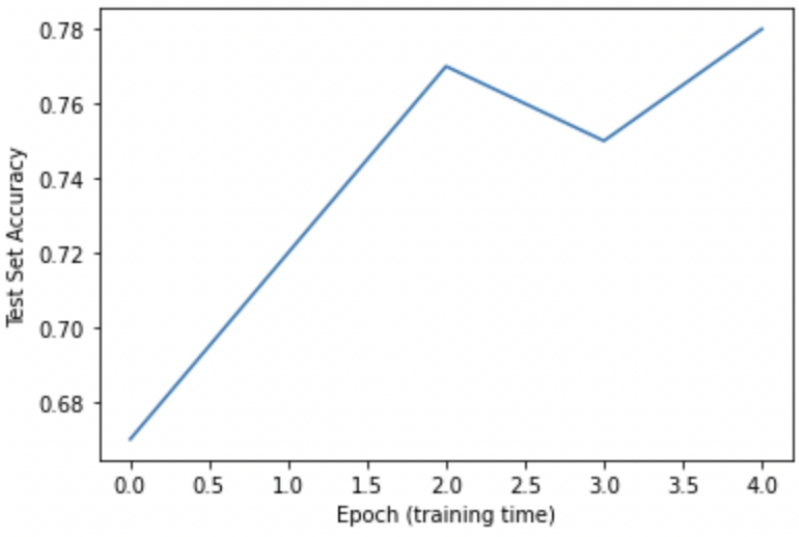
The smallest loss that I was able to achieve was 0.0043 with the following hyperparameters: learning rate = 0.001, number of epochs = 6000, momentum = 0.9 hidden size = 64, number of neural net layers = 3, optimizer choice = SGD.

Problem 5

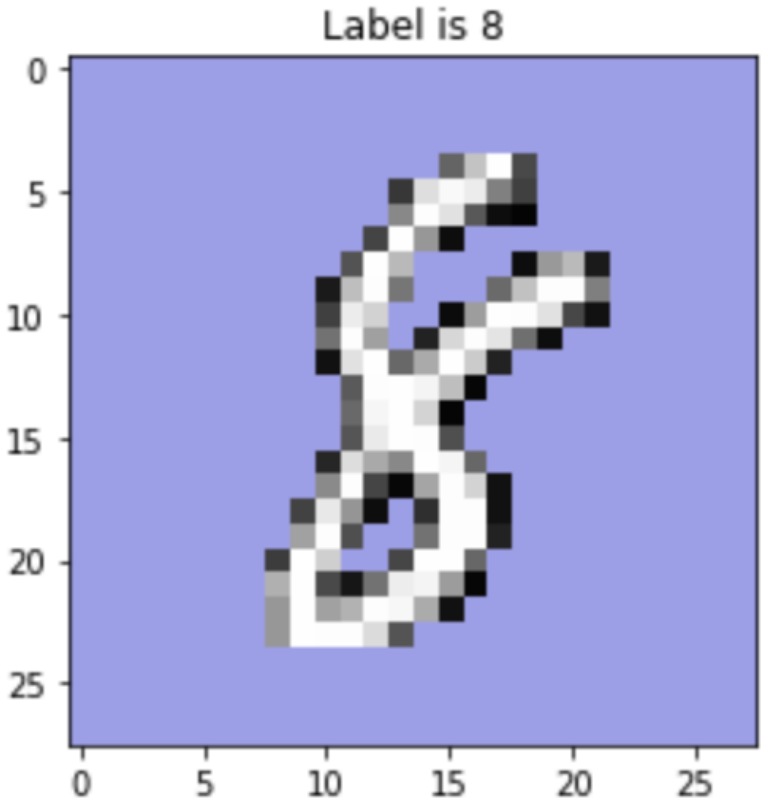
1

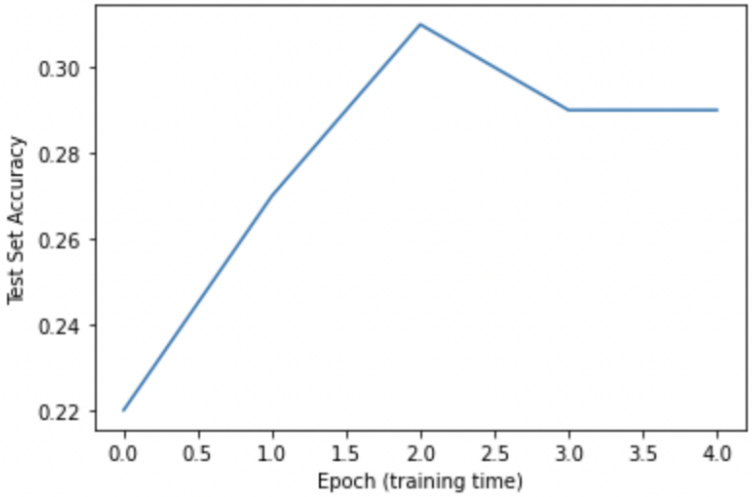


3

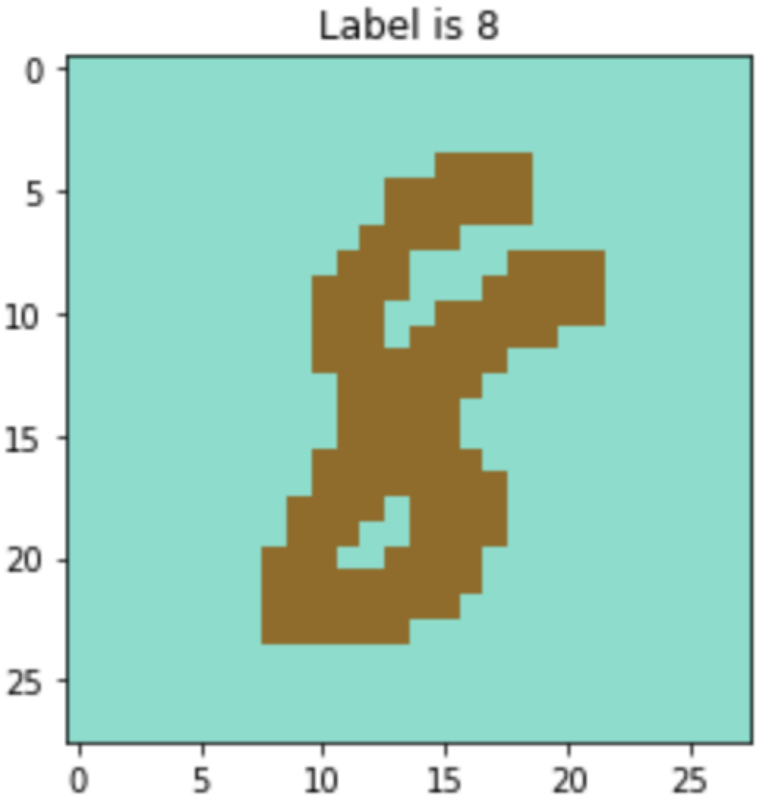

0.78
0.76
0.74
0.72
0.70
0.68
Test Set Accuracy
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Epoch (training time)

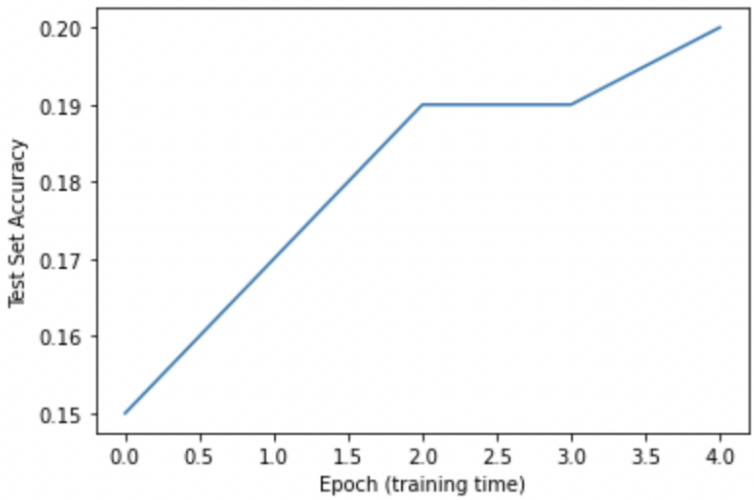
4


Label is 8
0
5
10
15
20
25
0
5
10
15
20
25


Test Set Accuracy
0.30
0.28
0.26
0.24
0.22
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Epoch (training time)

5

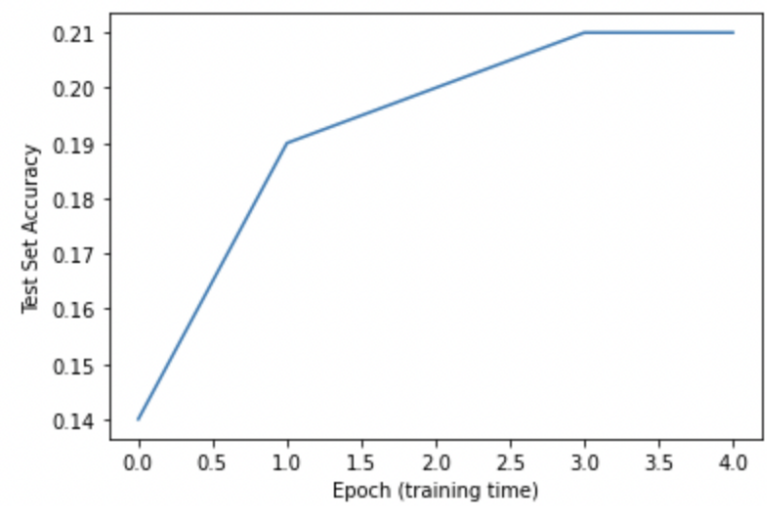




The convergence rate stays relatively constant across all three, but testing

accuracy decreases from vanilla MNIST to MNIST with modified backgrounds, to MNIST with modified digits and backgrounds, as expected.

6



There seems to be only slight improvement in convergence rate and testing accuracy.

Problem 6

2

Sin policy: Mean = 38.84, Variance = 232.14
Random policy: Mean = 22.76, Variance = 119.17

3

With a horizon of 500, my policy achieved a mean of 500 with a variance of 0.

Problem 7

1

```
A = [[0,1,0,0],[0,0,-1*(m*g)/M, 0],[0,0,0,1],[0,0,((M+m)*g)/(l*M),0]]
A = np.array(A)
B = [[0],[1/M],[0],[-1/(l*M)]]
B = np.array(B)
```

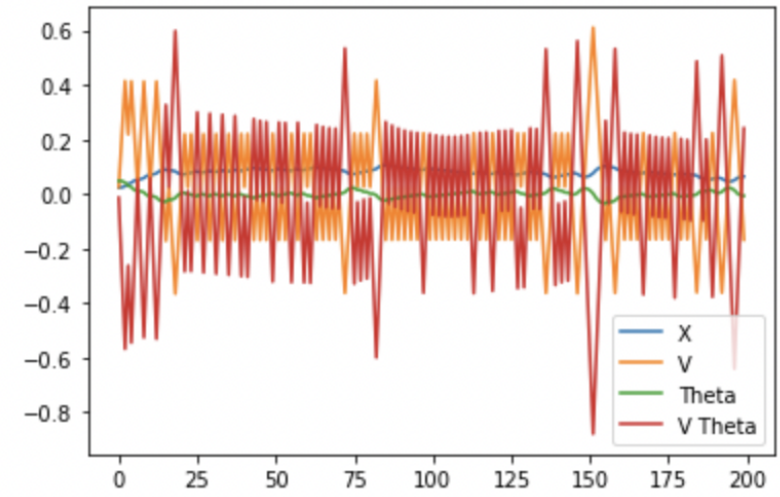
2

If the continuous action is non-negative, set the discrete action to 1. Otherwise, set the discrete action to 0.

3

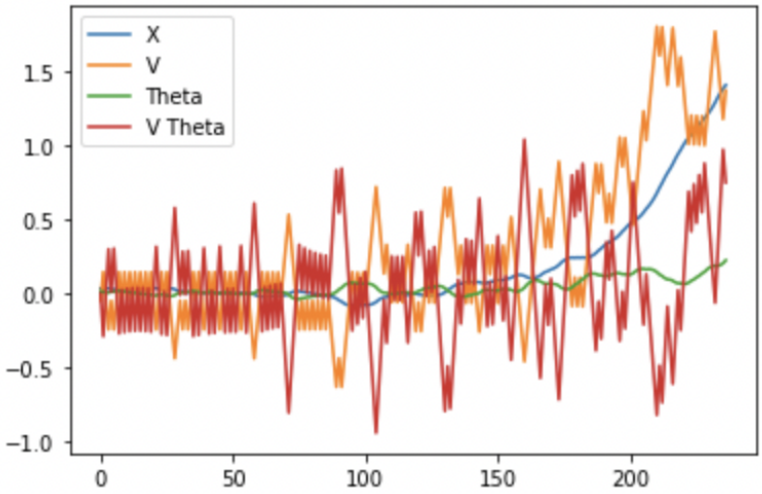
I believe this can be solved for an arbitrary number of steps - I achieved 10,000.

4



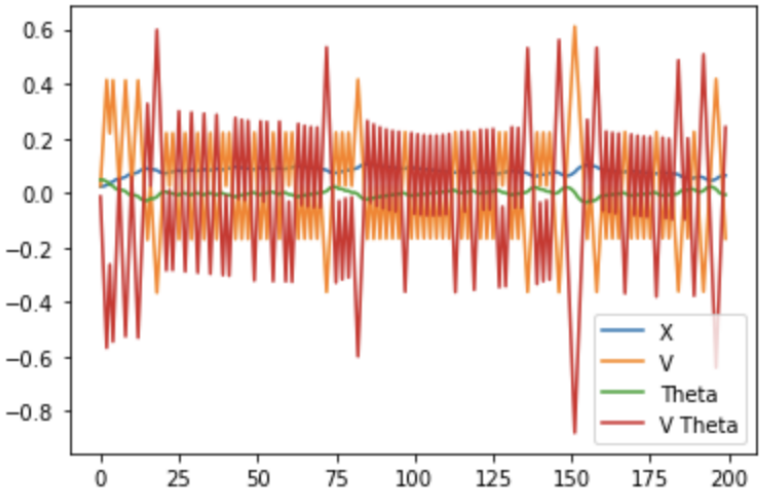
5

LQR still does reasonably well for a while, but fails eventually (mine fell at 236 steps).

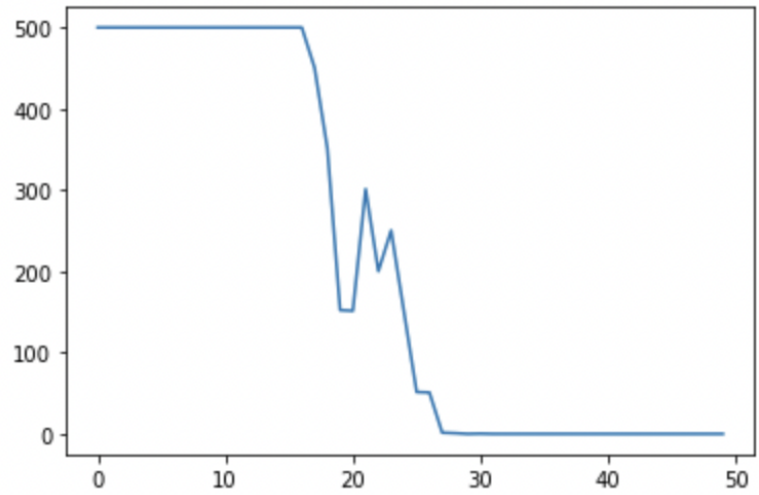


6

LQR was still able to solve the problem and reach the horizon.

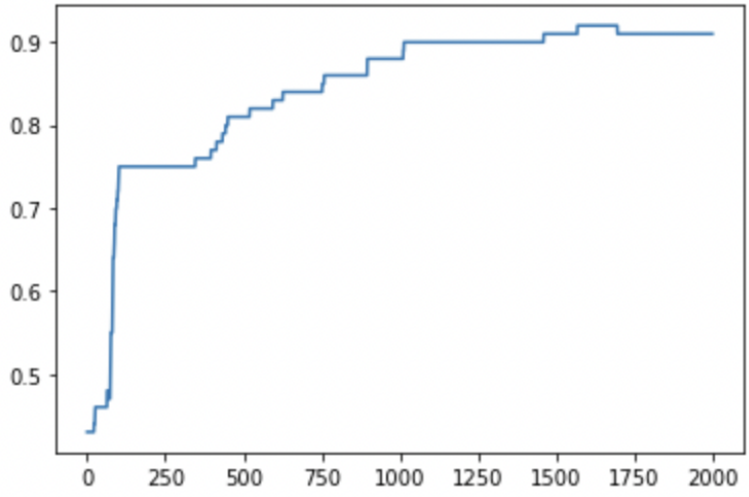


7

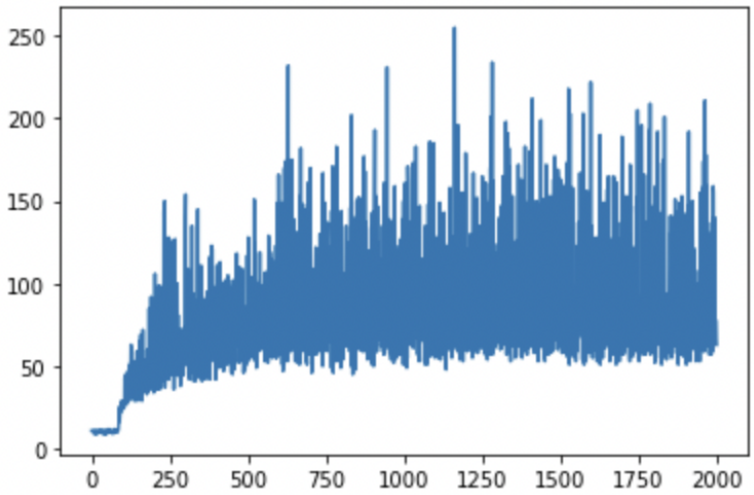

500
400
300
200
100
0
0
10
20
30
40
50

Problem 8

2

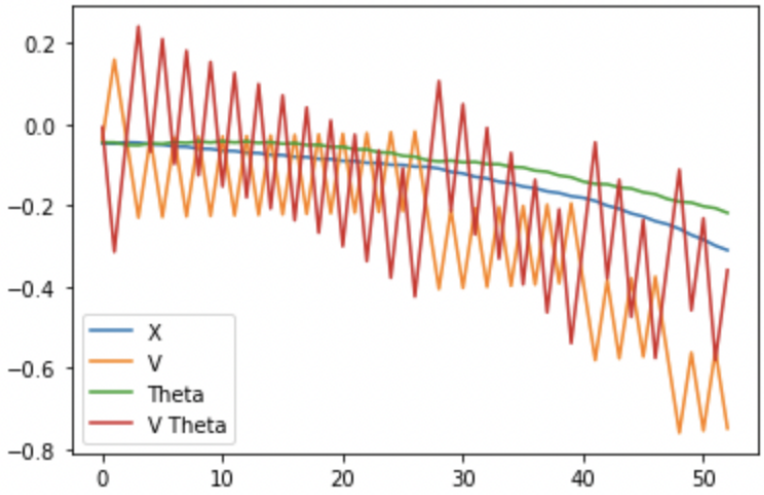

0.9
0.8
0.7
0.6
0.5
0
250
500
750
1000
1250
1500
1750
2000

3



At convergence, the imitation policy can last on average 97.295 steps.

4



5