Project 2 Reports

Yufei Zhang (1373240)

Part 1

Q1:

Each tile consists of $11 \times 11 = 121$ tiles starting from 0 and keep increasing, which includes 121 tiles in total.

The second tiling also contains 121 tiles; therefore, all of the tiles from all of the tilings are stored in one array of tiles from tiling 2 start at 120 to $120 + 121 = 241$.

Q2:

Each subsequent tiling is shifted in the x and y directions by 1 / number of tiles from the input space; for tiling N where N is in [0,7], (x, y) = (x + (N * 0.6)/8, y + (N * 0.6)/8).

If in1 = 0.1 and in2 = 0.1, the translated tiling coordinates are in range of (0,1) for tilings from 0 to 6. And because of that 0.1 + (6 * 0.6)/8 = 0.1 + 0.45 = 0.55 < 0.6 so the value is still in the first tile of the tiling. Therefore the input point (0.1, 0.1) should be in the first tile of each tiling from tiling 0 to 6. The first 7 tilings are 0, 121, 242, 363, 484, 605, and 726.

Q3 & Q4:

If in1 = 0.1 and in2 = 0.1, the coordinates in the 8th tiling are 0.1 + (7*0.6)/8 = 0.1 + 0.55 = 0.65. 0.65 is bigger than 0.6, the point is in the second tile of the second row of tiles in the 8th tiling. Therefore, the index of the first tile of this tiling is $121 * 7 + 13 - 1 = 859$.

Q5:

Science the 8th tiling with all 121 tiles will be mapped and we set the start to be 0. Then the maximum number of index should be $121 * 8 - 1 = 967$.

Q6*:*

The second and fourth example prints should have similar values because the input points for the two examples, (4.0, 2.0) and (4.0, 2.1), are close to each other. They have the same result for all tilings except the 5th and 6th tilings as is shown below. The point (4.0, 2.0) is 0.1 different from (4.0, 2.1). The point (4.0, 2.1) shifted over by 0.6/8=0.075 each tiling, which is smaller that 0.1. Therefore, point (4.0, 2.1) is classified in a new tile before (4.0, 2.0) and they remain in separate tiles until the point (4.0, 2.0) reclassifies into the same tile as the point (4.0, 2.1).

Output of Tilecoder.py:

```
yufei2@ud25:~/p2>python3 SuperLearn.py
Tile indices for input ( 0.1 , 0.1 ) are :    [0, 121, 242, 363, 484, 605, 726, 859]
Tile indices for input ( 4.0 , 2.0 ) are :    [39, 160, 281, 403, 524, 645, 777, 898]
Tile indices for input ( 5.99 , 5.99 ) are :    [108, 241, 362, 483, 604, 725, 846, 967]
Tile indices for input ( 4.0 , 2.1 ) are :    [39, 160, 281, 403, 535, 656, 777, 898]
```

Part 2

Output for 3:

yufei2@ud25:~/p2>python3 SuperLearn.py

Tile indices for input (0.1 , 0.1) are : [0, 121, 242, 363, 484, 605, 726, 859]

Tile indices for input (4.0 , 2.0) are : [39, 160, 281, 403, 524, 645, 777, 898]

Tile indices for input (5.99 , 5.99) are : [108, 241, 362, 483, 604, 725, 846, 967]

Tile indices for input (4.0 , 2.1) are : [39, 160, 281, 403, 535, 656, 777, 898]

Example (0.1 , 0.1 , 2.0): f before learning: 0.0 f after learning : 0.1914653659980762

Example (4.0 , 2.0 , -1.0): f before learning: 0.0 f after learning : -0.0957326829990381

Example (5.99 , 5.99 , 3.0): f before learning: 0.0 f after learning : 0.2871980489971143

Example (4.0 , 2.1 , -1.0): f before learning: -0.07210802296867311 f after learning : -0.16093761146316388

The forth points have some same tiles and the former value has been updated before the later value comes out; hence, the former value of the forth point is non-zero.

Output for 4:

```
yufei2@ud25:~/p2>python3 SuperLearn.py
Tile indices for input ( 0.1 , 0.1 ) are :    [0, 121, 242, 363, 484, 605, 726, 859]
Tile indices for input ( 4.0 , 2.0 ) are :    [39, 160, 281, 403, 524, 645, 777, 898]
Tile indices for input ( 5.99 , 5.99 ) are :    [108, 241, 362, 483, 604, 725, 846, 967]
Tile indices for input ( 4.0 , 2.1 ) are :    [39, 160, 281, 403, 535, 656, 777, 898]
The estimated MSE:    0.252751596629
The estimated MSE:    0.0689670007584
The estimated MSE:    0.0251791254506
The estimated MSE:    0.0152793886151
The estimated MSE:    0.0123254280905
The estimated MSE:    0.0119314248616
The estimated MSE:    0.0115319946975
The estimated MSE:    0.0116074273998
The estimated MSE:    0.0111939145072
The estimated MSE:    0.0112239870615
The estimated MSE:    0.0111107895115
```

The estimated MSE does not decrease further towards zero but converge around 0.01 because the generating the target includes a normally distributed random number term with a standard deviation of 0.1. It means: the weights are updated to better approximate the target as we learn. However, we can never predict the value produced by the random term in the target. We will always have an error of approximately 0.01 no matter how well the approximation we get. Furthermore, the value converges to 0.01 is because we consider the mean squared error is $(0.1)^2 = 0.01$.

Step 6

The output for learning after 20 examples does not similar to the target function, because as the output is not sufficient and the relatively small sample count of 20 does not fully sample the input space. Therefore, it hardly get an accurately enough approximation and we do not expect to get right approximation through only 20 examples.

In addition, weights for most locations in sample space were not updated and remain 0 because there were no samples in those locations of the input space. However, we want fewer peaks and valleys but also want the right result and there are about 5 peaks and 6 valleys in total. The height of these should be approximately 10% of the target of the size of the smallest tile and overlap of the first and last tiles. The width would change every time because the first 20 examples involves a lot random explorations.

If the input space were 11x21 with 20 samples, most peaks and valleys would still be created from one sample. Hence, the width of the peaks and valleys would not change and still close to 1 tile. However, because of 21 tiles in the in1 direction, the peaks would be thinner in the in1 dimension. If we continued, the plot would approach the target plot that using Wolfram Alpha to get. The change in tiling dimensions would not change the final shape of the learned function after a great number of examples as long as the target is still the same because the learned function would approximate the target no meter how the starting is.