Report of Supervised Learning with Tile Coding

Lingbo Tang

1353070

Introduction:

The objective of this project4 is to implement 2D tile coding and use it to construct the binary-feature representation for a simple case of supervised learning of a real-valued function. Our task will be to write the Python function f(in1,in2) which returns a guess at the corresponding input.

Part1

Q1:

Each tile consists of 11*11 = 121 tiles, it means that the first tiling stops at index 120, and the second tiling will continue the index until the edge, which the index number will be 121 + 121-1 = 241 index.

Q2:

The size of each tile is 0.6 * 0.6 and the step size is 0.6/8 on each direction. This will cause all the points lie in the region of [0, 0.15] in both in1 and in2 direction. They will fall to the first tile for the first seven tilings and then move on to the next tile. That's why we always got them in the multiplication of 121.

Q3:

All points lies in [0.075, 0.675] (the first and the eighth) in both x and y directions should be in the 13^{th} in the 8^{th} tiling. Therefore, the index number is 121*7+13-1 = 859.

Q4:

Since the 8^{th} tiling with all 121 tiles will be mapped and we set the start to be 0, then maximum index is equal to 121*8-1= 967

Q5:

Because they are very close to each other. The actual distance between them is smaller than the size of the tile.

Part 2:

3.

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 : 0.19999999999999998

Example (4.0 , 2.0 , -1.0): f before learning: 0.0 f after learning : -0.09999999999999999

Example (5.99 , 5.99 , 3.0): f before learning: 0.0 f after learning : 0.30000000000000004

Example (4.0 , 2.1 , -1.0): f before learning: -0.072108022 f after learning : -0.16093761146316

The forth point has a lot of the tiles in common with the second point. The function value learned has been updated before the forth point learning value came out. Therefore, the "f before learning" is not zero.

4.

The estimated MSE: 0.247438771649

The estimated MSE: 0.0596576640336

The estimated MSE: 0.0220131258502

The estimated MSE: 0.0140155160088

The estimated MSE: 0.0125174140258

The estimated MSE: 0.012052096727

The estimated MSE: 0.0116132149878

The estimated MSE: 0.0113038524762

The estimated MSE: 0.0112575583097

The estimated MSE: 0.0110595329776

The estimated MSE: 0.0110947570837

There is some fluctuation caused by the normal function. Therefore the MSE will not decrease to 0, it will be a value between 0 to 0.1.

The function after 20 trains does not look similar to the target function because it's not sufficient and accurate enough to make an accurate approximation.

The final function is to approximate the samples we have observed. There are approximately 4 peaks and 5 valleys in total. The height of these should be approximately 10% of the target height. But the thing is

the approximation is not consistent. The width also got difference in the range of the size of the unit tile and the unit overlap of the first and last tilings because only 20 trains still involves too much uncertainty.

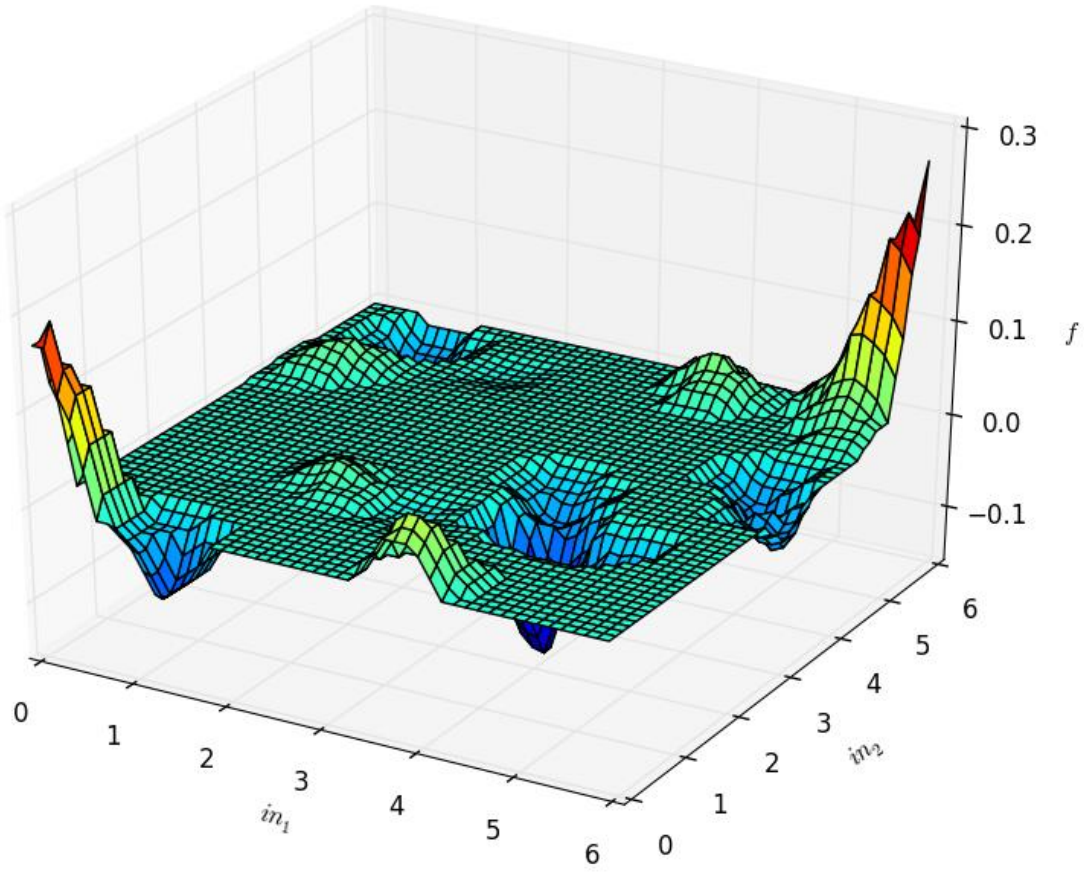

Figure f20

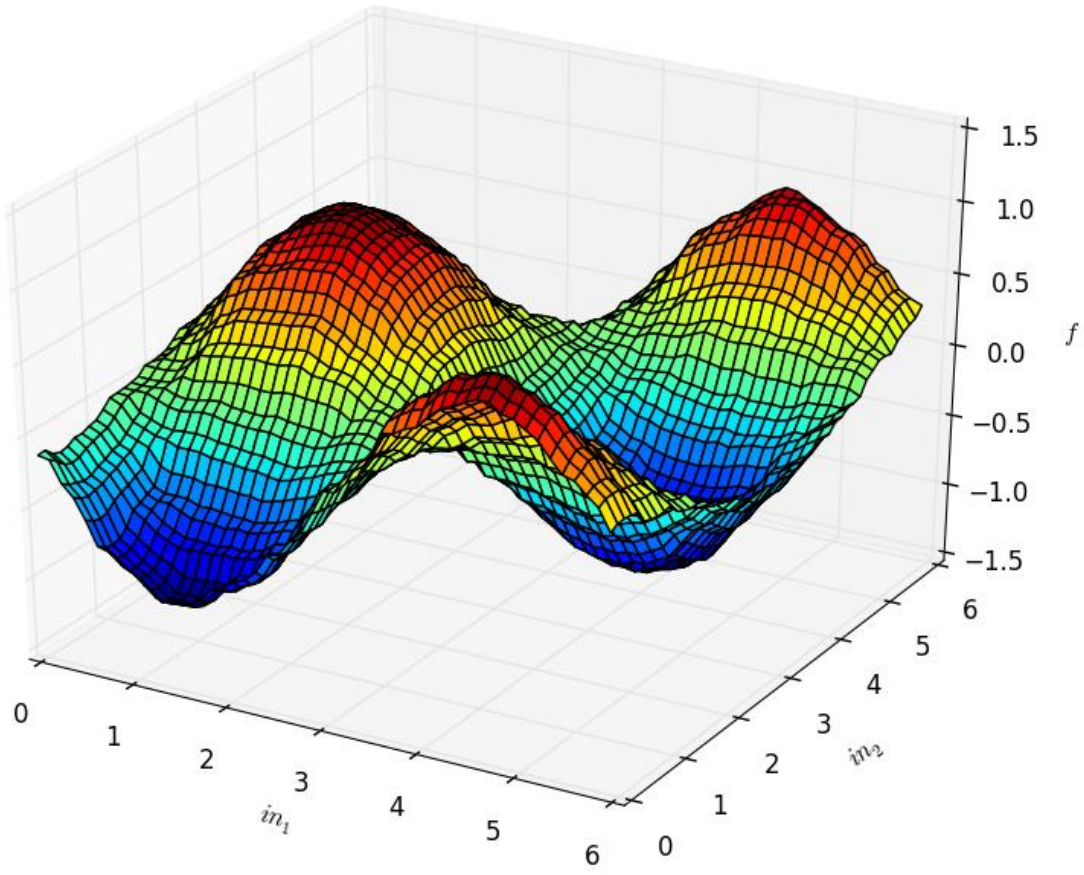

Figure f10000

If the new grid was used, then the scale of the peaks and valleys in the in1 direction will be half of the scale in the in2 direction. It will still be similar to the one that we get in our results but with extended scale towards the in2 direction. But since we learns more properties through more tiles, the MSE will be smaller than 11*11 grid.