THEANO TILE CODING: A GPU-BASED FRAMEWORK FOR REPRESENTATION OF CONTINUOUS VARIABLES AS BINARY FEATURES

Mohammad Pezeshki
Montreal Institute for Learning Algorithms
University of Montreal
mohammad.pezeshki@umontreal.ca

ABSTRACT

In this technical report, we have presented a GPU-based framework for mapping a continuous feature space into a binary representation using Tile Coding. We have also provided an example of function approximation using the presented framework. Of necessity, we only consider tilings that are uniformly distributed in a mesh shaped structure. Our main contribution is to accelerate the computations of Tile Coding function approximation by a vector-form implementation which enables us to use GPUs.

1 INTRODUCTION

Reinforcement Learning (RL) has gained great deal of attention over the last decade as an area of powerful algorithms for sequential decision making. However, in a high-dimensional, continuous feature space, compelling function approximations are necessary in order to scale RL to real-world applications. Tile Coding in one of the famous function approximations that reduces the computations while achieving high performance.

This paper aims to accelerate Tile Coding by proposing a vector-form implementation. Vector-form computations could be performed in parallel using GPUs. We present a framework on top of Theano that enables us to easily construct the computational graph for the task of Tile Coding. We also provide an example of function approximation to further illustrate it functionality.

2 TILE CODING

In Reinforcement Learning applications, an agent interacts with an environment by making a set of actions A, moving between a set of states S, and achieving rewards R. In many real-world cases, any of states or actions could be continuous variables. Evaluating the value function ($V(S)$ or $Q(S, A)$) over a large continuous space is not computationally feasible unless using a function approximator. One typical function approximation is Tile Coding.

Tile Coding quantizes a continuous feature space into tiles. A quantization is called a tiling and each tile in a tiling is associate with a weight. By having multiple overlapping tilings, each point could have multiple weights. The approximate value for a given point is achieved by summing the weights of tiles in all tilings which contain the point. Figure ? depicts the concept of tilings and tiles.

Training procedure for a given training example (x, y) is done by adjusting weights using the following update rule:

$$index = \text{quantize}(x) \tag{1}$$

$$\hat{y} = \text{sum}(\text{weights}[index]) \tag{2}$$

$$\text{weights}[index] = \text{weights}[index] + \alpha(y - \hat{y}) \tag{3}$$

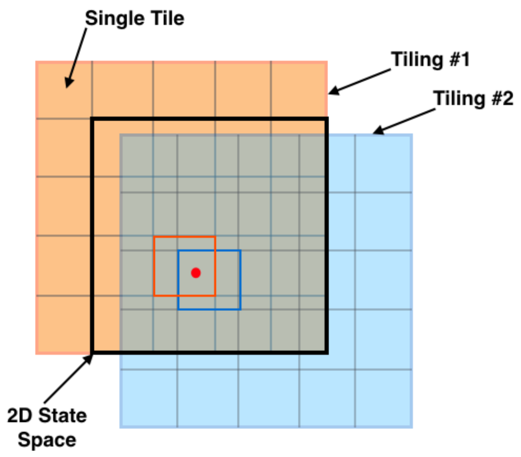

Figure 1: An example of two tilings on a 2D state space. The approximate value for the shown red point is the sum of two tiles containing the point.

3 The Framework

The code in available in author's github repository.

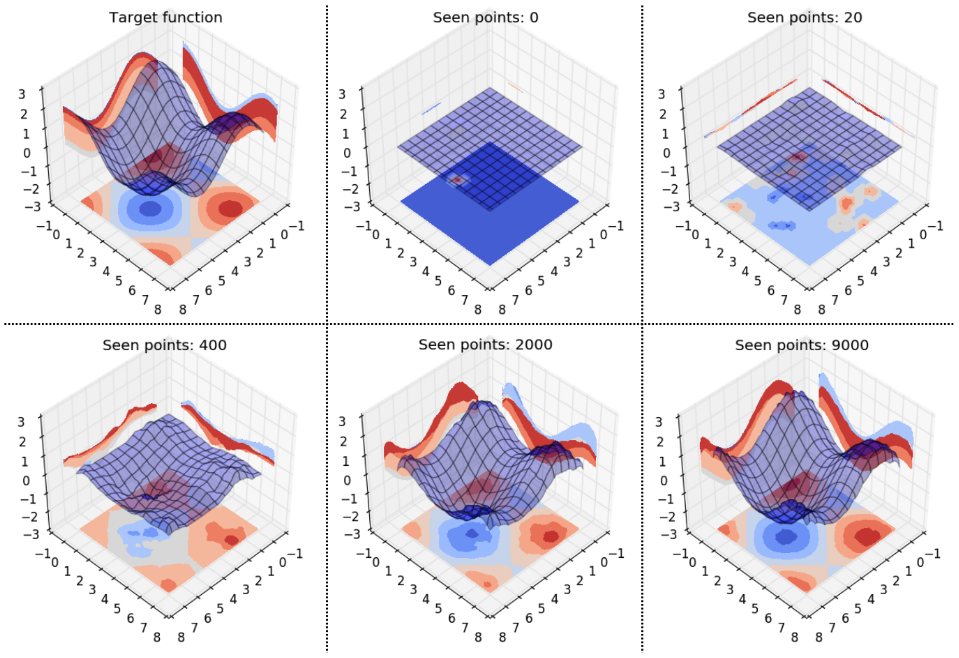

Figure 2: The Figure shows an example of approximation of the function $z = \sin(x) + \cos(y)$. Snapshots of the training procedure is shown as the number of seen data points grows.

4 Conclusion

In this work, we provided a framework for function approximation using the well-known method of Tile Coding in Reinforcement Learning literature. This framework in coded on the top of the Theano library that provides a high-level programming environment for computational graph construction.