Revisiting the Fuzzy Tilling Activation and How to Set its Hyperparameters

Muhammad Gohar Javed
Department of Electrical and Computer Engineering
University of Alberta
javed4@ualberta.ca

Tian Xiang Du
Department of Computer Science
University of Alberta
tdu@ualberta.ca

Amir Bahmani
Department of Computer Science
University of Alberta
bahmani1@ualberta.ca

Vlad Tkachuk
Department of Computer Science
University of Alberta
vtkachuk@ualberta.ca

Abstract

Sparsity has been shown to improve model performance on decision making problems with non-stationary data, such as online supervised learning and reinforcement learning (RL). Sparsity is when a large number of weights in a neural network are approximately zero. The fuzzy tiling activation (FTA) has been proposed to enforce sparsity by design, and has been shown to outperform other activations, such as ReLU and Tanh, that do not enforce sparsity. However, a difficulty of using the FTA is that it is sensitive to a new *tiling bound* hyperparameter, which currently requires a search to be set effectively. In this work we do two things. First, we reproduce experiments comparing FTA to ReLU using deep Q-learning (DQN) on the LunarLander RL environment, showing that indeed, for this environment FTA is no better than ReLU if the tiling bound is not set appropriately, Second, we empirically test if a simple technique of normalizing the activation values passed into the FTA cell can remove the need for setting the tiling bound.

1 Introduction

Neural networks (NN) have been shown to perform well on several challenging problems Mnih et al. [2013], Silver et al. [2017] Part of the reason for their success is due to their ability to learn good representations of the input data, which can then be used to effectively generalize. A representation of an input is defined as the values of all the neurons of the NN when the input is passed into the NN. The values of all the neurons are commonly called *features*. Unfortunately, when the NN is trained online (data recieved sequentially), interference can occur between the representations that are learned CITE. Interference is loosly defined as when a new learned representation affects the performance of the NN on previously seen data in a negative way (i.e. the old representation becomes worse due to the newly learned one). A potentially useful intuition for when interference can occur is when the representation for some input x_1 is dense (a large number of features are non-zero), since then, if another input x_2 is sufficiently different (uncorolated) from x_1, its representation should likely be quite different. However, in order for a different representation to be learned for x_1 the NN will likely have to update the weights that contribute to the representation of x_1, thus likely making the representation for x_1 worse in order to learn a good representation for x_2.

A method that has been found to reduce interference, is to enforce sparse representations (a large number of features are zero) CITE. Revisiting the example above with inputs x_1, x_2, if the representation for x_1 is sparse, then the representation that is learned for x_2 can use weights that are unused

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

for the representation of x_1, thus causing no interference between thkkke features for x_1 and x_2. The best way to enforce sparsity is an active topic of research A discussion of related works can be found in Section 2.

One way to enforce sparsity is by design, a method which does this is using a *fuzzy tiling activation* (FTA) [Pan et al., 2019]. FTA is an activation function that can be used at each layer of a network, similar to ReLU or Tanh. Different from ReLU and Tanh is that FTA has a vector output instead of a scalar output. FTA takes as input a scalar and outputs a one-hot vector. The output of FTA can be thought of as mapping a scalar $z \in \mathbb{R}$ to one of k evenly spaced bins in $\mathbb{R}^k$. A critical step to using FTA effectively is setting the tiling bound parameter u. This is because z should be in $[-u, u]$ to work well. However, just satisfying this is usually not enough for good performance (as one could easily set u very large). For instance if $u = 10$ and $k = 4$ then the bins would be $[-10, -5), [-5, 0), [0, 5), [5, 10]$ and if z is really takes values in $[0, 1]$ then it will always map to the same bin (bin 3 since $[0, 1] \subset [0, 5)$). Thus, as discussed by [Pan et al., 2019], setting the tiling bound is important; however, no good solution exists for setting it other than performing a search. The sensitivity to the tiling bound was especially prominant in the work of [Pan et al., 2019] when using FTA in deep Q-network (DQN) [Mnih et al., 2013] in the LunarLander RL environment.

As such, in this work we aim to do two things:

1. We aim to reproduce the FTA vs ReLU using DQN in LunarLander experiments presented in [Pan et al., 2019], to confirm that indeed FTA is sensitive to the tiling bound in this setting.
2. Next, we aim to test if the simple technique of normalizing the values (using $\tanh$, or batch norm) passed to FTA can remove the need for tuning the tiling bound parameter in FTA.

Experiments related to the first point are shown in section 4.1.

To answer the question in the second point we normalize the values passed into FTA using two methods: passing them into a $\tanh$, using batch norm. The method of using $tanh$ to normalize the input values was discussed in Pan et al. [2019], but not tested. Pan et al. [2019] mentioned that using this method might suffer from vanishing gradients, which we hope to confirm from this experiment. We hypothesis that normalizing the values should solve the need to set tiling bounds using search, since normalizing will ensure that then values passed into FTA are in $[-1, 1]$, thus the tiling bound can always be set to $[-1, 1]$. Experiments related to the second point are shown in section 4.2.

2 Background

Some background about sparsity and how it is useful in RL. Mention FTA paper here of course and maybe find any new papers that use FTA as well?

3 Preliminaries

In this section we introduce the Fuzzy Tiling Activation (FTA) and formally define the RL setting.

3.1 Fuzzy Tiling Activation

First we introduce the simpler *tiling activation* (TA) and then extend it to the FTA. A TA is a function $\phi : \mathbb{R} \to \mathbb{R}^k$, which expects inputs $z \in [-u, u]$ and maps them to one-hot vectors (standard basis) in $\mathbb{R}^k$. For example, if $u = 10$ and $k = 4$, then any $z \in [-10, -5)$ would be mapped to $(1, 0, 0, 0) \in \mathbb{R}^k$, any $z \in [-5, 0)$ would be mapped to $(0, 1, 0, 0)$, any $z \in [0, 5)$ would be mapped to $(0, 0, 1, 0)$, and any $z \in [5, 10]$ would be mapped to $(0, 0, 0, 1)$. Pan et al. [2019] show that the TA can implemented efficientely as follows. Assume you want evenly spaced bins of size $\delta > 0$, and $k = 2u/\delta$, where WLOG u was chosen such that it is divisable by δ. Define the tiling vector

$$\mathbf{c} = (-u, -u + \delta, l + 2\delta, \ldots, u - 2\delta, u - \delta) \in \mathbb{R}^k$$

Then the TA can be defined as

$$\phi(z) = \mathbb{1} - I_+(\max(\mathbf{c} - z\mathbb{1}, 0) + \max((z - \delta)\mathbb{1} - \mathbf{c}, 0))$$

where $\mathbb{1} \in \mathbb{R}^k$, and $I_+(\cdot)$ is an indicator function which returns 1 if the input is positive and zero otherwise, and is applied element wise to vectors.

An issue with the TA is that it has zero derivative almost everywhere. In order to solve this issue Pan et al. [2019] proposed modifying the TA for a fuzzy tiling activation (FTA). The FTA can be implemented as follows. Define a fuzzy version of the indicator function as

$$I_{\eta,+} = I_+(\eta - x)x + I_+(x - \eta)$$

where $\eta \geq 0$. Then the FTA can be defined as

$$\phi_\eta(z) = \mathbb{1} - I_{\eta,+}(\max(\mathbf{c} - z\mathbb{1}) + \max((z - \delta)\mathbb{1} - \mathbf{c}, 0)).$$

where $I_{\eta,+}$ is applied element wise to vectors The reason for the use of fuzzy in the name of FTA can be understood from the definition of $I_{\eta,+}$. The term $I_+(\eta - x)$ is 1 when $x < \eta$, while when $x > \eta$, $I_+(x - \eta)$ is 1. Thus, when $x < \eta$ we have that $I_{\eta,+}$ evaluates to x, giving smoother (fuzzy) indicator function. Notice, that this means for $x < \eta$ the derivative is non-zero, allowing us to use backpropagation to train out networks. Also, when $\eta = 0$ the original indicator function I_+ can be recovered.

3.2 Reinforcement Learning

The reinforcement learning (RL) setting can be formulated as an markov decision process (MDP). Formally an MDP is characterized by the tuple $(\mathcal{S}, \mathcal{A}, \mathbb{P}, R, \gamma)$, where $\mathcal{S}$ is that state space, cA is the action space, $\mathbb{P}$ is the transition probability kernel, $R : \mathcal{S} \times \mathcal{A}\mathcal{S} \to \mathbb{R}$ is the reward function, and $\gamma \in [0, 1]$ is the discount factor. A policy $\pi : \mathcal{S} \to \mathcal{M}_1$ is a map from states to distributions over actions, where $\mathcal{M}_1$ is the set of all probability measures over actions $\mathcal{A}$. Then, an agent (policy) interacts with an MDP as follows. At each time step $t \in 1, 2 \cdots$ the agent observes a state $s_t \in \mathcal{S}$. The agent takes an action according to its policy $a_t \sim \pi(s_t)$. The agent then transits to its next state according to $s_{t+1} \sim \mathbb{P}(s, a)$, and recieves a reward $R(s_t, a_t, s_{t+1})$.

An action value function under policy π is defined as

$$Q_\pi(s, a) = \mathbf{E}[G_t | S_t = s, A_t = a; A_{t+1:\infty} \sim \pi]$$

where $G_t = \sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1})$ is the return. The goal of the agent is maximize its expected reward from each state in $\mathcal{S}$.

4 Experiments

Explain the environment we test on. How the LunarLander env works (i.e. state, action space, and reward structure) Explain how DQN works since we use it as our algorithm.

4.1 Reproducibility Experiment

Explain the network structure and how we change only final layer with FTA and ReLU. Then explain what hyper-parameter sweeps we did and add the plot showing them (should discuss patterns we saw here for FTA hyper-params and how they confirm those discussed in [Pan et al., 2019]). Show the plot of FTA vs ReLU using the same hyper-params as in [Pan et al., 2019] and discuss how we see basically same results as them hopefully.

4.2 Normalizing Experiment

Explain how we plan to normalize the activation value going into FTA (i.e. $z = \tanh(Xw)$). I think we can leave this as TODO for final report.

5 Discussion

Discuss how the reproducibility experiment basically matches the results in [Pan et al., 2019] paper Discuss what we find from normalizing activations (i.e. if it worked or not, and if not any hypothesis why might have as to why)

References

V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Y. Pan, K. Banman, and M. White. Fuzzy tiling activations: A simple approach to learning sparse representations online. *arXiv preprint arXiv:1911.08068*, 2019.

D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676): 354–359, 2017.