Summary on Value Function Approximation

Christopher Mutschler

Value Function Approximation

- In practice we often need to approximate the value function, because:
  - (Explicit) tabular representations require too much space
  - We want to generalize information across state (see also: POMDPs!)
- For linear function approximation almost all convergence guarantees hold
  - For non-linear function approximation such guarantees cannot be given
  - But careful scheduling and several tricks help to stabilize training
- But:
  - Non-linear function approximation is very sensitive to hyper-parameter tuning!
  - See also: https://www.youtube.com/watch?v=Vh4H0gOwdIg
    (not directly related but definitely worth watching!)
  - And also: https://www.alexirpan.com/2018/02/14/rl-hard.html
    (but please read with humor)

The Deadly Triad

- Stability in RL is a very serious thing!
- Instability and divergence in RL mainly stem from
  1. Function Approximation.
  2. Bootstrapping.
  3. Off-policy training.
- Unfortunately, in most of the case we really use the full combination.

Another way to try to prevent instability is to use special methods for function approximation. In particular, stability is guaranteed for function approximation methods that do not extrapolate from the observed targets. These methods, called *averagers*, include nearest neighbor methods and locally weighted regression, but not popular methods such as tile coding and artificial neural networks (ANNs).

Exercise 11.3 (programming) Apply one-step semi-gradient Q-learning to Baird's counterexample and show empirically that its weights diverge. □

11.3 The Deadly Triad

Our discussion so far can be summarized by saying that the danger of instability and divergence arises whenever we combine all of the following three elements, making up what we call *the deadly triad*:

Function approximation A powerful, scalable way of generalizing from a state space much larger than the memory and computational resources (e.g., linear function approximation or ANNs).

Bootstrapping Update targets that include existing estimates (as in dynamic programming or TD methods) rather than relying exclusively on actual rewards and complete returns (as in MC methods).

Off-policy training Training on a distribution of transitions other than that produced by the target policy. Sweeping through the state space and updating all states uniformly, as in dynamic programming, does not respect the target policy and is an example of off-policy training.

ontrol or to generalized policy iteration.
nstability arises in the simpler prediction
ne deadly triad. The danger is also *not*
nvironment, because it occurs just as
rogramming, in which the environment

sent, but not all three, then instability
h the three and see if there is any one

clearly cannot be given up. We need
eat expressive power. We need at least
s and parameters. State aggregation or
with data are too weak or too expensive.
dratic complexity and are therefore too

ost of computational and data efficiency.
itational efficiency. Monte Carlo (non-
verything that happens between making

Google

the deadly triad

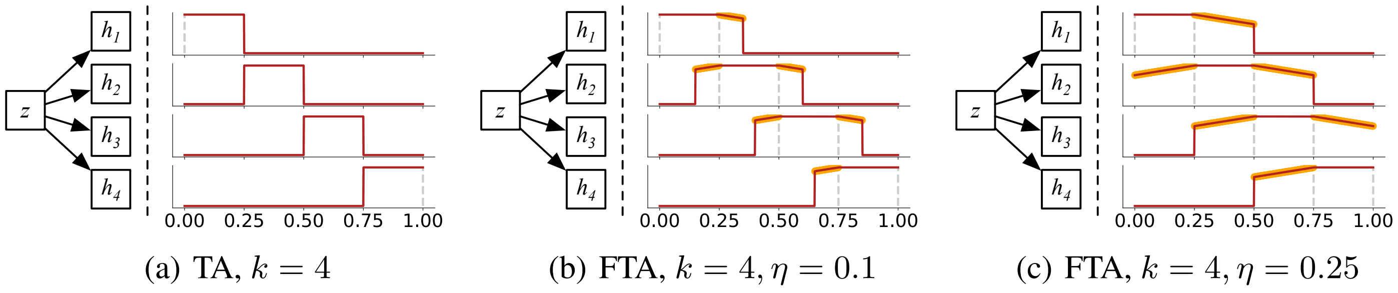
Alle | Bilder | Videos | Shopping | News | Mehr | Einstellungen | Suchfilter

Ungefähr 1.860.000 Ergebnisse (0,46 Sekunden)

arxiv.org › cs ▾ Diese Seite übersetzen
Deep Reinforcement Learning and the Deadly Triad
von H van Hasselt - 2018 - Zitiert von: 19 - Ähnliche Artikel
06.12.2018 - Sutton and Barto (2018) identify a **deadly triad** of function approximation, bootstrapping, and off-policy learning. When these three properties ...
Du hast diese Seite 2 Mal aufgerufen. Letzter Besuch: 24.07.19

medium.com › defeating-the-deadly-triad-f5a8e3... ▾ Diese Seite übersetzen
Defeating the Deadly Triad - David Sanwald - Medium
I thought this might by interesting but as long as medium doesn't provide proper support for typesetting formulas, I have to point to the original version on github: ...

Fuzzy Tiling Activations



- DQNs need target networks to reduce the chance of divergence
- Main reason:
  - The Q-Targets are non-stationary and moving
  - over subsequent gradient descent steps
- Idea:
  - Introduce a special activation function that produces sparse representations! i.e., updates on a particular Q-value does no affect nearby Q-values that much.
  - FTA layers stack k-dimensional sparse encodings for each element $h_1 W_2$ ($h_1 = xW_1$)

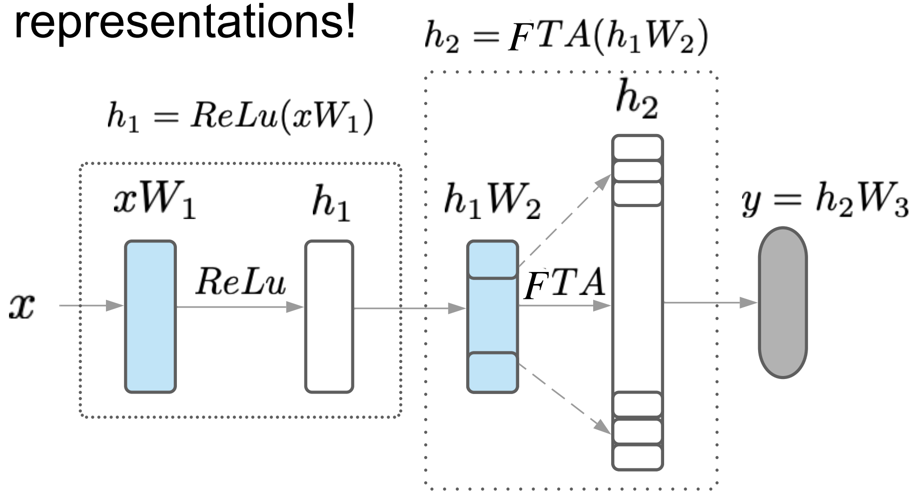


Figure 2: A visualization of an FTA layer

Pan et al.: Fuzzy Tiling Activations: A Simple Approach to Learning Sparse Representations Online. ICLR 2021.

Lesson of today

“Be careful with (non-linear) function approximation”

References

Books:

- Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.

Lectures:

- Pieter Abbeel: CS 188 Introduction to Artificial Intelligence. Fall 2018
- UCL Course on RL. http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html

Web:

- https://medium.com/init27-labs/understanding-q-learning-the-cliff-walking-problem-80198921abbc
- https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/
- https://danieltakeshi.github.io/2016/10/31/going-deeper-into-reinforcement-learning-understanding-q-learning-and-linear-function-approximation/
- http://www0.cs.ucl.ac.uk/staff/d.silver/web/Resources_files/deep_rl.pdf