Advanced Topics in Reinforcement Learning

Lecture 14: Deep Reinforcement Learning I

Josiah Hanna
University of Wisconsin — Madison

Announcements

- Homework done!
- Read Chapter 13 for next week. Policy-based RL!
- Upcoming dates:
  - Literature survey due next week: October 30
  - Exam: November 6

Learning Outcomes

After this week, you will be able to:

1. Describe key benefits and challenges of using neural networks as function approximators in RL.

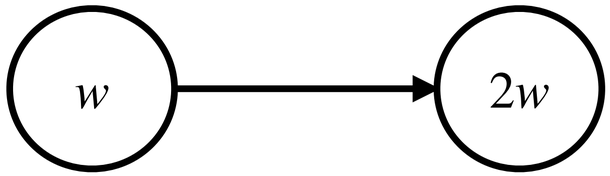
2. Describe how deep learning techniques are used within RL environments.

3. Implement key deep RL techniques such as target networks and experience replay.

Linear Function Approximation Review

- Assume value estimate is a linear function of state-action features.
  - $\hat{q}(s, a, \mathbf{w}) = \mathbf{w}^\top x(s, a) = \sum_{i=1}^{d} w_i x_i(s, a)$
- The features, $x_i(s, a)$, can be non-linear functions of state variables and actions.
  - Expressive choices for $\mathbf{x}(s, a)$ make linear methods more powerful than they first appear.
  - What if we instead learn $\mathbf{x}$ while learning $\hat{q}$? —> Deep RL!!!
- Semi-Gradient Q-learning:
  - $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \alpha(R_{t+1} + \gamma \max_{a'} \hat{q}(S_{t+1}, a', \mathbf{w}_t) - \hat{q}(S_t, A_t, \mathbf{w}_t)) \nabla \hat{q}(S_t, A_t \mathbf{w}_t)$

Divergence Example #1



- Initialize $w = 10$, $\gamma = 0.99$, $\alpha = 0.1$, and the transition gives zero reward.
- What happens with semi-gradient TD after you've seen this transition?
  - w increases to try and match bootstrapping target of $2\gamma w$.
- How can we fix divergence here?
  - First extend example to full MDP, then remove off-policy, bootstrapping, or function approximation.

Off-Policy Divergence

- In general, we lack convergence or even stability results for the simplest and most practical off-policy, semi-gradient methods.
- Includes Q-learning which is one of the most widely used algorithms in RL.
  - Maybe OK if behavior and target policy are close?
  - State distributions will then be close.

The Deadly Triad

1. Function Approximation: changing the value estimate at one state affects the value estimate at other states.

2. Bootstrapping: using existing estimated values as part of the learning target instead of only using actual returns.

3. Off-Policy Learning: using a distribution of transitions (s, a, s', r) other than that of the target policy.

Do we need the deadly triad?

- Why use function approximation?
  - Too many states to represent explicitly; need generalization.
- Why bootstrap?
  - Memory and computation requirements; learning in non-episodic tasks; faster learning.
- Why use off-policy learning?
  - Separate exploration and exploitation; general purpose learning agents must learn about multiple reward signals and target policies at the same time.

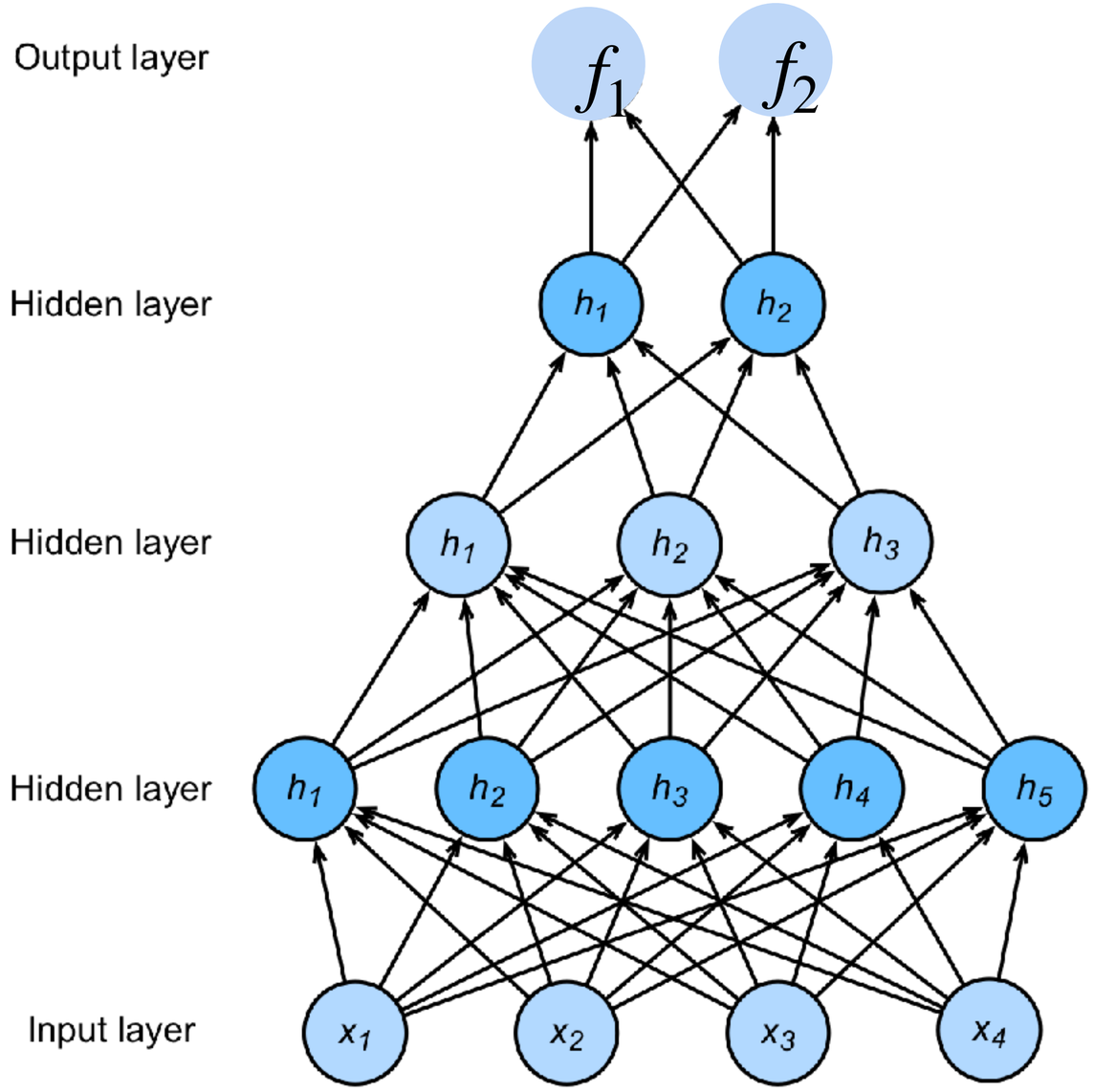
The Deadly Triad in Deep RL

- In practice, each component of the deadly triad is not binary.
- Bootstrapping: can use n-step returns or target networks to decrease amount of bootstrapping.
- Function approximation: larger neural networks decrease over-generalization.
- Off-Policy learning: controlling distribution of samples from the replay buffer modulates how off-policy updates are.

“Deep Reinforcement Learning and the Deadly Triad.” Van Hasselt et al. 2018.

Neural Network Function approximations

$$\mathbf{h}_1 = \sigma(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1)$$
$$\mathbf{h}_2 = \sigma(\mathbf{W}_2\mathbf{h}_1 + \mathbf{b}_2)$$
$$\mathbf{h}_3 = \sigma(\mathbf{W}_3\mathbf{h}_2 + \mathbf{b}_3)$$
$$\mathbf{f} = \mathbf{W}_4\mathbf{h}_3 + \mathbf{b}_4$$



Hierarchical Representations

- Each layer of a neural network transforms the output of the layer before it (the first layer transforms the input).
- We can say that each layer is producing a new representation of the values at the previous layer.
- Chaining layers together allows the network to learn progressively more complex representations of the data.
  - Pixels —> Edge detectors —> Shape detectors —> object detectors

Neural Network Training in RL

- Semi-Gradient Q-learning:
  - $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \alpha(R_{t+1} + \gamma \max_{a'} \hat{q}(S_{t+1}, a', \mathbf{w}_t) - \hat{q}(S_t, A_t, \mathbf{w}_t)) \nabla \hat{q}(S_t, A_t \mathbf{w}_t)$
- The parameter, $\mathbf{w}_t$, is all weights and biases of the neural network.
- Backpropagation algorithm: use chain rule of calculus to derive gradient of network outputs with respect to each weights or bias of the network.
- Adjust each weight in proportion to gradient of output times TD-error.

Problems with Backpropagation

- With many layers, gradients for parameters in the initial layers are either very large or very small. Why?
  - Chain rule means the gradient is a product of many factors. Gradient magnitude can grow or decay exponentially in network depth.
- Alternative #1: Find a different learning rule.
  - Evolution; Reinforcement Learning.
- Alternative #2: Make our networks more backprop-friendly.

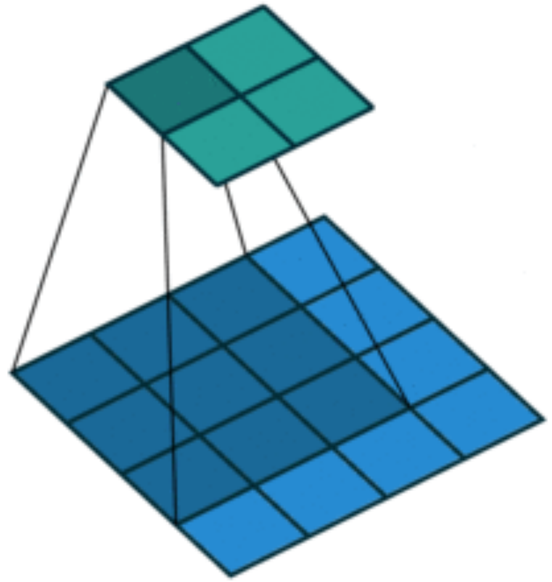
Training Improvements

- Regularization: methods that encourage learning simple models over more complex ones.
  - Add $||\mathbf{w}||_2^2$ as an additional objective; drop-out
- Better initializations: start gradient descent at a good location in weight space.
  - Pre-training; special initialization rules (e.g., Glorot/Xavier initializations)
- Architecture improvements
  - Residual connections, normalize inputs and layer activations.
- Not all of these ideas work well for reinforcement learning! (E.g., scaling)

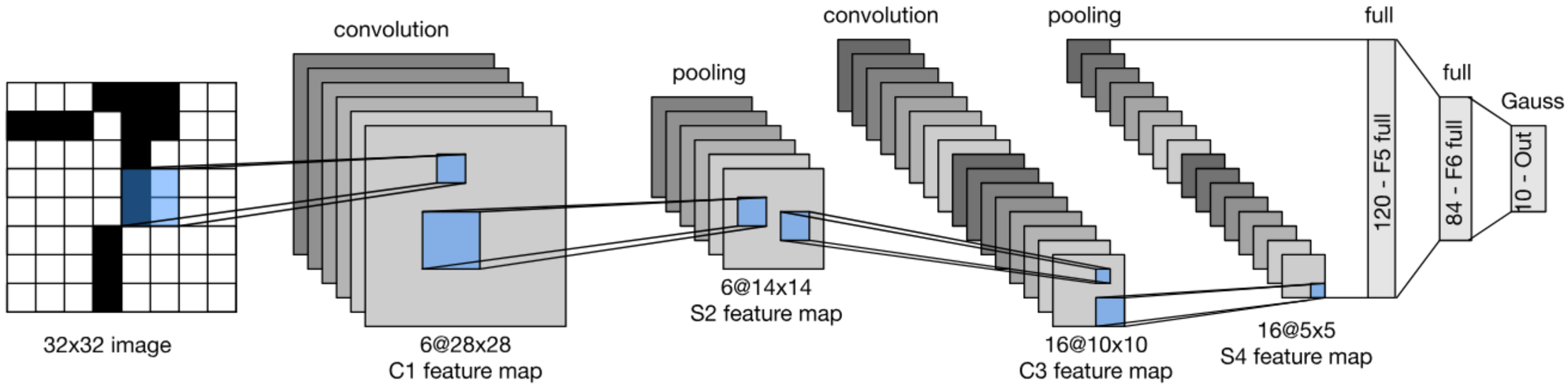
http://joschu.net/docs/nuts-and-bolts.pdf

Convolutional Neural Networks

- Special architecture primarily for processing visual inputs.
- Local connections between inputs and outputs at next layer.
- Learn “filters” that have the same weights no matter where applied on an image.

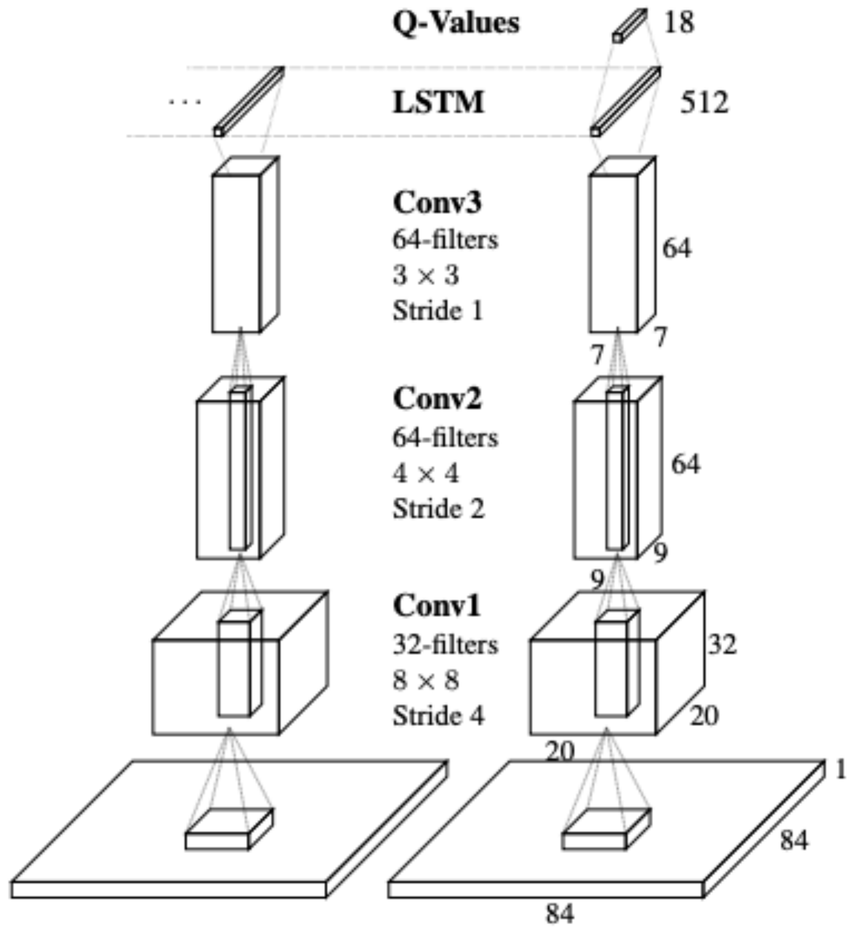


(vdumoulin@ Github)



Recurrent Neural Networks

- Recurrent neural networks update a hidden state that is an input to computations at the next time-step.
- Allows networks to remember previously seen inputs when computing future outputs.
- In RL, provides a method to learn a Markov state.
- Alternative to frame-stacking.



Deep Recurrent Q-Learning for Partially Observable MDPs. Hausknecht and Stone. 2015.

Harry’s Presentation

DR3: Value-Based Deep Reinforcement Learning Requires Explicit Regularization

Kumar et al. 2021

- Slides

Summary

- Neural networks are differentiable, non-linear function approximations that can be easily (in principle) used with semi-gradient reinforcement learning.
- Instead of learning a linear function of fixed, non-linear features, neural networks also learn the non-linear features.
- Removes the need for feature engineering though may require careful architecture and training considerations.

Action Items

- Complete literature review.
- Read Chapter 13 (skip 13.5).