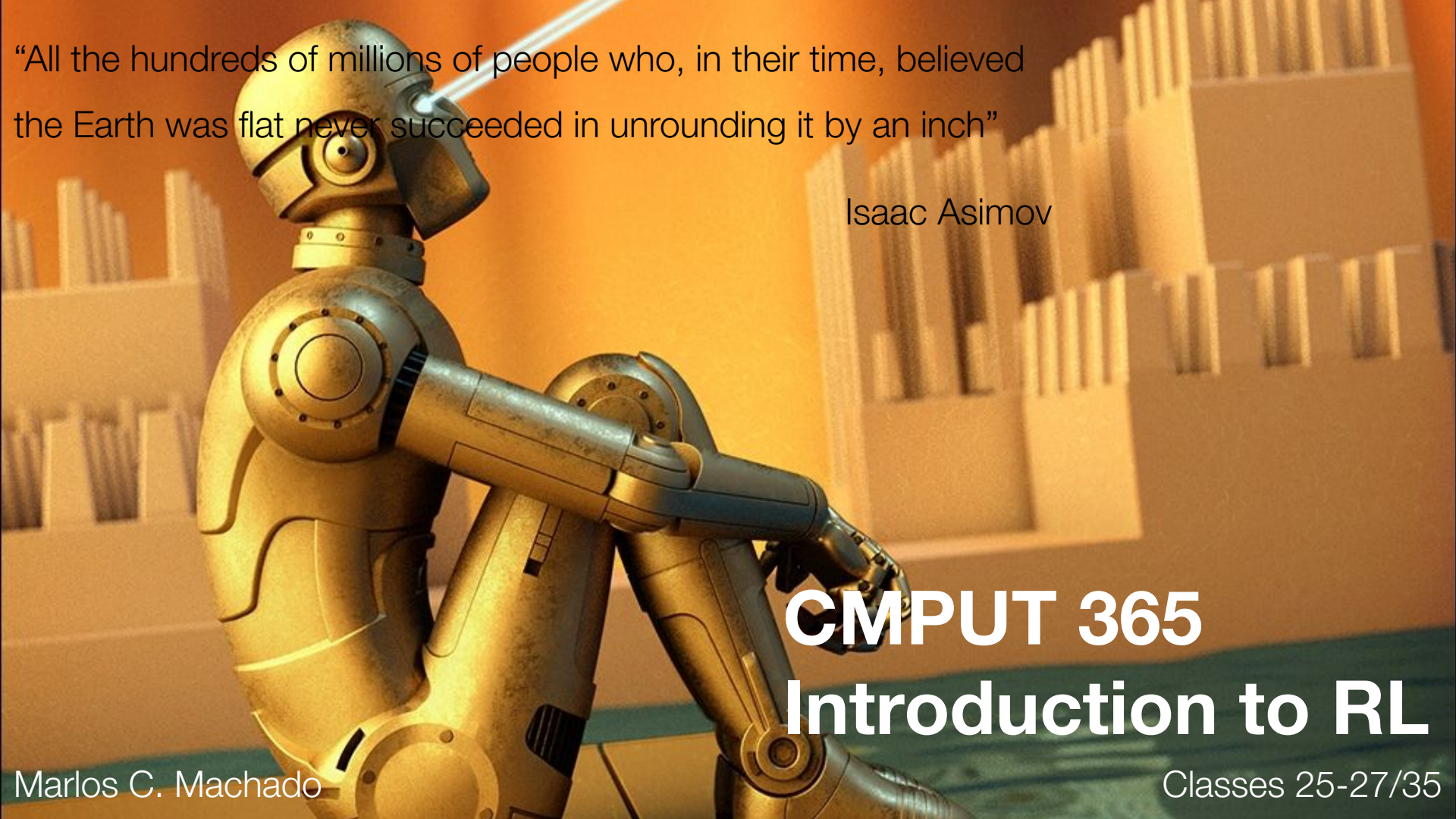

“All the hundreds of millions of people who, in their time, believed
the Earth was flat never succeeded in unrounding it by an inch”
Isaac Asimov
CMPUT 365
Introduction to RL
Marlos C. Machado
Classes 25-27/35

Reminder I

You **should be enrolled in the private session** we created in Coursera for CMPUT 365.

I **cannot** use marks from the public repository for your course marks.

You **need** to **check**, **every time**, if you are in the private session and if you are submitting quizzes and assignments to the private section.

The deadlines in the public session **do not align** with the deadlines in Coursera.

If you have any questions or concerns, **talk with the TAs** or email us `cmput365@ualberta.ca`.

Reminders and Notes

- The practice quiz and programming assignment for “Constructing features for prediction” are due on Wednesday.
- Exam viewing for Midterm 2:
  - Tuesday (March 17): 13:00 - 15:00 @ UCOMM-3-480
  - Wednesday (March 18): 15:00 - 17:00 @ UCOMM-4-104
  - Friday (March 20): 15:00 - 17:00 @ UCOMM-4-104
  - Monday (March 23): 15:00 - 17:00 @ UCOMM-4-104
- The grades for the last Coursera activity will be (are?) available on Canvas soon.
- Unfortunately, I can’t stay after class today.

Please, interrupt me at any time!

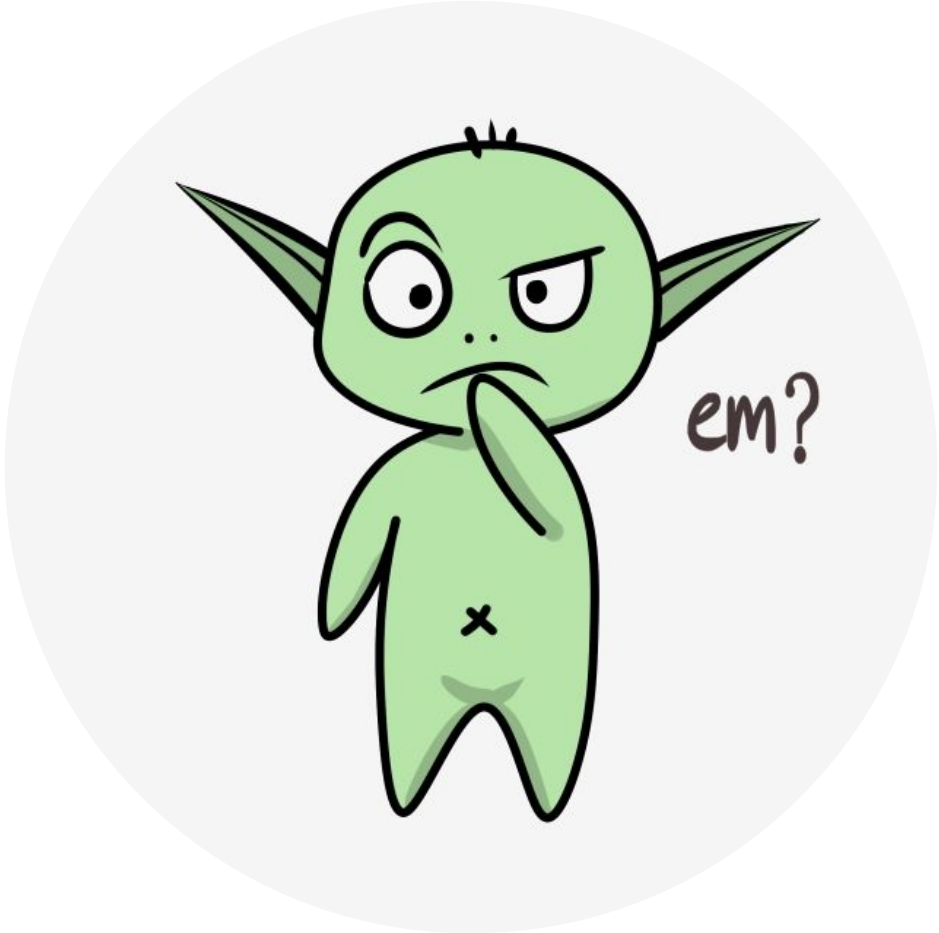
https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

Previous topic: Semi-gradient TD with Function Approximation

Semi-gradient TD(0) for estimating $\hat{v} \approx v_\pi$

Input: the policy π to be evaluated **What approximation should we use?!**
Input: a differentiable function $\hat{v} : \mathcal{S}^+ \times \mathbb{R}^d \to \mathbb{R}$ such that $\hat{v}(\text{terminal},\cdot) = 0$
Algorithm parameter: step size $\alpha > 0$
Initialize value-function weights $\mathbf{w} \in \mathbb{R}^d$ arbitrarily (e.g., $\mathbf{w} = \mathbf{0}$)

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose $A \sim \pi(\cdot|S)$
    Take action A, observe R, S'
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha\big[R + \gamma\hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})\big]\nabla\hat{v}(S,\mathbf{w})$
    $S \leftarrow S'$
  until S is terminal



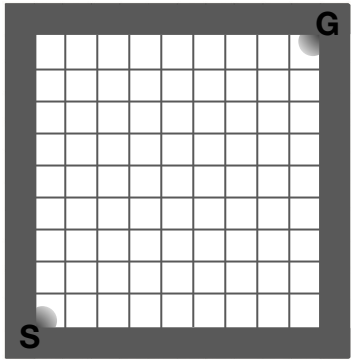
https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

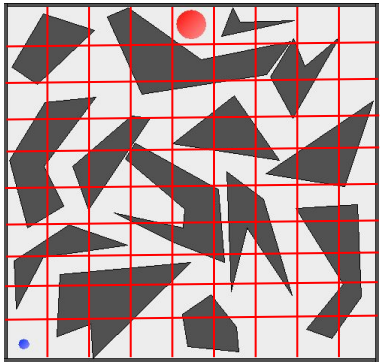
Feature Construction for Linear Methods

- Linear methods can be effective, but they heavily rely on how states are represented in terms of features.
- Feature construction is a way of adding domain knowledge; but at the same time, it went out of fashion because of *deep reinforcement learning*.
- Naïve linear function approximation methods do not take into consideration the interaction between features.

State Aggregation

- Simplest form of representation
- States are grouped together (one component of the vector **w**) for each group.
- State aggregation is a special case of SGD in which the gradient, $\nabla \hat{v}(S_t, \mathbf{w}_t)$, is 1 for S_t's group's component and 0 for the other components.





http://irl.cs.brown.edu/pinball/

Polynomials

- Doesn’t work so well, but they are one of the simplest families of features.

Polynomials

- Doesn’t work so well, but they are one of the simplest families of features.
- Suppose an RL problem has states with two numerical dimensions.

$$\mathbf{x}(s) = (s_1, s_2)^\top$$

Polynomials

- Doesn’t work so well, but they are one of the simplest families of features.
- Suppose an RL problem has states with two numerical dimensions.

$$\mathbf{x}(s) = (s_1, s_2)^\top$$

But what about interactions? What if both features were zero?

$$\mathbf{x}(s) = (1, s_1, s_2, s_1 s_2)^\top$$

Polynomials

- Doesn't work so well, but they are one of the simplest families of features.
- Suppose an RL problem has states with two numerical dimensions.

$$\mathbf{x}(s) = (s_1, s_2)^\top$$

But what about interactions? What if both features were zero?

$$\mathbf{x}(s) = (1, s_1, s_2, s_1 s_2)^\top$$

And we can keep going…

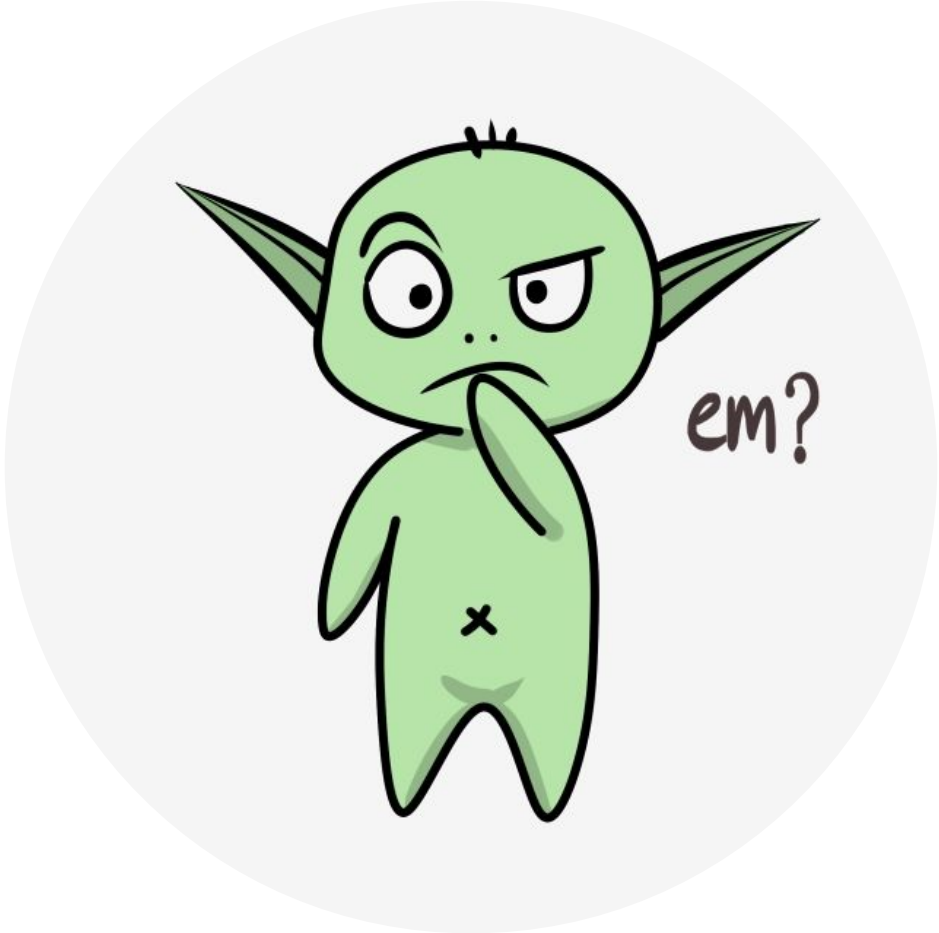
$$\mathbf{x}(s) = (1, s_1, s_2, s_1 s_2, s_1^2, s_2^2, s_1 s_2^2, s_1^2 s_2, s_1^2 s_2^2)^\top$$

Polynomials

Suppose each state s corresponds to k numbers, s_1, s_2, ..., s_k, with each $s_i \in \mathbb{R}$. For this k-dimensional state space, each order-n polynomial-basis feature x_i can be written as

$$x_i(s) = \Pi_{j=1}^{k} s_j^{c_{i,j}}, \tag{9.17}$$

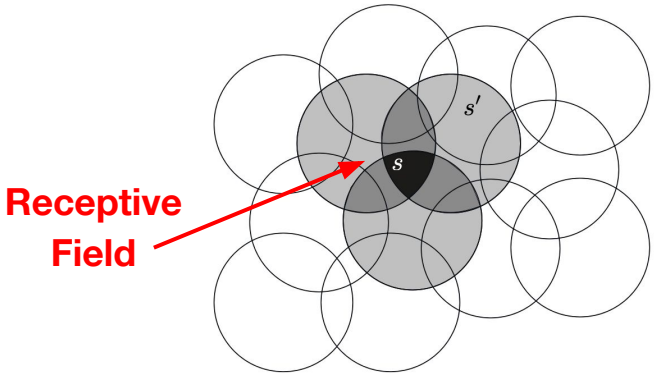
where each $c_{i,j}$ is an integer in the set $\{0, 1, \ldots, n\}$ for an integer $n \geq 0$. These features make up the order-n polynomial basis for dimension k, which contains $(n+1)^k$ different features.



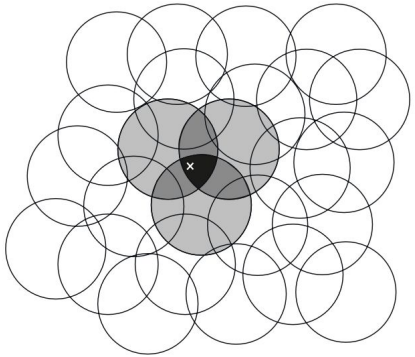
https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

Coarse Coding

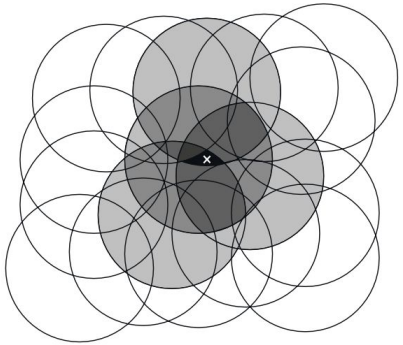
- Consider a task in which the natural representation of the state set is a continuous two-dimensional space.
- We define binary features indicating whether a state is present or not in a specific circle.



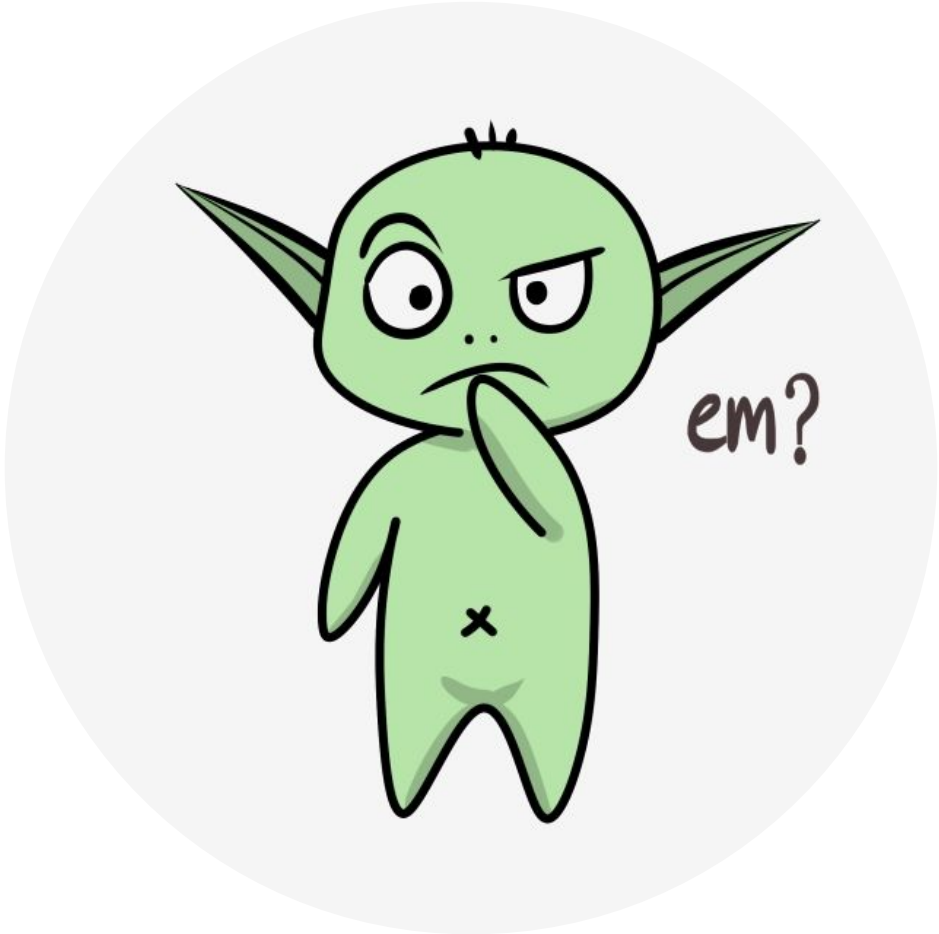
The shape defines generalization



Narrow generalization



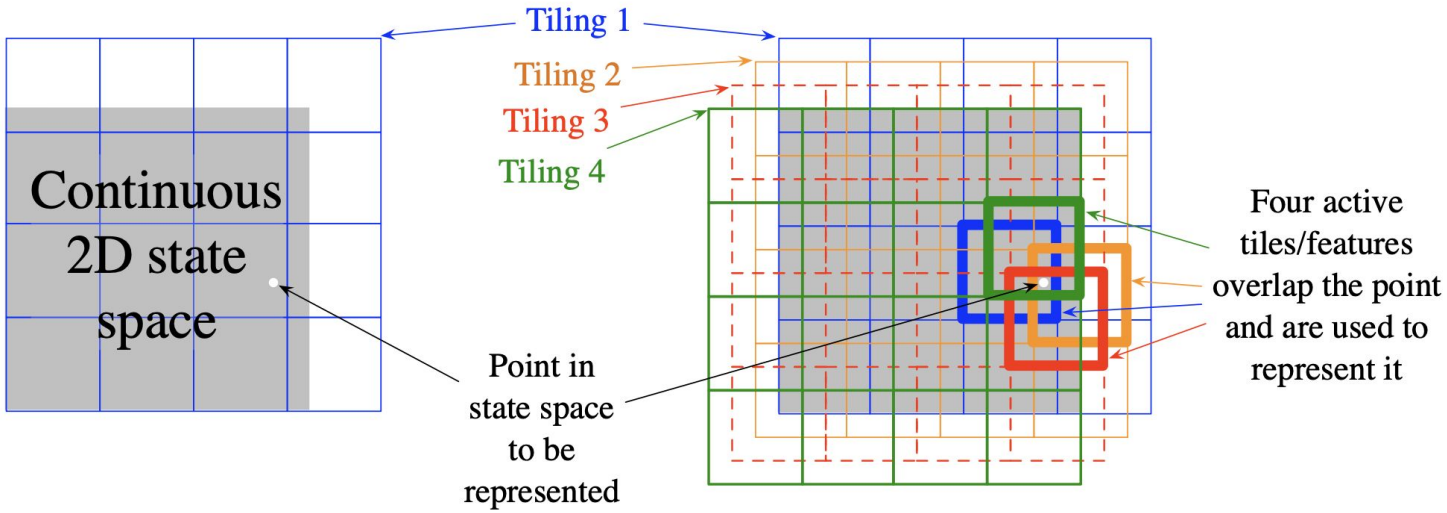
Broad generalization



https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

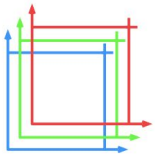
Tile Coding

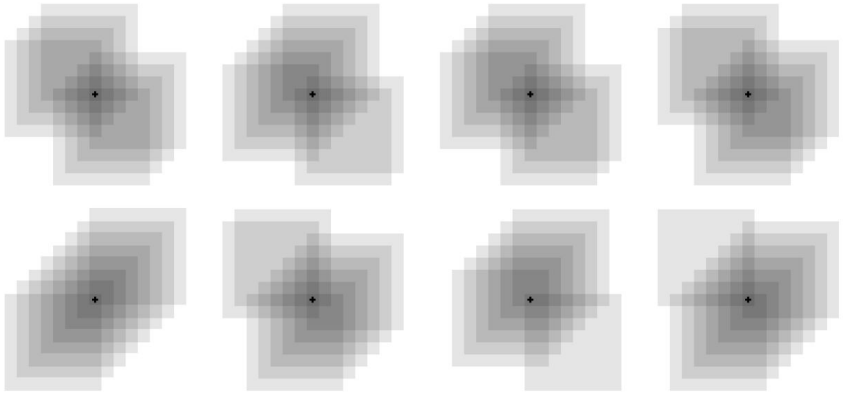
- Tile coding is a form of coarse coding for multi-dimensional continuous spaces (with a fixed number of active features per timestep).



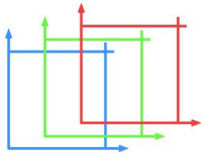
Tile Coding

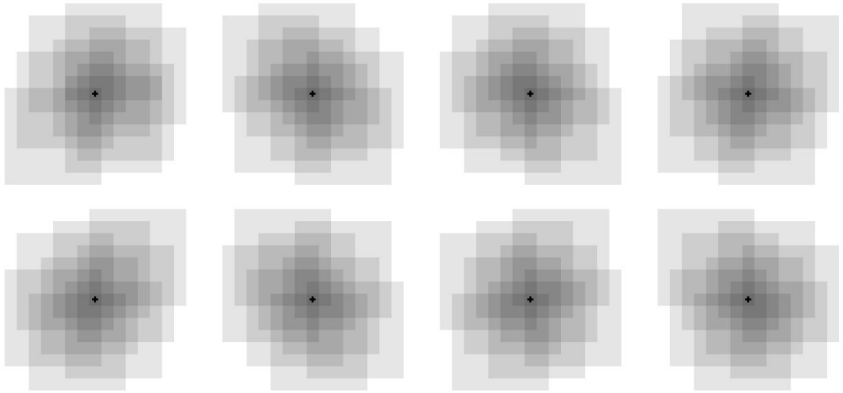
Possible generalizations for uniformly offset tilings

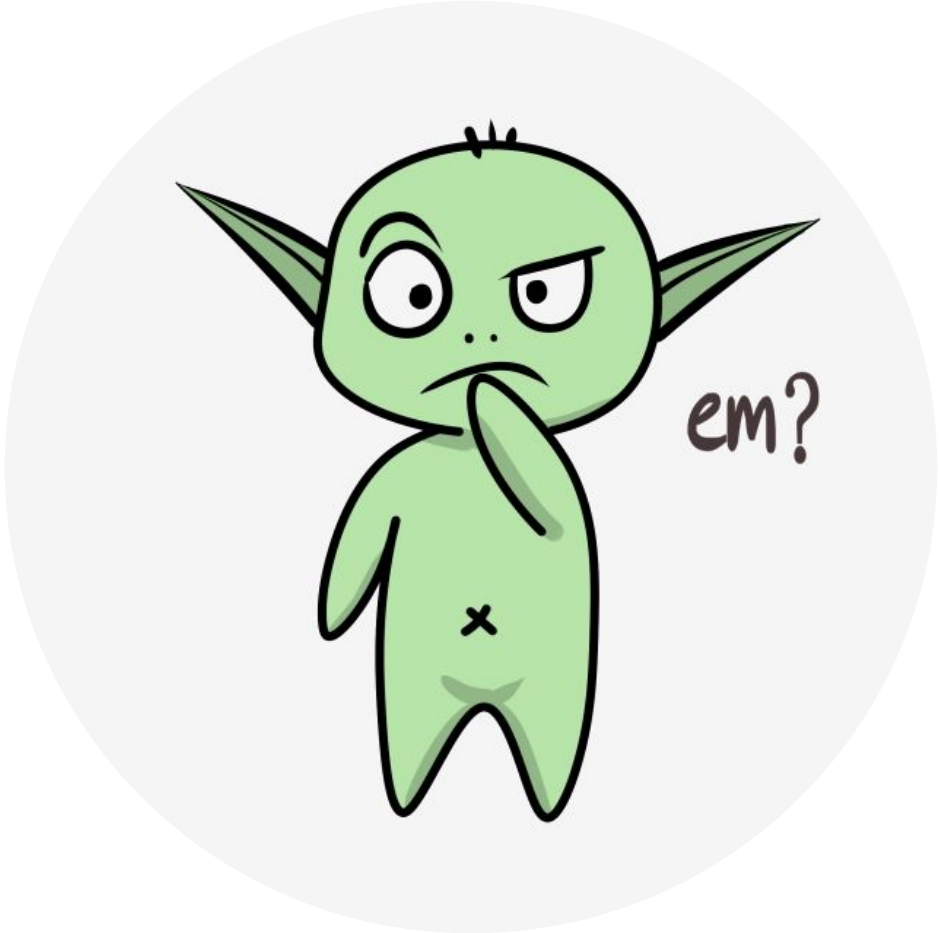




Possible generalizations for asymmetrically offset tilings





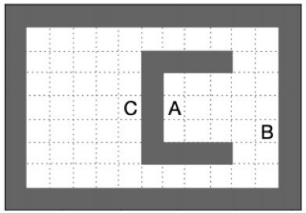


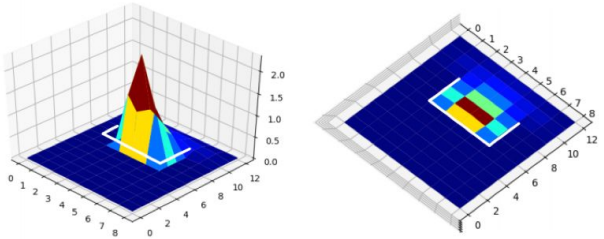
https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

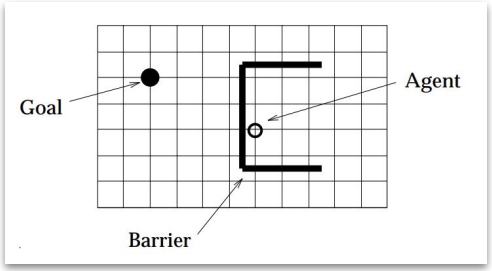
It Isn’t that We do Function Approximation Because We Cannot do Tabular Reinforcement Learning

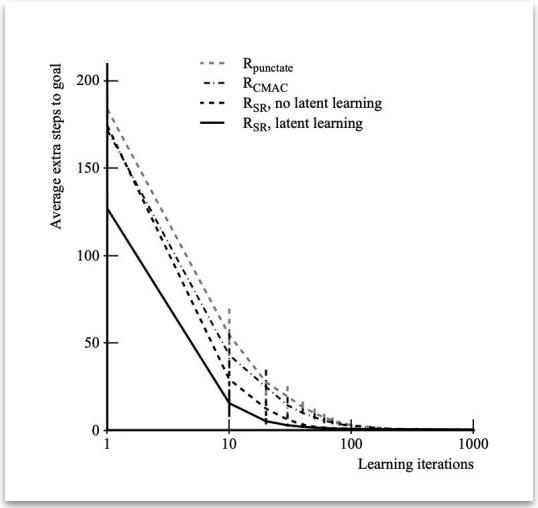
- Successor Representation [Dayan, Neural Computation 1993].

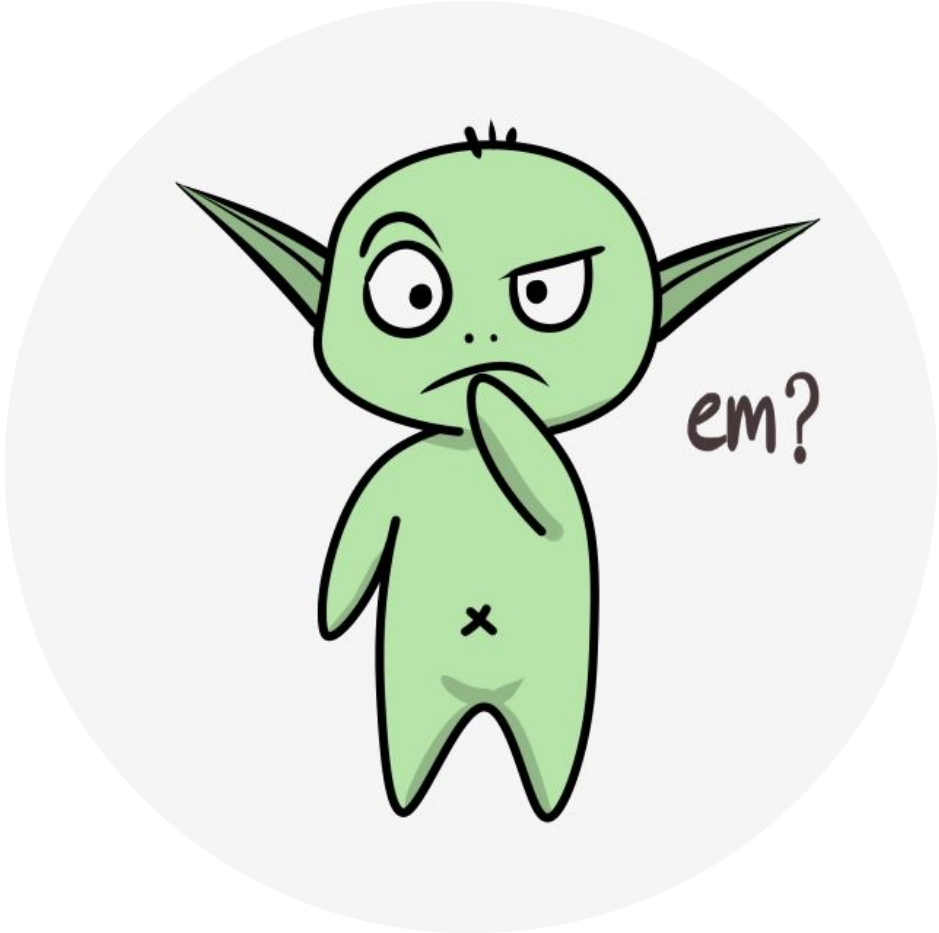
$$\Psi_\pi(s, s') = \mathbb{E}_\pi \left[\sum_t \gamma^t \mathbf{1}_{S_t = s'} | S_0 = s \right]$$









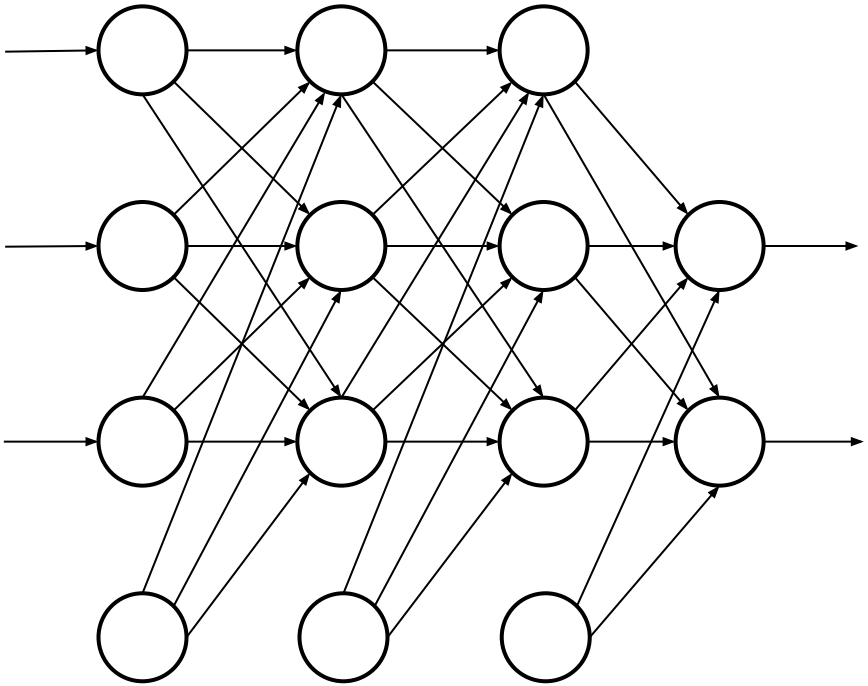


https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

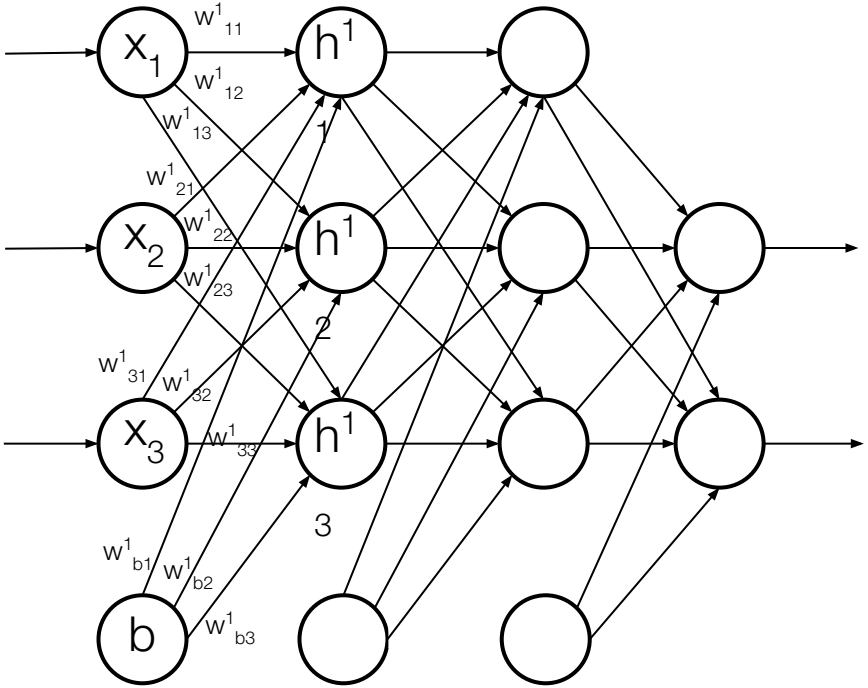
Nonlinear Function Approximation: Artificial Neural Networks

- The basics of deep reinforcement learning.
- Idea: Instead of using linear features, we feed the “raw” input to a neural network and ask it to predict the state (or state-action) value function.

Neural Networks



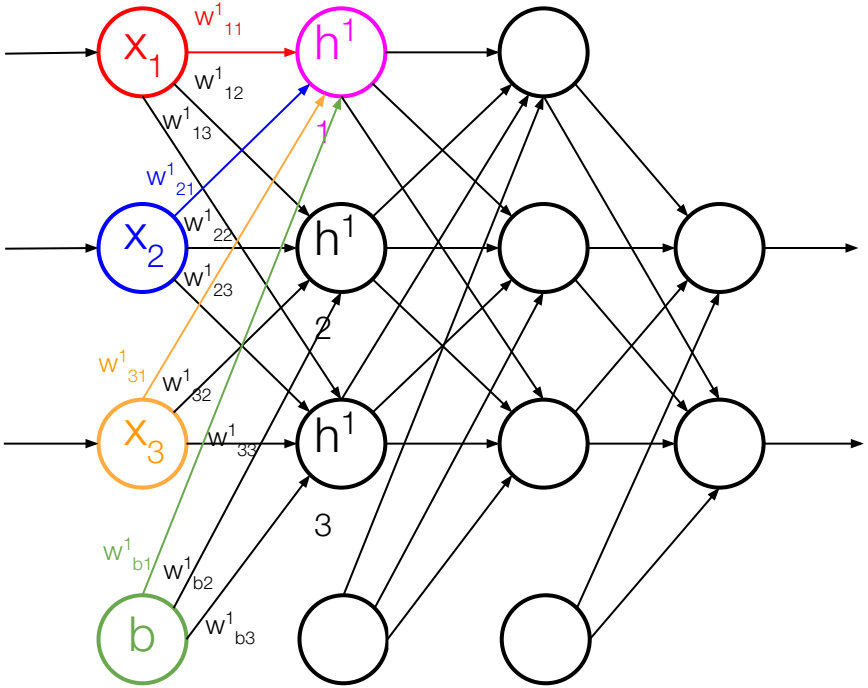
Neural Networks



$\mathbf{h^1} = \text{act}(\mathbf{xW^1})$

s.t. $h^1_1 = x_1 w^1_{11} + x_2 w^1_{21} + x_3 w^1_{31} + b w^1_{b1}$

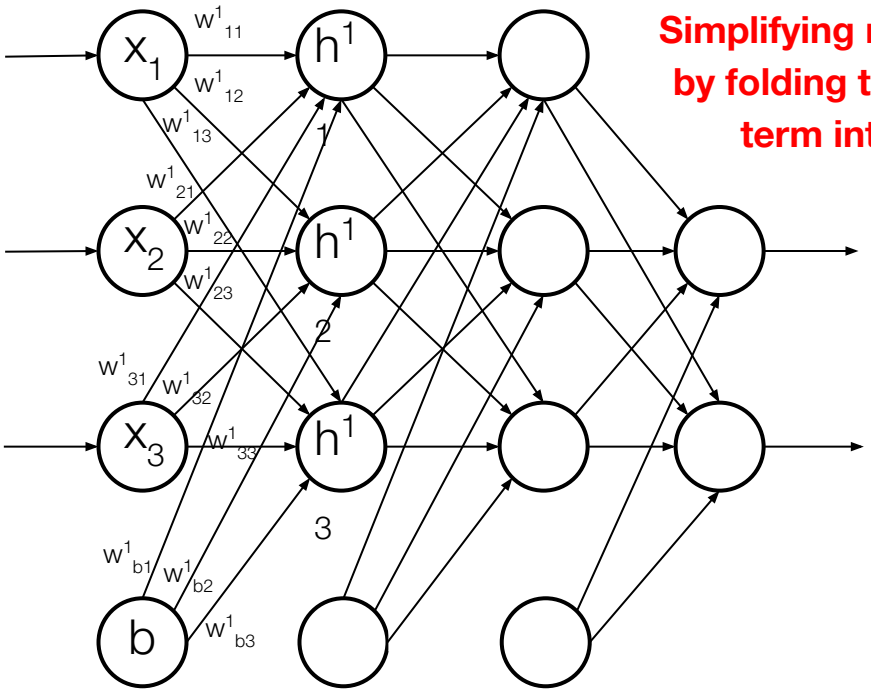
Neural Networks



$\mathbf{h^1} = \text{act}(\mathbf{x}\mathbf{W^1})$

s.t. $h^1_1 = x_1 w^1_{11} + x_2 w^1_{21} + x_3 w^1_{31} + b w^1_{b1}$

Neural Networks



Simplifying notation by folding the bias term into x

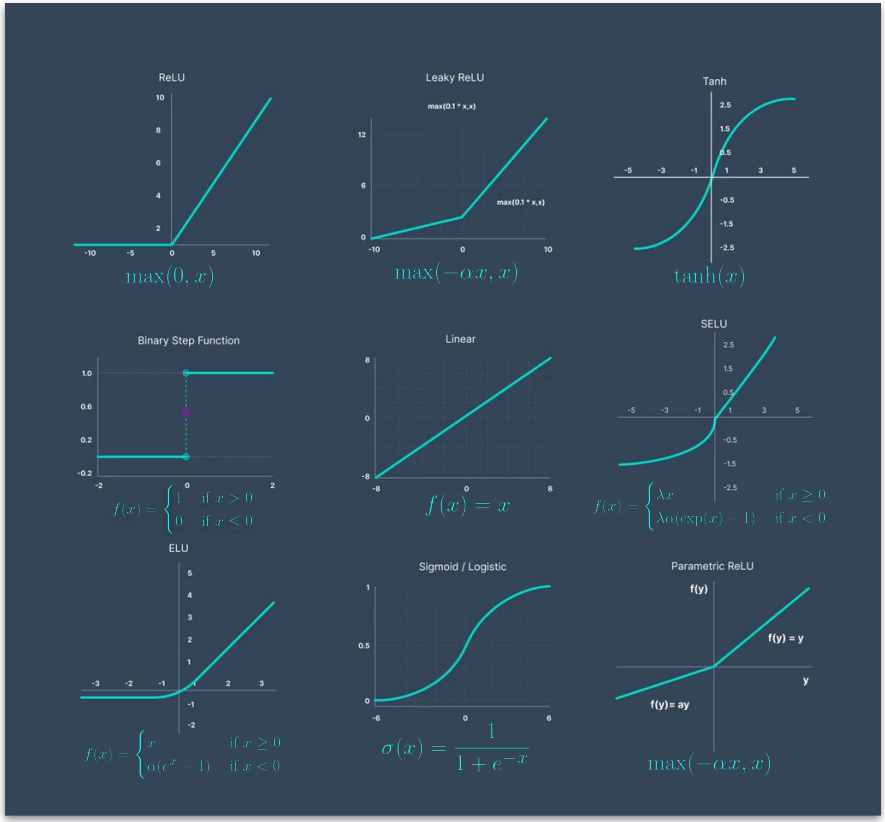
$\mathbf{h^1} = act(\mathbf{xW^1})$

s.t. $h^1_1 = x_1 w^1_{11} + x_2 w^1_{21} + x_3 w^1_{31} + b w^1_{b1}$

The activation function introduces non-linearity

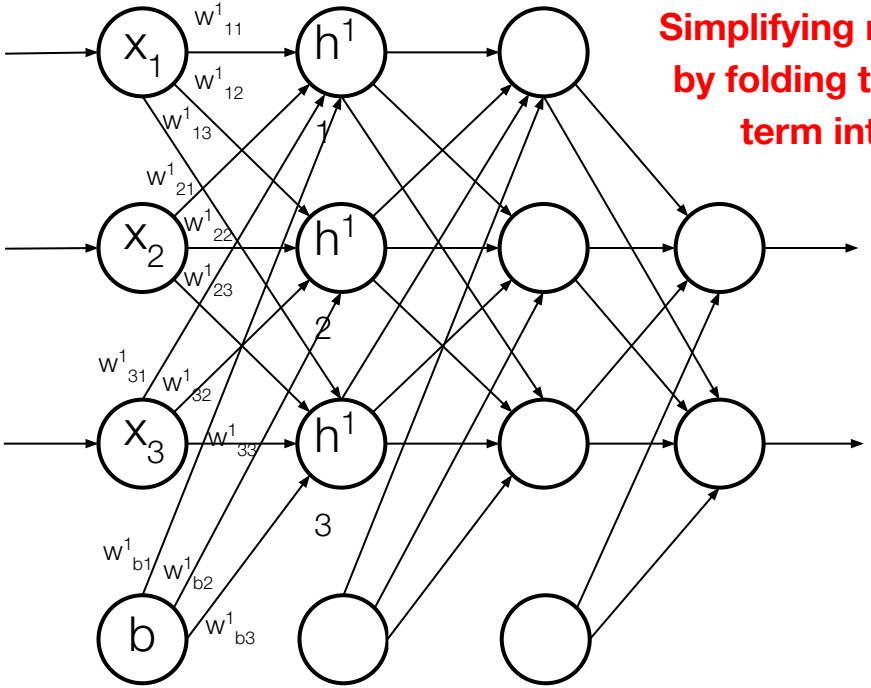
E.g.: *f(x) = max(0, x)*

Main Types of Activation Functions



https://www.v7labs.com/blog/neural-networks-activation-functions

Neural Networks



Simplifying notation by folding the bias term into x

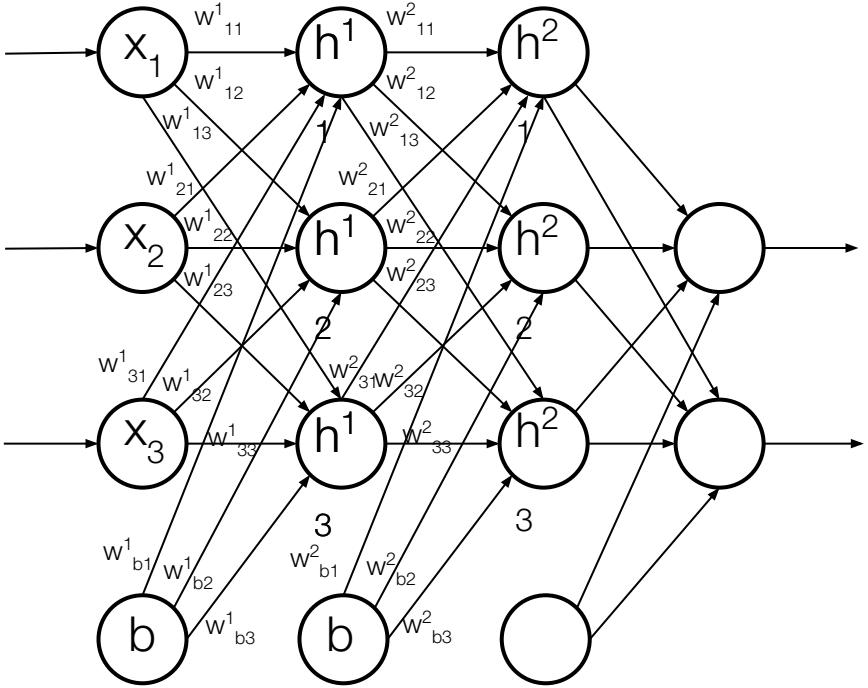
$\mathbf{h^1} = act(\mathbf{xW^1})$

s.t. $h^1_1 = x_1 w^1_{11} + x_2 w^1_{21} + x_3 w^1_{31} + b w^1_{b1}$

The activation function introduces non-linearity

E.g.: *f(x) = max(0, x)*

Neural Networks



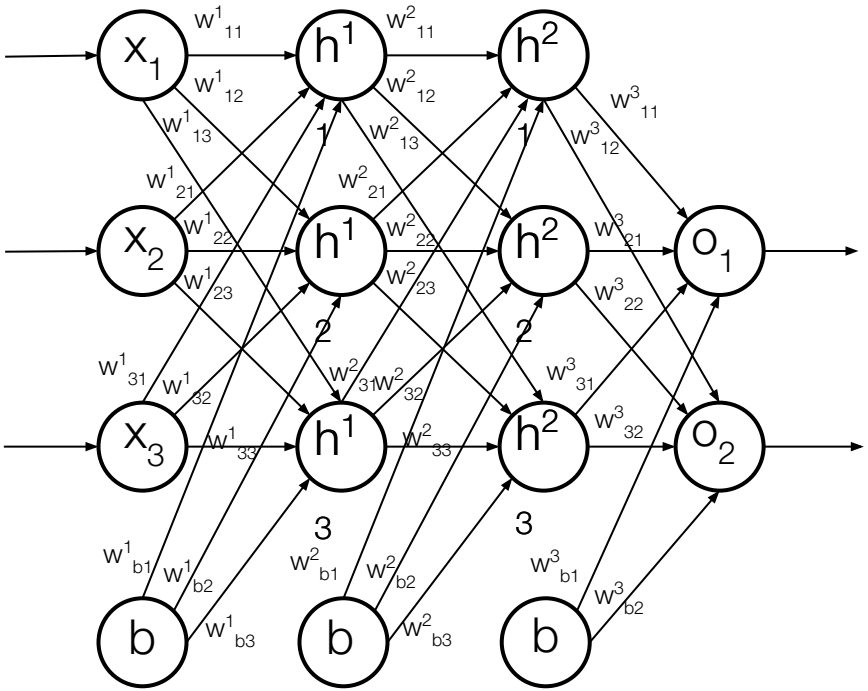
$\mathbf{h^1} = \text{act}(\mathbf{x}\mathbf{W^1})$

s.t. $h^1_1 = x_1 w^1_{11} + x_2 w^1_{21} + x_3 w^1_{31} + b w^1_{b1}$

$\mathbf{h^2} = \text{act}(\mathbf{h^1}\mathbf{W^2})$

s.t. $h^2_1 = h^1_1 w^2_{11} + h^1_2 w^2_{21} + h^1_3 w^2_{31} + b w^2_{b1}$

Neural Networks



$\mathbf{h^1} = \text{act}(\mathbf{xW^1})$

s.t. $h^1_1 = x_1 w^1_{11} + x_2 w^1_{21} + x_3 w^1_{31} + b w^1_{b1}$

$\mathbf{h^2} = \text{act}(\mathbf{h^1 W^2})$

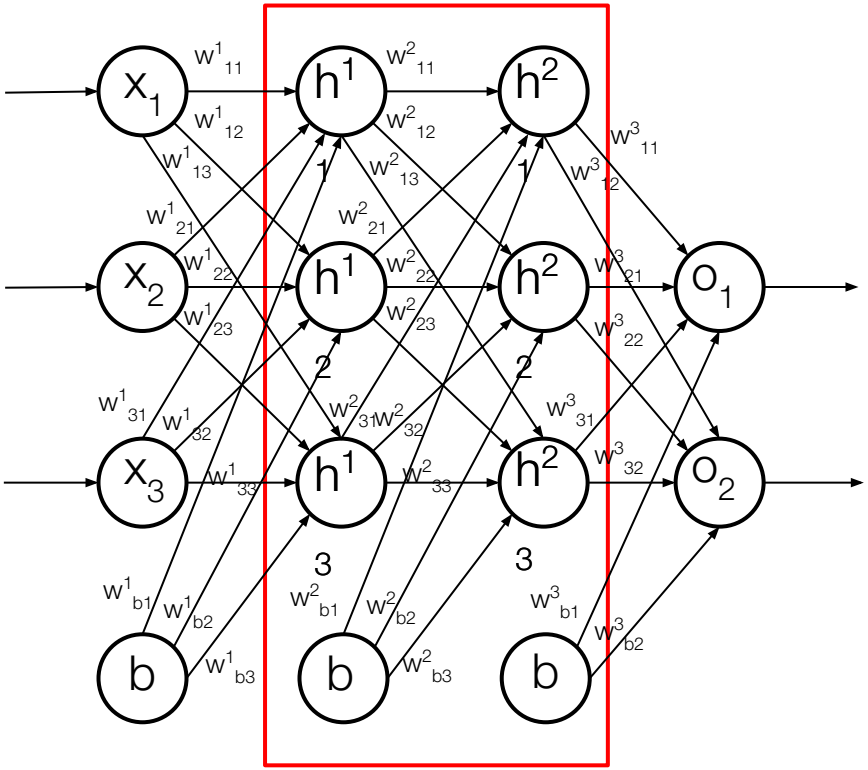
s.t. $h^2_1 = h^1_1 w^2_{11} + h^1_2 w^2_{21} + h^1_3 w^2_{31} + b w^2_{b1}$

$\mathbf{o} = \text{act}(\mathbf{h^2 W^3})$

s.t. $o_1 = h^2_1 w^3_{11} + h^2_2 w^3_{21} + h^2_3 w^3_{31} + b w^3_{b1}$

$\mathbf{o} = \text{act}(\text{act}(\text{act}(\mathbf{xW^1})\mathbf{W^2})\mathbf{W^3})$

Neural Networks



$\mathbf{h^1} = act(\mathbf{xW^1})$

s.t. $h^1_1 = x_1 w^1_{11} + x_2 w^1_{21} + x_3 w^1_{31} + b w^1_{b1}$

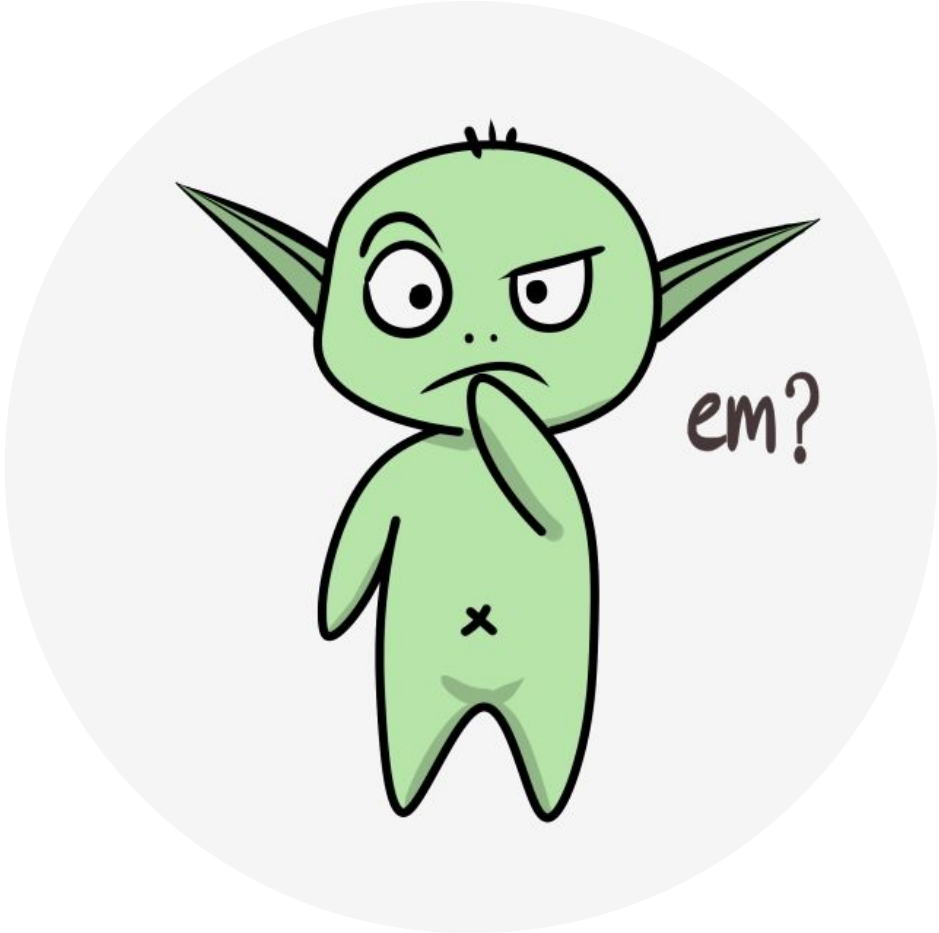
$\mathbf{h^2} = act(\mathbf{h^1 W^2})$

s.t. $h^2_1 = h^1_1 w^2_{11} + h^1_2 w^2_{21} + h^1_3 w^2_{31} + b w^2_{b1}$

$\mathbf{o} = act(\mathbf{h^2 W^3})$

s.t. $o_1 = h^2_1 w^3_{11} + h^2_2 w^3_{21} + h^2_3 w^3_{31} + b w^3_{b1}$

$\mathbf{o} = act(act(act(\mathbf{xW^1})\mathbf{W^2})\mathbf{W^3})$



https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

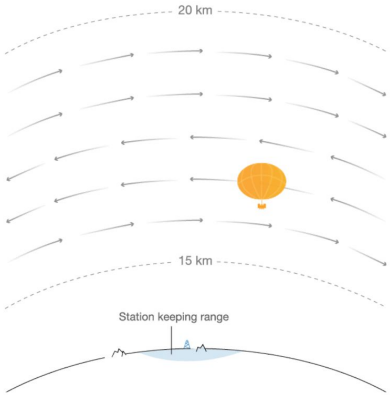
A Note from the Textbook

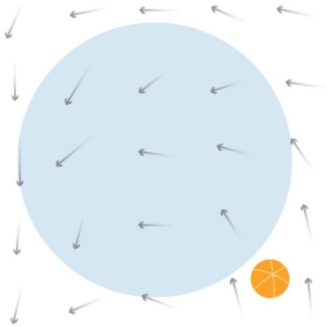
The backpropagation algorithm can produce good results for shallow networks having 1 or 2 hidden layers, but it may not work well for deeper ANNs. In fact, training a network with $k + 1$ hidden layers can actually result in poorer performance than training a network with k hidden layers, even though the deeper network can represent all the functions that the shallower network can (Bengio, 2009). Explaining results like these is not easy, but several factors are important. First, the large number of weights in a typical deep ANN makes it difficult to avoid the problem of overfitting, that is, the problem of failing to generalize correctly to cases on which the network has not been trained. Second, backpropagation does not work well for deep ANNs because the partial derivatives computed by its backward passes either decay rapidly toward the input side of the network, making learning by deep layers extremely slow, or the partial derivatives grow rapidly toward the input side of the network, making learning unstable. Methods

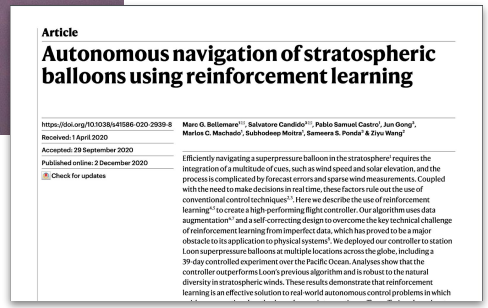
Things change quickly...

What we feed to a neural network still matters (a lot!)

https://www.nature.com/articles/s41586-020-2939-8







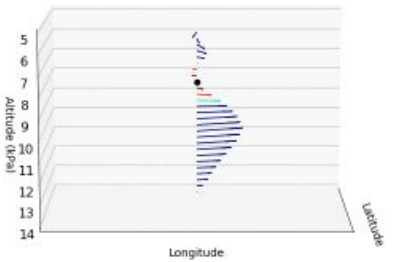
Article

Autonomous navigation of stratospheric balloons using reinforcement learning

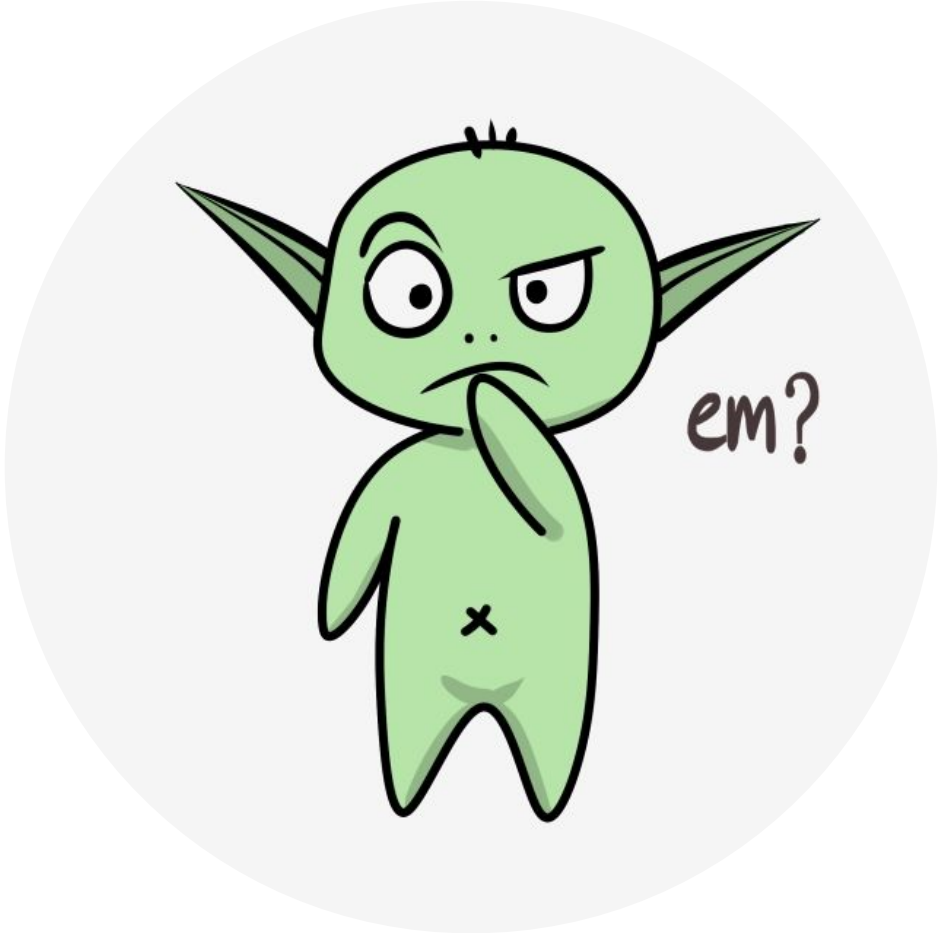
https://doi.org/10.1038/s41586-020-2939-8
Received: 1 April 2020
Accepted: 29 September 2020
Published online: 2 December 2020
Check for updates

Marc G. Bellemare[1,2], Salvatore Candido[1,2], Pablo Samuel Castro[1], Jun Gong[1], Marlos C. Machado[1], Subhodeep Moitra[1], Sameera S. Ponda[1] & Ziyu Wang[1]

Efficiently navigating a superpressure balloon in the stratosphere[1] requires the integration of a multitude of cues, such as wind speed and solar elevation, and the process is complicated by forecast errors and sparse wind measurements. Coupled with the need to make decisions in real time, these factors rule out the use of conventional control techniques[2,3]. Here we describe the use of reinforcement learning[4,5] to create a high-performing flight controller. Our algorithm uses data augmentation[6,7] and a self-correcting design to overcome the key technical challenge of reinforcement learning from imperfect data, which has proved to be a major obstacle to its application to physical systems[8]. We deployed our controller to station Loon superpressure balloons at multiple locations across the globe, including a 39-day controlled experiment over the Pacific Ocean. Analyses show that the controller outperforms Loon's previous algorithm and is robust to the natural diversity in stratospheric winds. These results demonstrate that reinforcement learning is an effective solution to real-world autonomous control problems in which



[Bellemare, Candido, Castro, Gong, Machado, Moitra, Ponda, & Wang, Nature 2020]



https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

Deep Convolutional Network

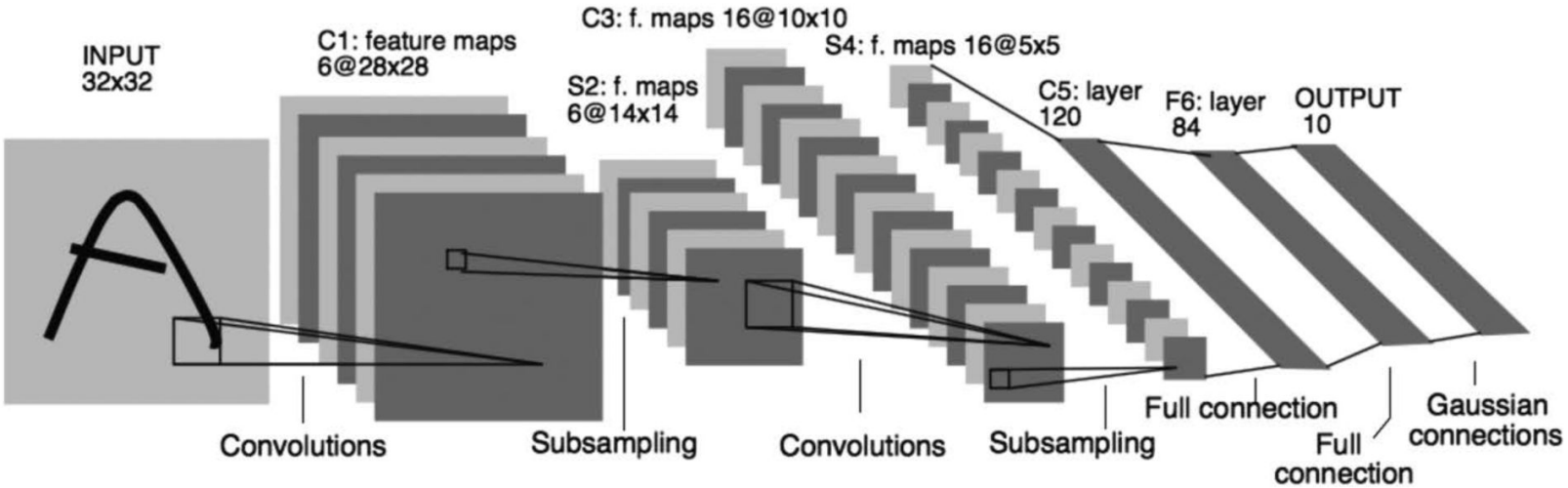
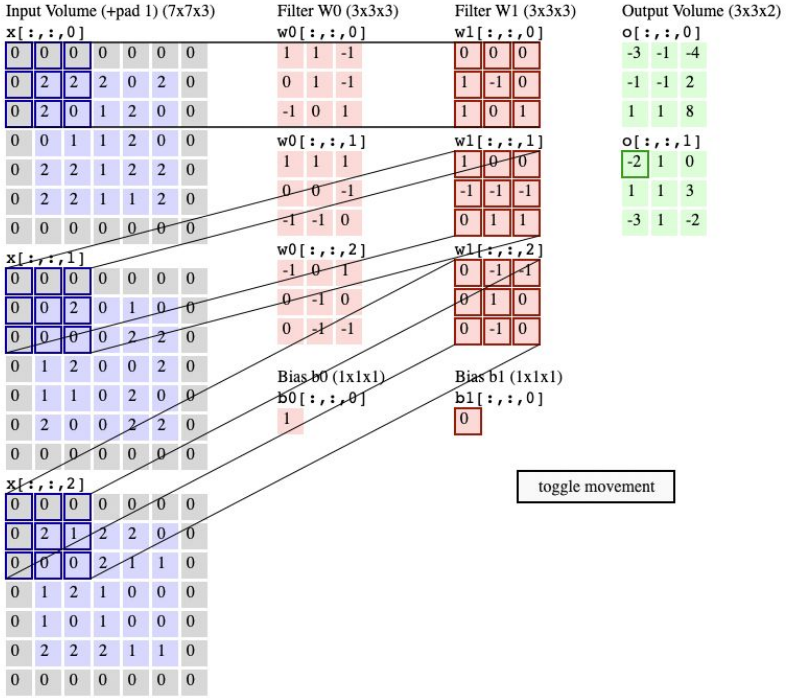


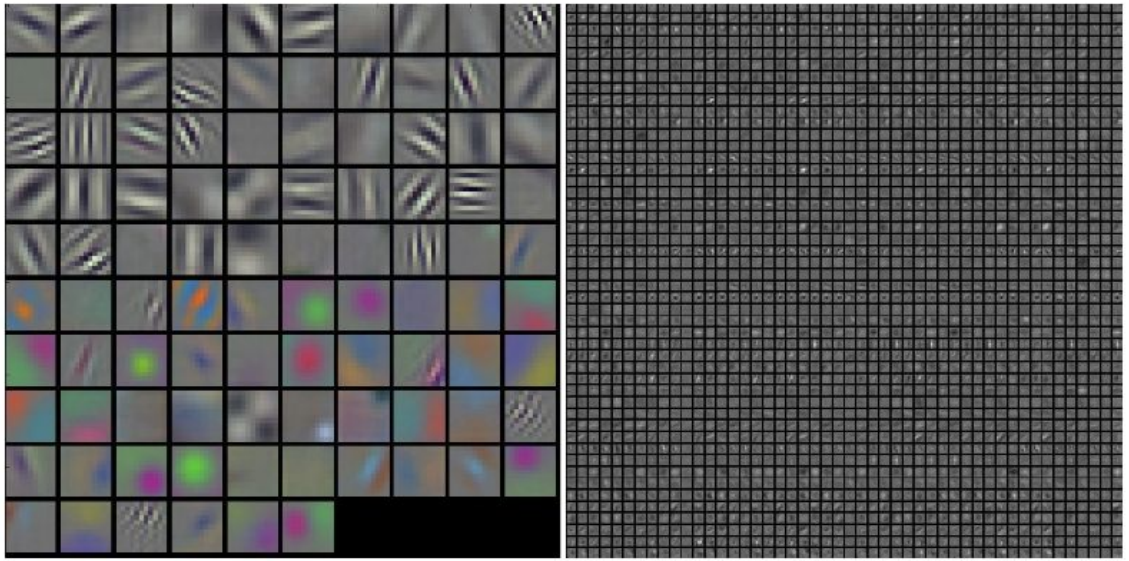
Figure 9.15: Deep Convolutional Network. Republished with permission of Proceedings of the IEEE, from Gradient-based learning applied to document recognition, LeCun, Bottou, Bengio, and Haffner, volume 86, 1998; permission conveyed through Copyright Clearance Center, Inc.

Deep Convolutional Network



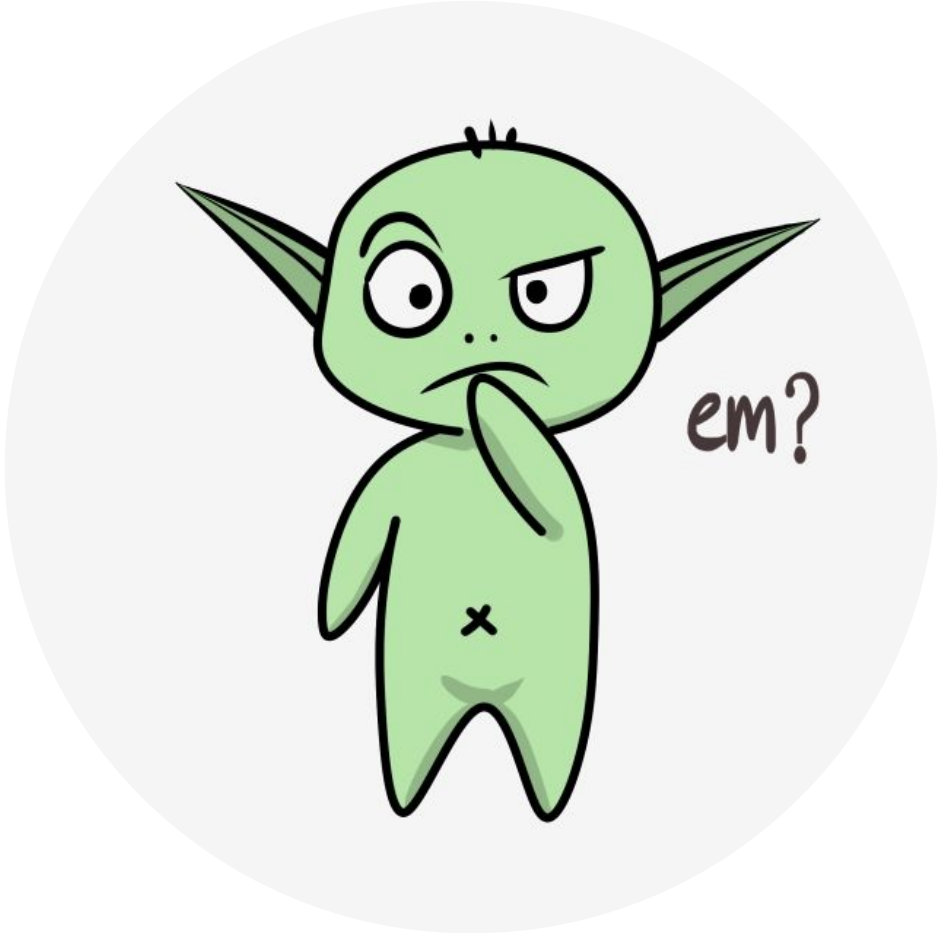
[Figure from demo in https://cs231n.github.io/convolutional-networks/]

Learned Representations



Typical-looking filters on the first CONV layer (left), and the 2nd CONV layer (right) of a trained AlexNet. Notice that the first-layer weights are very nice and smooth, indicating nicely converged network. The color/grayscale features are clustered because the AlexNet contains two separate streams of processing, and an apparent consequence of this architecture is that one stream develops high-frequency grayscale features and the other low-frequency color features. The 2nd CONV layer weights are not as interpretable, but it is apparent that they are still smooth, well-formed, and absent of noisy patterns.

[Figure from https://cs231n.github.io/understanding-cnn/]



https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

Are we improving anything? From features to architecture…

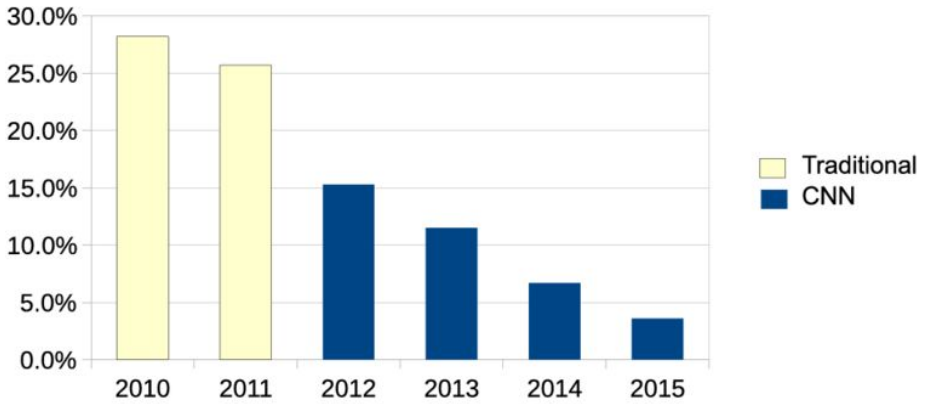
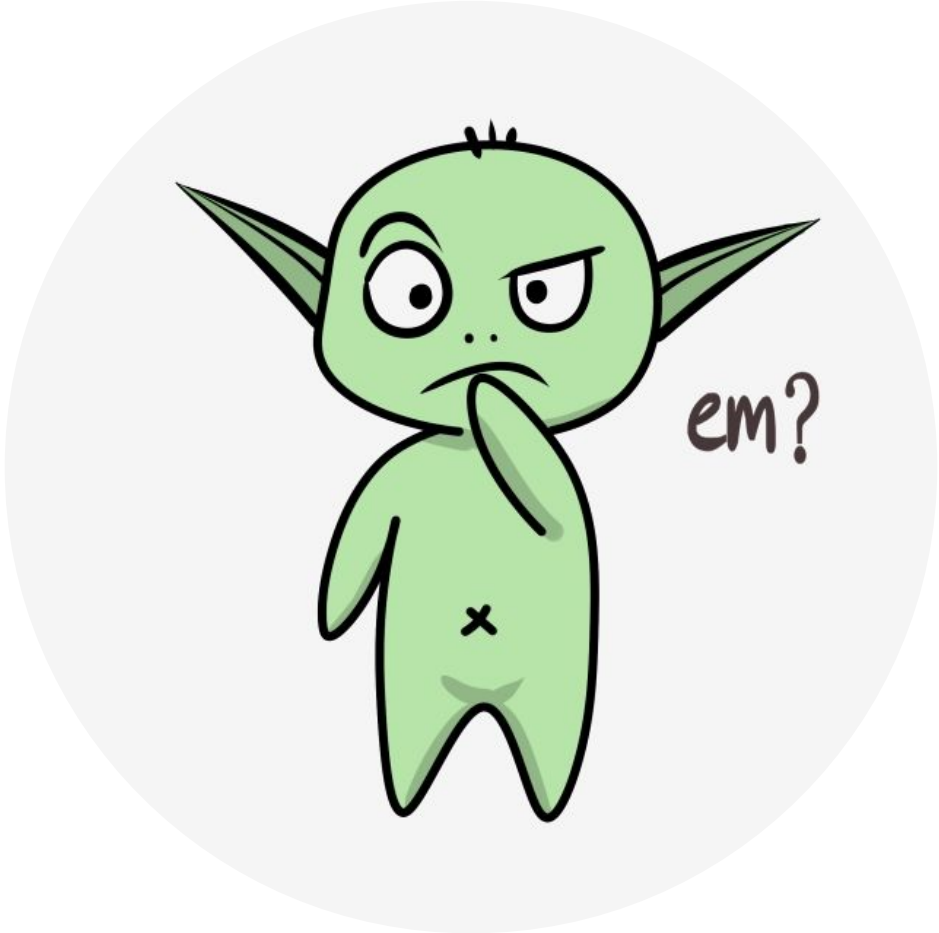


Image by Bottou, Curtis, and Nocedal (2016)



https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

Journal of Artificial Intelligence Research 47 (2013) 253–279 Submitted 02/13; published 06/13

The Arcade Learning Environment: An Evaluation Platform for General Agents

Marc G. Bellemare MG17@CS.UALBERTA.CA
University of Alberta, Edmonton, Alberta, Canada

Yavar Naddaf YAVAR@EMPIRICALRESULTS.CA
Empirical Results Inc., Vancouver, British Columbia, Canada

Joel Veness VENESS@CS.UALBERTA.CA
Michael Bowling BOWLING@CS.UALBERTA.CA
University of Alberta, Edmonton, Alberta, Canada

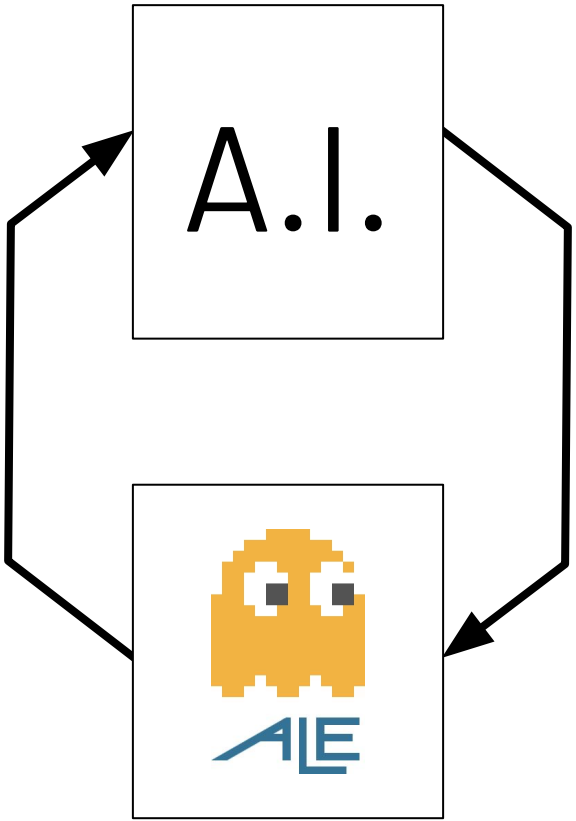
Abstract

In this article we introduce the Arcade Learning Environment (ALE): both a challenge problem and a platform and methodology for evaluating the development of general, domain-independent AI technology. ALE provides an interface to hundreds of Atari 2600 game environments, each one different, interesting, and designed to be a challenge for human players. ALE presents significant research challenges for reinforcement learning, model learning, model-based planning, imitation learning, transfer learning, and intrinsic motivation. Most importantly, it provides a rigorous testbed for evaluating and comparing approaches to these problems. We illustrate the promise of ALE by developing and benchmarking domain-independent agents designed using well-established AI techniques for both reinforcement learning and planning. In doing so, we also propose an evaluation

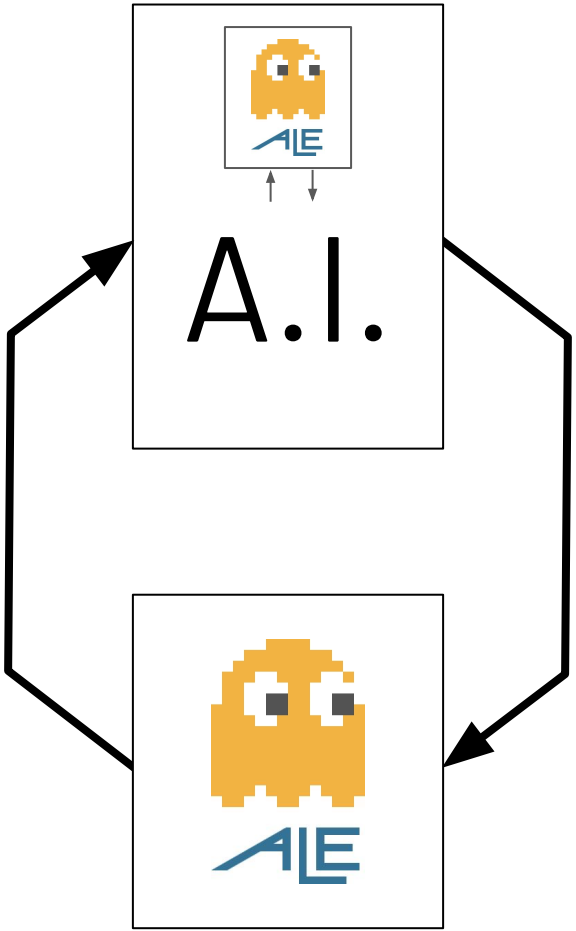
Arcade Learning Environment

Over 50 Domains in 8 Minutes 23 Seconds

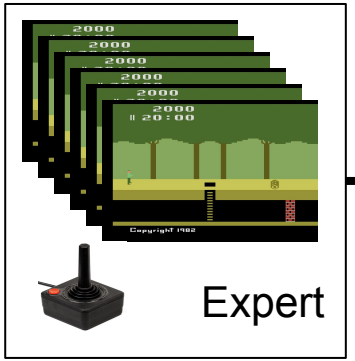
Reinforcement Learning

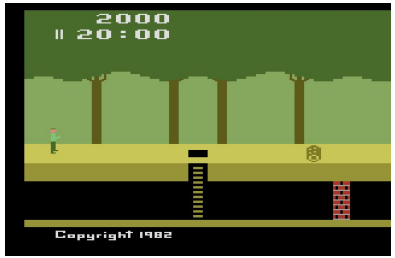


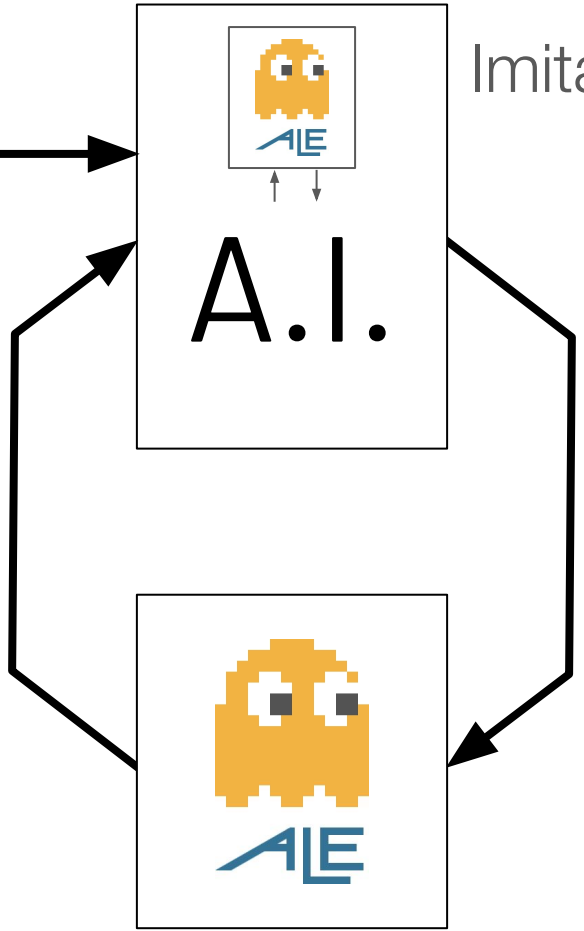
Model Learning



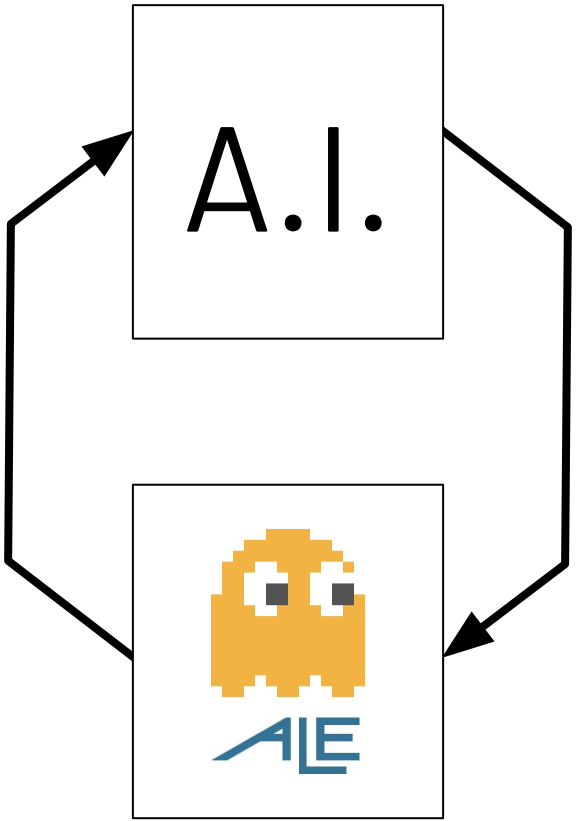
Imitation/Apprenticeship Learning





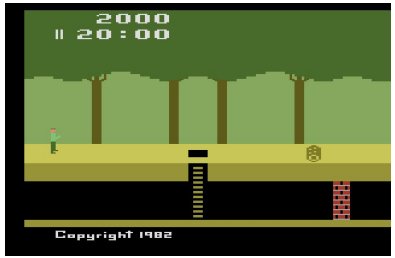


Exploration



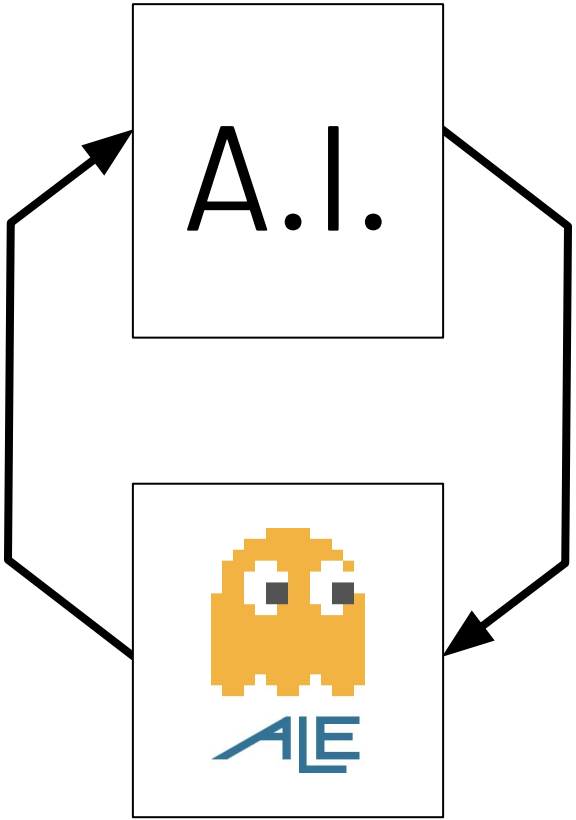
Transfer Learning

Pitfall!

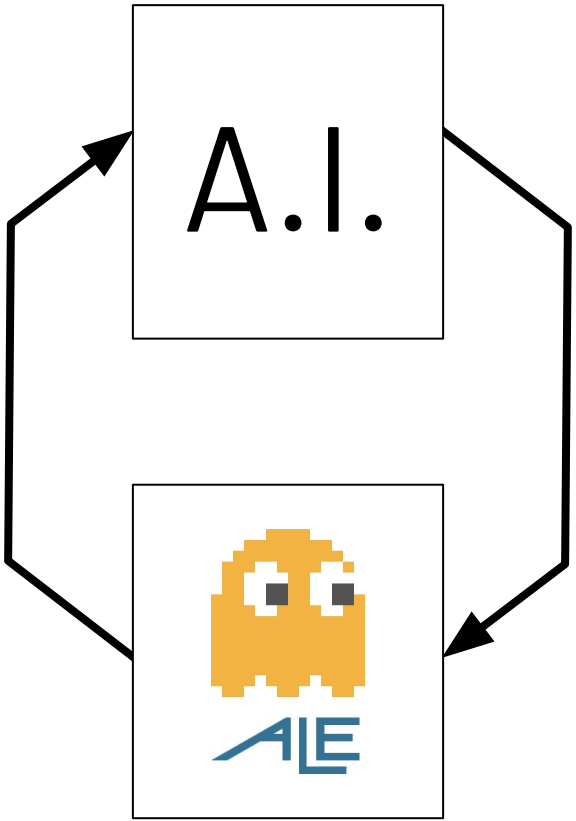


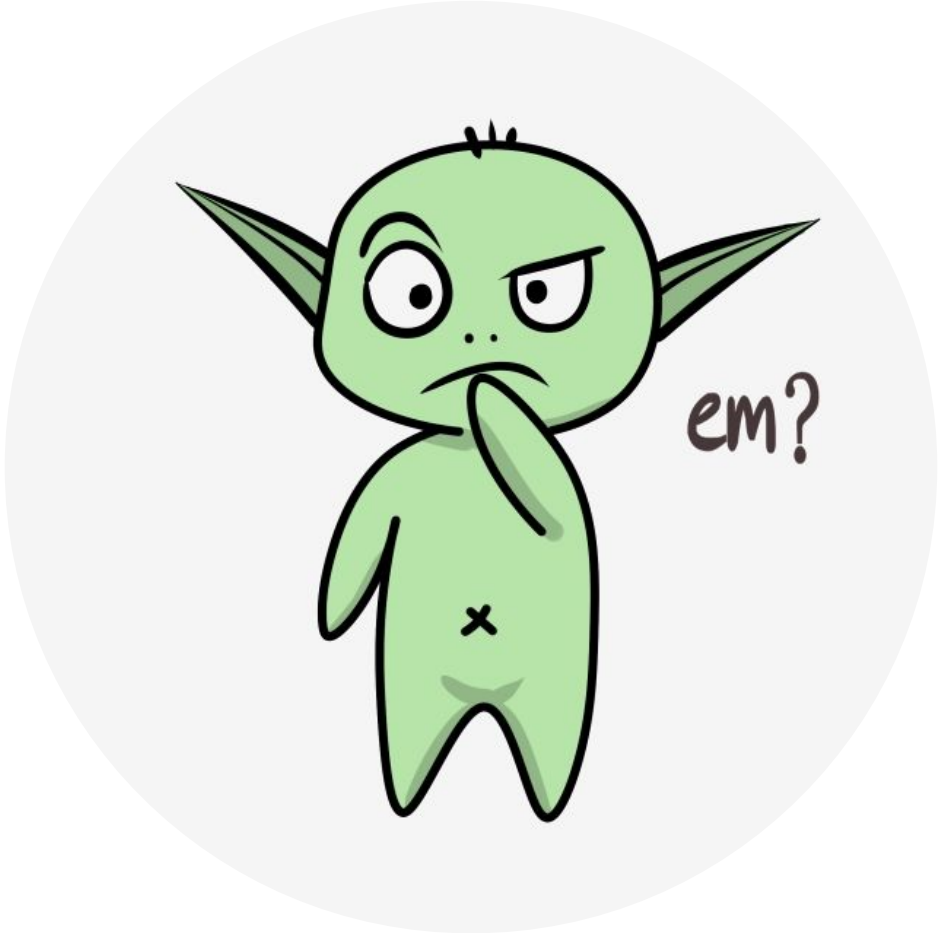
. .
.

Pitfall II



Intrinsic Motivation

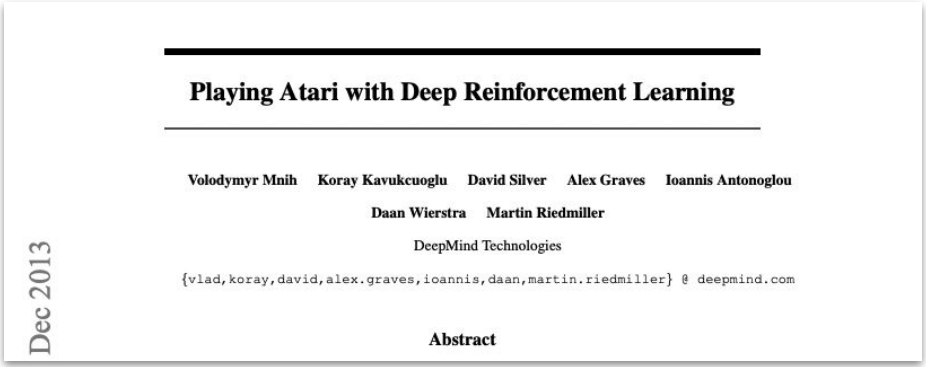




https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

Deep Q-Network (and Deep RL)

[Mnih et al., 2013, 2015]



Dec 2013

Playing Atari with Deep Reinforcement Learning

Volodymyr Mnih Koray Kavukcuoglu David Silver Alex Graves Ioannis Antonoglou

Daan Wierstra Martin Riedmiller

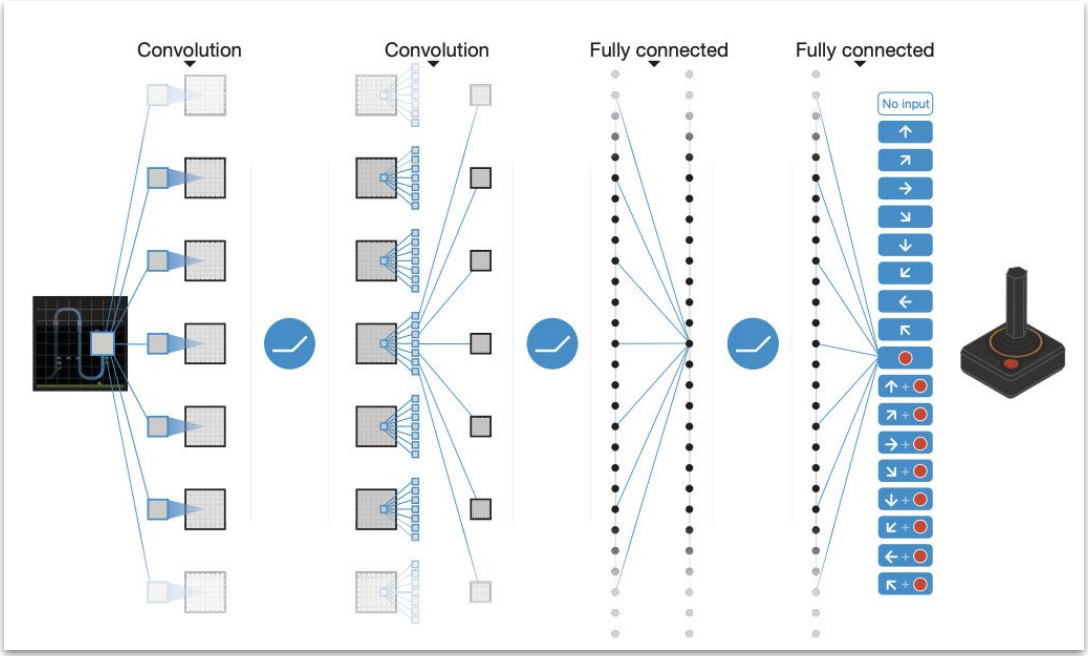
DeepMind Technologies

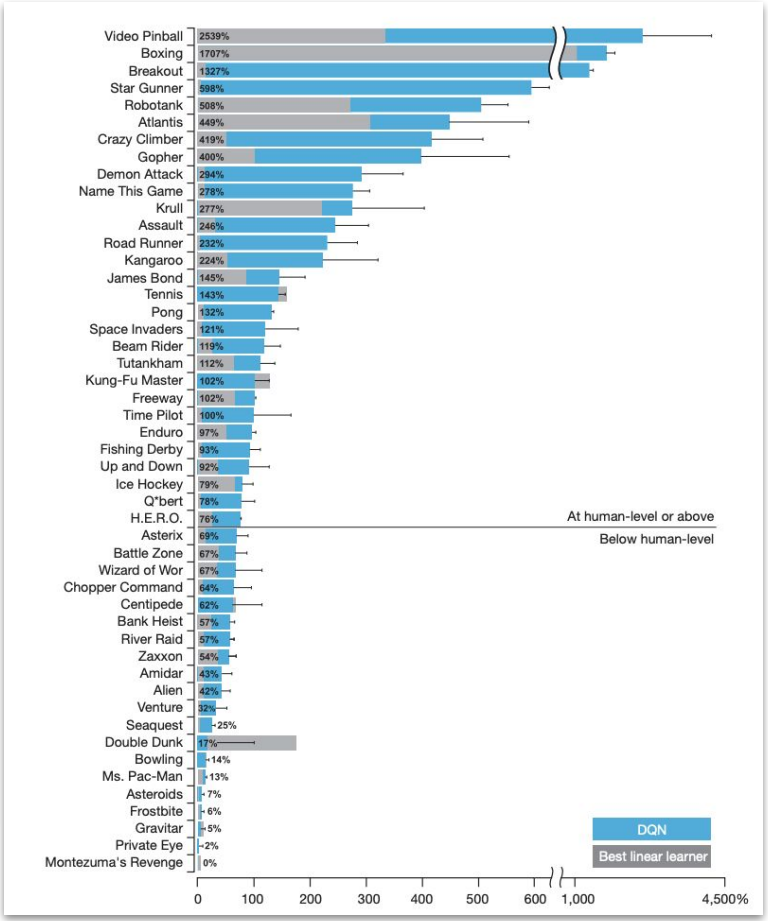
{vlad,koray,david,alex.graves,ioannis,daan,martin.riedmiller} @ deepmind.com

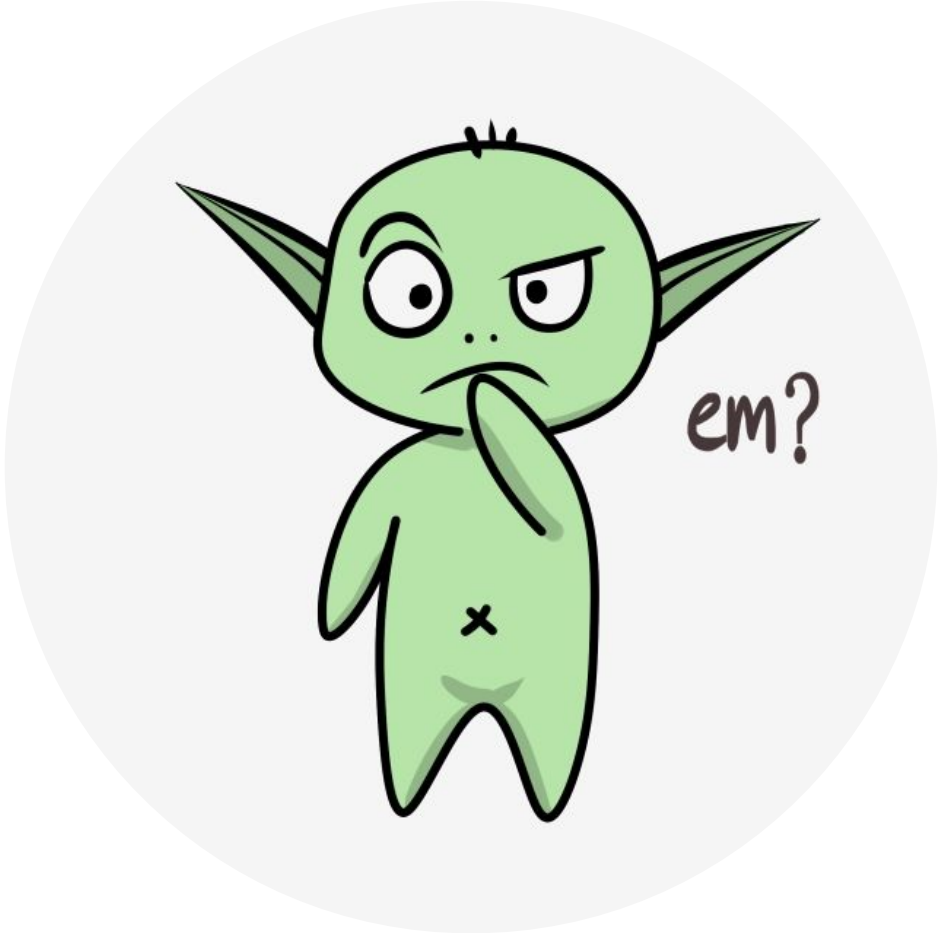
Abstract

Deep Q-Network (and Deep RL)

[Mnih et al., 2013, 2015]







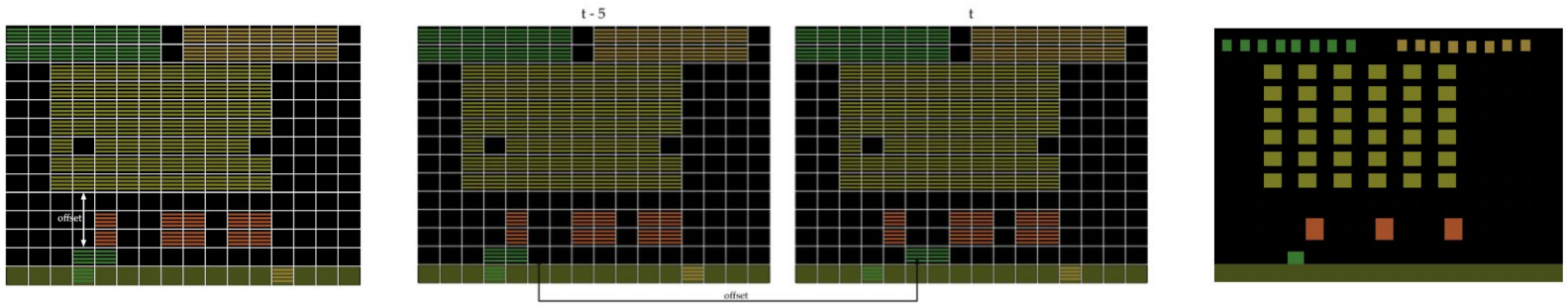
https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

What if we were to design features instead?

What if we were to design features instead?

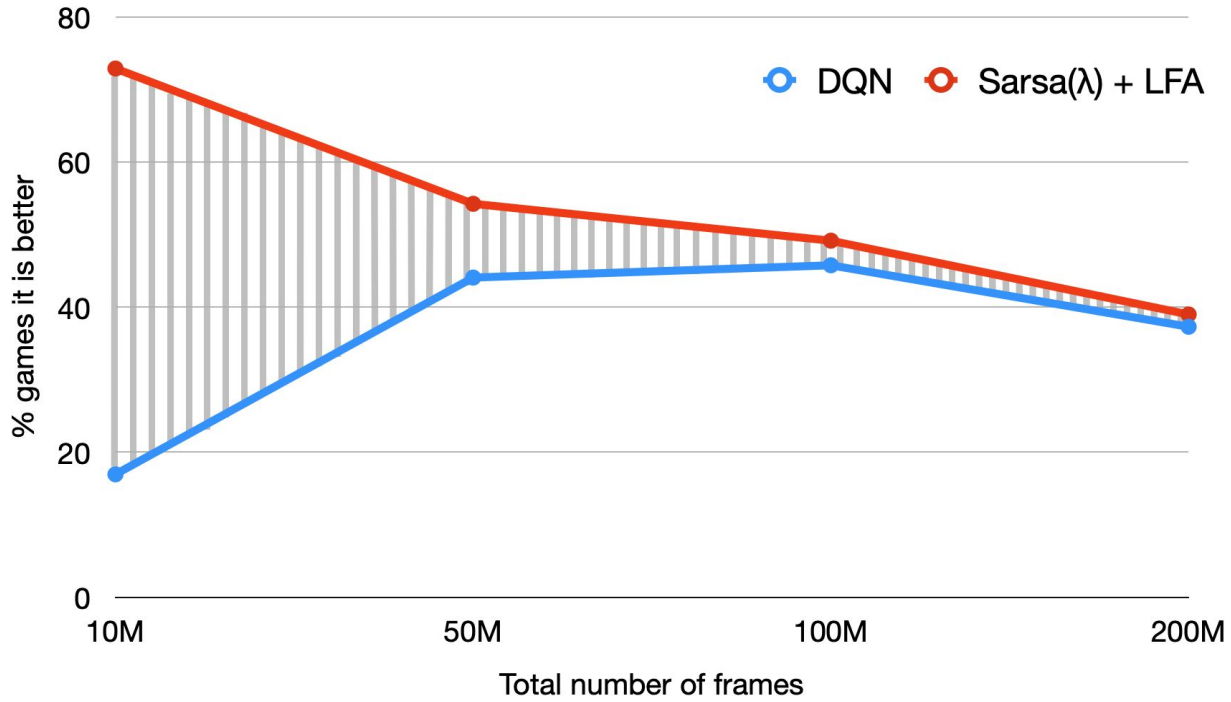
State of the Art Control of Atari Games Using Shallow Reinforcement Learning

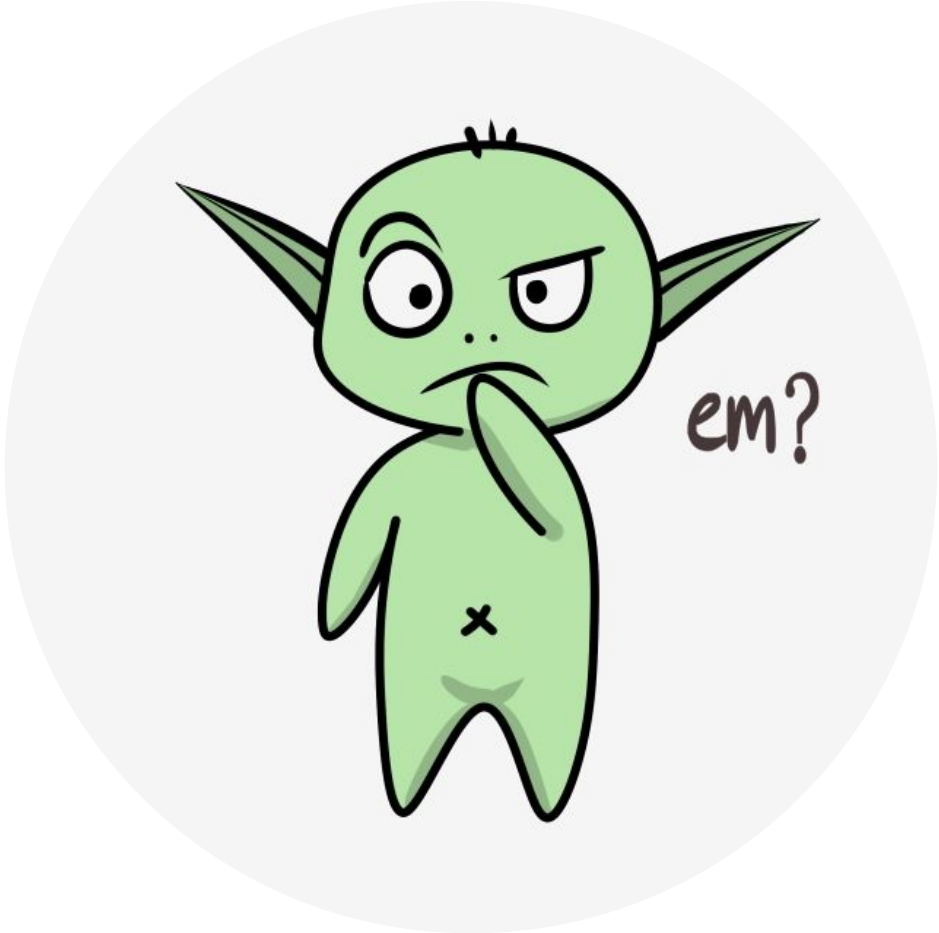
Yitao Liang[†], Marlos C. Machado[‡], Erik Talvitie[†], and Michael Bowling[‡]
[†]Franklin & Marshall College
Lancaster, PA, USA
{yliang, erik.talvitie}@fandm.edu
[‡]University of Alberta
Edmonton, AB, Canada
{machado, mbowling}@ualberta.ca



There are many trade-offs, we need to understand them

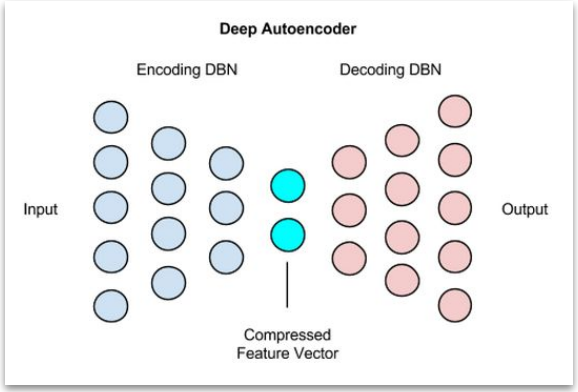
[Liang et al., 2016; Machado et al. 2018]





https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

There are many many inductive biases one can use



https://wiki.pathmind.com/deep-autoencoder

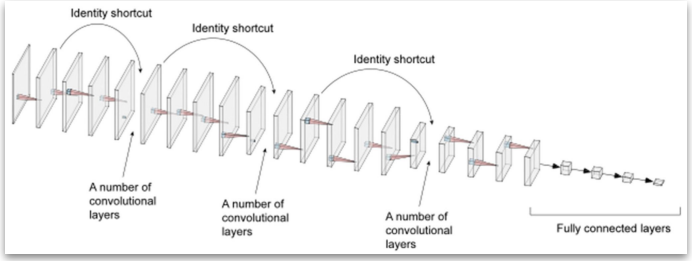
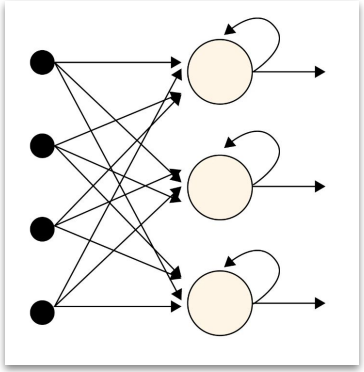


Image by Yu, Miao, and Wang (2022)



https://www.scaler.com/topics/deep-learning/rnn/

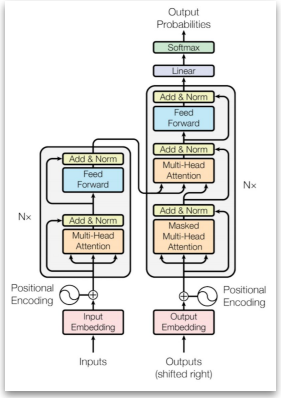


Image by Vaswani et al. (2017)