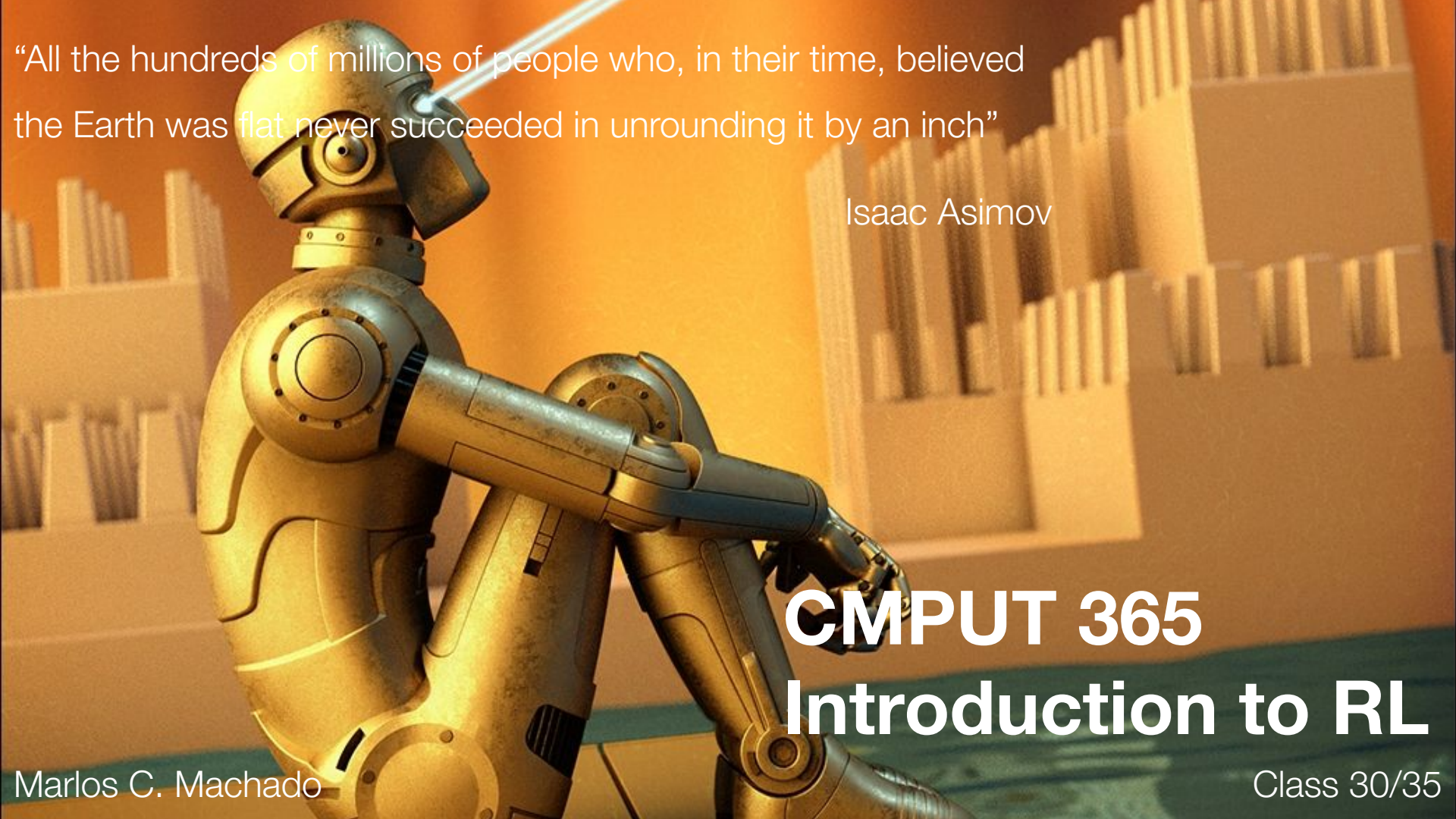

“All the hundreds of millions of people who, in their time, believed
the Earth was flat never succeeded in unrounding it by an inch”
Isaac Asimov
CMPUT 365
Introduction to RL
Marlos C. Machado
Class 30/35

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`.

Reminder II / Updates

- The practice quiz for “Constructing features for prediction” is due today.
- The Student Perspectives of Teaching (SPOT) Survey is now available.
- Rich Sutton will give a guest lecture Dec 1st (this Friday). Spread the word.

Please, interrupt me at any time!

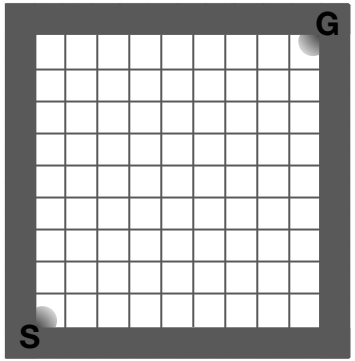
https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

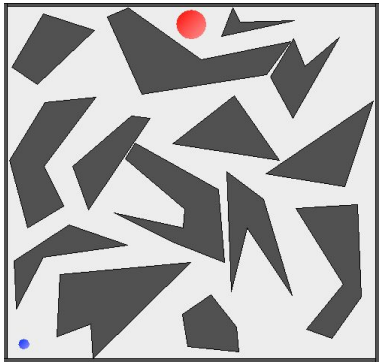
Feature Construction for Linear Methods

- 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…

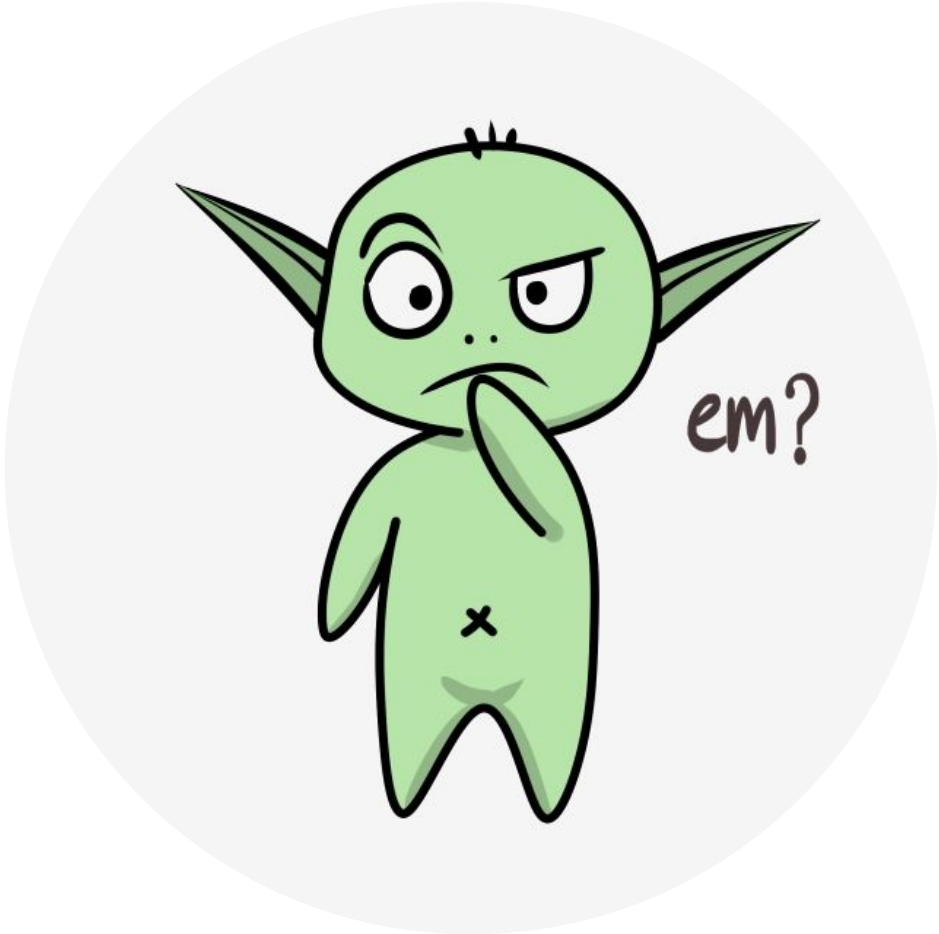
$$\mathbf{x}(s) = (1, s_1, s_2, s_1 s_2, s_1^2, s_2^2, s_1 s_2^2, s_1^2 s_2, s_1^2 s_2^2)^\top$$

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

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

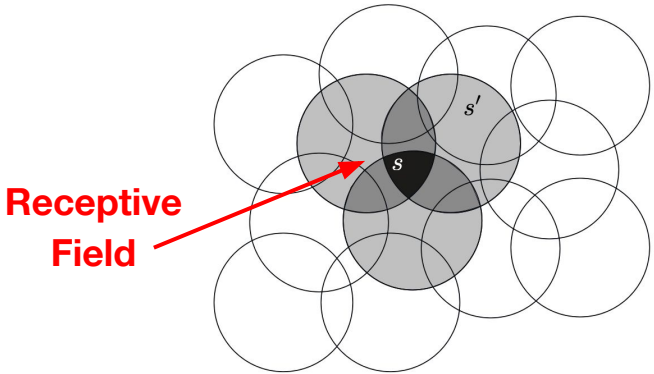
where each $c_{i,j}$ is an integer in the set $\{0, 1, \ldots, n\}$ for an integer $n \geq 0$. These features make up the order-n polynomial basis for dimension k, which contains $(n+1)^k$ different features.



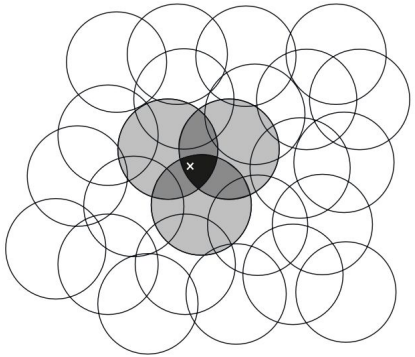
https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

Coarse Coding

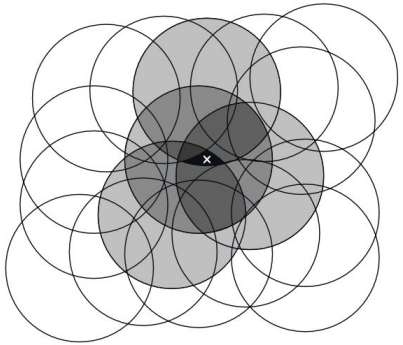
- Consider a task in which the natural representation of the state set is a continuous two- dimensional space.
- We define binary features indicating whether a state is present or not in a specific circle.



The shape defines generalization



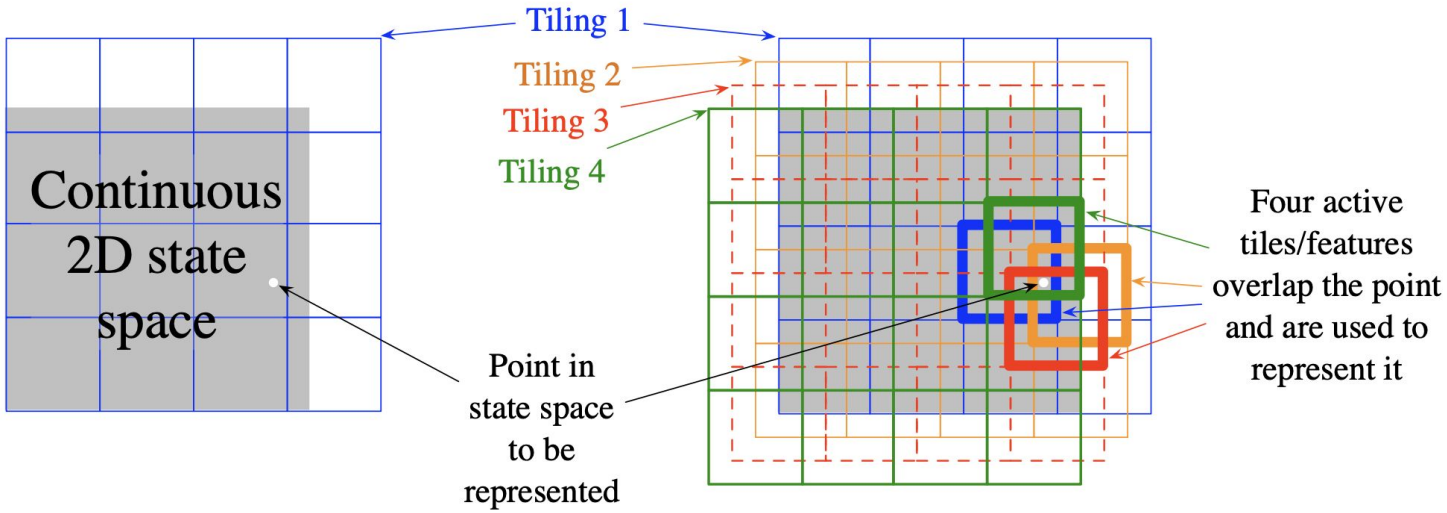
Narrow generalization



Broad generalization

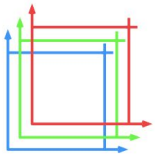
Tile Coding

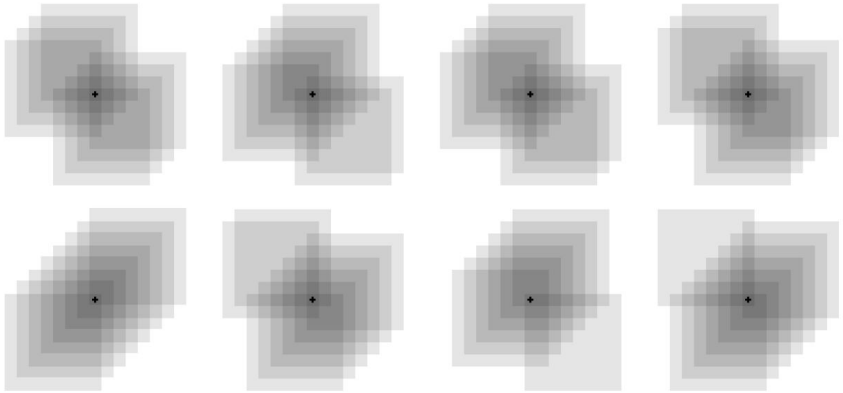
- Tile coding is a form of coarse coding for multi-dimensional continuous spaces (with a fixed number of active features per timestep).



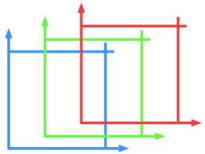
Tile Coding

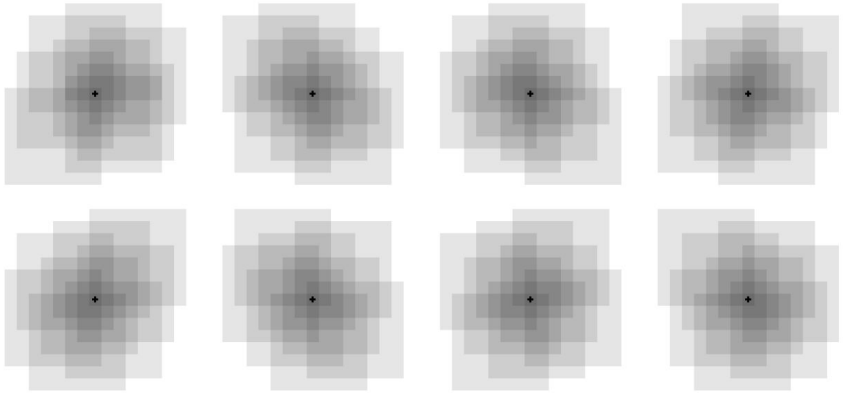
Possible generalizations for uniformly offset tilings

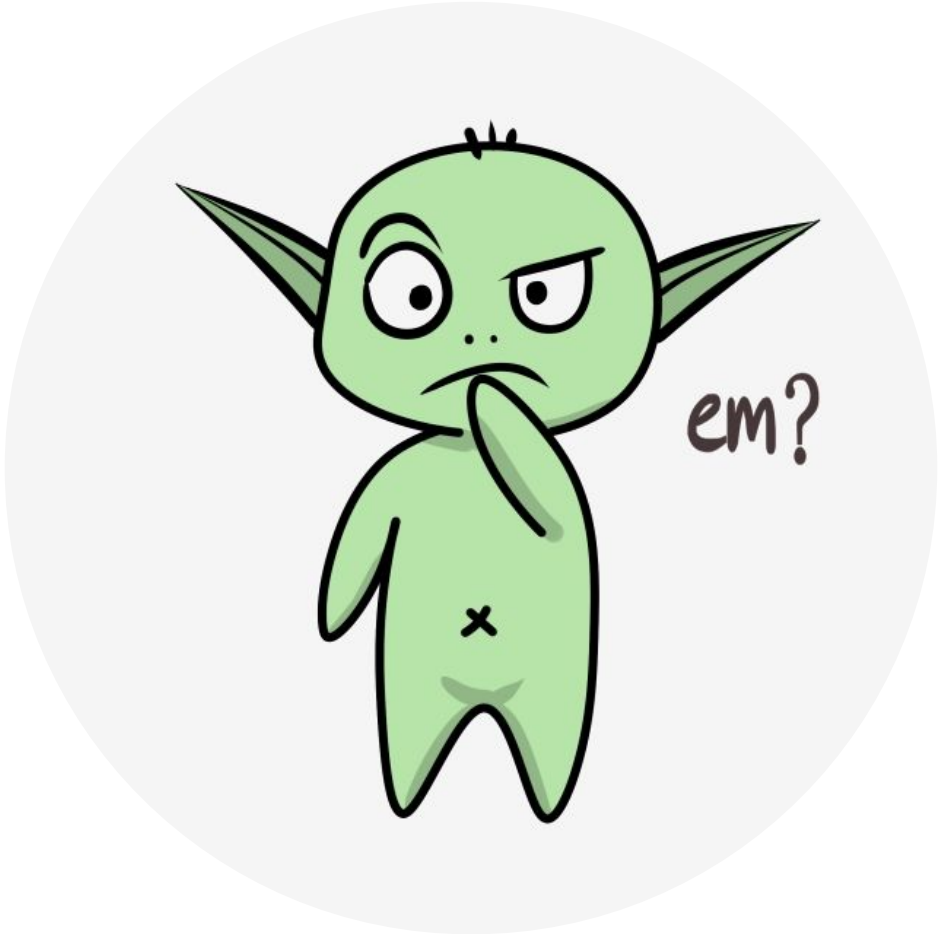




Possible generalizations for asymmetrically offset tilings





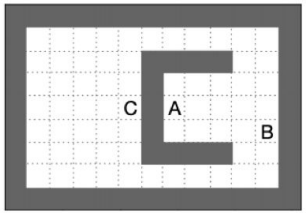


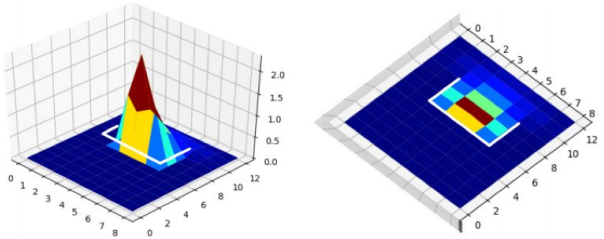
https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

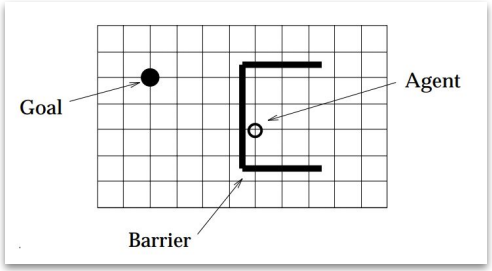
It Isn’t that We do Function Approximation Because We Cannot do Tabular Reinforcement Learning

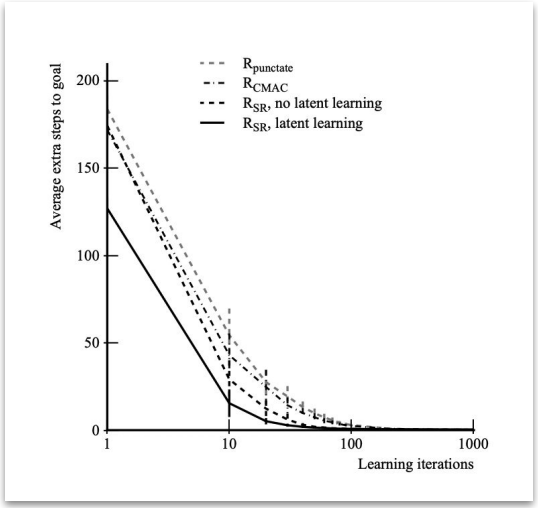
- Successor Representation [Dayan, Neural Computation 1993].

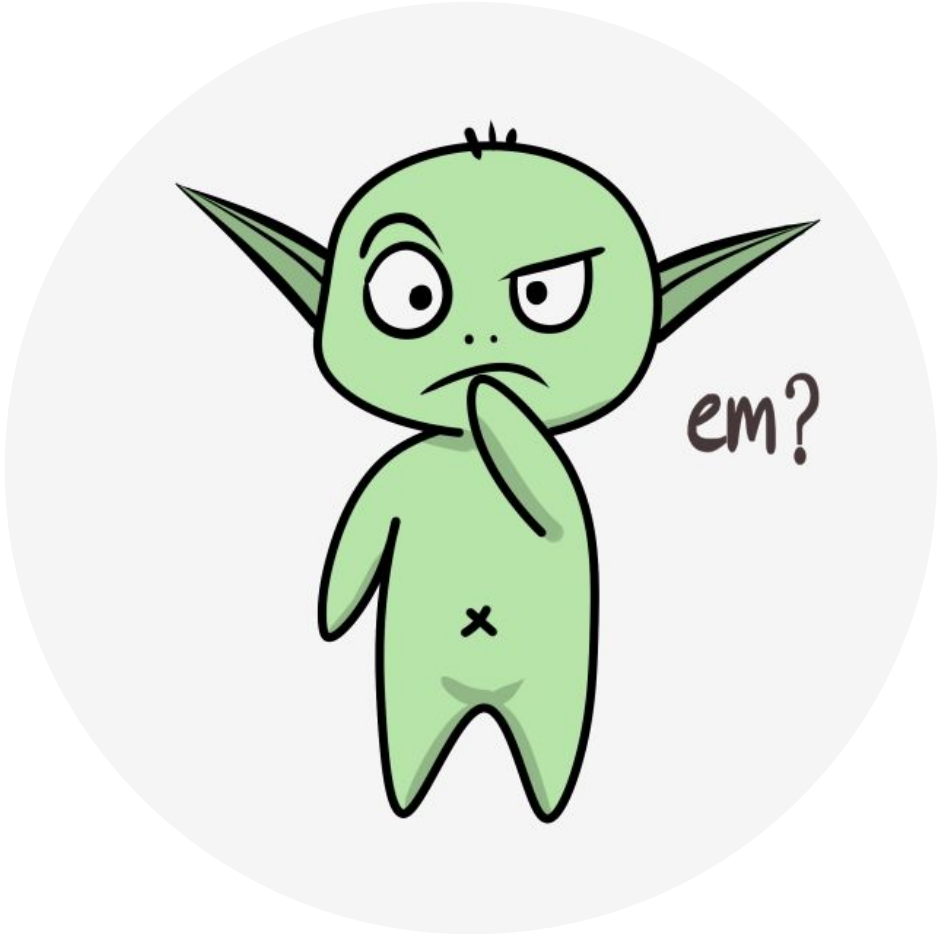
$$\Psi_\pi(s, s') = \mathbb{E}_\pi \left[\sum_t \gamma^t \mathbf{1}_{S_t = s'} | S_0 = s \right]$$









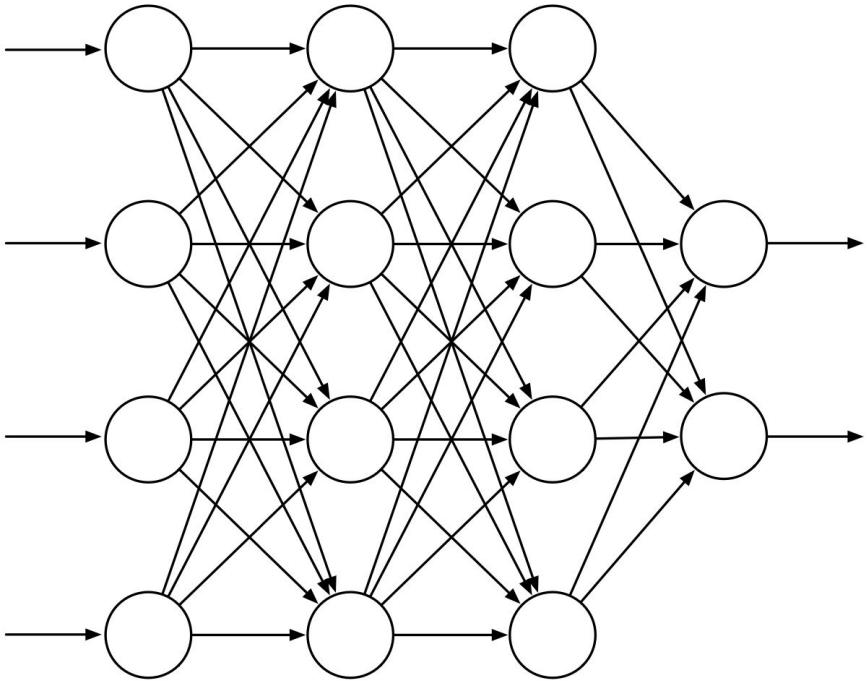


https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

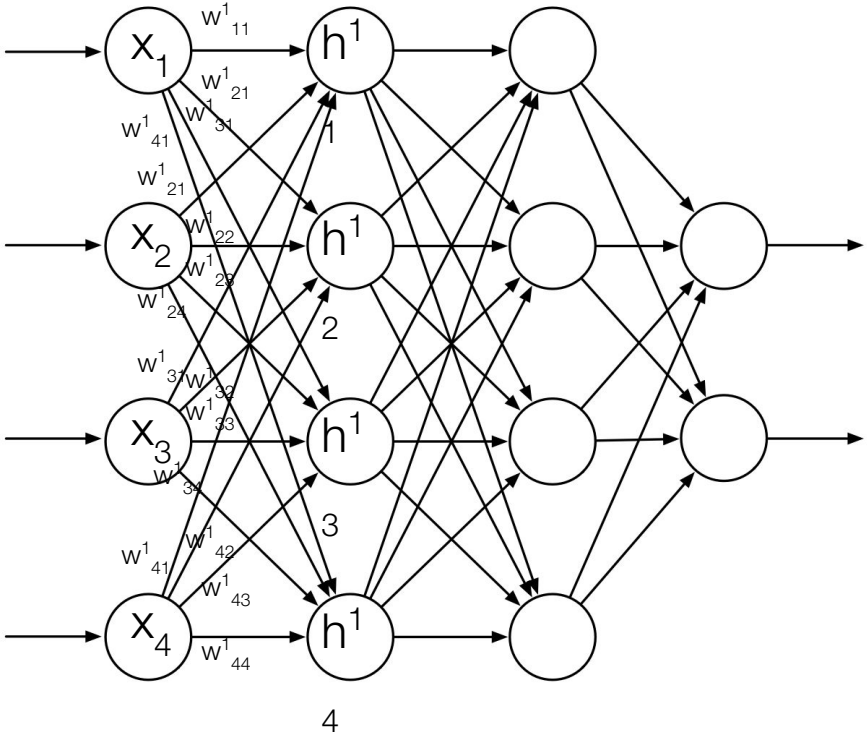
Nonlinear Function Approximation: Artificial Neural Networks

- 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



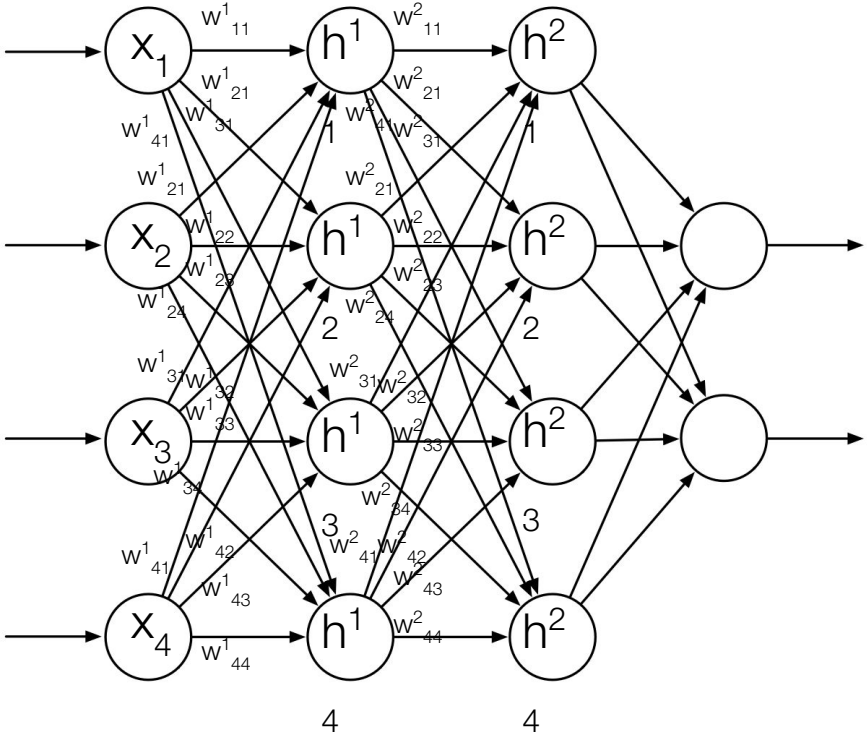
The activation function introduces non-linearity

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

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

s.t. $h^1_1 = x_1 w^1_{11} + x_2 w^1_{21} + x_3 w^1_{31} + x_4 w^1_{41} + B^1$

Neural Networks



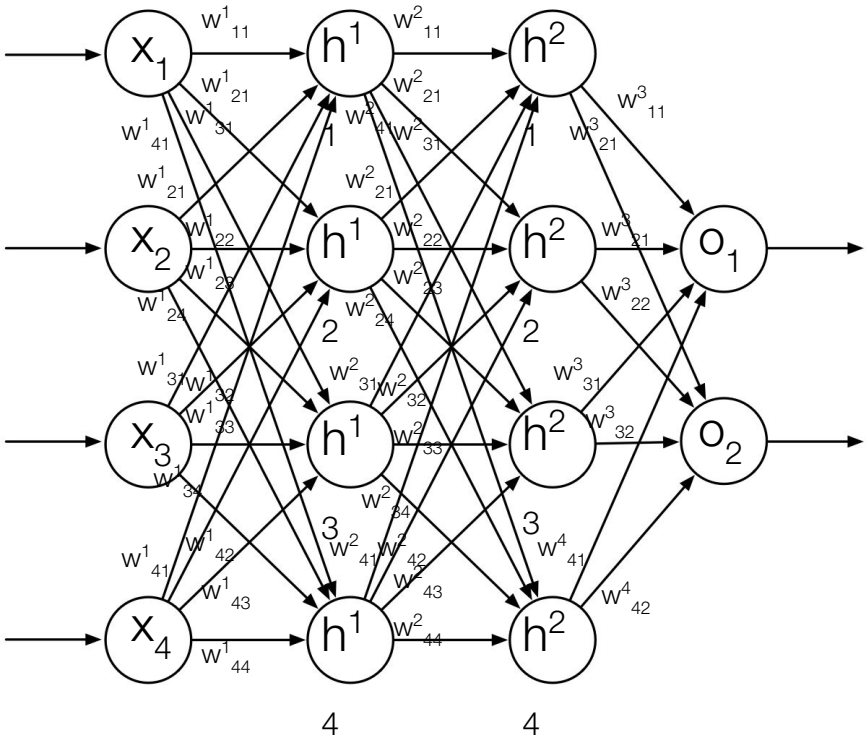
$\mathbf{h^1} = \text{act}(\mathbf{xW^1} + \mathbf{b^1})$

s.t. $h^1_1 = x_1 w^1_{11} + x_2 w^1_{21} + x_3 w^1_{31} + x_4 w^1_{41} + B^1$

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

s.t. $h^2_1 = h^1_1 w^2_{11} + h^1_2 w^2_{21} + h^1_3 w^2_{31} + h^1_4 w^2_{41} + B^2$

Neural Networks



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

s.t. $h^1_1 = x_1 w^1_{11} + x_2 w^1_{21} + x_3 w^1_{31} + x_4 w^1_{41} + B^1$

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

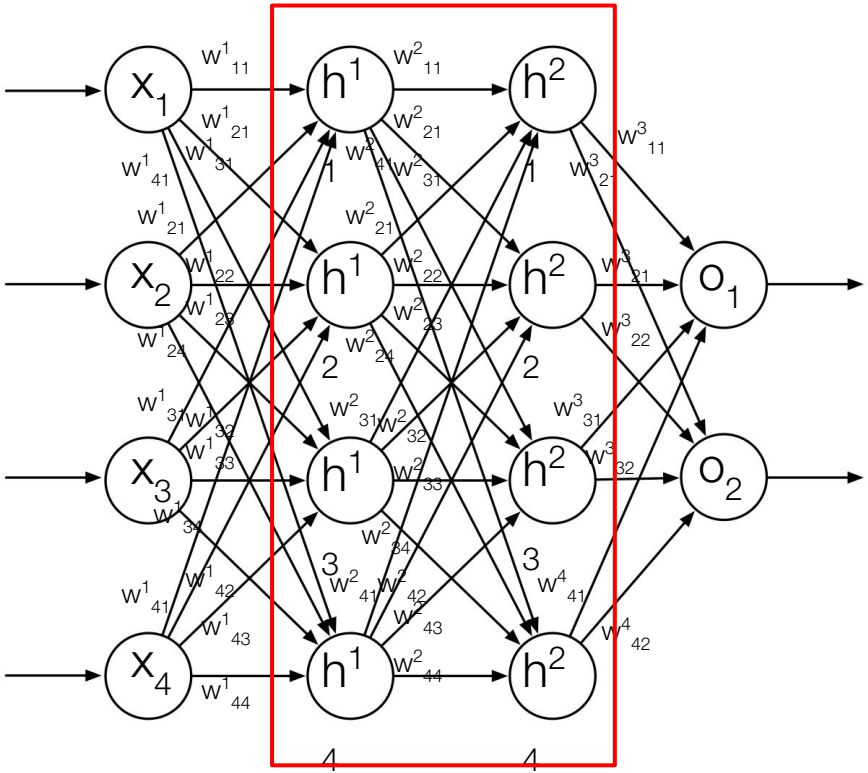
s.t. $h^2_1 = h^1_1 w^2_{11} + h^1_2 w^2_{21} + h^1_3 w^2_{31} + h^1_4 w^2_{41} + B^2$

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

s.t. $o_1 = h^2_1 w^3_{11} + h^2_2 w^3_{21} + h^2_3 w^3_{31} + h^2_4 w^3_{41} + B^2$

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

Neural Networks



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

s.t. $h^1_1 = x_1 w^1_{11} + x_2 w^1_{21} + x_3 w^1_{31} + x_4 w^1_{41} + B^1$

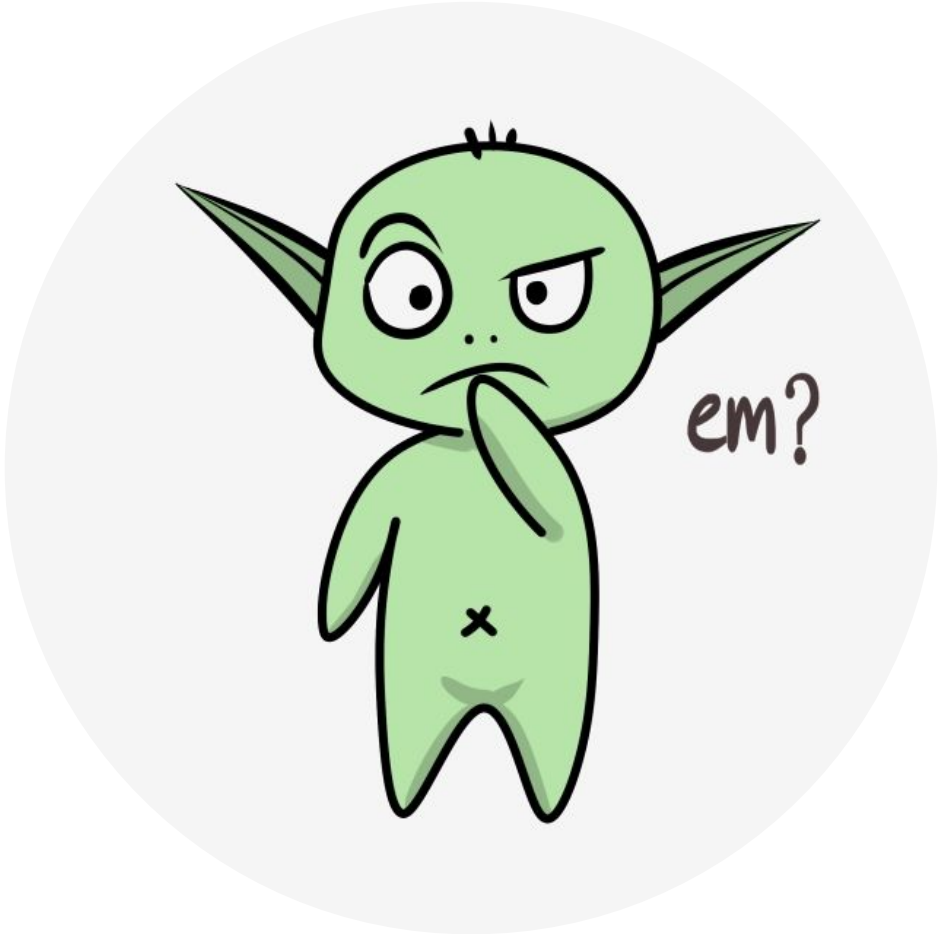
$\mathbf{h^2} = act(\mathbf{h^1 W^2} + \mathbf{b^2})$

s.t. $h^2_1 = h^1_1 w^2_{11} + h^1_2 w^2_{21} + h^1_3 w^2_{31} + h^1_4 w^2_{41} + B^2$

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

s.t. $o_1 = h^2_1 w^3_{11} + h^2_2 w^3_{21} + h^2_3 w^3_{31} + h^2_4 w^3_{41} + B^2$

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



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...

Deep Convolutional Network

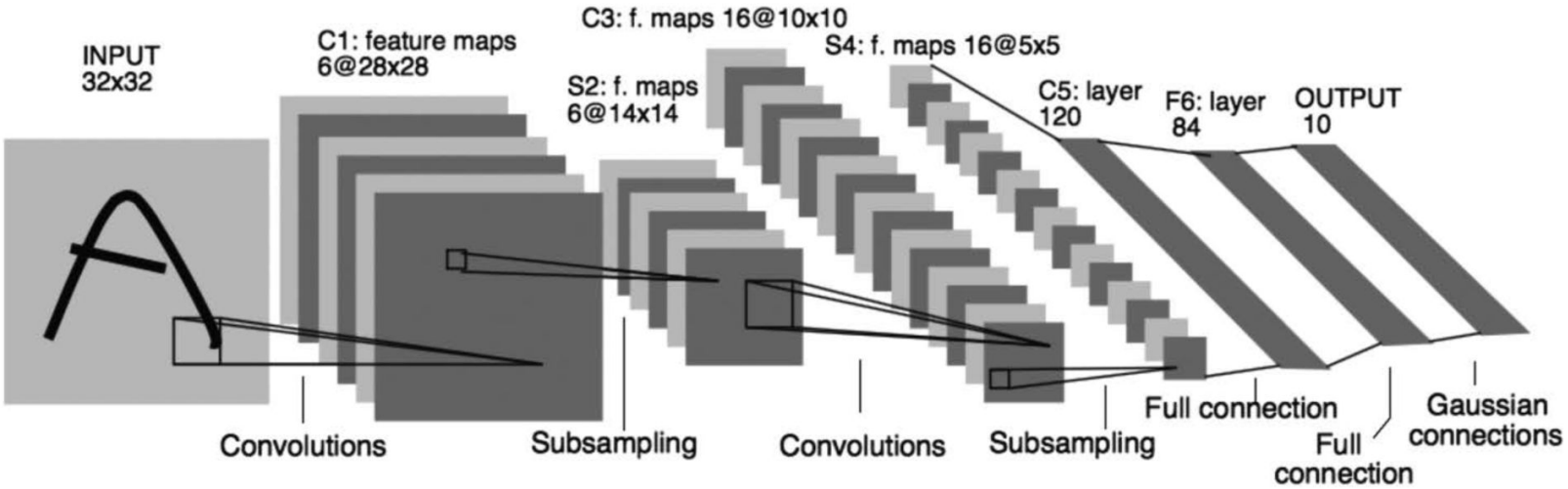
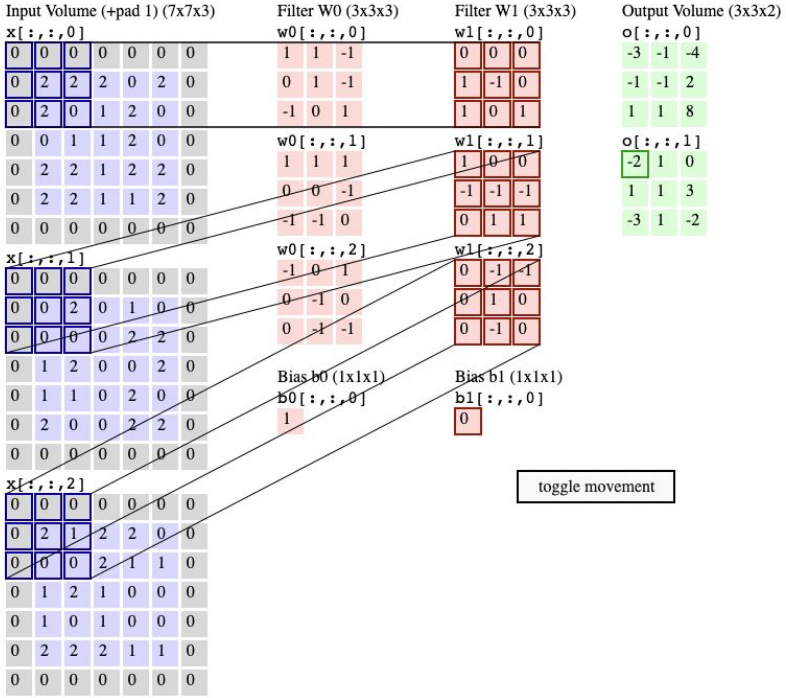


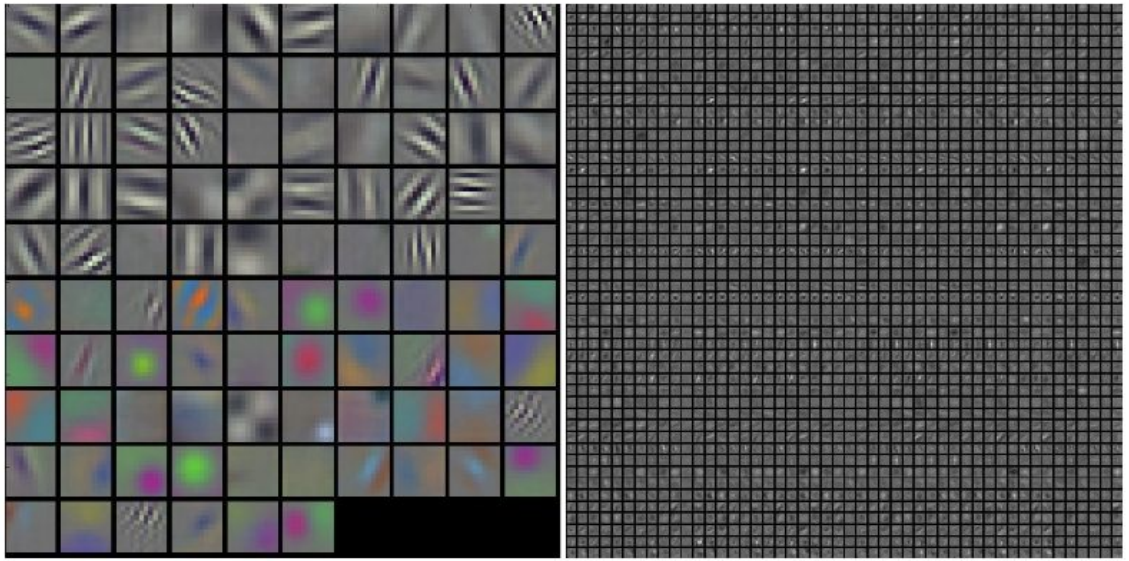
Figure 9.15: Deep Convolutional Network. Republished with permission of Proceedings of the IEEE, from Gradient-based learning applied to document recognition, LeCun, Bottou, Bengio, and Haffner, volume 86, 1998; permission conveyed through Copyright Clearance Center, Inc.

Deep Convolutional Network



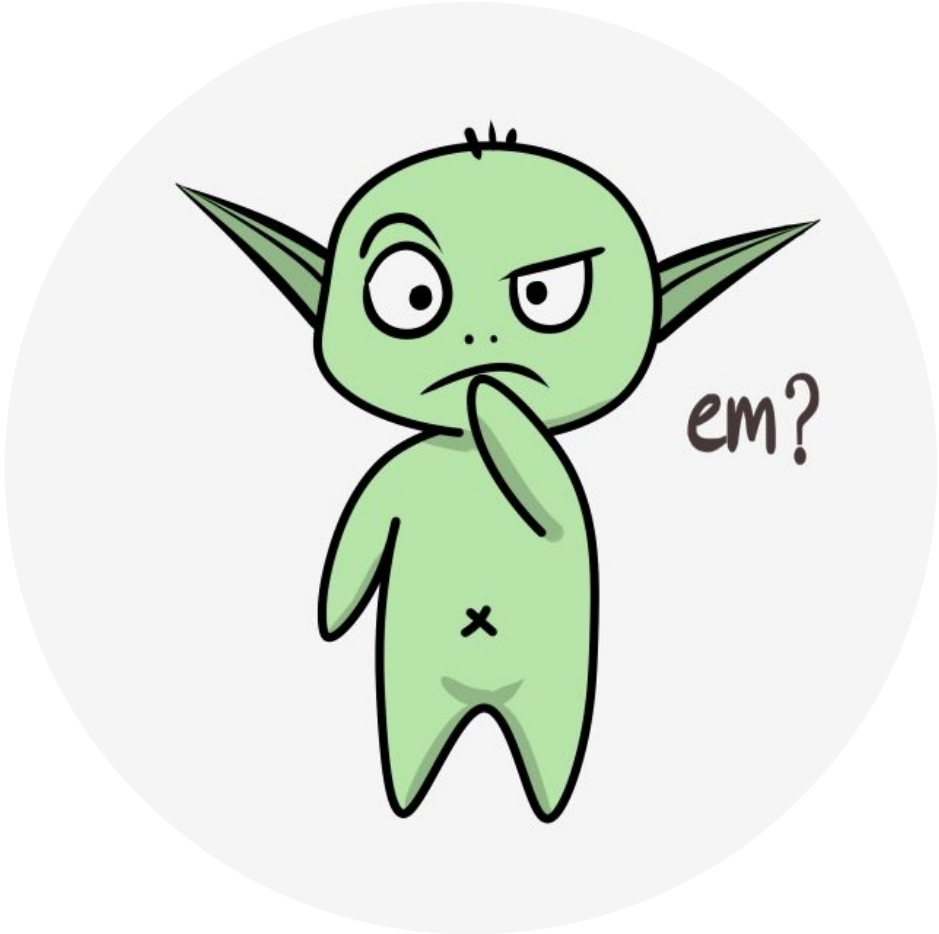
[Figure from demo in https://cs231n.github.io/convolutional-networks/]

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