Lesson 4: Neural Q-Learning

Lesson 4: Where are we?

We are here

Established
(30 years old)

Tabular Methods

Very theoretically justified, strong convergence guarantees, well understood. Very limited in terms of applicability. Empirically poor performance on complex problems.

Developing
(15 years old)

Neural Q-Learning

The “bridge” between Tabular Methods and DQN. Integrates basic Neural Networks with Reinforcement Learning. Idea is old, performance only been demonstrated recently.

Recent
(Last 5 years)

Deep Q-Learning

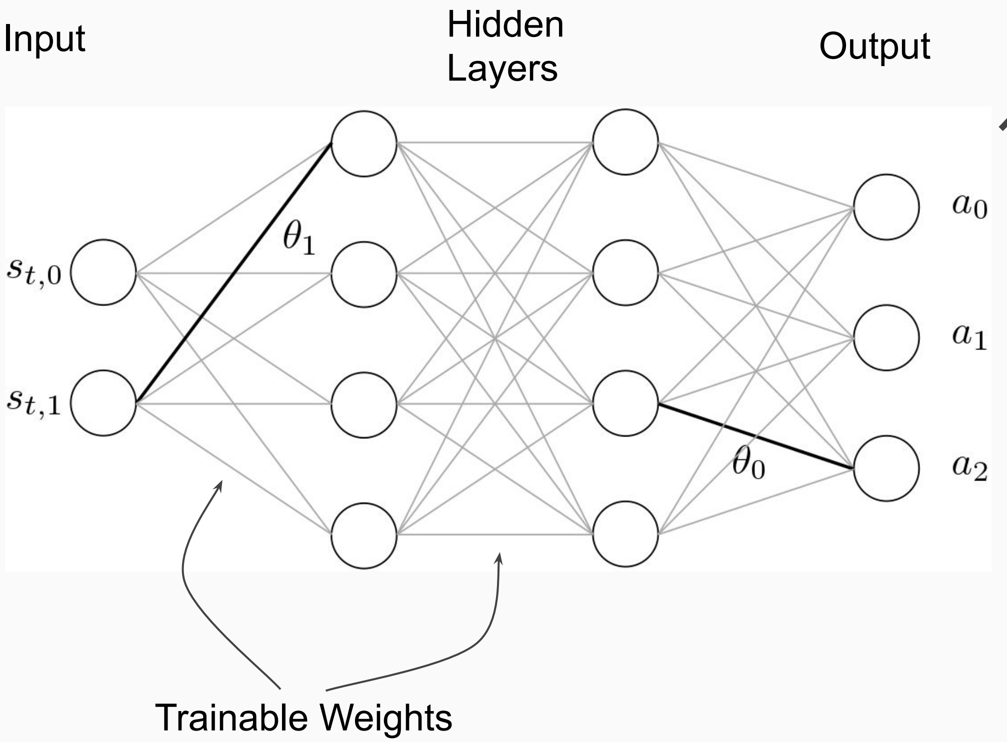
Breakthrough paper - CNN’s with RL. Many little “tricks” like experience replay, target networks etc.

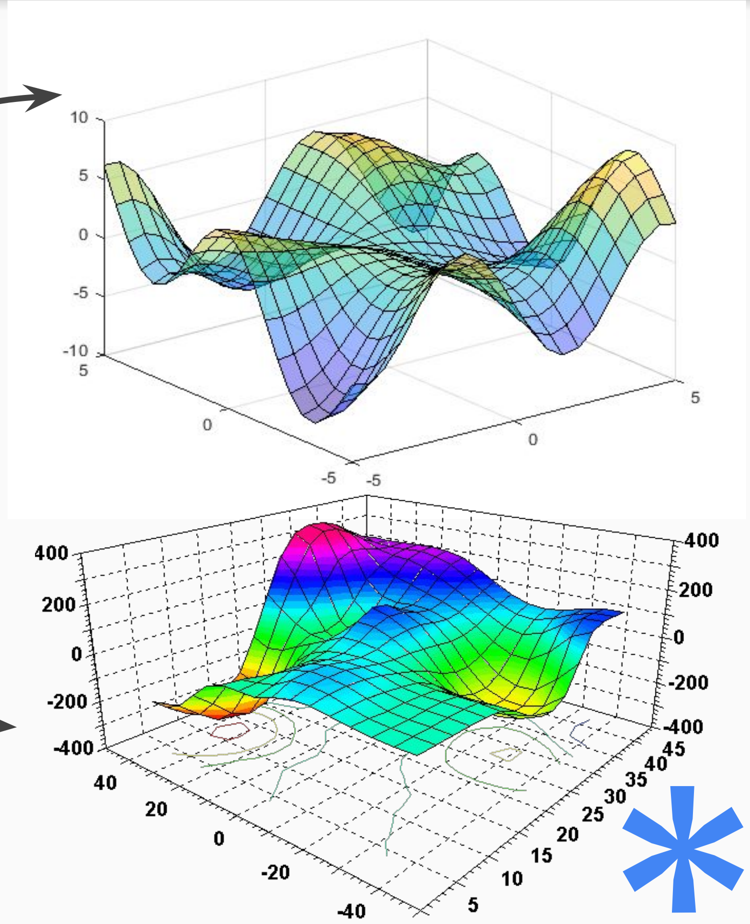
State of the Art

Policy Gradients/Trust region methods

Also an old idea. Newer algorithms currently SOTA e.g. PPO, TRPO, EPG, A3C etc. Far and away best performers in continuous action spaces.

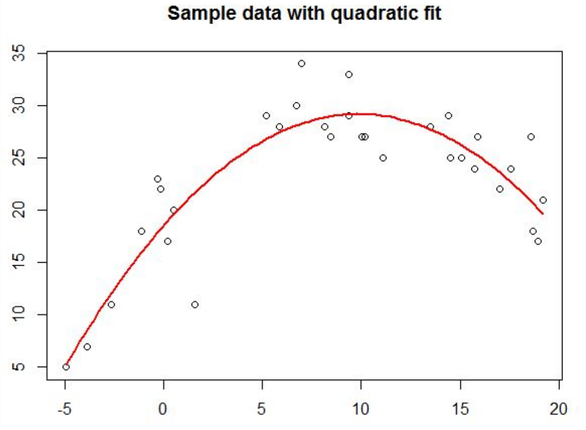
Neural Networks Recap: Function Approximation



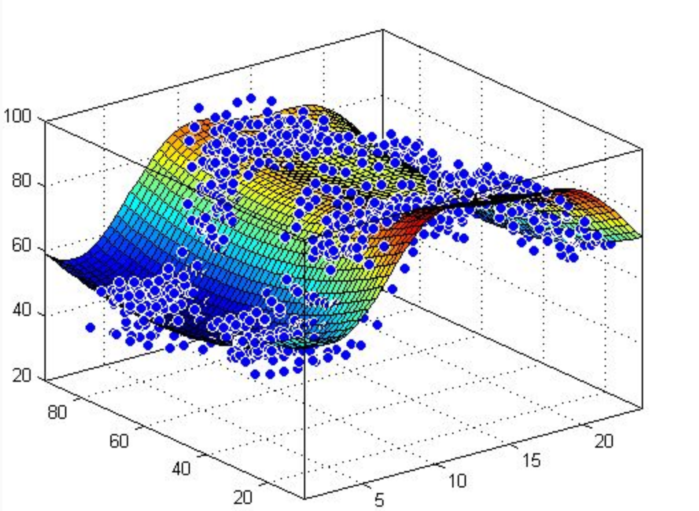


Neural Networks Recap

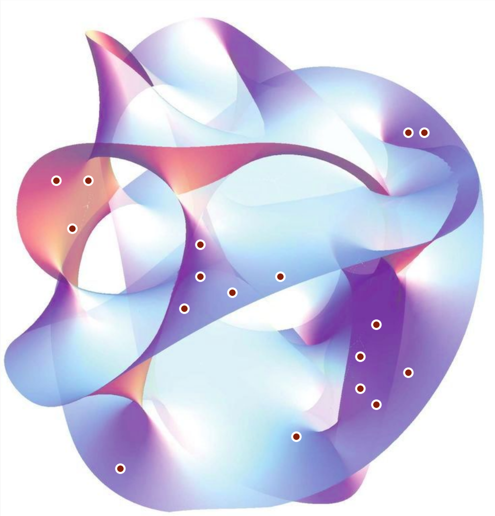
1D Autofit



2D Autofit



N-D Autofit



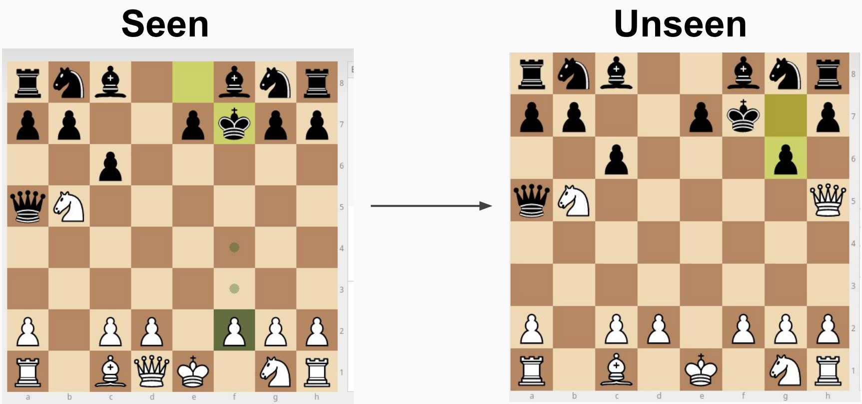
Problems with Tabular Approach

What happens if we require fine-grained control?

- Q-table increases exponentially with action resolution if state representation remains constant
- Will run out of memory

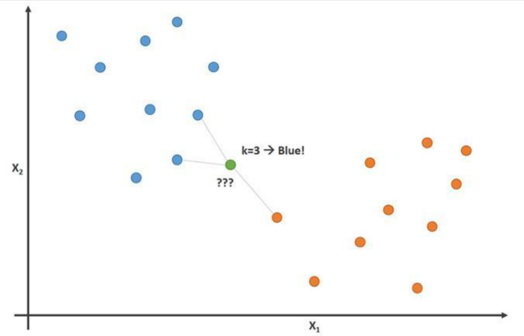
What happens if we encounter a super similar state that we've never seen before?

- Algorithm has no idea
- No "generalization" capability

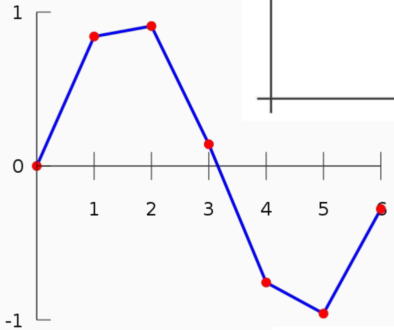


Sidetrack - Why not just use K-Nearest Neighbors /Interpolation for generality?

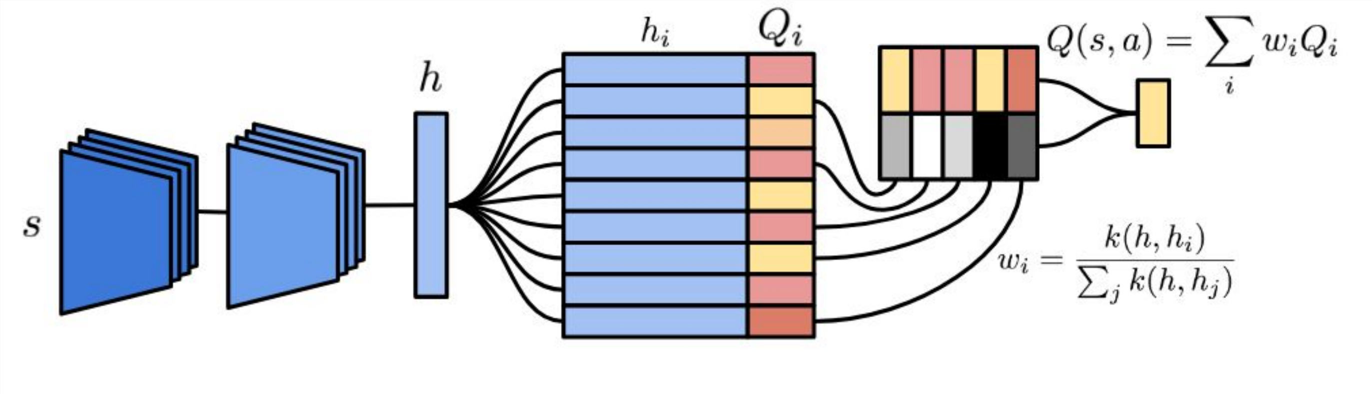
K-NN



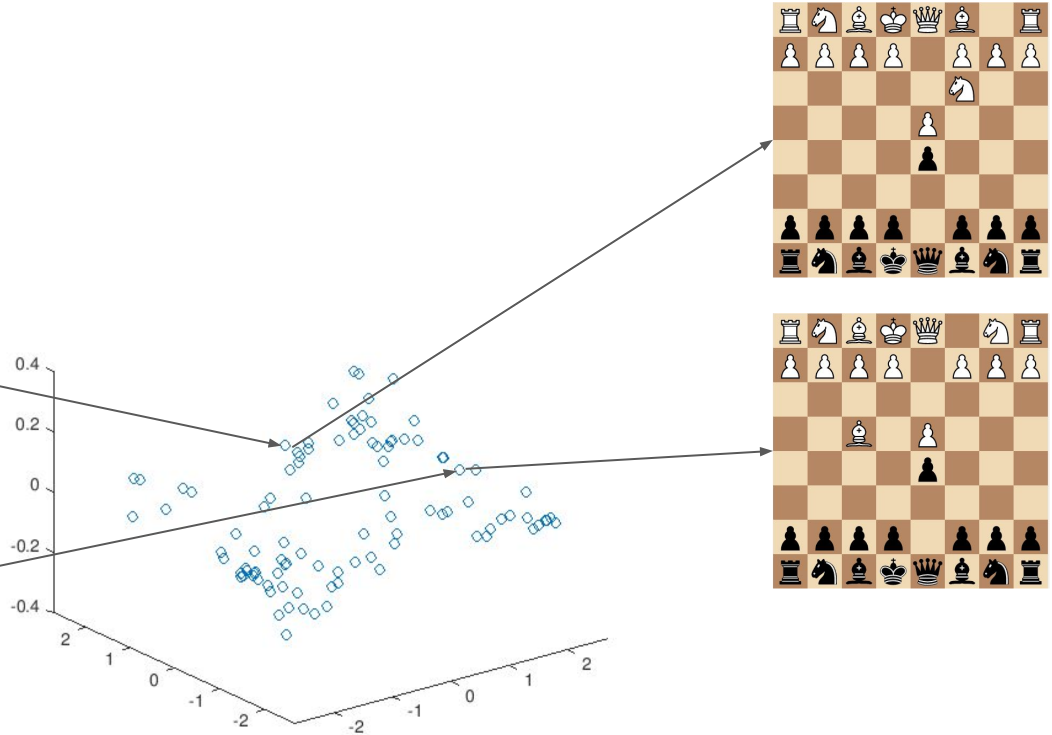
Linear Interpolation



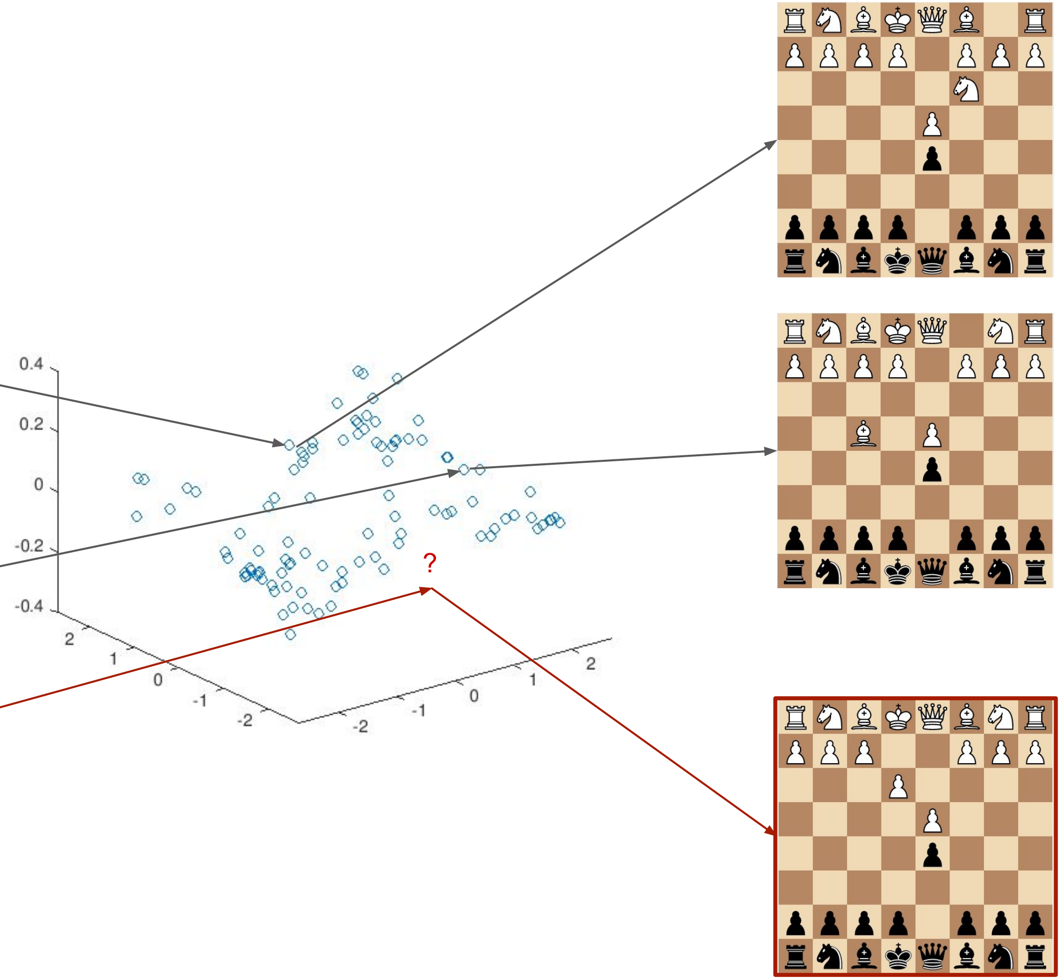
- Extend the K-NN concept to use a soft-max lookup over all present values
- Let the values be representation in the latent space output of a CNN.
- Wire everything up to be differentiable
- You've just invented Neural Episodic Control.



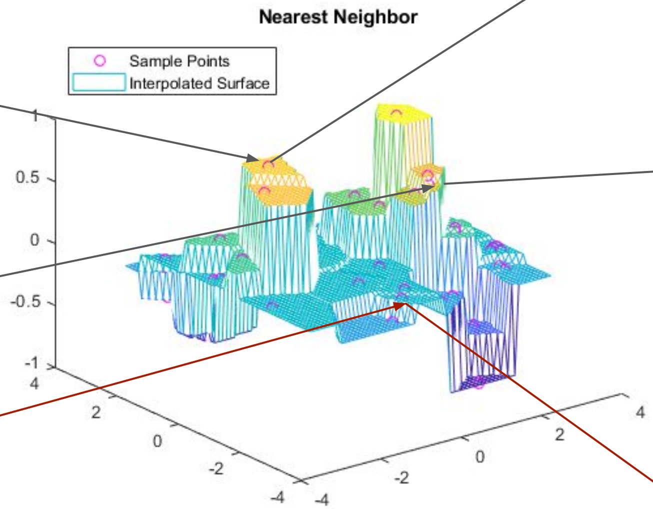
3.5	5.1	0.222222
3	4.9	0.166667
3.2	4.7	0.111111
3.1	4.6	0.083333
3.6	5	0.194444
3.9	5.4	0.305556
3.4	4.6	0.083333
3.4	5	0.194444
2.9	4.4	0.027778
3.1	4.9	0.166667
3.7	5.4	0.305556

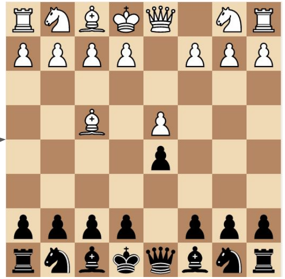


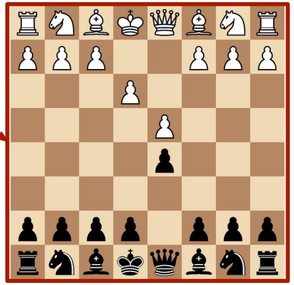
3.5	5.1	0.222222
3	4.9	0.166667
3.2	4.7	0.111111
3.1	4.6	0.083333
3.6	5	0.194444
3.9	5.4	0.305556
3.4	4.6	0.083333
3.4	5	0.194444
2.9	4.4	0.027778
3.1	4.9	0.166667
3.7	5.4	0.305556
2.8	3.1	?



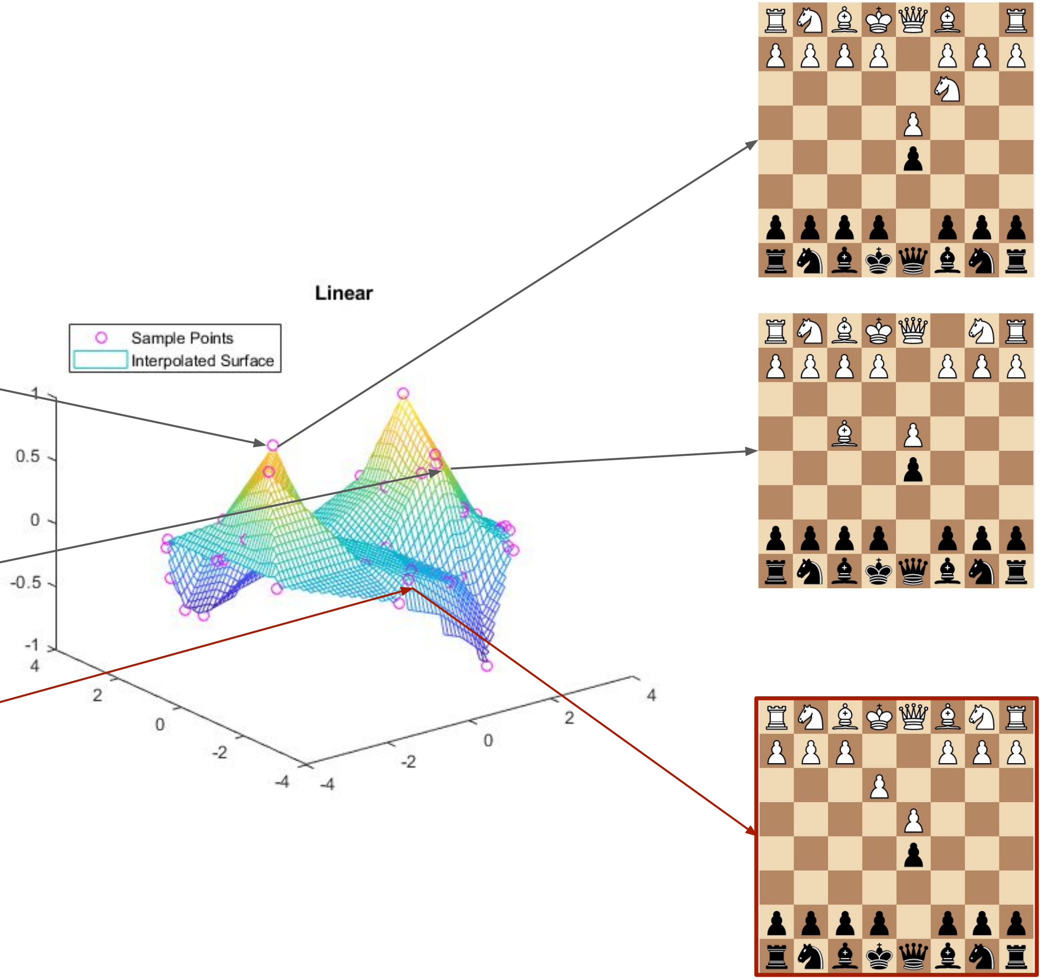
3.5	5.1	0.222222
3	4.9	0.166667
3.2	4.7	0.111111
3.1	4.6	0.083333
3.6	5	0.194444
3.9	5.4	0.305556
3.4	4.6	0.083333
3.4	5	0.194444
2.9	4.4	0.027778
3.1	4.9	0.166667
3.7	5.4	0.305556
2.8	3.1	0.305



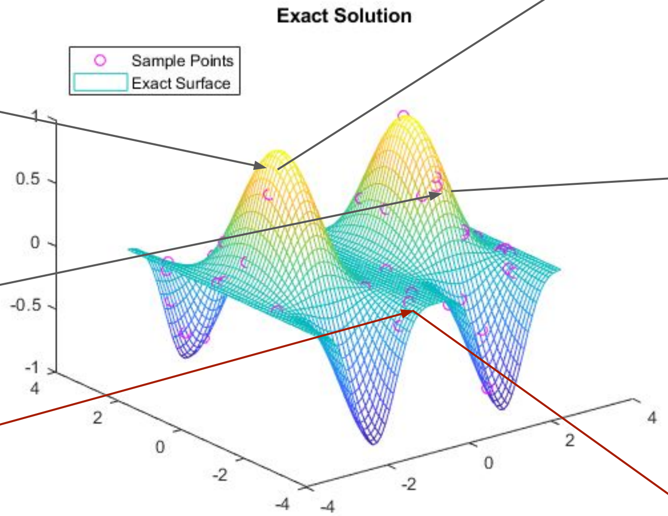


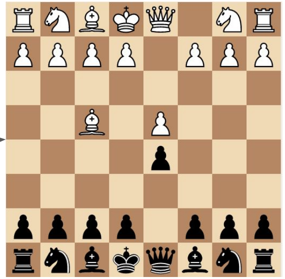


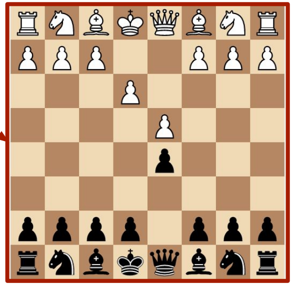
3.5	5.1	0.222222
3	4.9	0.166667
3.2	4.7	0.111111
3.1	4.6	0.083333
3.6	5	0.194444
3.9	5.4	0.305556
3.4	4.6	0.083333
3.4	5	0.194444
2.9	4.4	0.027778
3.1	4.9	0.166667
3.7	5.4	0.305556
2.8	3.1	0.285



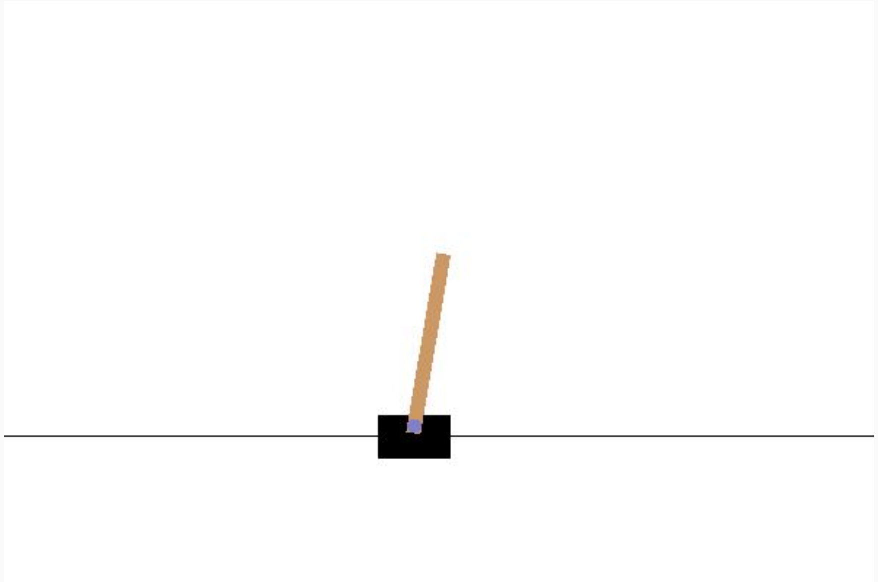
3.5	5.1	0.222222
3	4.9	0.166667
3.2	4.7	0.111111
3.1	4.6	0.083333
3.6	5	0.194444
3.9	5.4	0.305556
3.4	4.6	0.083333
3.4	5	0.194444
2.9	4.4	0.027778
3.1	4.9	0.166667
3.7	5.4	0.305556
2.8	3.1	0.285







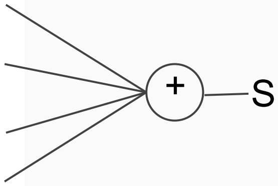
CartPole Environment: Intuition



Observation

Type: Box(4)

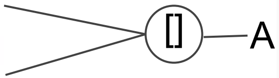
Num	Observation	Min	Max
0	Cart Position	-2.4	2.4
1	Cart Velocity	-Inf	Inf
2	Pole Angle	~ -41.8°	~ 41.8°
3	Pole Velocity At Tip	-Inf	Inf



Actions

Type: Discrete(2)

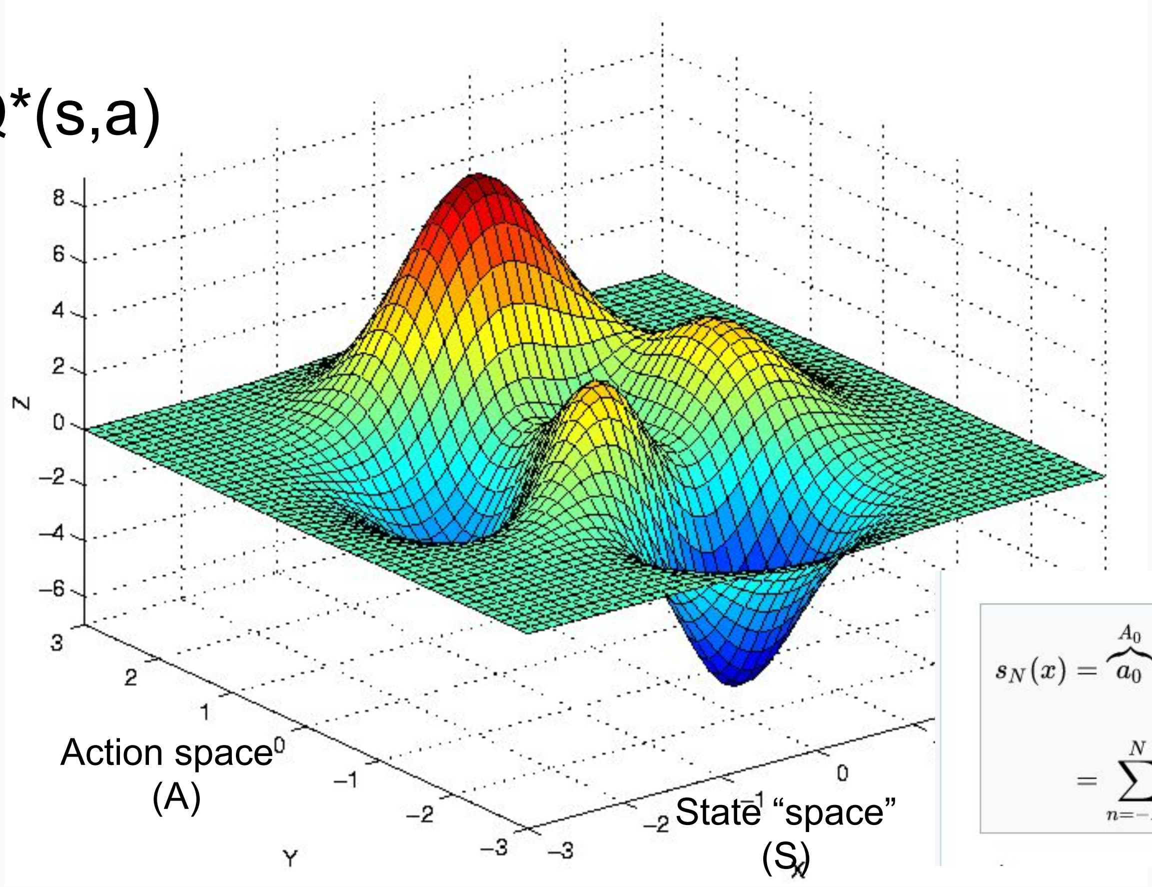
Num	Action
0	Push cart to the left
1	Push cart to the right



- Reduce our statespace to one continuous variable
- Set our action space to be one continuous variable

Intuition: The “Q-Surface”

Q*(s,a)



Action space (A)

State “space” (S)

Code letter	Sample size	Acceptance quality limits (in %)										
		0.0	0.1	0.15	0.25	0.4	0.65	1.0	1.5	2.5	4.0	6.5
A	2	≤ 0	≤ 0	≤ 0	≤ 0	≤ 0	≤ 0	≤ 0	≤ 0	≤ 0	≤ 0	≤ 0
B	3	≤ 0	≤ 0	≤ 0	≤ 0	≤ 0	≤ 0	≤ 0	≤ 0	≤ 0	≤ 0	≤ 0
C	5	≤ 0	≤ 0	≤ 0	≤ 0	≤ 0	≤ 0	≤ 0	≤ 0	≤ 0	≤ 0	≤ 1
D	8	≤ 0	≤ 0	≤ 0	≤ 0	≤ 0	≤ 0	≤ 0	≤ 0	≤ 0	≤ 1	≤ 1
E	13	≤ 0	≤ 0	≤ 0	≤ 0	≤ 0	≤ 0	≤ 0	≤ 0	≤ 1	≤ 1	≤ 2
F	20	≤ 0	≤ 0	≤ 0	≤ 0	≤ 0	≤ 0	≤ 0	≤ 1	≤ 1	≤ 2	≤ 3
G	32	≤ 0	≤ 0	≤ 0	≤ 0	≤ 0	≤ 0	≤ 1	≤ 1	≤ 2	≤ 3	≤ 5
H	50	≤ 0	≤ 0	≤ 0	≤ 0	≤ 0	≤ 1	≤ 1	≤ 2	≤ 3	≤ 5	≤ 7
J	80	≤ 0	≤ 0	≤ 0	≤ 0	≤ 1	≤ 1	≤ 2	≤ 3	≤ 5	≤ 7	≤ 10
K	125	≤ 0	≤ 0	≤ 0	≤ 1	≤ 1	≤ 2	≤ 3	≤ 5	≤ 7	≤ 10	≤ 14
L	200	≤ 0	≤ 0	≤ 1	≤ 1	≤ 2	≤ 3	≤ 5	≤ 7	≤ 10	≤ 14	≤ 21
M	315	≤ 0	≤ 1	≤ 1	≤ 2	≤ 3	≤ 5	≤ 7	≤ 10	≤ 14	≤ 21	≤ 21
N	500	≤ 0	≤ 1	≤ 2	≤ 3	≤ 5	≤ 7	≤ 10	≤ 14	≤ 21	≤ 21	≤ 21
P	800	≤ 0	≤ 2	≤ 3	≤ 5	≤ 7	≤ 10	≤ 14	≤ 21	≤ 21	≤ 21	≤ 21
Q	1,250	≤ 0	≤ 3	≤ 5	≤ 7	≤ 10	≤ 14	≤ 21	≤ 21	≤ 21	≤ 21	≤ 21
R	2,000	≤ 0	≤ 5	≤ 7	≤ 10	≤ 14	≤ 21	≤ 21	≤ 21	≤ 21	≤ 21	≤ 21

Note: if the sample size exceeds lot size, carry out 100% inspection.

Tabular approximation (sampling)

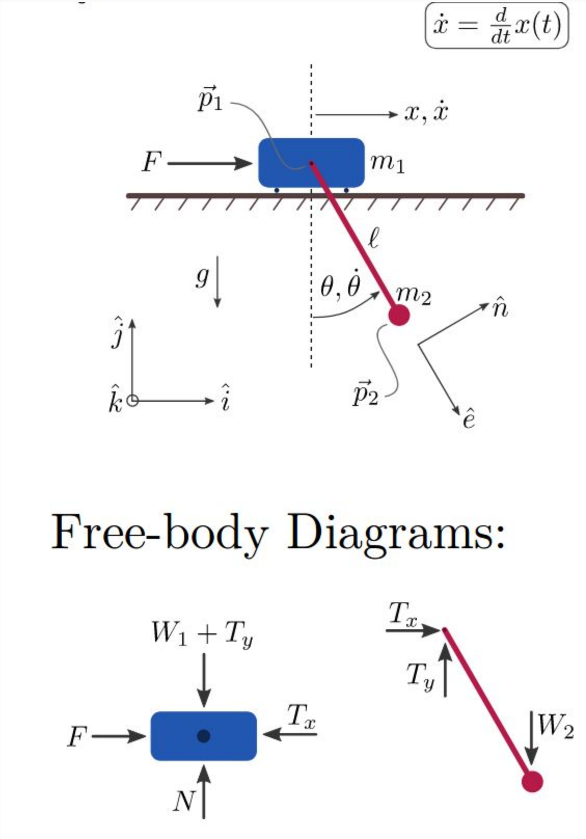
vs.

Functional approximation (Parameterization)

$$s_N(x) = \overbrace{a_0}^{A_0}/2 + \sum_{n=1}^{N}\left(\overbrace{a_n}^{A_n\,\sin(\phi_n)}\cos\left(\frac{2\pi nx}{P}\right) + \overbrace{b_n}^{A_n\,\cos(\phi_n)}\sin\left(\frac{2\pi nx}{P}\right)\right)$$

$$= \sum_{n=-N}^{N} c_n \cdot e^{i\frac{2\pi nx}{P}},$$

What decides the shape of our Q-surface?



→ Combine **Eqn 1** and **Eqn 2** to cancel tension:

$$F - m_1\,\ddot{x} = m_2\,\left(\ddot{x} + \ell\,\ddot{\theta}\,\cos\theta - \ell\,\dot{\theta}^2\,\sin\theta\right)$$

$$F = (m_1 + m_2)\,\ddot{x} + m_2\,\ell\,\ddot{\theta}\,\cos\theta - m_2\,\ell\,\dot{\theta}^2\,\sin\theta$$

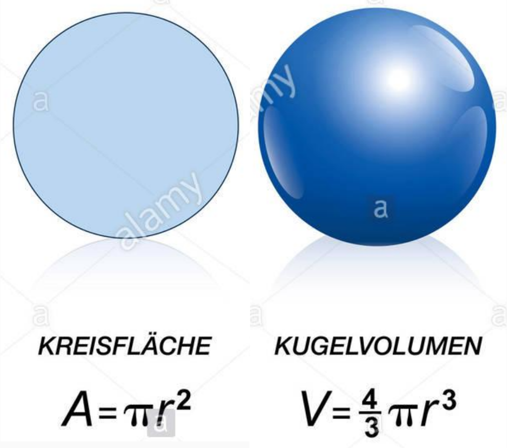
→ Write equations of motion in matrix form:

$$\boxed{\begin{pmatrix} \cos\theta & \ell \\ m_1 + m_2 & m_2\ell\cos\theta \end{pmatrix} \begin{pmatrix} \ddot{x} \\ \ddot{\theta} \end{pmatrix} = \begin{pmatrix} -g\sin\theta \\ F + m_2\,\ell\,\dot{\theta}^2\,\sin\theta \end{pmatrix}}$$

Relatively, this is considered a “simple” environment

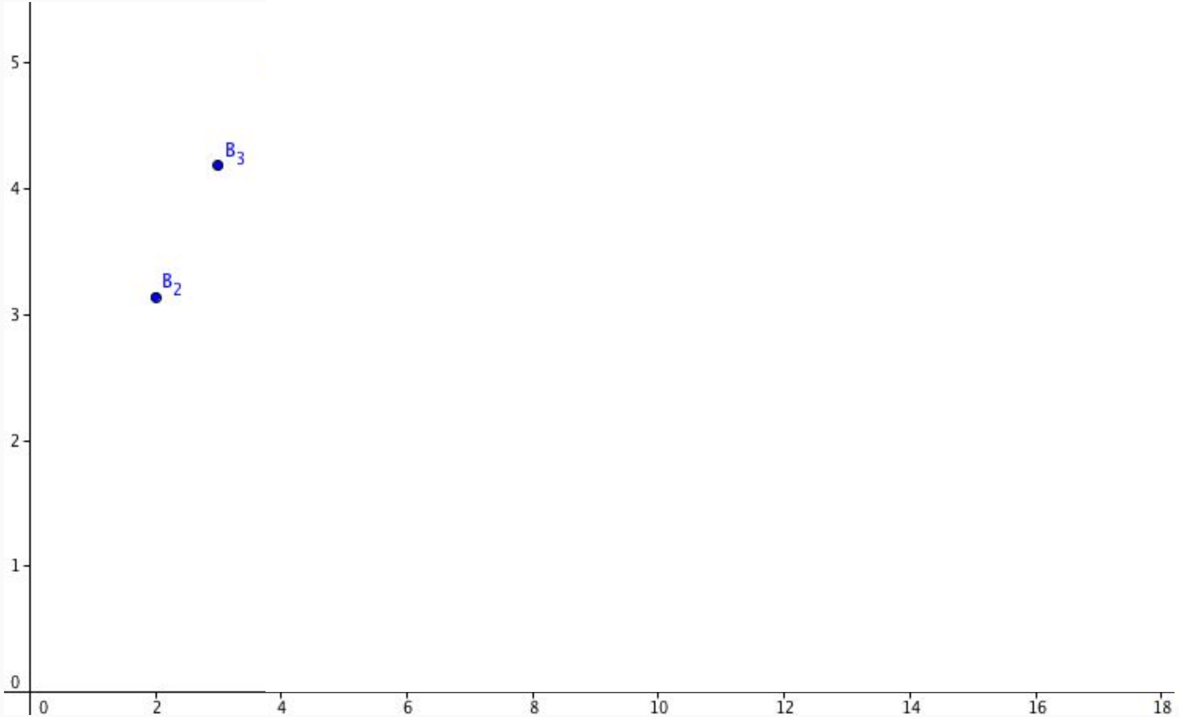
Caveat: Intuition in Low dimensions does not necessarily extend to High dimensions

2D 3D 4D ? n-D ???

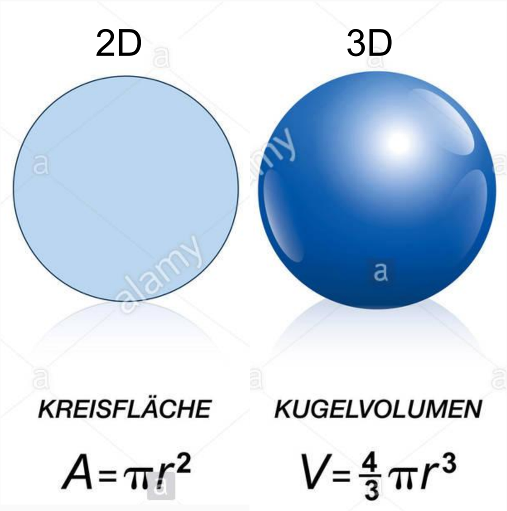


$$V_n(R) = \frac{2^{\frac{n+1}{2}} \pi^{\frac{n-1}{2}} R^n}{1 \cdot 3 \cdot 5 \cdots n}$$

$$V_n(R) = \frac{\pi^{\frac{n}{2}} R^n}{\Gamma\left(\frac{n}{2}+1\right)}.$$



Caveat: Intuition in Low dimensions does not necessarily extend to High dimensions

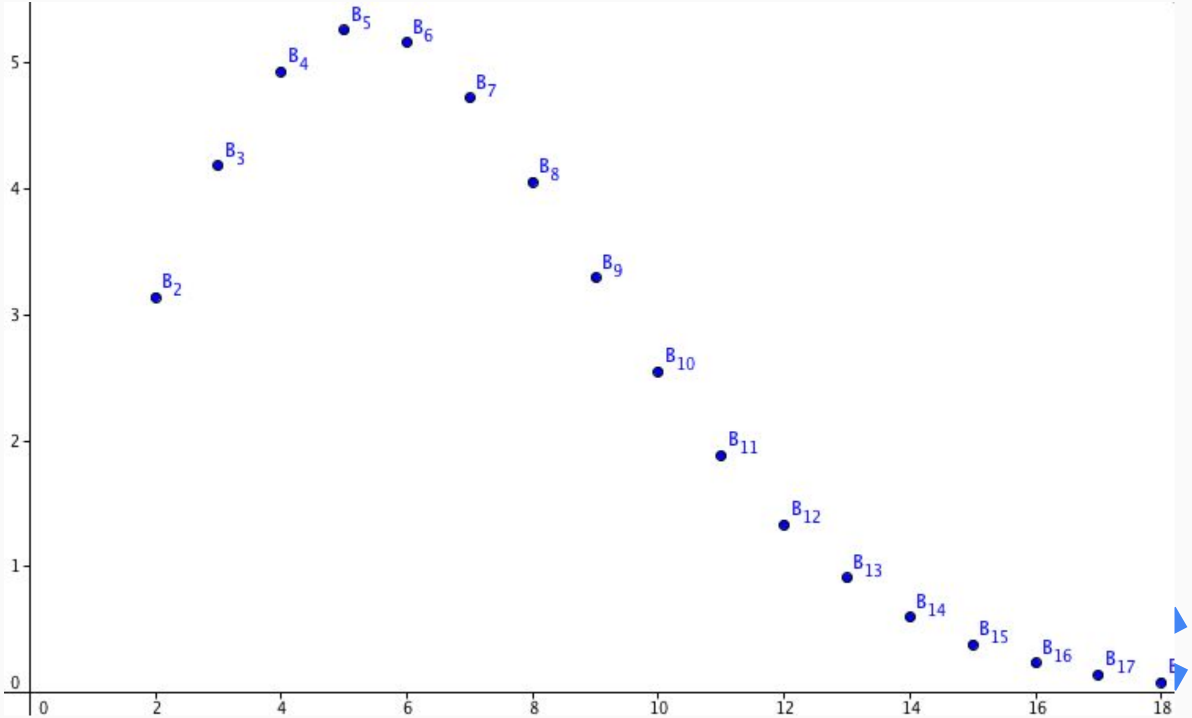


4D
?

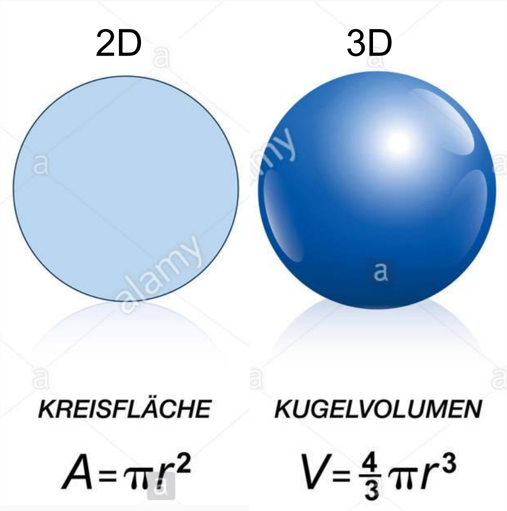
n-D
???

$$V_n(R) = \frac{2^{\frac{n+1}{2}} \pi^{\frac{n-1}{2}} R^n}{1 \cdot 3 \cdot 5 \cdots n}$$

$$V_n(R) = \frac{\pi^{\frac{n}{2}} R^n}{\Gamma\left(\frac{n}{2}+1\right)}.$$



Caveat: Intuition in Low dimensions does not necessarily extend to High dimensions

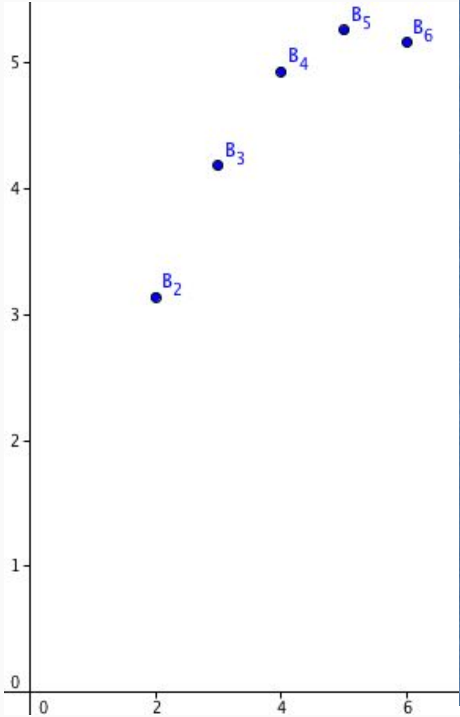


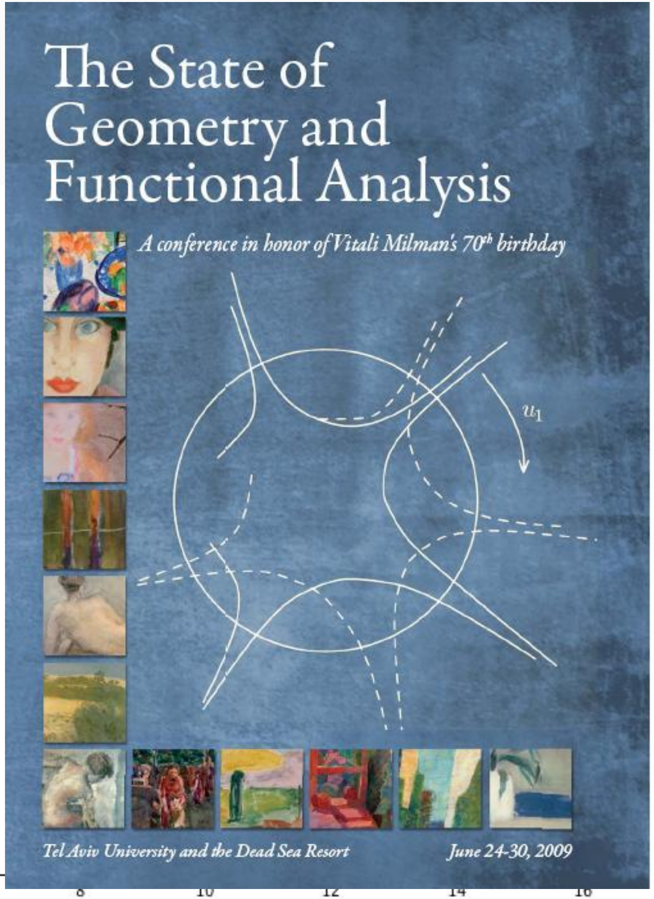
4D
?

n-D
???

$$V_n(R) = \frac{2^{\frac{n+1}{2}} \pi^{\frac{n-1}{2}} R^n}{1 \cdot 3 \cdot 5 \cdots n}$$

$$V_n(R) = \frac{\pi^{\frac{n}{2}} R^n}{\Gamma\left(\frac{n}{2}+1\right)}.$$





Why Function Approximation?

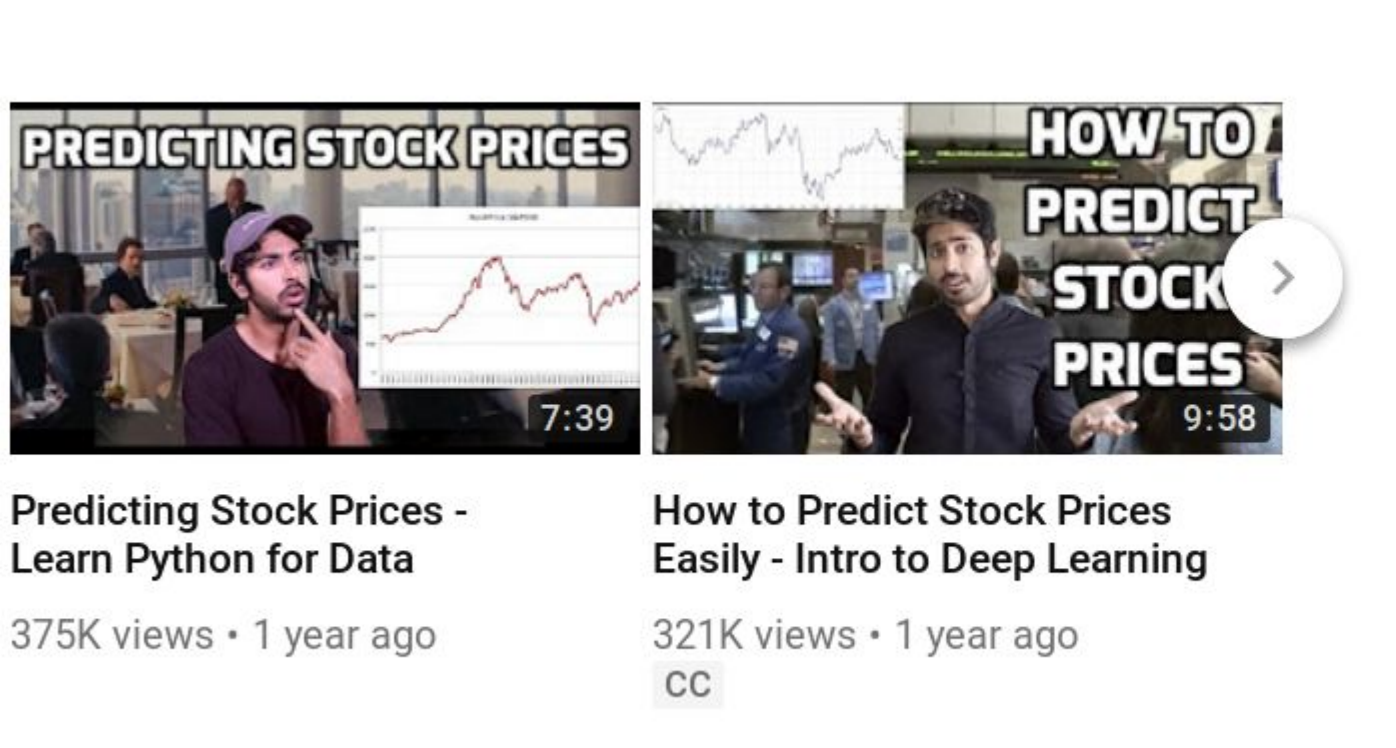
Base assumption: The environment has some underlying dynamics that can be more efficiently represented as a (maybe complex) function than by sampling.

“Now I know about RL I can make a stock trading bot and it’ll learn to trade for me, catchya from my island scrubs” - anon

Why Function Approximation?

Base as … can be more effi… ing.

“Now I k… for me, catchya



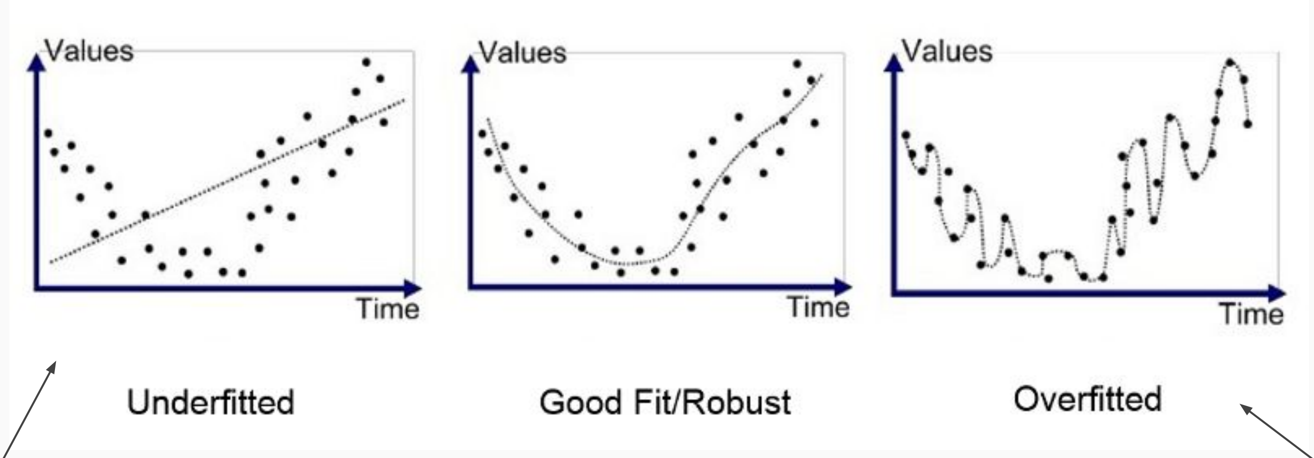
Why Function Approximation?

Base assumption: The environment has some underlying dynamics that can be more efficiently represented as a (maybe complex) function than by sampling.

“Now I know about RL I can make a stock trading bot and it’ll learn to trade for me, catchya from my island scrubs” - anon

Neural Networks *are* universal function approximators in theory - but in practice we’re constrained by memory, computational power, and simulation accuracy/data.

Over/Underfitting in Deep-RL



1 - layer NN (i.e Linear Regression)

2 - layer NN with ReLU

Convnet/Resnet/some other monstrosity

Easy to train (stable)

Learns quickly

Good for: simple dynamics

Good for: complex dynamics

Hard to train (unstable) (for now)

Learns slowly

Why Neural Networks?

- Hype
- Backpropagation is a cool idea - using gradients is appealing since we make use of more information than an uninformed search
- But really, its possible to use RL with any sufficiently complex function approximation method
- Some work on interpretability uses decision trees instead - it works
- Key point: RL is a *General Framework* which we can use Neural Networks within.