

Deep Reinforcement Learning

Aaron Schumacher

Data Science DC

2017-11-14

Aaron Schumacher

- planspace.org has these slides



DEEP

LEARNING

ANALYTICS

Deep Reinforcement Learning Bootcamp

26-27 August 2017

Berkeley, CA

Reinforcement Learning

An Introduction



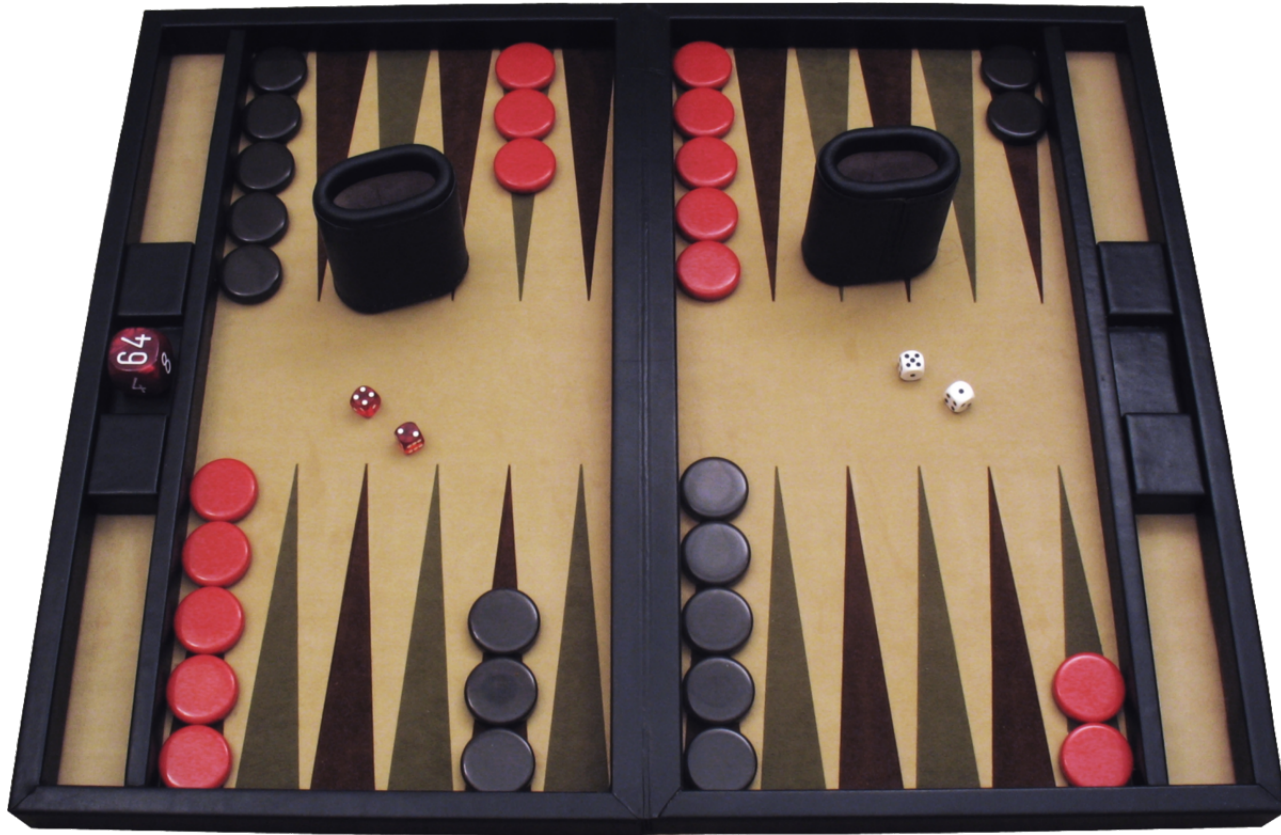
Richard S. Sutton and Andrew G. Barto

Plan

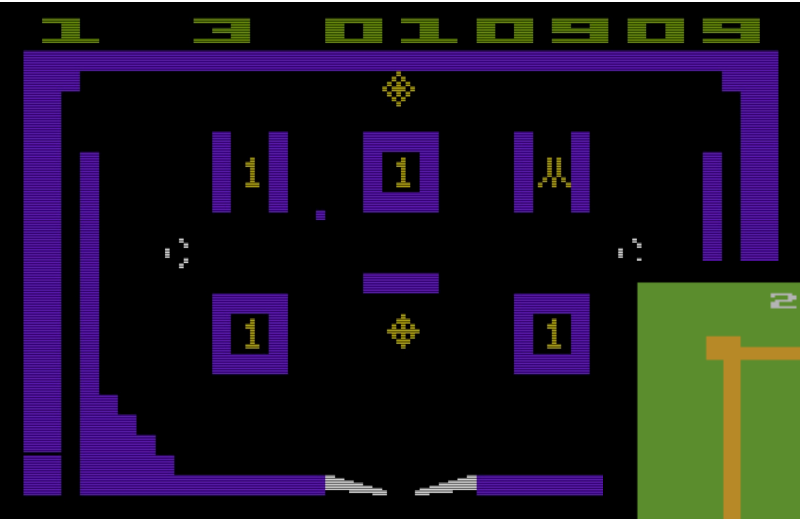
- applications: what
 - theory
- applications: how
- onward

applications: what

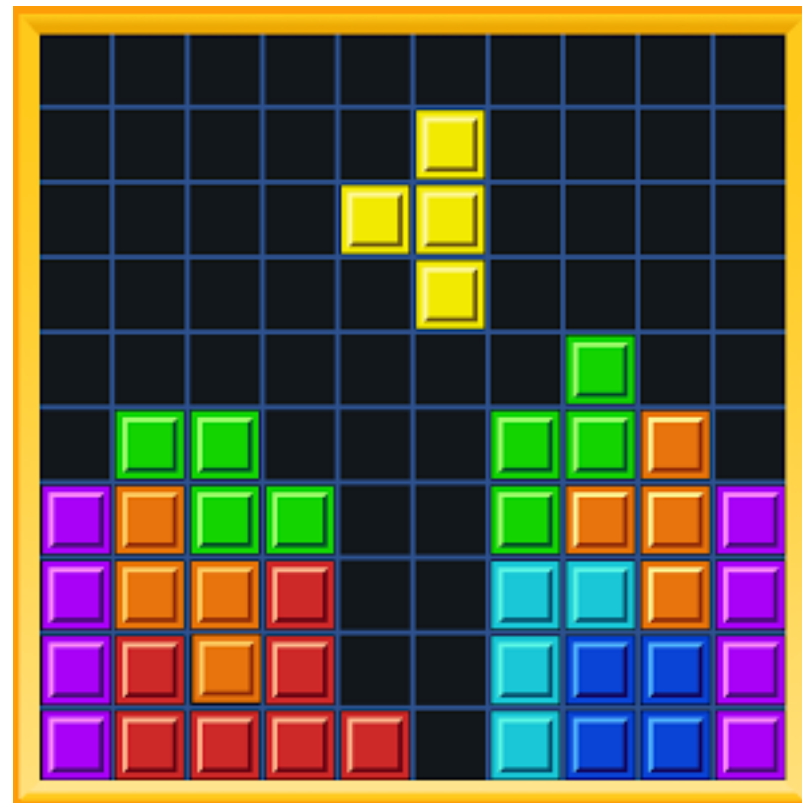
Backgammon



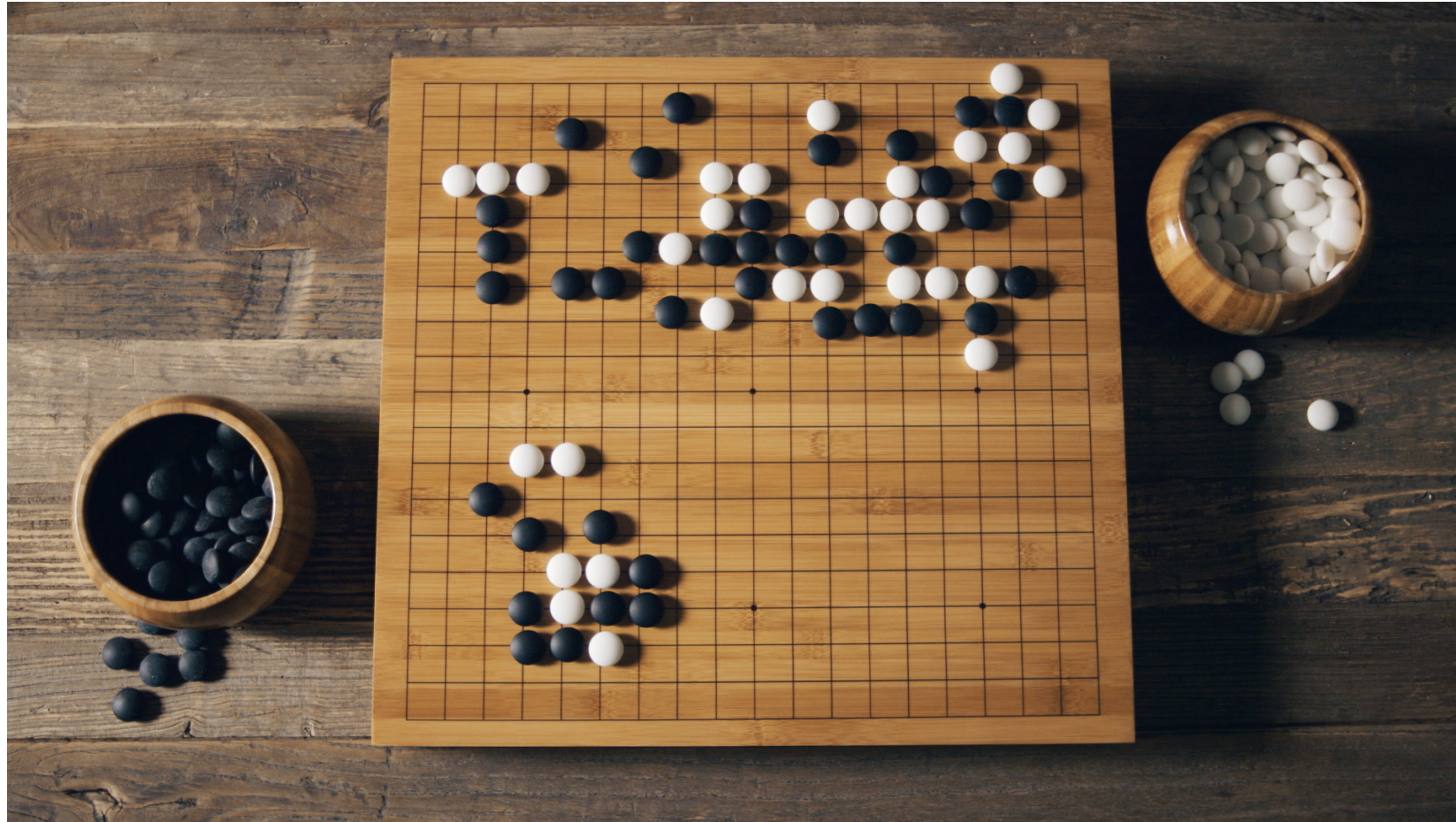
Atari



Tetris



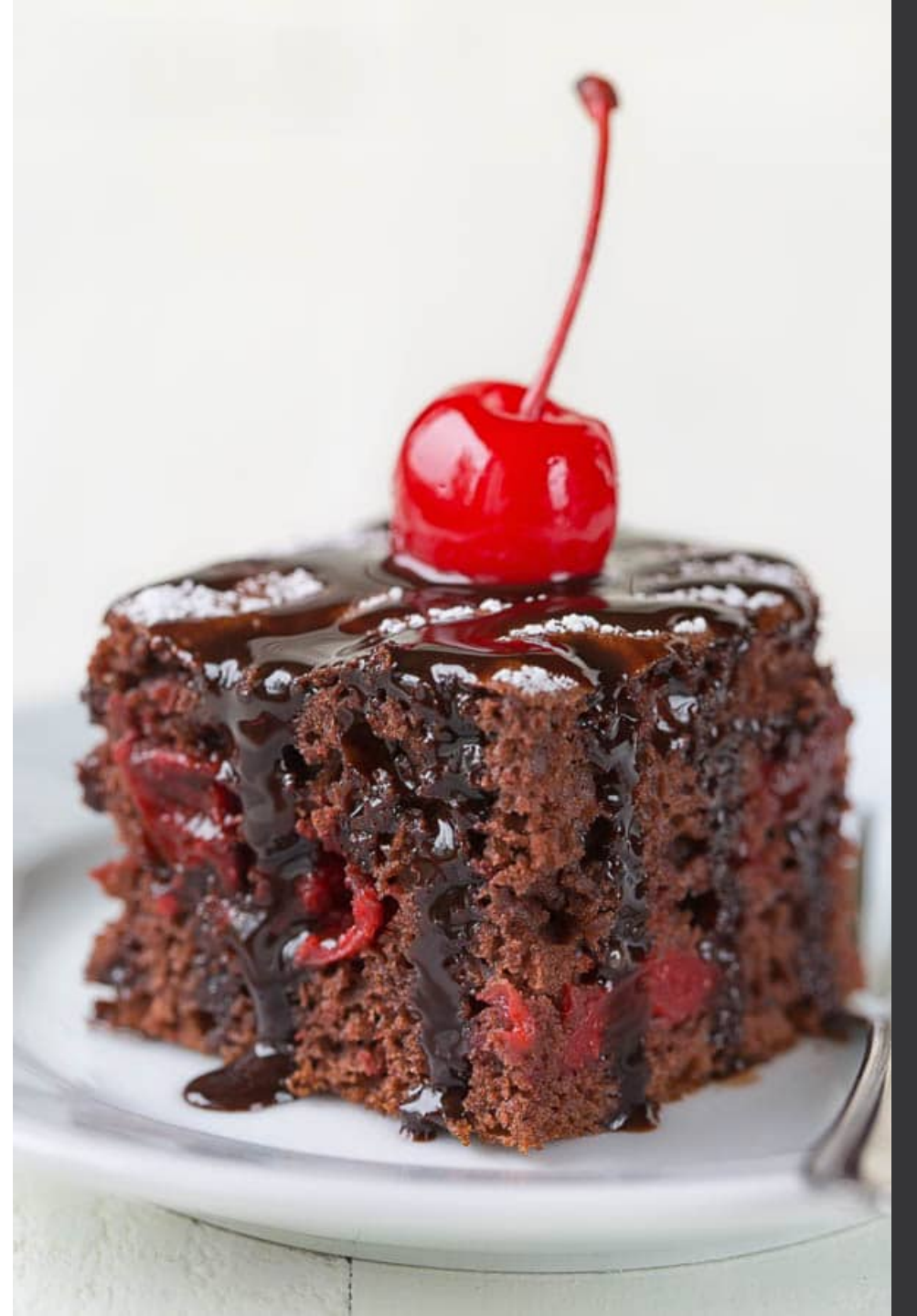
Go



theory

Yann LeCun's cake

- cake: unsupervised learning
- icing: supervised learning
- cherry: reinforcement learning



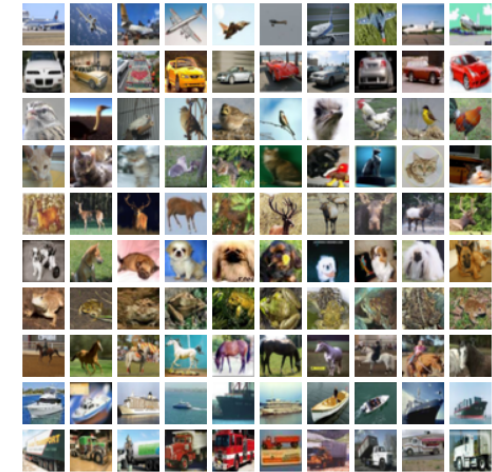
unsupervised learning

- S

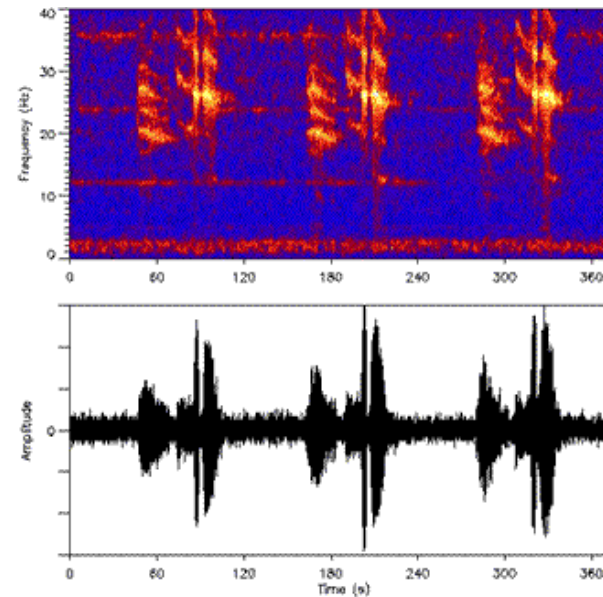
state s

NUMBERS

1 1 5 4 3
7 5 3 5 3
5 5 9 0 6
3 5 2 0 0



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras quis cursus purus. Aliquam nisl purus, posuere id venenatis et, posuere non leo. Proin commodo, arcu vel commodo pellentesque, erat sapien mattis justo, nec semper urna lacus vel nisi. Nullam nisi odio, interdum id dictum vel, ultricies quis leo. Aliquam sed porttitor lacus. Suspendisse vulputate, leo vitae tempus accumsan, justo ex hendrerit magna, vel suscipit eros purus non magna. Vestibulum ac interdum ipsum. Sed nibh sem, pharetra euismod purus non, finibus mattis sapien.



unsupervised (?) learning

- given s
- learn $s \rightarrow \text{cluster_id}$
- learn $s \rightarrow s$



unsupervised learning

- S



supervised learning

- $s \rightarrow a$

action *a*

17.3

NUMBERS

[2.0, 11.7, 5]

“cat”/“dog”

“left”/“right”

4.2V

supervised learning


- given s, a
- learn $s \rightarrow a$



deep

supervised learning

• $S \rightarrow a$



with deep neural nets

reinforcement learning

- $r, s \rightarrow a$

reward r

NUMBER

-3

0

1

7.4

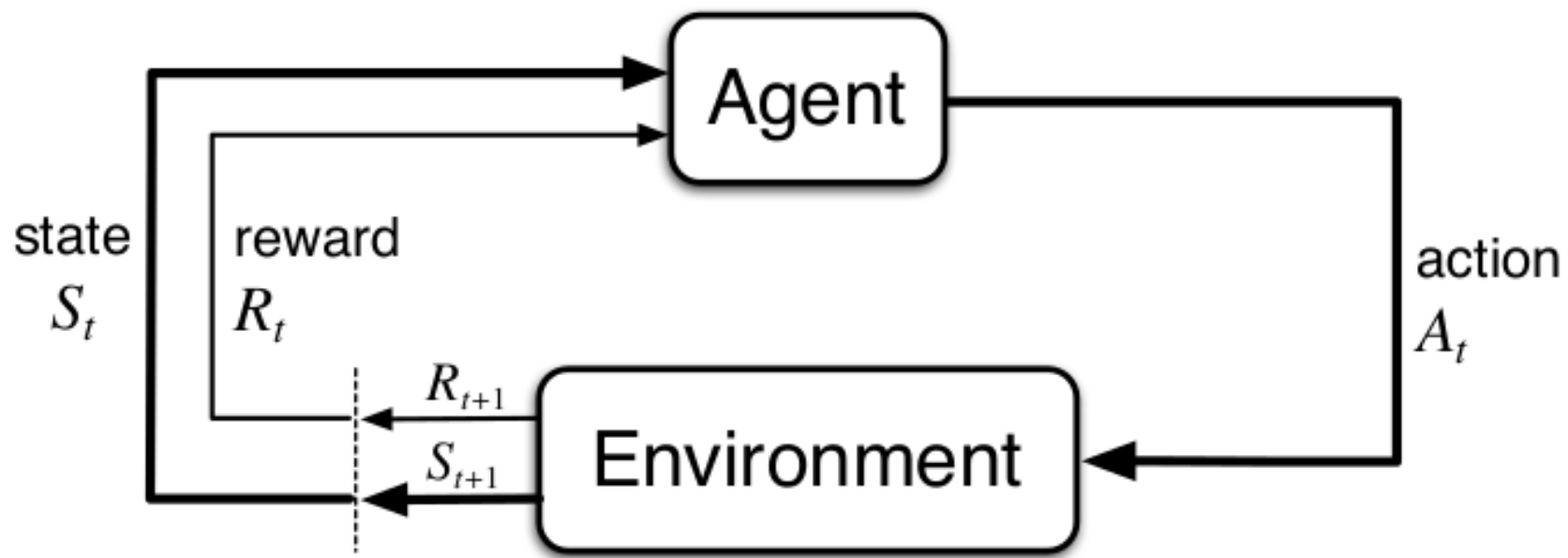
optimal control /
reinforcement learning

r, s → *a*



$$r', s' \rightarrow a'$$





reinforcement learning

- “given” $r, s, a, r', s', a', \dots$
- learn “good” $s \rightarrow a$

Question

- Can you formulate supervised learning as reinforcement learning?

supervised as reinforcement?

- reward 0 for incorrect, reward 1 for correct
 - beyond binary classification?
- reward deterministic
- next state random

Question

- Why is the Sarsa algorithm called Sarsa?

reinforcement learning


- $r, s \rightarrow a$



deep

reinforcement learning

• $r, s \rightarrow a$



with deep neural nets

Markov property

- s is enough

		Make choices?	
		no	yes
Completely observable?	yes	Markov Chain	MDP Markov Decision Process
	no	HMM Hidden Markov Model	POMDP Partially Observable MDP

usual RL problem elaboration

policy π

• $\pi: S \rightarrow a$

return

- $\sum r$

- into the future (episodic or ongoing)

start			goal

reward

0	0	0	0
0	0	0	0
0	0	0	1
0	0	0	0

return

1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1

Question

- How do we incentivize efficiency?

start			goal

negative reward at each time step

-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	0
-1	-1	-1	-1

discounted rewards

- $\sum \gamma^t r$

0	0	0	0
0	0	0	0
0	0	0	1
0	0	0	0

average rate of reward

$$\bullet \frac{\sum r}{t}$$

0	0	0	0
0	0	0	0
0	0	0	1
0	0	0	0

value functions

- $v: s \rightarrow \sum r$

- $q: s, a \rightarrow \sum r$

Question

- Does a value function specify a policy (if you want to maximize return)?

- $v: s \rightarrow \sum r$

- $q: s, a \rightarrow \sum r$

trick question?

- v_π

- q_π

Question

- Does a value function specify a policy (if you want to maximize return)?

- $v: s \rightarrow \sum r$

- $q: s, a \rightarrow \sum r$

environment dynamics

- $s, a \rightarrow r', s'$

Going “right”
from “start,”
where will
you be next?

start			goal

model

• $s, a \rightarrow r', s'$

learning and acting

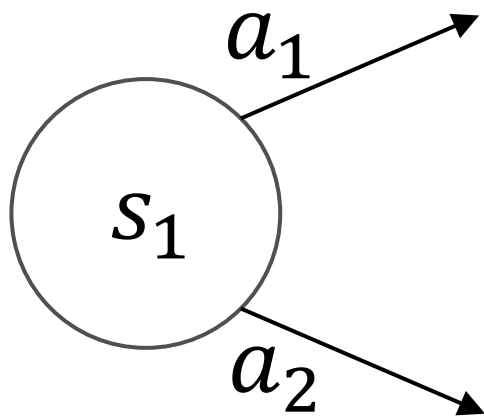
learn model of dynamics

- from experience
- difficulty varies

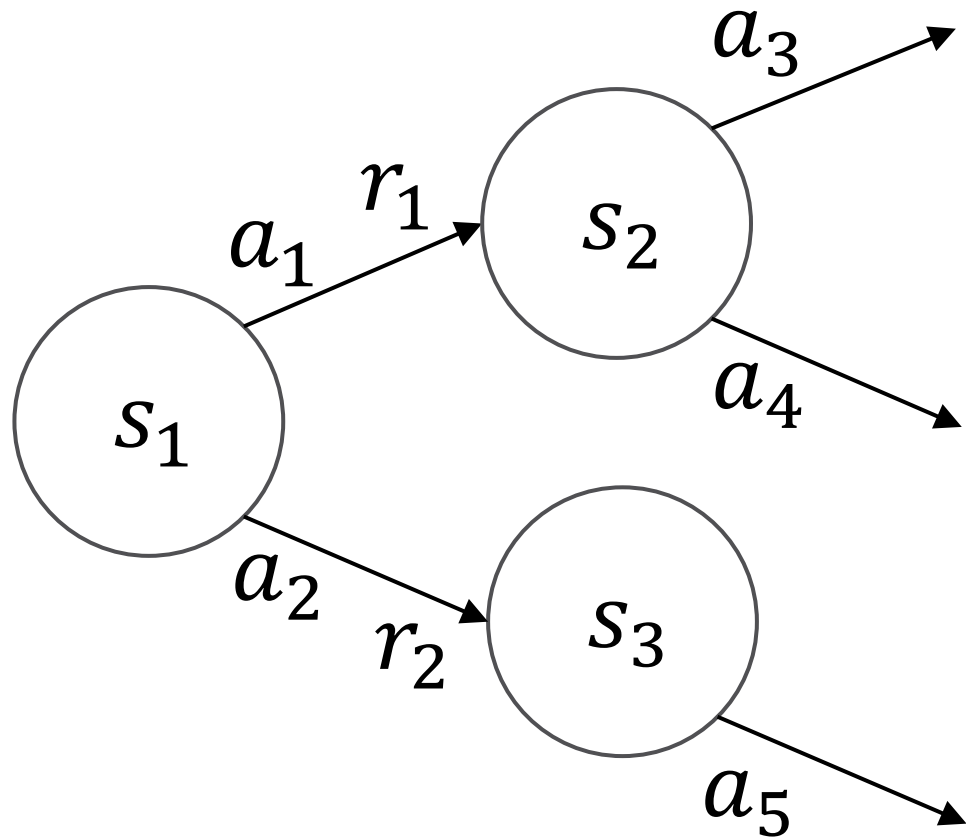
learn value function(s) with planning

- assuming we have the environment dynamics or a good model already

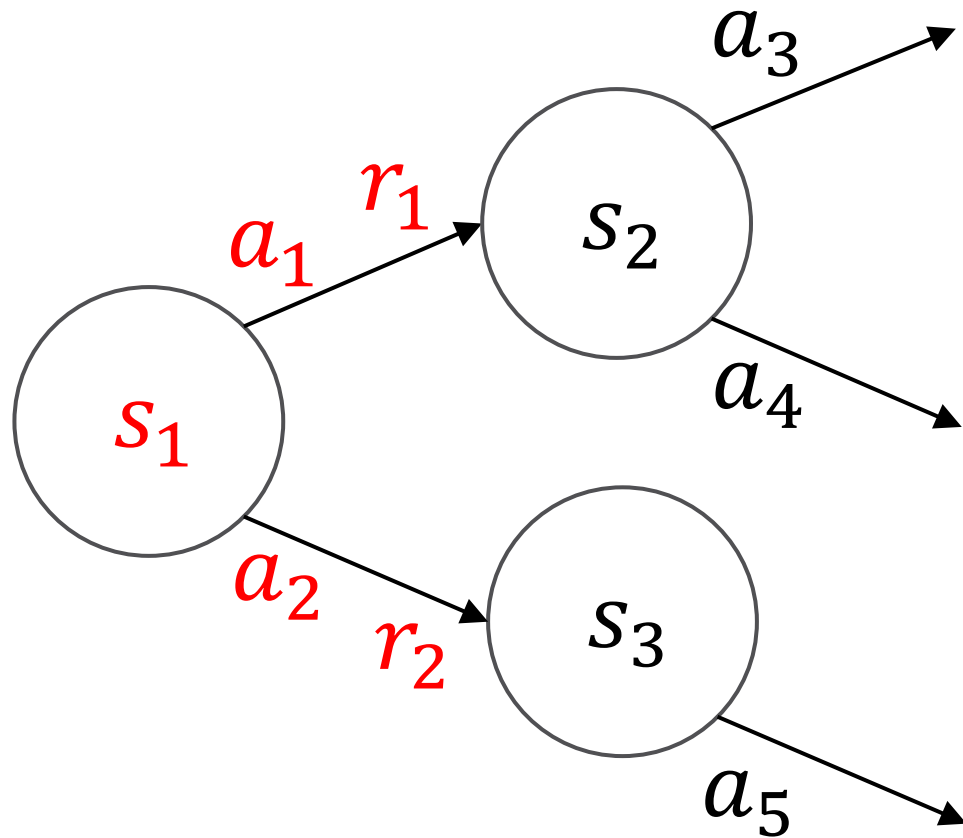
planning



planning

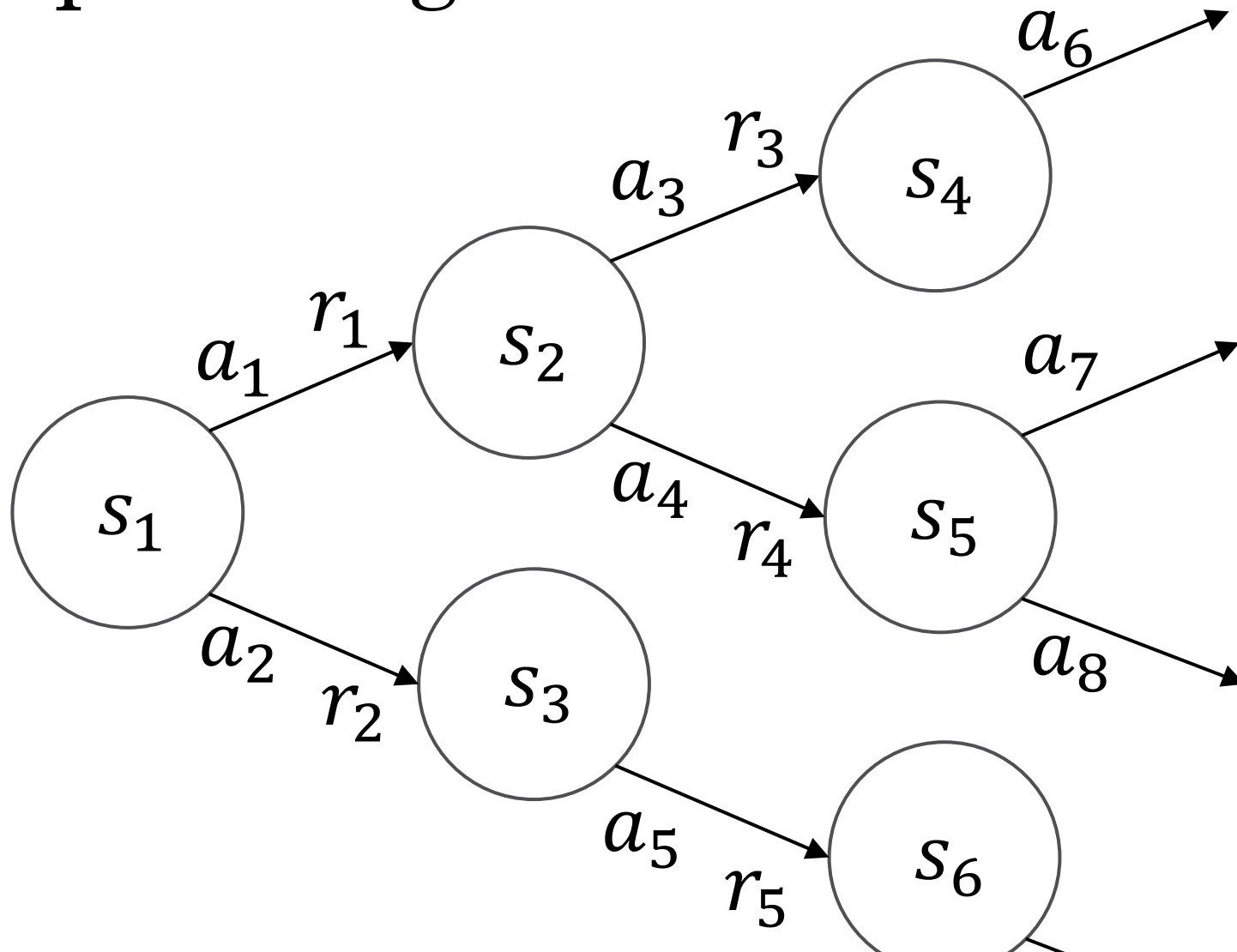


planning

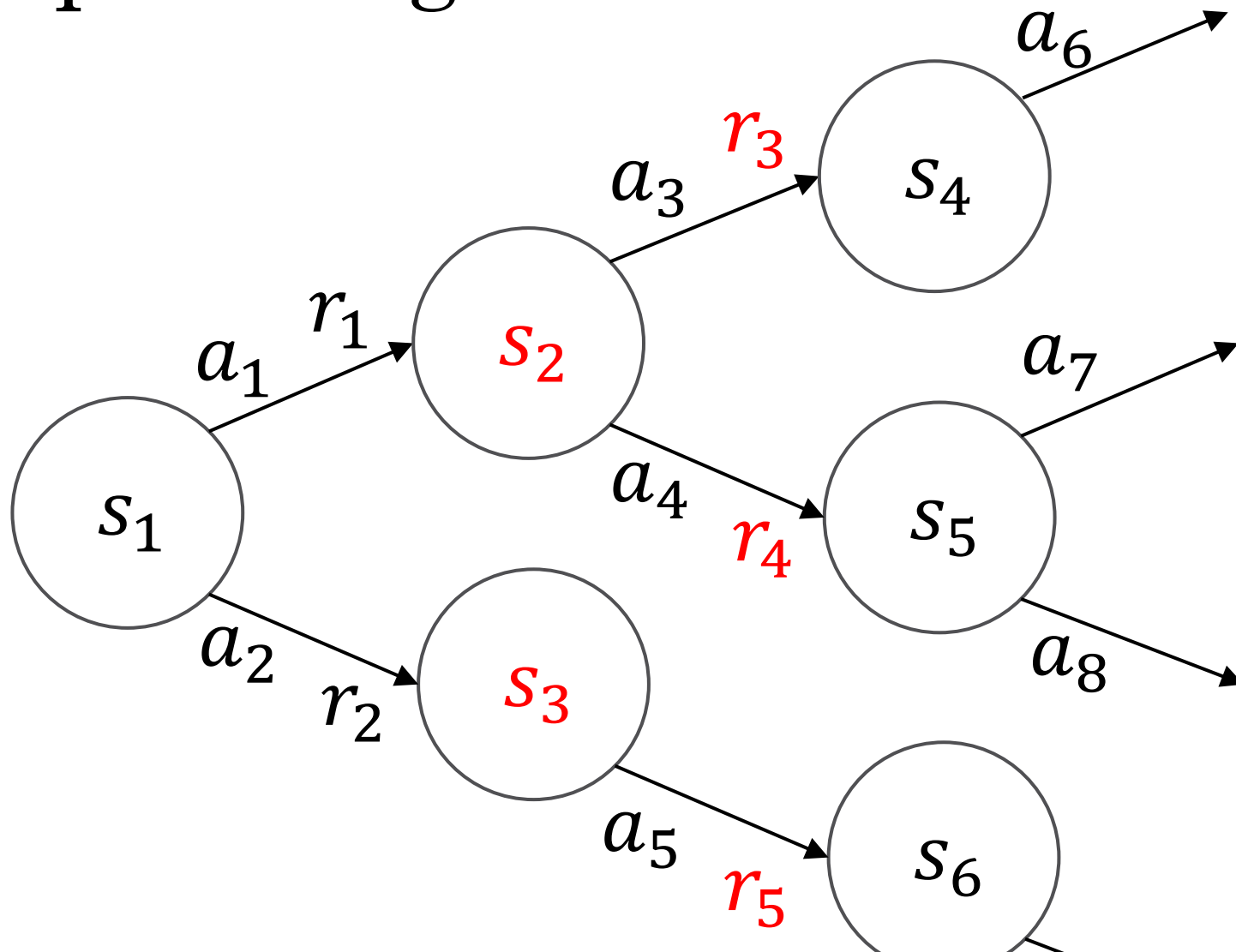


Start
estimating
 $q(s, a)$!

planning

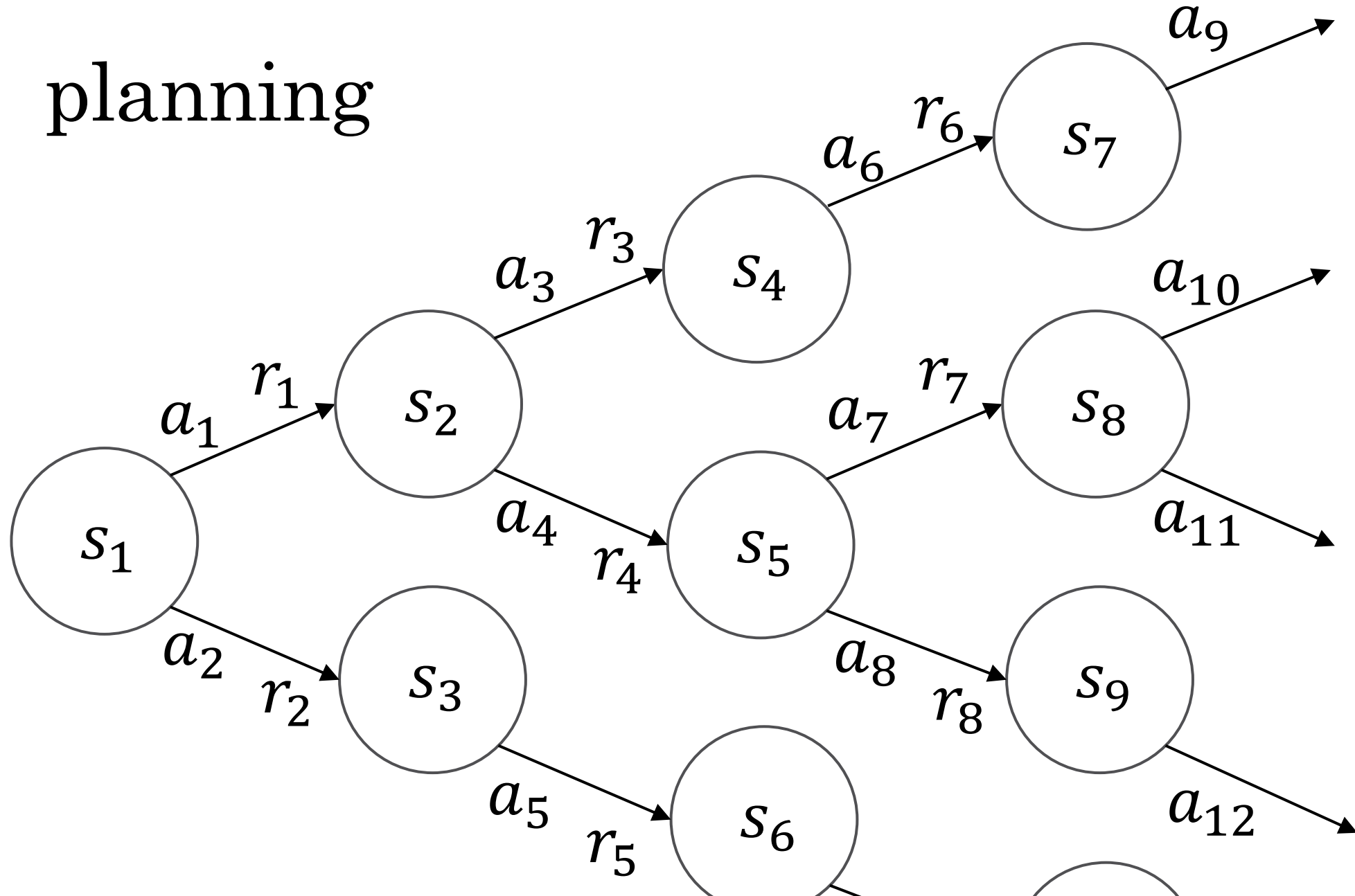


planning

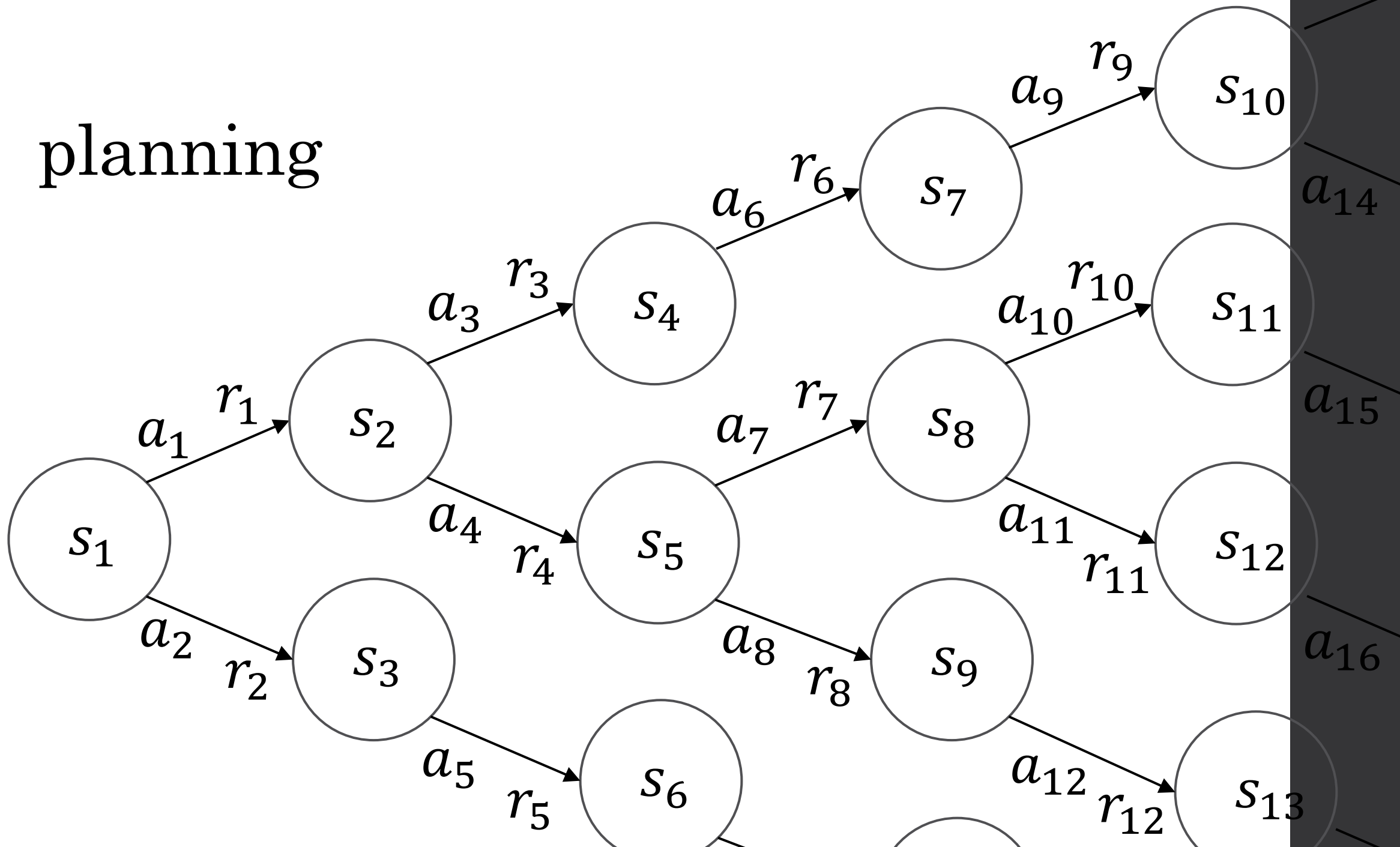


Start
estimating
 $v(s)$!

planning



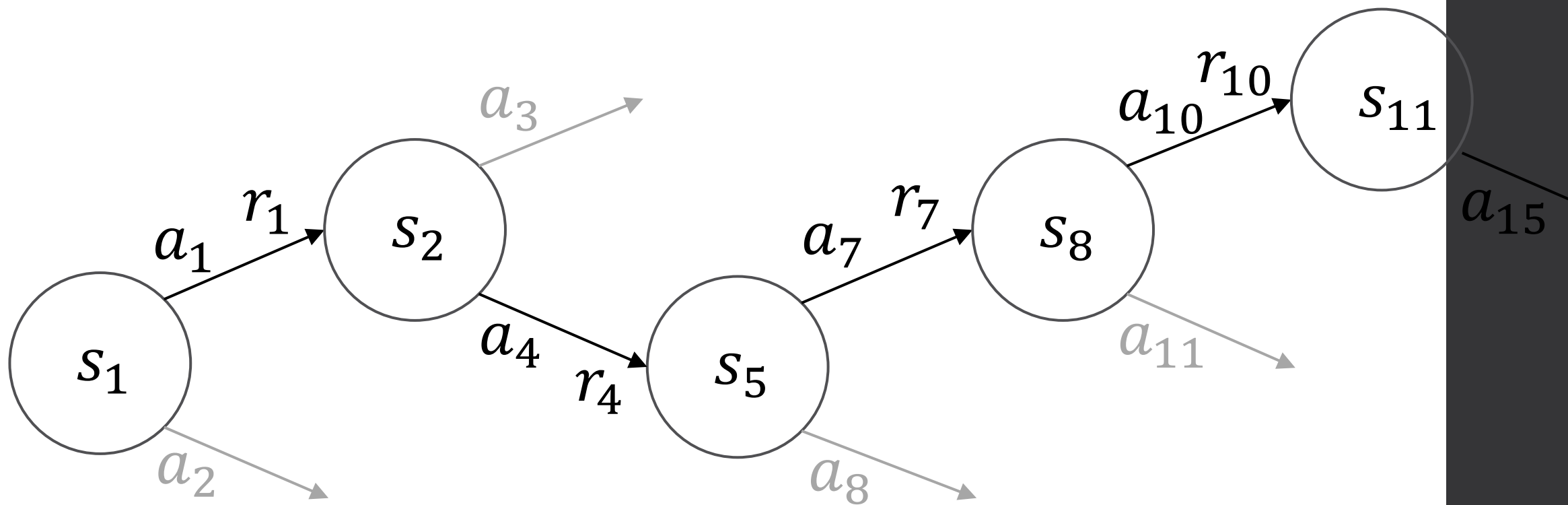
planning



connections

- Dijkstra's algorithm
- A* algorithm
- minimax search
 - two-player games not a problem
- Monte Carlo tree search

roll-outs



roll-outs



everything can be stochastic

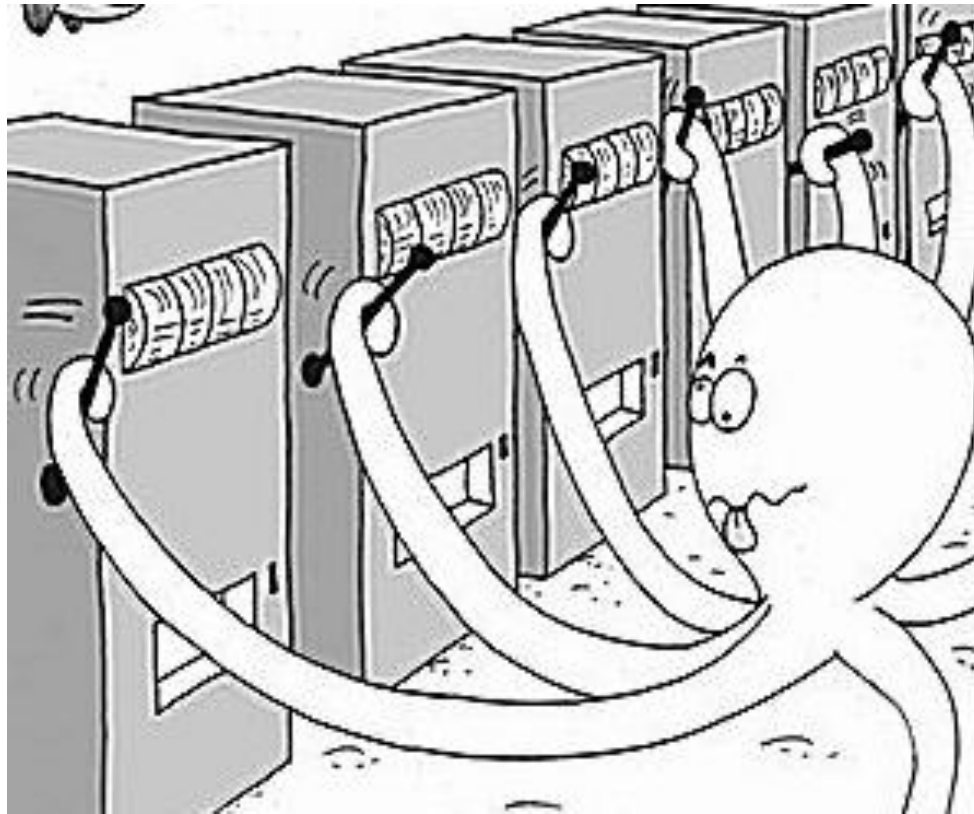
stochastic s'

“icy pond”
20% chance you
“slip” and go
perpendicular to
your intent

-1	-1	-1	-1
-1	-1	-1	-1
start	-1	-1	goal
-10	-10	-10	-10

stochastic r'

multi-armed
bandit



exploration vs. exploitation

- ϵ -greedy policy
 - usually do what looks best
 - ϵ of the time, choose random action

Question

- How does randomness affect π , v , and q ?
 - $\pi: s \rightarrow a$
 - $v: s \rightarrow \sum r$
 - $q: s, a \rightarrow \sum r$

Question

- How does randomness affect π , v , and q ?
 - $\pi: \mathbb{P}(a|s)$
 - $v: s \rightarrow \mathbb{E} \sum r$
 - $q: s, a \rightarrow \mathbb{E} \sum r$

model-free: Monte Carlo returns

- $v(s) = ?$
- keep track and average
- like “planning” from experienced “roll-outs”

non-stationarity

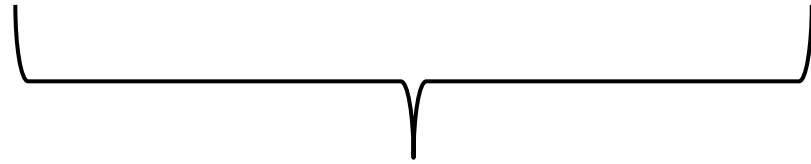
- $v(s)$ changes over time!

moving average

- new mean = old mean + α (new sample - old mean)

moving average

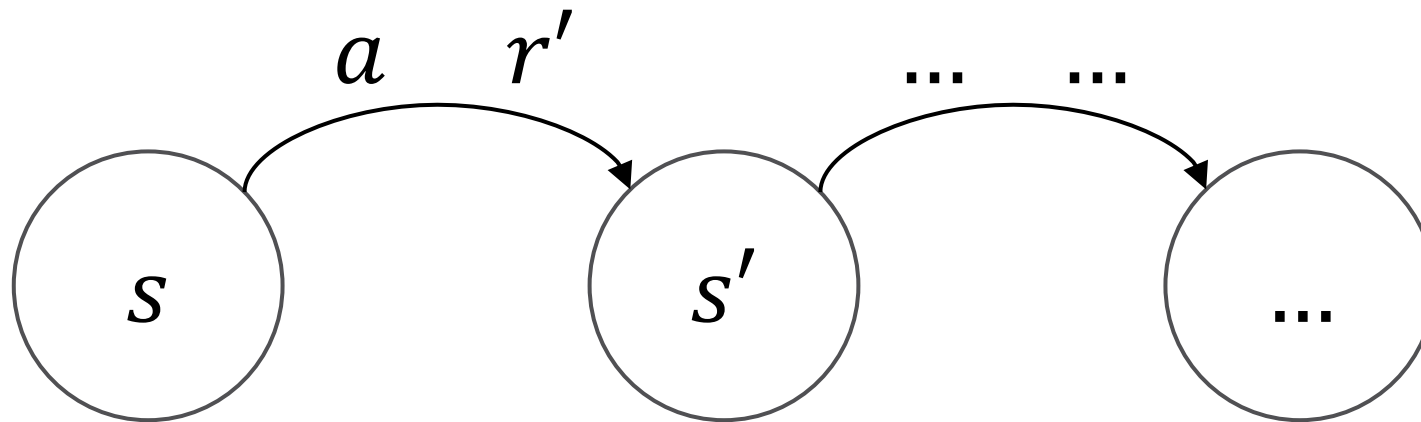
- new mean = old mean + $\alpha(\text{new sample} - \text{old mean})$



$$\nabla \frac{1}{2} (\text{sample} - \text{mean})^2$$

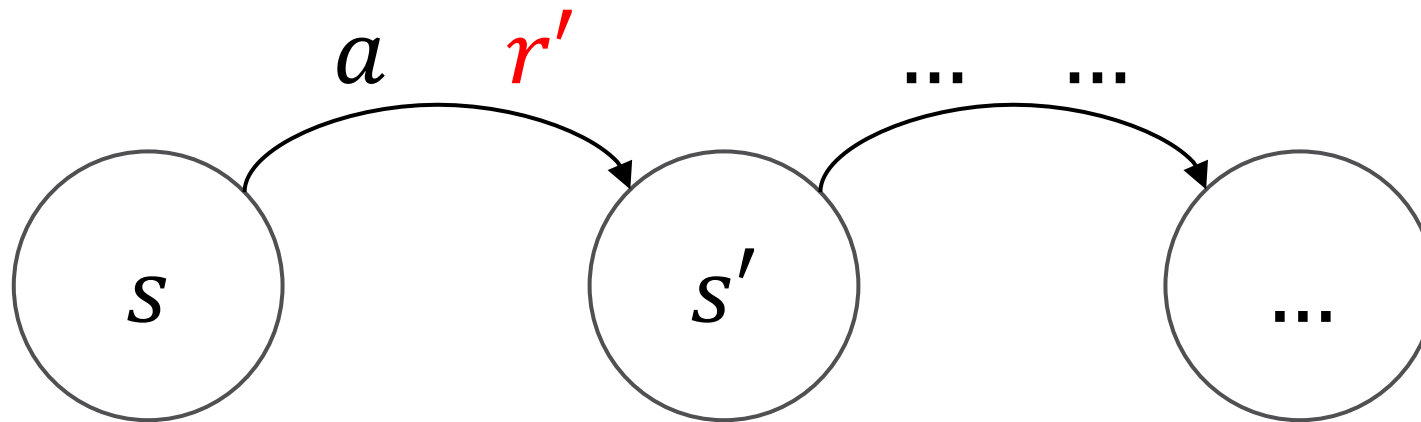
Question

- What's the difference between $v(s)$ and $v(s')$?



Question

- What's the difference between $v(s)$ and $v(s')$?



Bellman equation

- $v(s) = r' + v(s')$

Bellman equation

- $v(s) = r' + v(s')$



Bellman equation

- $v(s) = r' + v(s')$

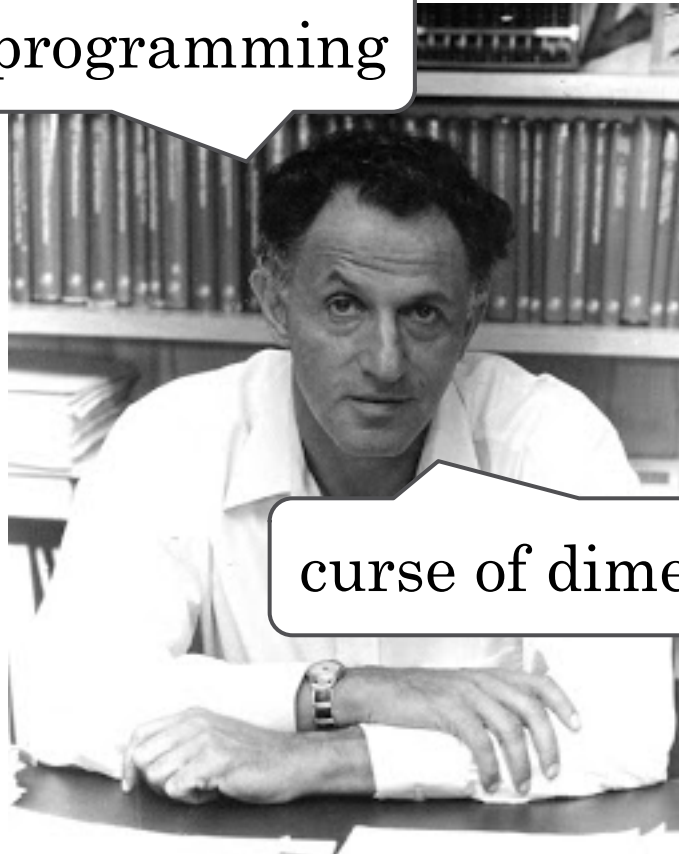
dynamic programming



Bellman equation

- $v(s) = r' + v(s')$

dynamic programming



curse of dimensionality

temporal difference (TD)

- $v(s) = r' + v(s')$
- $0 =? r' + v(s') - v(s)$
 - “new sample”: $r' + v(s')$
 - “old mean”: $v(s)$

Q-learning

- new $q(s, a) = q(s, a) + \alpha(r' + \gamma \max_a q(s', a) - q(s, a))$

on-policy / off-policy

estimate v, q

- with a deep neural network

back to the π

- parameterize π directly
- update based on how well it works
- **REINFORCE**
 - REward Increment = Nonnegative Factor times Offset
Reinforcement times Characteristic Eligibility

policy gradient

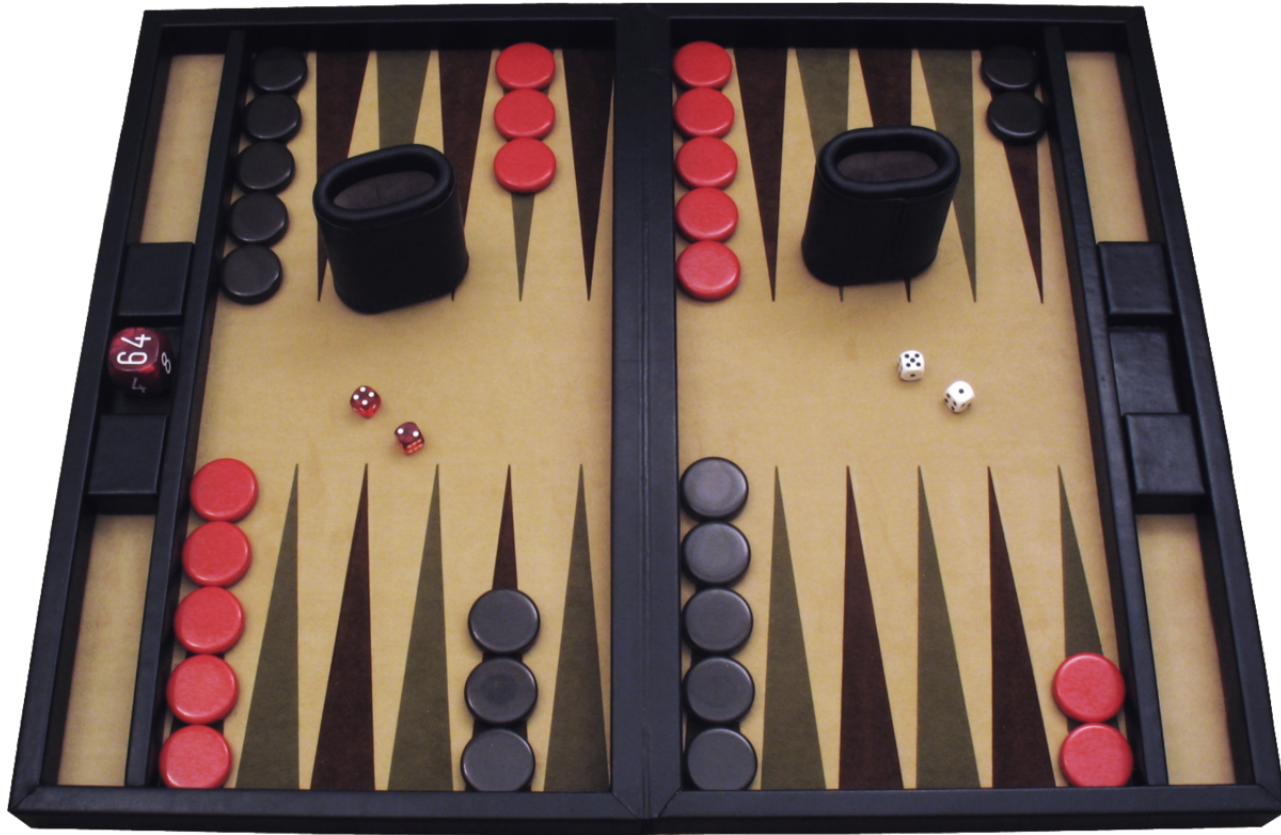
- $\nabla \log(\pi(a|s))$

actor-critic

- π is the actor
- v is the critic
- train π by policy gradient to encourage actions that work out better than v expected

applications: how

Backgammon

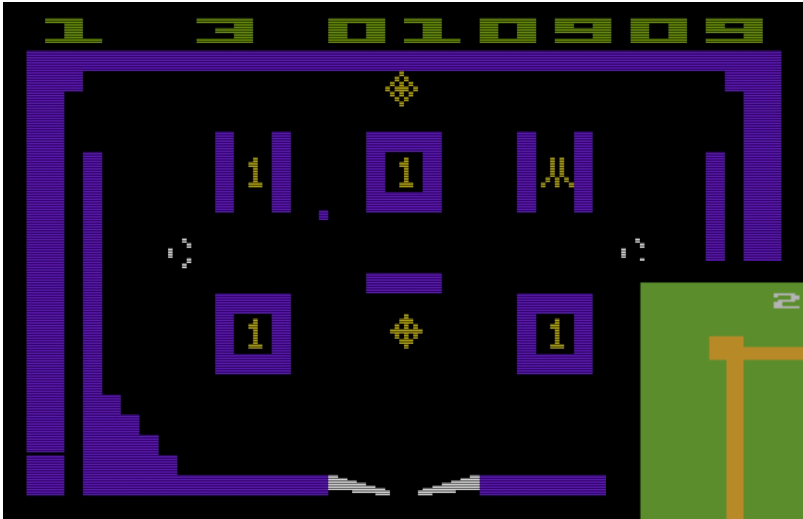


TD-Gammon (1992)

- s is custom features
- v with shallow neural net
- r is 1 for a win, 0 otherwise
- TD(λ)
 - eligibility traces
- self-play
- shallow forward search

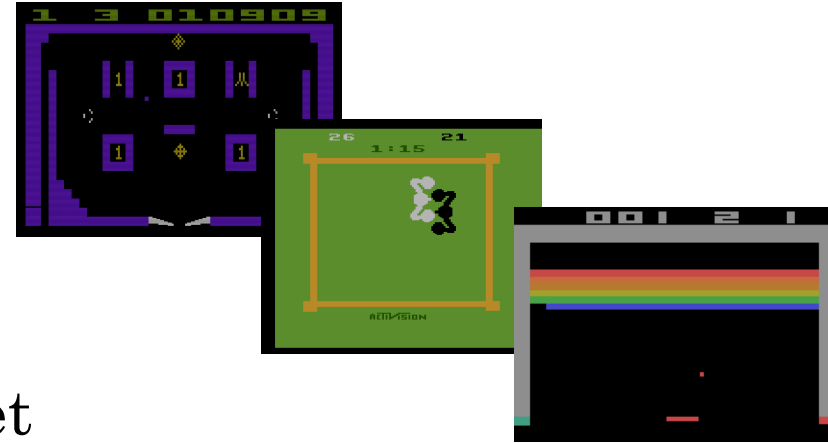


Atari

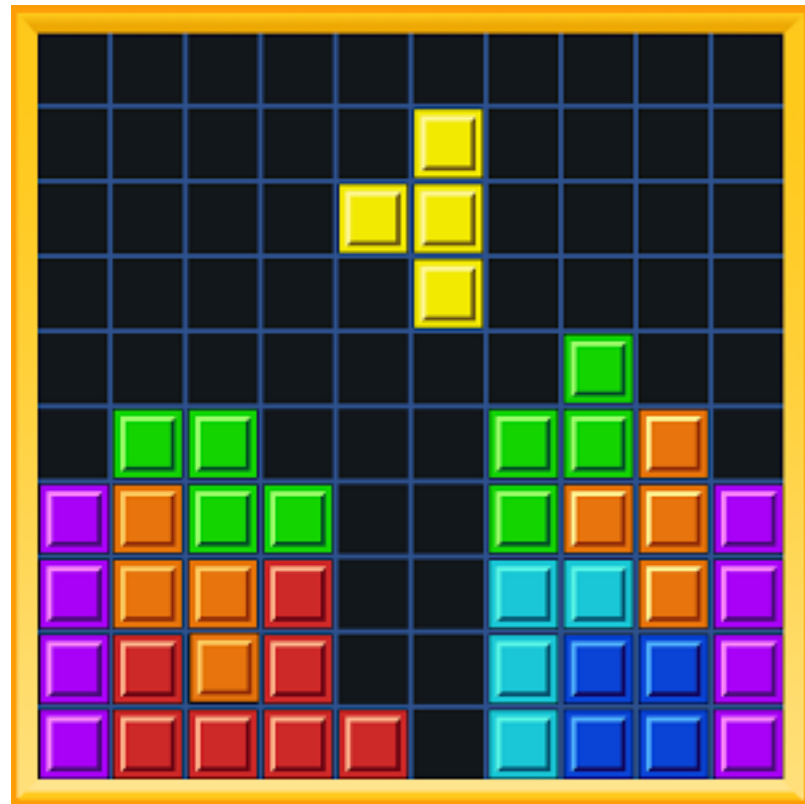


DQN (2015)

- s is four frames of video
- q with deep convolutional neural net
- r is 1 if score increases, -1 if decreases, 0 otherwise
- Q-learning
 - usually-frozen target network
 - clipping update size etc.
- ϵ -greedy
- experience replay

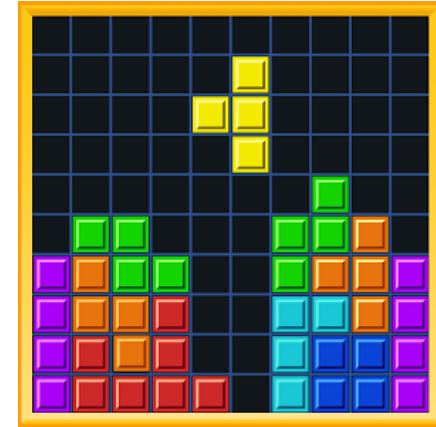


Tetris

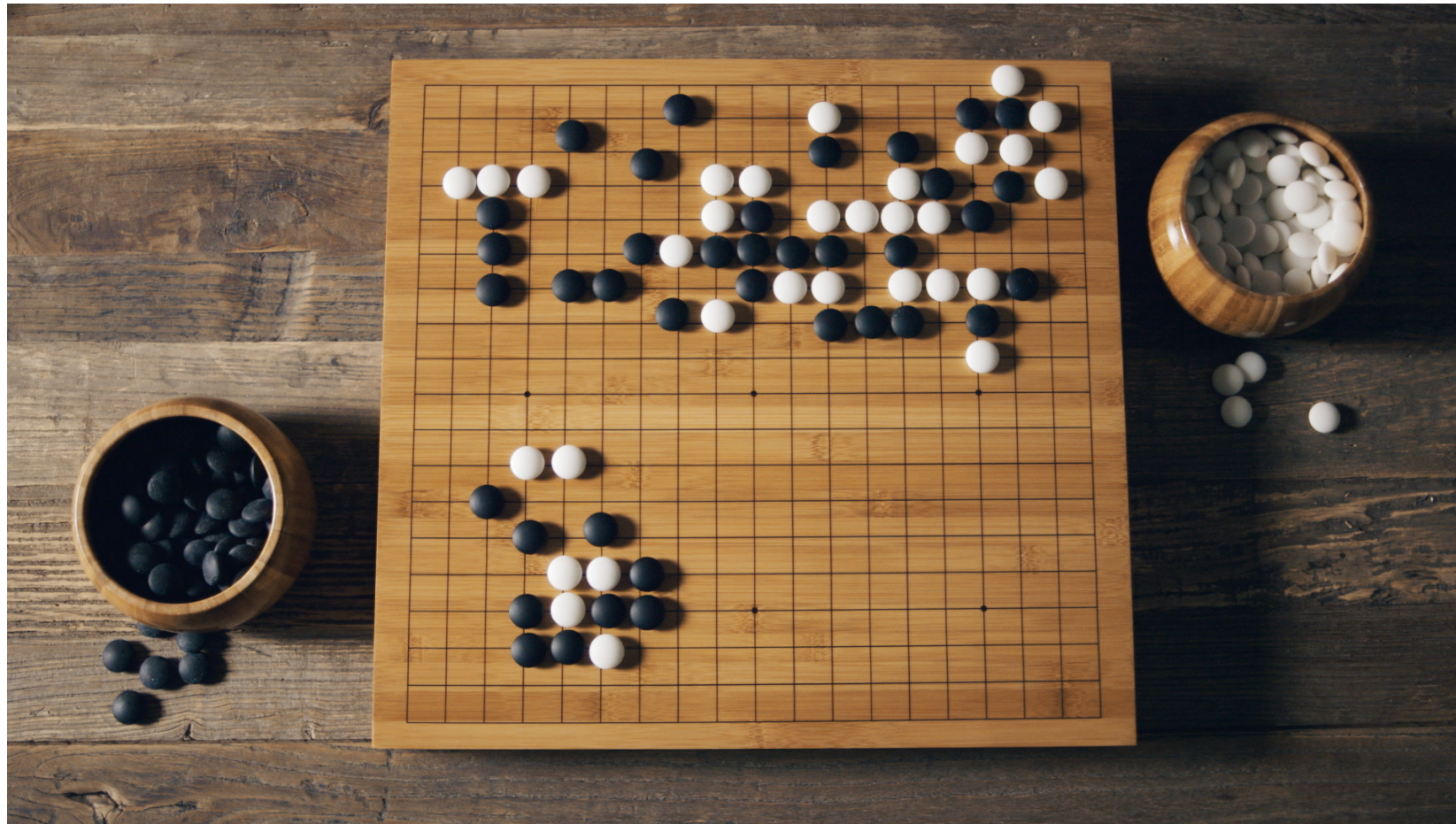


evolution (2006)

- s is custom features
- π has a simple parameterization
- evaluate by final score
- cross-entropy method

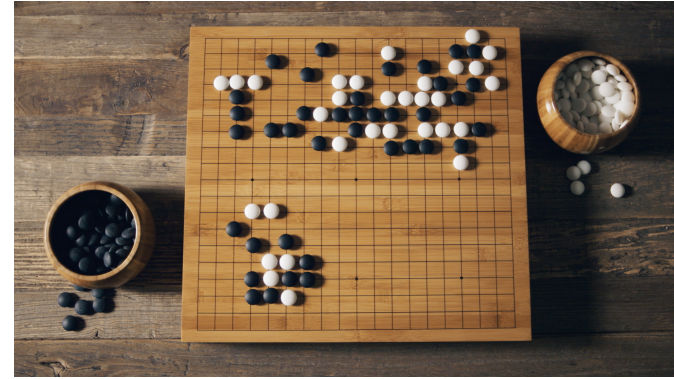


Go



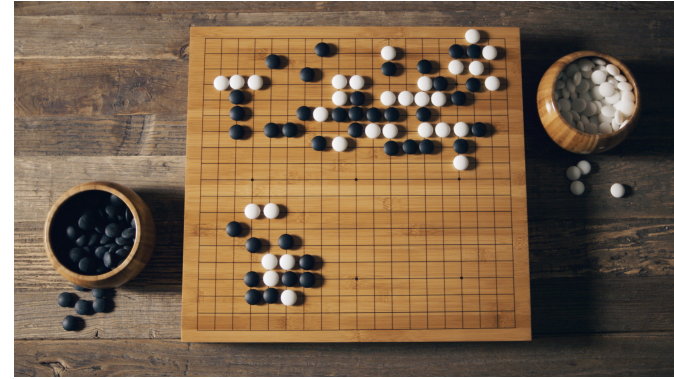
AlphaGo (2016)

- s is custom features over geometry
 - big and small variants
- π_{SL} with deep convolutional neural net, supervised training
- $\pi_{rollout}$ with smaller convolutional neural net, supervised training
- r is 1 for a win, -1 for a loss, 0 otherwise
- π_{RL} is π_{SL} refined with policy gradient peer-play reinforcement learning
- v with deep convolutional neural net, trained based on π_{RL} games
- asynchronous policy and value Monte Carlo tree search
 - expand tree with π_{SL}
 - evaluate positions with blend of v and $\pi_{rollout}$ rollouts



AlphaGo Zero (2017)

- s is simple features over time and geometry
- π, v with deep residual convolutional neural net
- r is 1 for a win, -1 for a loss, 0 otherwise
- Monte Carlo tree search for self-play training and play



onward

Neural Architecture Search (NAS)

- policy gradient where the actions design a neural net
- reward is designed net's validation set performance

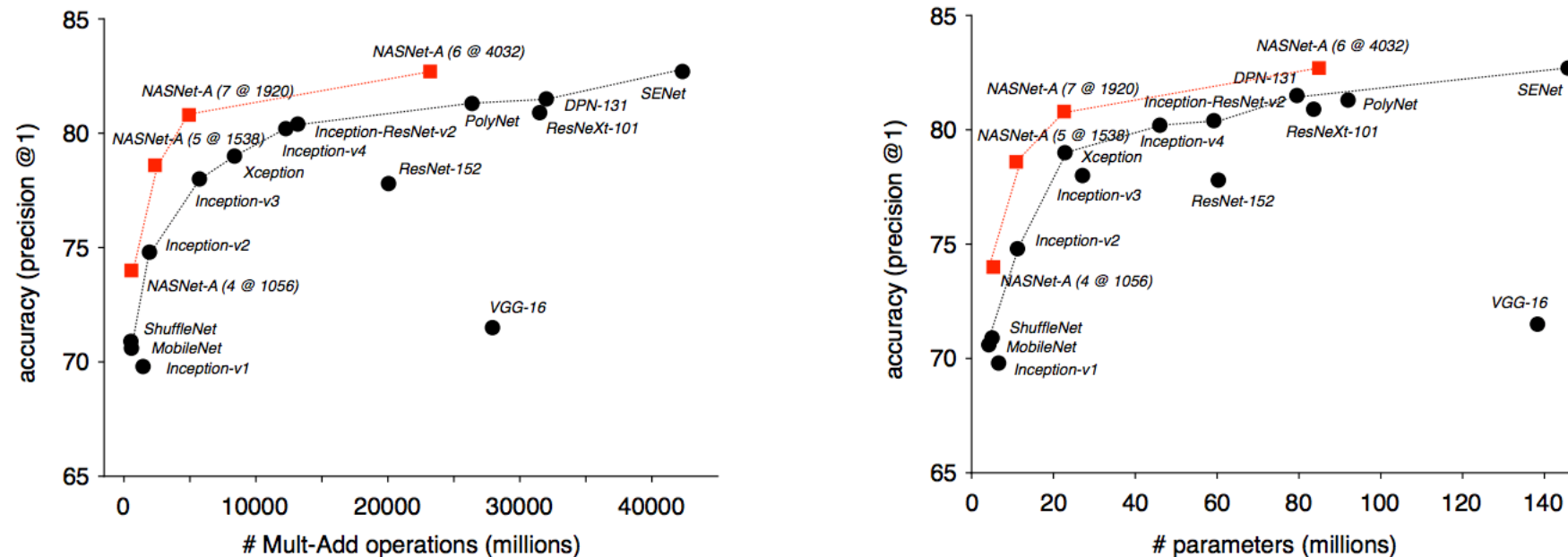


Figure 5. Accuracy versus computational demand (left) and number of parameters (right) across top performing published CNN architectures on ImageNet 2012 ILSVRC challenge prediction task. Computational demand is measured in the number of floating-point multiply-add operations to process a single image. Black circles indicate previously published work and red squares highlight our proposed models.

language



A Deep Reinforcement Learning Chatbot

Iulian V. Serban, Chinnadhurai Sankar, Mathieu Germain, Saizheng Zhang, Zhouhan Lin, Sandeep Subramanian, Taesup Kim, Michael Pieper, Sarath Chandar, Nan Rosemary Ke, Sai Mudumba, Alexandre de Brebisson Jose M. R. Sotelo, Dendi Suhubdy, Vincent Michalski, Alexandre Nguyen, Joelle Pineau and Yoshua Bengio
Montreal Institute for Learning Algorithms, Montreal, Quebec, Canada

Abstract

We present MILABOT: a deep reinforcement learning chatbot developed by the Montreal Institute for Learning Algorithms (MILA) for the Amazon Alexa Prize competition. MILABOT is capable of conversing with humans on popular small talk topics through both speech and text. The system consists of an ensemble of natural language generation and retrieval models, including template-based models, bag-of-words models, sequence-to-sequence neural network and latent variable neural network models. By applying reinforcement learning to crowdsourced data and real-world user interactions, the system has been trained to select an appropriate response from the models in its ensemble. The system has been evaluated through A/B testing with real-world users, where it performed significantly better than competing systems. Due to its machine learning architecture, the system is likely to improve with additional data.

Emergence of Grounded Compositional Language in Multi-Agent Populations

Igor Mordatch¹ Pieter Abbeel^{1,2}

Abstract

By capturing statistical patterns in large corpora, machine learning has enabled significant advances in natural language processing, including in machine translation, question answering, and sentiment analysis. However, for agents to intelligently interact with humans, simply capturing the statistical patterns is insufficient. In this paper we investigate if, and how, grounded compositional language can emerge as a means to achieve goals in multi-agent populations. Towards this end, we propose a multi-agent learning environment and learning methods that bring about emergence of a basic compositional language. This language is represented as streams of abstract discrete symbols uttered by agents over time, but nonetheless has a coherent structure that possesses a defined vocabulary and syntax. We also observe emergence of non-verbal communication such as pointing and guiding when language communication is unavailable.

can capture structural and statistical relationships in language, but they do not capture its functional aspects, or that language happens for purposes of successful coordination between humans. Evaluating success of such imitation-based approaches on the basis of linguistic plausibility also presents challenges of ambiguity and requirement of human involvement.

Recently there has been a surge of renewed interest in the pragmatic aspects of language use and it is also the focus of our work. We adopt a view of (Gauthier & Mordatch, 2016) that an agent possesses an understanding of language when it can use language (along with other tools such as non-verbal communication or physical acts) to accomplish goals in its environment. This leads to evaluation criteria that can be measured precisely and without human involvement.

In this paper, we propose a physically-situated multi-agent learning environment and learning methods that bring about emergence of a basic compositional language. This language is represented as streams of abstract discrete symbols uttered by agents over time, but nonetheless has a coherent structure that possesses a defined vocabulary and

robot control



self-driving cars

- interest
- results?



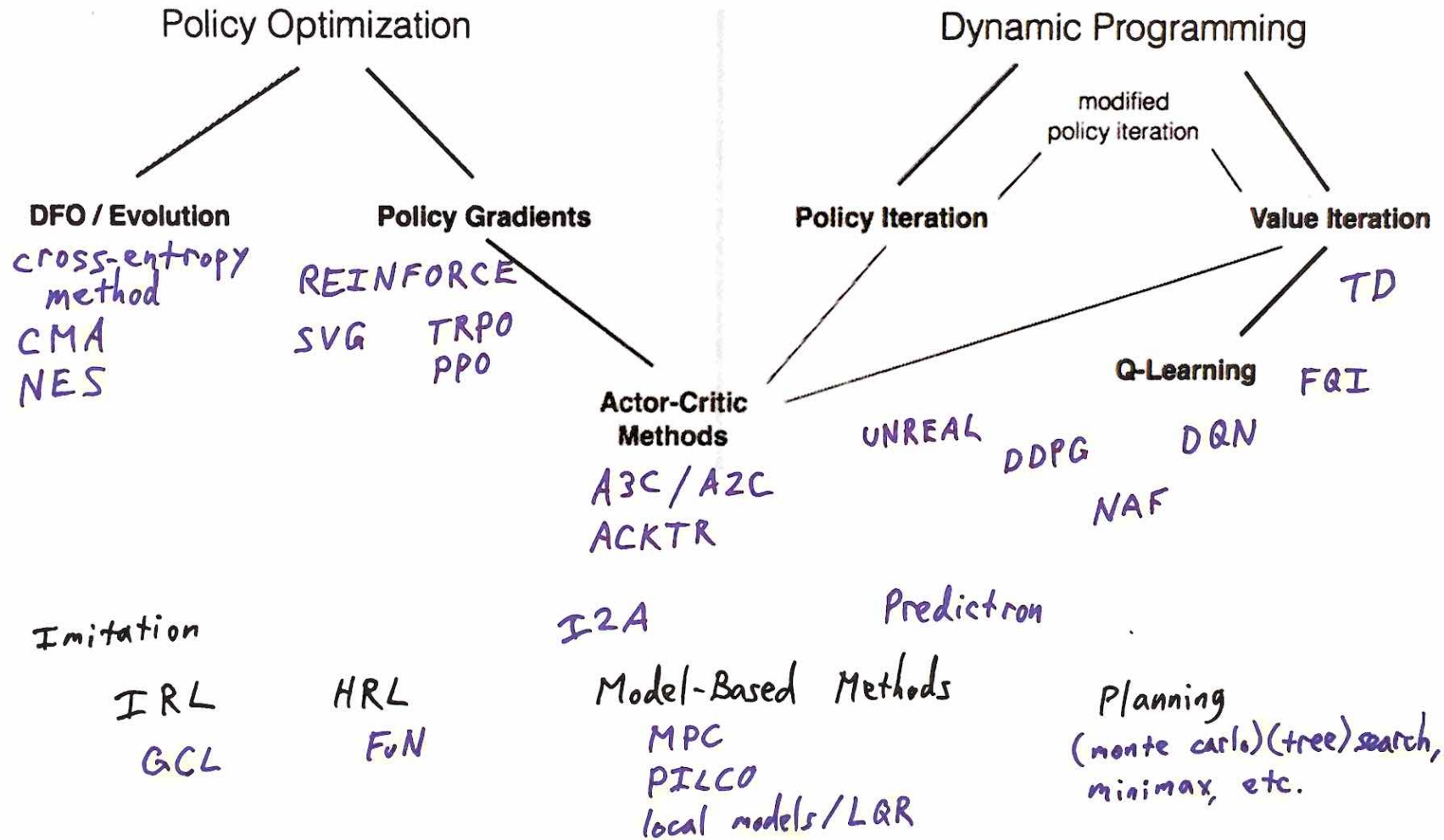
DotA 2



conclusion

(PO)MDP

RL Algorithms Landscape



Tools: GP, GMM, (D)(C)(R)NN, LSTM, ...

reinforcement learning

- $r, s \rightarrow a$

- $\pi: s \rightarrow a$

- $v: s \rightarrow \sum r$

- $q: s, a \rightarrow \sum r$

network architectures for deep RL

- feature engineering can still matter
- if using pixels
 - often simpler than state-of-the-art for supervised
 - don't pool away location information if you need it
- consider using multiple/auxiliary outputs
- consider phrasing regression as classification
- room for big advancements

the lure and limits of RL

- seems like AI (?)
- needs so much data

Question

- Should you use reinforcement learning?

Thank you!

further resources

- planspace.org has these slides and links for all resources
- A Brief Survey of Deep Reinforcement Learning (paper)
- Karpathy's Pong from Pixels (blog post)
- Reinforcement Learning: An Introduction (textbook)
- David Silver's course (videos and slides)
- Deep Reinforcement Learning Bootcamp (videos, slides, and labs)
- OpenAI gym / baselines (software)
- National Go Center (physical place)
- Hack and Tell (fun meetup)