Unsupervised Learning

CS4000: Harnessing AI

October 16, 2019

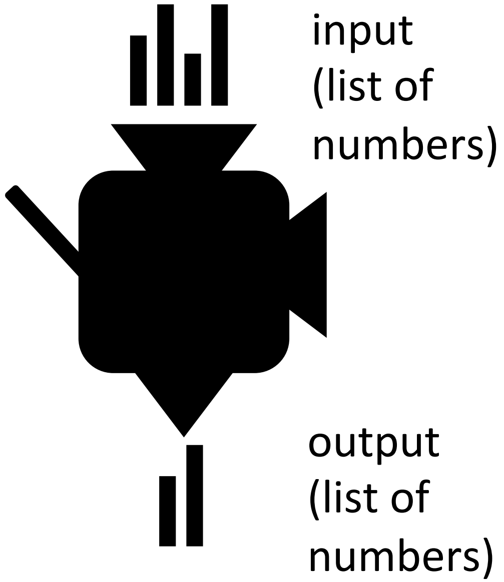
Chris Darken

Outline

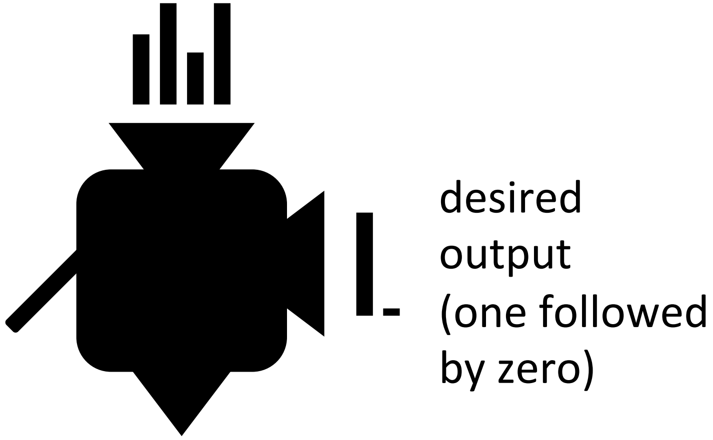
- The data required for supervised learning is not always available
- In some of these situations, it is possible to solve the problem anyway
- These techniques are called "unsupervised learning"
- They often depend upon using the same algorithms as supervised learning, but applied in a clever way
- We will discuss two of the most significant algorithms in this family, anomaly detection and deep reinforcement learning
- We will discuss what these algorithms do and the main principle that makes them work

Black Box Supervised Learning

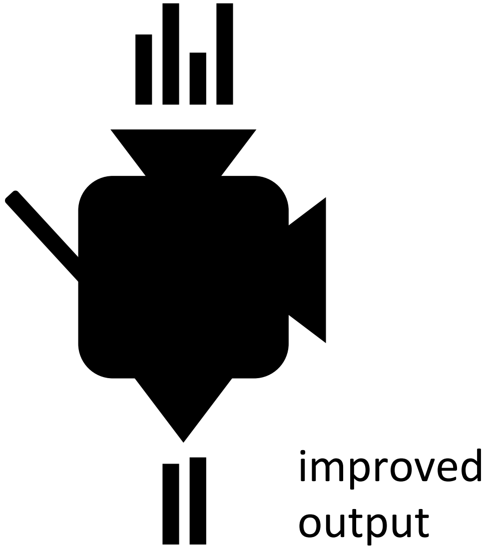
Before Learning



During Learning



After Learning

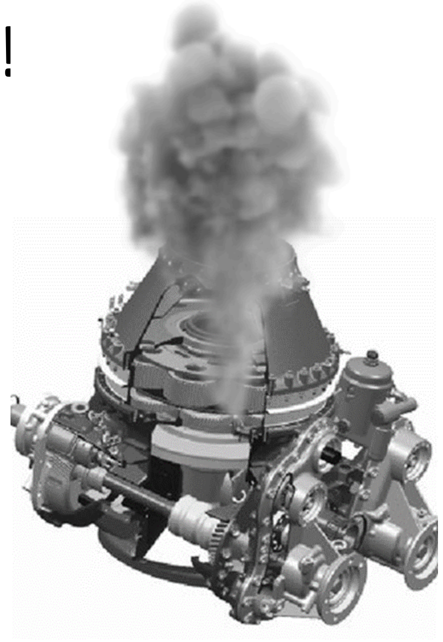


Part 1: Anomaly Detection

Example Task: Gearbox Failure Prediction

- Given a characterization of the vibration of a helo gearbox, determine whether the gearbox is healthy or about to fail
- Despite the smoke in my cartoon, there was no easy way to determine which were about to fail!

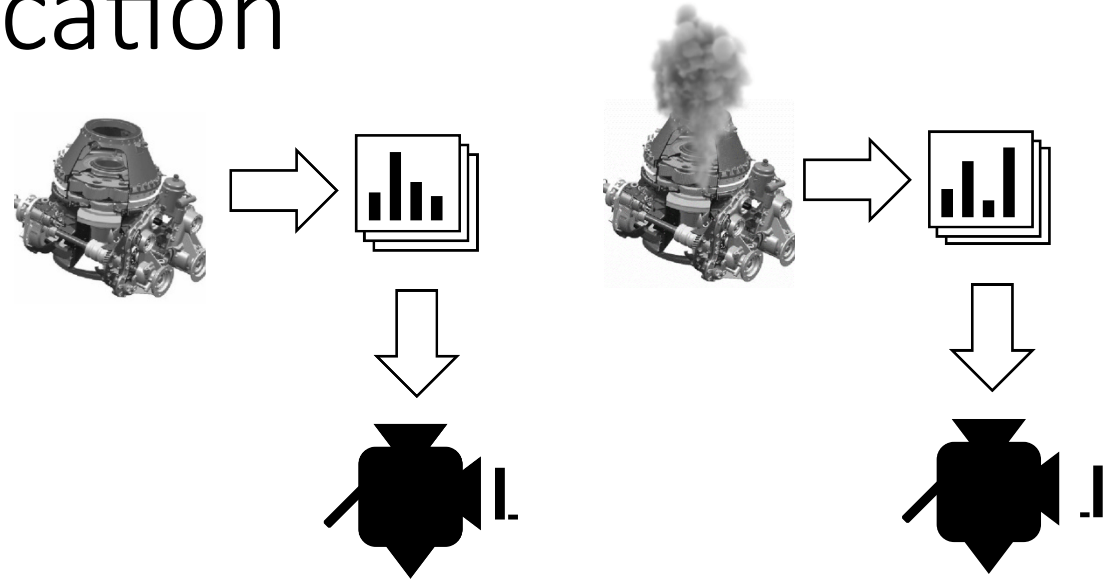
Healthy gearbox



Failing gearbox

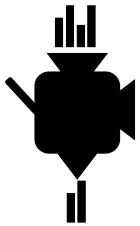
Supervised Learning Approach: Gearbox Classification

Gather a large amount of healthy and failing data



Train neural net on both

Neural net will now classify data from unknown gearboxes



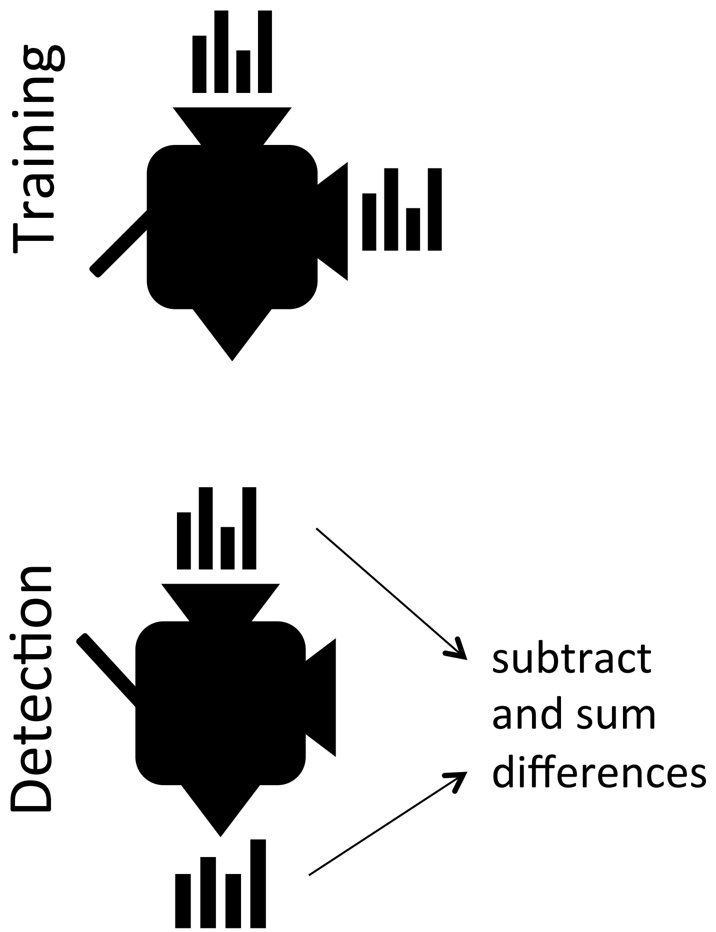
- Roadblock
  - We’re only sure that a gearbox is failing when a helo fails, and more helo failures is the last thing we want

The Unsupervised Approach: Anomaly Detection

- Given a set of data records from **healthy gearboxes only**, determine how similar a new record is
- If it is similar, we consider it normal, otherwise anomalous
- The anomalous records are reported to a human user who makes the determination as to whether "anomalous" means "failing"
- This type of system is called an **anomaly detector**
- The trick here is to find a good measure of similarity. The simplest ones are often not the best.
- Neural autoencoders are one of the most successful measures.

One Anomaly Detector: Neural Autoencoder

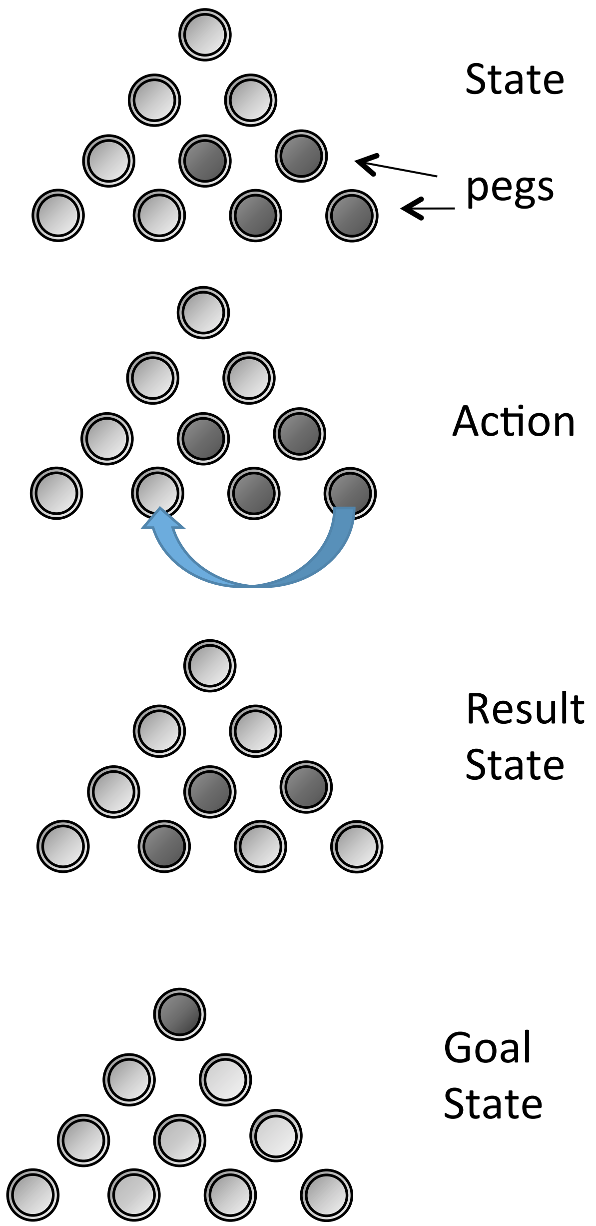
- Input the healthy vibration data into a neural net, and train it to output the **exact same data that was input**
- Neural net is limited so as to make learning the identity function impossible
- After training, the neural net does better on records like the ones it trained on
- More error in the neural net's prediction indicates that new data is different from the training data, i.e. is anomalous



Part 2: Reinforcement Learning

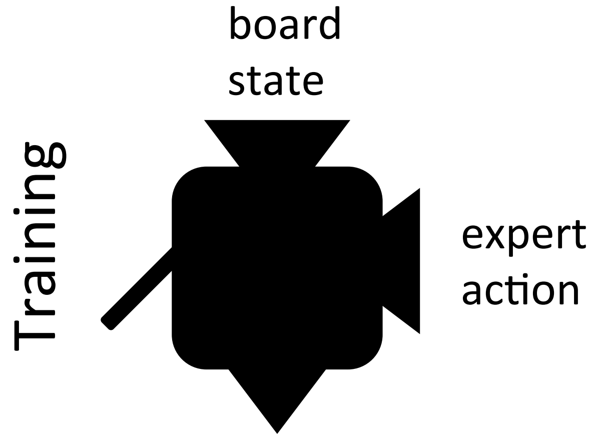
Example Action Selection Task: Peg Jump Puzzle

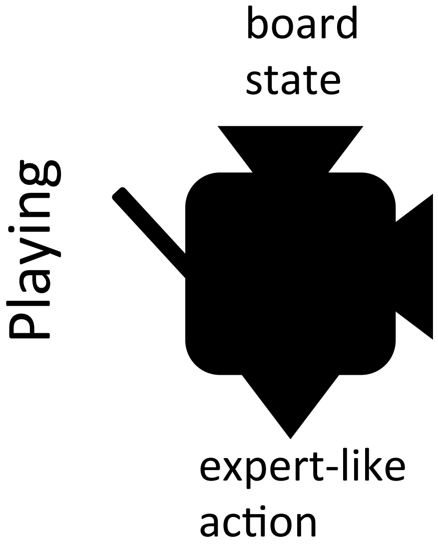
- State
  - Each board position is a state.
- Action
  - Jump one peg over another and remove the jumped peg
- Reward
  - Maximize the long term discounted reward
  - Maximum reward of 1 for achieving the goal state
  - “Reward” of -1 for getting stuck
  - **Zero reward otherwise (almost all the time)**
  - We reduce the reward by a fraction f each move to encourage quick solutions!



Supervised Learning Approach: Behavioral Cloning

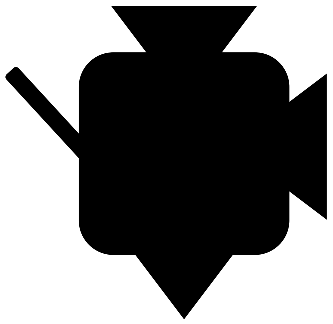
- Procedure
  - Let an expert play the game.
  - Record the states and the actions the expert chooses in those states
  - Use supervised learning to create a neural net that predicts actions from states
  - Then use the neural net to choose actions, imitating the expert's behavior
- Roadblock
  - The neural net isn't perfect copy of the expert's behavior
  - So there will be differences in action choice from the expert
  - This will eventually result in states which an expert would never encounter
  - The neural net's choices on such states will generally be very poor

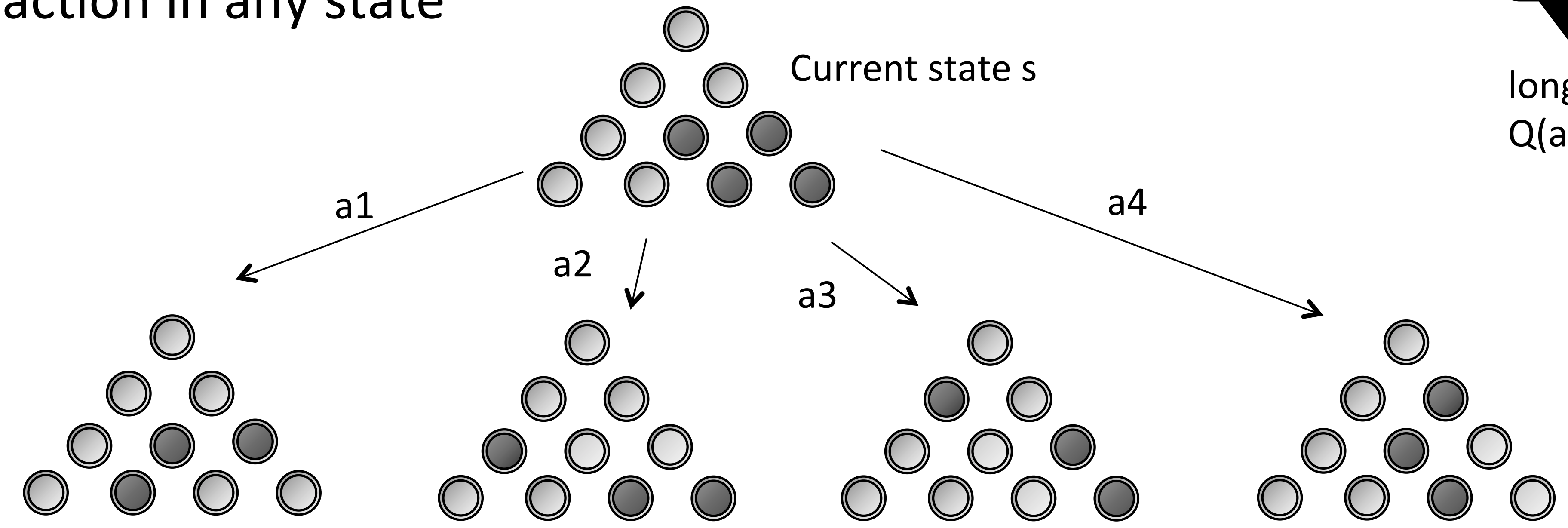




Neural (“Deep”) Reinforcement Learning (1/4)

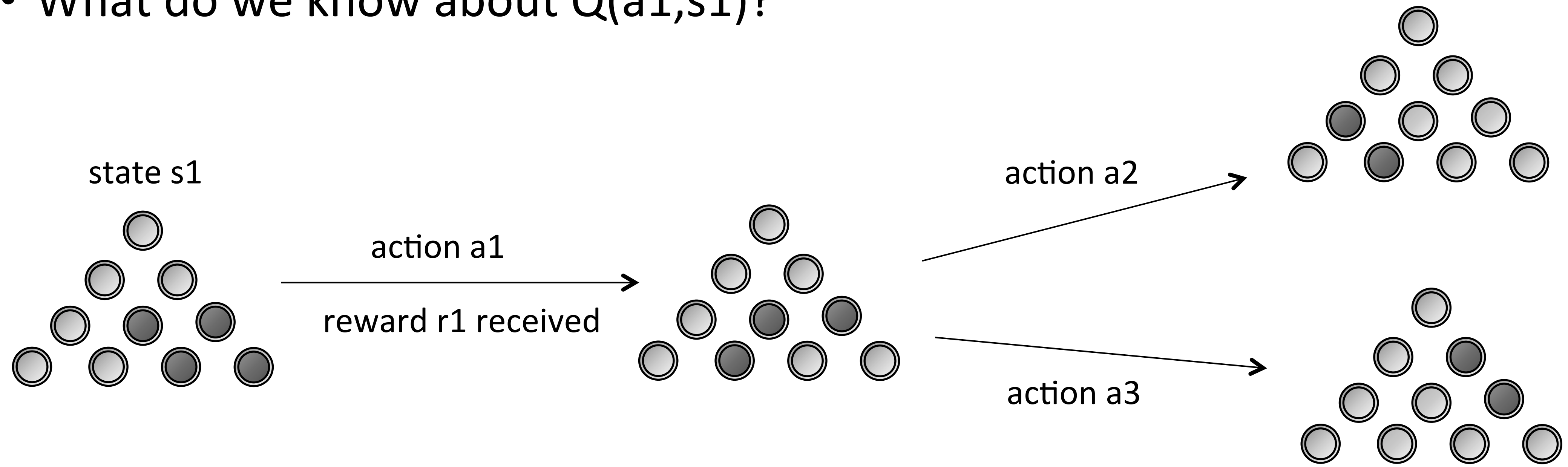
- Key Idea
    - Use a neural net to represent the long term reward function: Q(a,s) where a is an action and s is the current state.
- Such a function would allow easy determination of the best action in any state





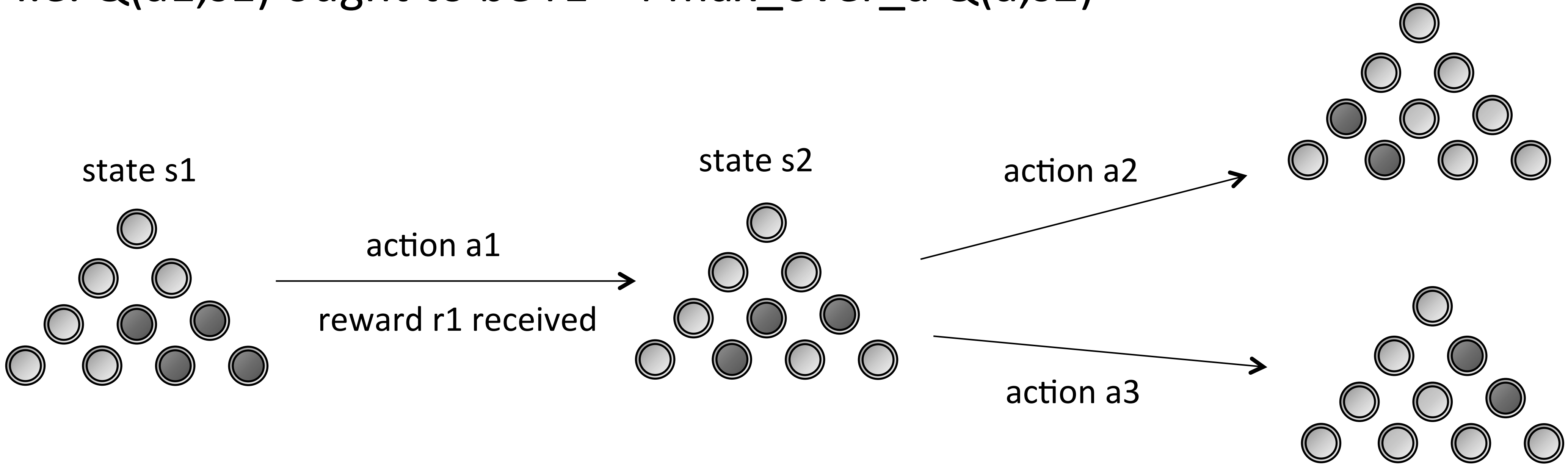
Neural ("Deep") Reinforcement Learning (2/4)

- But how can the neural net be trained?
- Assume that in state s1 action a1 is taken, with immediate reward r1 and ending up in state s2
- In state s2, we have a choice of either action a2 or a3
- What do we know about Q(a1,s1)?



Neural ("Deep") Reinforcement Learning (3/4)

- Q(a1,s1) is the long term reward we get from taking action a1 in state s1
- But the long term reward is just the immediate reward, r1, plus…
- The reward we get later, which will be the discounted long term reward from taking a2 or a3, whichever is better
- I.e. Q(a1,s1) ought to be r1 + f max_over_a Q(a,s2)

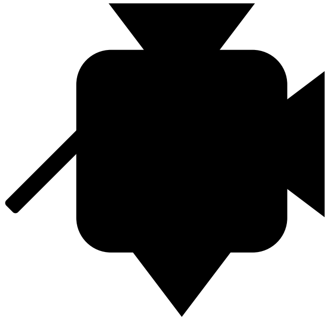


Neural (“Deep”) Reinforcement Learning (4/4)

- Since we know what Q(a1,s1) should be, we can train the neural net to produce it
- Then we can use the corrected neural net to choose our next action
- As we take actions, see new states, and get rewards, we continue to train the neural net, which will become more and more accurate
- And that’s the principle that makes neural reinforcement learning work!

Training

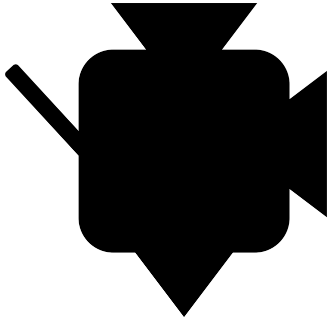
action a1,
state s1



r1 + f max_over_a Q(a,s2)

Playing

action a,
state s

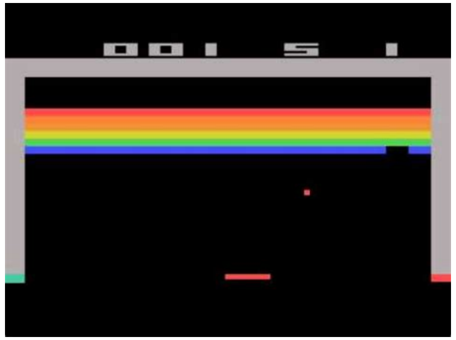


estimate of
Q(a,s)

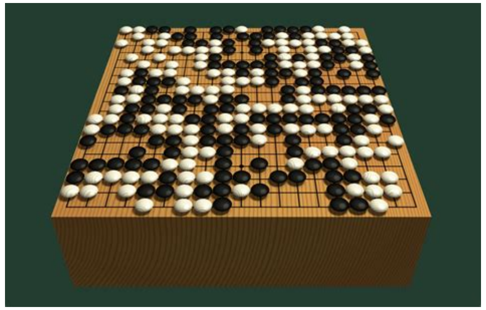
Example of the Algorithm in Action

- https://youtu.be/aX9S6MGh90Y

Superhuman Reinforcement Learners



- DQN (2015)
  - Superhuman play in dozens of Atari 2600 games (subhuman in others)
  - Insightful play in Breakout surprised its developers
- Alpha Go (2016)
  - Beat Lee Sedol (second in international titles at the time) four games to one.
  - Move 37 of the second game is an example of insightful AI play



- Alpha Zero (2017-18)
  - Single system that can learn chess, Shogi, or Go
  - Learns entirely from self-play
- Alpha Star (ongoing)
  - Beat a strong professional StarCraft player (Grzegorz “MaNa" Komincz) 5-0

These are all deep reinforcement learners built by Alphabet’s (formerly Google’s) DeepMind.

Fitness for Military Applications

- Input/output matches many military tasks
- Flexibility (e.g. multiple video games/chess, Shogi, or Go)
- Superhuman performance
- Tactics that surprise all human experts

Issues

- Creating the state representation can be difficult
  - Recurrency (to try to capture how the state depends on older information automatically) and cross training on related tasks (including prediction)
- Reliability
  - There are dozens of algorithm variants and each has dozens of consequential parameters whose values must be set (typically by human trail and error)
- Speed
  - One run can take hours or days on a fast computer, and many runs may be required to achieve success

Questions?