



S&DS 265 / 565
Introductory Machine Learning

Reinforcement Learning (continued)

November 14

Yale

Goings on

- Assignment 5 is out; due November 30
- Quiz 5: posted Thursday (posted in the morning)
- Class on Thursday will be short
- Final exam, Dec 15 at 2pm

Outline

- Quick recap
 - Q -learning
 - Illustration on taxi problem
 - RL concepts
- Deep reinforcement learning
- Learning to play Atari games
- Neuroscience connection

Reinforcement learning

- An agent interacts with an environment
- The actions the agent takes change the state of the environment
- The agent receives rewards for each action, and seeks to maximize the total cumulative reward

Reinforcement learning is a framework for sequential decision making to achieve a long-term goal.

Reinforcement learning: Formalization

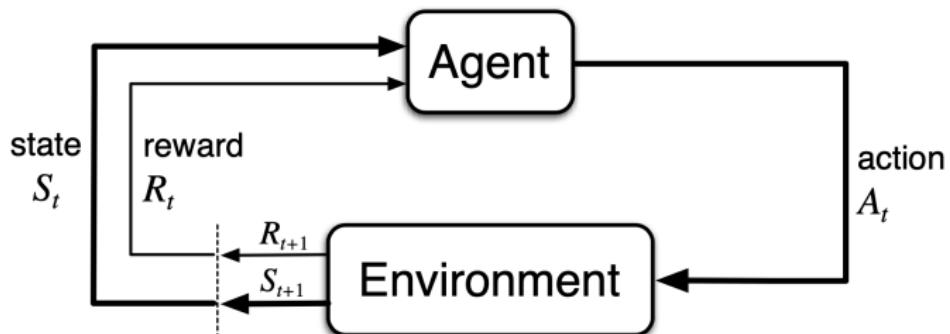
- The environment is in state s at a given time
- The agent takes action a
- The environment transitions to state $s' = \text{next}(s, a)$
- The agent receives reward $r = \text{reward}(s, a)$

Reinforcement learning: Formalization

- The environment is in state s at a given time
- The agent takes action a
- The environment transitions to state $s' = \text{next}(s, a)$
- The agent receives reward $r = \text{reward}(s, a)$

This is said to be a *Markov decision process*. It's "Markov" because the next state only depends on the current state and the action selected. It's a "decision process" because the agent is making choices of actions in a sequential manner.

RL setup



Recall: Taxi problem

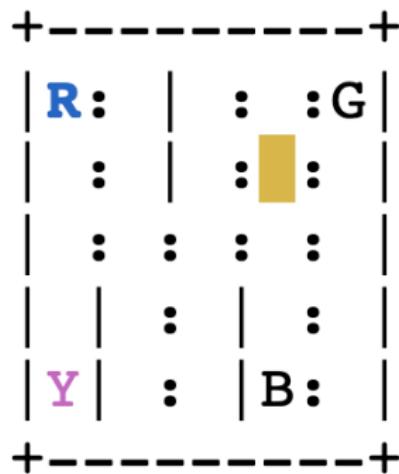
A taxicab drives around the environment, picking up and delivering a passenger at four locations



Recall: Taxi problem

A taxicab drives around the environment, picking up and delivering a passenger at four locations

"Ascii art" rendition:



+ Code + Text

Reinforcement Learning

In reinforcement learning, an agent interacts with the environment, experiencing a series of rewards based on its actions. The agent seeks to maximize its rewards by developing a strategy that learns to choose appropriate actions in each state.



In this notebook we demo the Q-learning algorithm, one of the fundamental algorithms of reinforcement learning. We illustrate Q-learning on the taxicab problem formulated by Tom Dietterich in the paper "Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition", as developed in the code from [OpenAI gym](#). Our presentation follows [this tutorial](#).

We'll need the OpenAI gym package. This can be installed as shown below. We'll display some simple graphics using `IPython.display`.

```
[ ] #!pip install gym
import gym
import numpy as np
from IPython.display import clear_output
from time import sleep
```

The environment is a simple grid, with some barriers inserted to make things more interesting. A taxicab drives around the environment, picking up and delivering a passenger at four locations. A graphic of the environment is shown below.



Q -learning

Policy:

When in state s :

with probability ε :

take a random action

with probability $1 - \varepsilon$:

take action a^* that maximizes $Q(s, a)$

Q -learning update

Update:

$$Q(s, a) \leftarrow$$

$$Q(s, a) + \alpha \left(\text{reward}(s, a) + \gamma \max_{a'} Q(\text{next}(s, a), a') - Q(s, a) \right)$$

Q -learning update

Update:

$$Q(s, a) \leftarrow$$

$$Q(s, a) + \alpha \left(\text{reward}(s, a) + \gamma \max_{a'} Q(\text{next}(s, a), a') - Q(s, a) \right)$$

- When action a is taken in state s , reward $\text{reward}(s, a)$ is given
- Then, the algorithm moves to a new state $\text{next}(s, a)$

Q -learning update

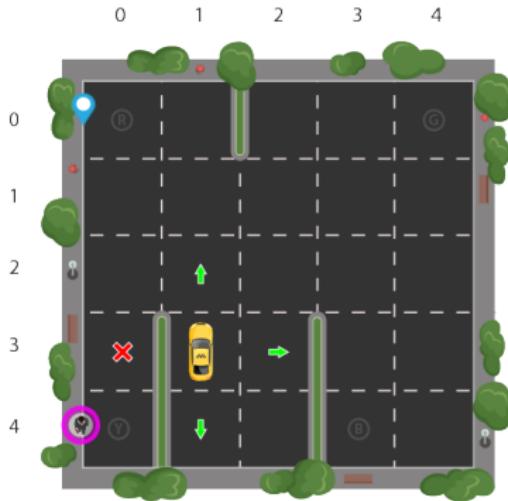
Update:

$$Q(s, a) \leftarrow$$

$$Q(s, a) + \alpha \left(\text{reward}(s, a) + \gamma \max_{a'} Q(\text{next}(s, a), a') - Q(s, a) \right)$$

- Cumulative future reward of this action is $\max_{a'} Q(\text{next}(s, a), a')$
- Future rewards discounted by factor $\gamma < 1$
- Trades off short-term against long-term rewards

No map? No problem!



In Q-learning, the agent learns to navigate optimally without learning an explicit map.

Important RL concepts

Policy: A mapping from states to actions. An algorithm/rule to make decisions at each time step, designed to maximize the long term reward.

Important RL concepts

Value function: A mapping from states to total reward. The total reward the agent can expect to accumulate in the future, starting from that state.

Rewards are short term. Values are predictions of future rewards.

Bellman equation



The optimal value function is the largest expected discounted long term reward starting from that state.

Bellman equation: Deterministic case

The optimality condition for the Q -function is

$$Q_*(s, a) = \text{reward}(s, a) + \gamma \max_{a'} Q_*(\text{next}(s, a), a')$$

and then $v_*(s) = \max_{a'} Q_*(s, a')$

Bellman equation: Deterministic case

The optimality condition for the value function v_* is

$$v_*(s) = \max_a \{ \text{reward}(s, a) + \gamma v_*(\text{next}(s, a)) \}$$

Q -learning update

Note how this makes sense in terms of the update rule:

$$Q(s, a) \leftarrow$$

$$Q(s, a) + \alpha \left(\text{reward}(s, a) + \gamma \max_{a'} Q(\text{next}(s, a), a') - Q(s, a) \right)$$

Bellman equation: Deterministic case

If we know Q_ , we know v_* :*

Bellman equation: Deterministic case

If we know Q_* , we know v_* :

$$\begin{aligned} v_*(s) &= \max_a Q_*(s, a) \\ &= \max_a \left\{ \text{reward}(s, a) + \gamma \max_{a'} Q_*(\text{next}(s, a), a') \right\} \\ &= \max_a \left\{ \text{reward}(s, a) + \gamma v_*(\text{next}(s, a)) \right\} \end{aligned}$$

which is the Bellman equation

Random environments: Assn 5



- Problem 4: Frozen: Navigating a random environment (25 points)

In class we introduce the Q-learning algorithm using the Taxi problem from the OpenAI gym package. I toy reinforcement learning problem, called "Frozen Lake". In this problem you need to walk over a grid being careful not to fall through holes in the ice. (The author of this problem fell through the ice in a fr was quite an experience...) The environment is a simple 4x4 grid, and the goal is to walk from one corn

Will help gain intuition for deterministic vs. random environments

Bellman equation: Random environments

Q-function optimality

$$Q_*(s, a) = \mathbb{E} \left[R_{t+1} + \gamma \max_{a'} Q_*(S_{t+1}, a') \mid S_t = s, A_t = a \right]$$

Bellman equation: Random environments

Value function optimality

$$v_*(s) = \max_a \mathbb{E} [R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a]$$

Comment on Q-learning

- Q-learning is an example of *temporal difference (TD) learning*
- It is an “off-policy” approach that is practical if the space of actions is small
- Value iteration is analogous approach for learning the value function for a given policy π , a (possibly random) choice of action for each state

Deep reinforcement learning: Motivation

- Direct implementation of Q-learning only possible for small state and action spaces
- For large state spaces we need to map states to “features”
- Deep RL uses a multilayer neural network to learn these features and the Q -function

Starting point: Bellman equation

$$Q(s, a; \theta) = \mathbb{E} \left[R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a'; \theta) \mid S_t = s, A_t = a \right]$$

- The parameters θ are weights in a neural network
- The state S_{t+1} is the input to the network
- Each possible action a is assigned a value by the network

How do we solve this implicit equation for the network parameters?

Strategy

Let y_t be expected reward given the current model:

$$y_t = R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a'; \theta_{\text{current}})$$

Adjust the parameters θ to make the squared error small (SGD):

$$(y_t - Q(s, a; \theta))^2$$

Strategy

Let y_t be expected reward given the current model:

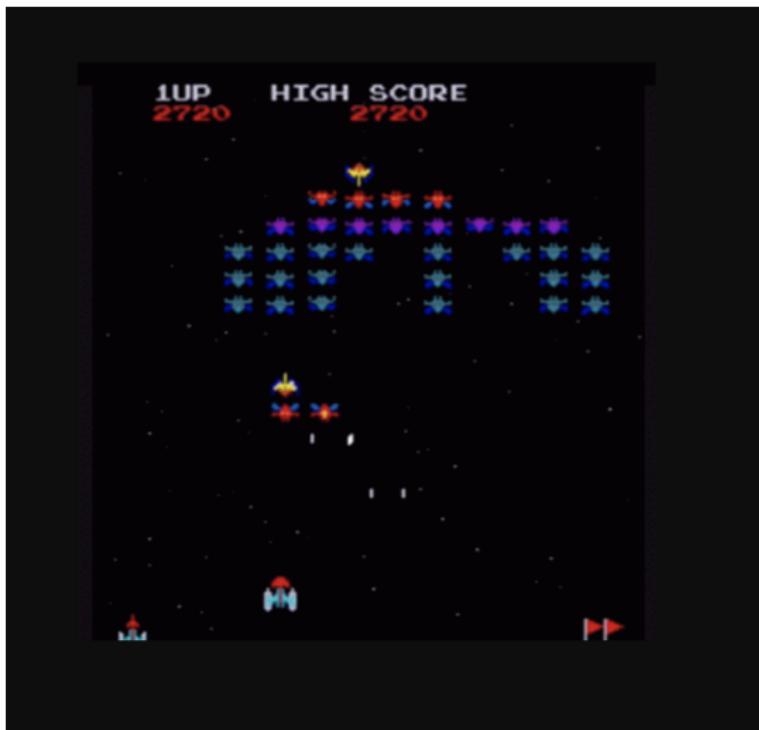
$$y_t = R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a'; \theta_{\text{current}})$$

Adjust the parameters θ to make the squared error small (SGD):

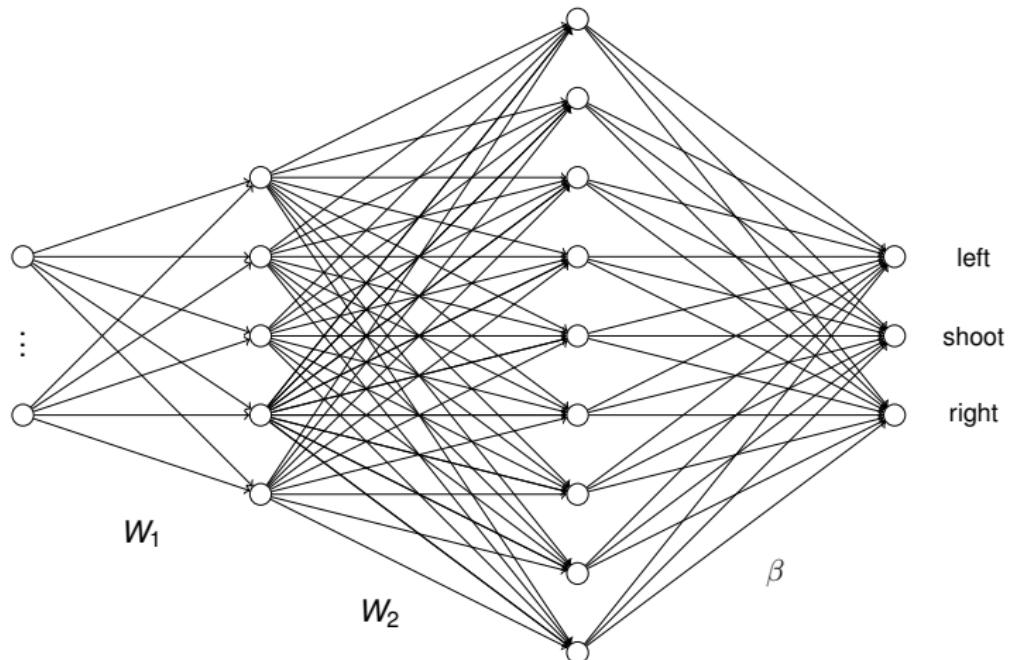
$$(y_t - Q(s, a; \theta))^2$$

How? Carry out SGD with backpropagation

Space Invaders



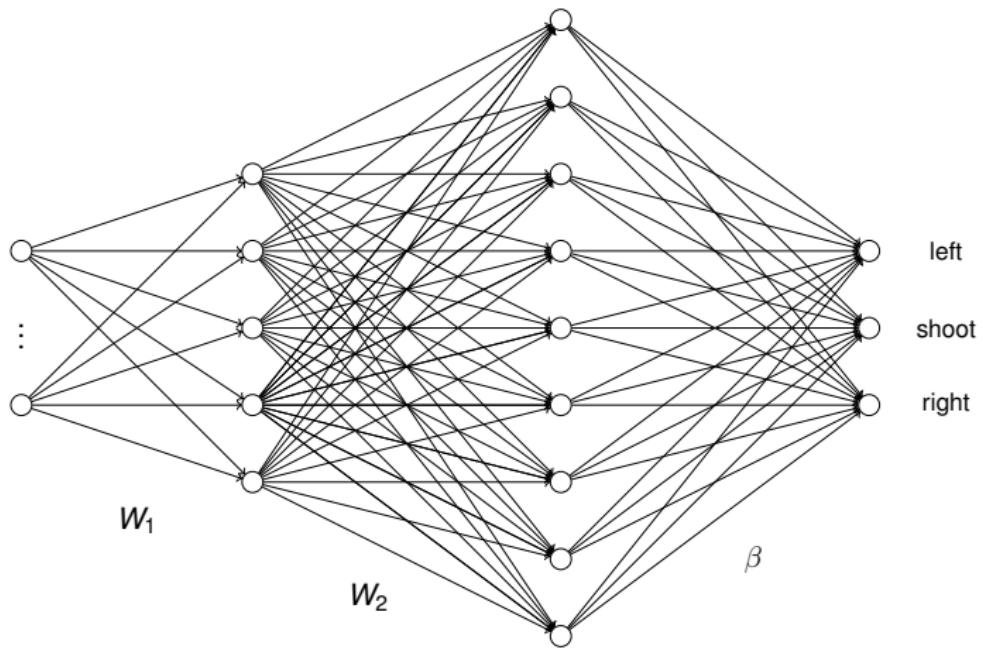
Space Invaders: Q-learning framework



Space Invaders: Q-learning framework



(multiple frames)



DeepMind work

- Images cropped to 84×84 pixels; 128 color palette; input sequence of 4 frames; reward is score
- 3-layer convolutional neural network, ReLU nonlinearity, final layer fully connected, 256 neurons
- Q-learning carried out over minibatches of playing sequences that are “remembered and replayed”

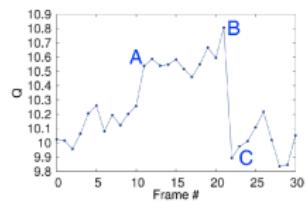
“Playing Atari with Deep Reinforcement Learning,” Mnih et al., DeepMind Technologies, 2013. Larger versions reached human level performance on 29/49 games. Google acquired DeepMind and used Deep Q-learning to save energy (and money) in data centers.

DeepMind work

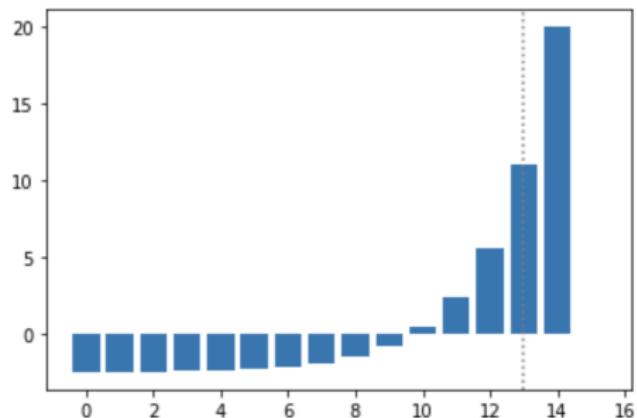
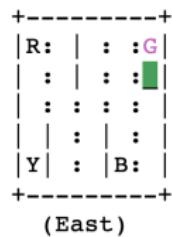
- Identical architecture used for seven games (Beam Rider, Breakout, Enduro, Pong, Q*bert, Seaquest, Space Invaders)
- Each game trained for 100 epochs (50,000 minibatch weight updates / epoch), about 50 hours
- Surpasses human expert on three of seven games

"Playing Atari with Deep Reinforcement Learning," Mnih et al., DeepMind Technologies, 2013. Larger versions reached human level performance on 29/49 games. Google acquired DeepMind and used Deep Q-learning to save energy (and money) in data centers.

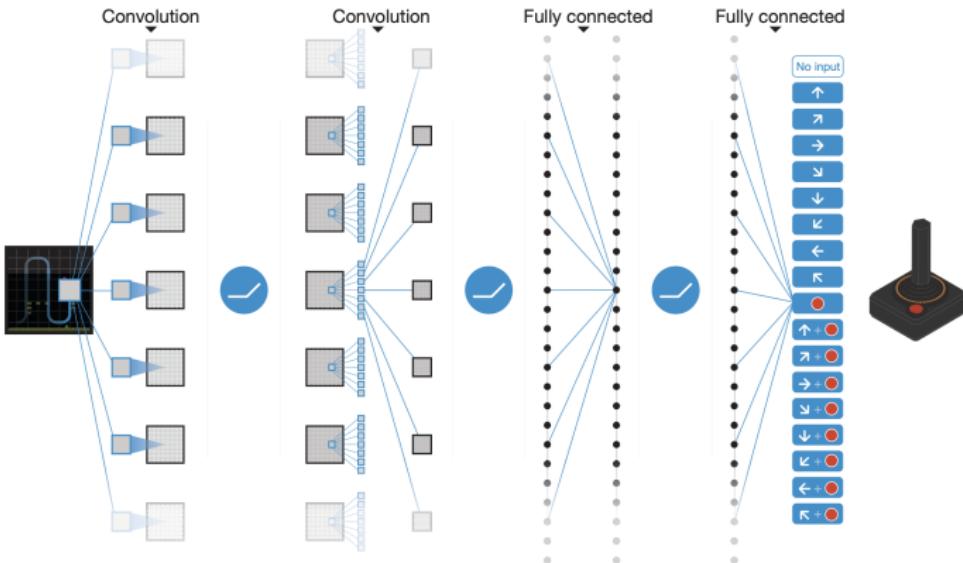
Visualizing value function



Visualizing value function: Taxi

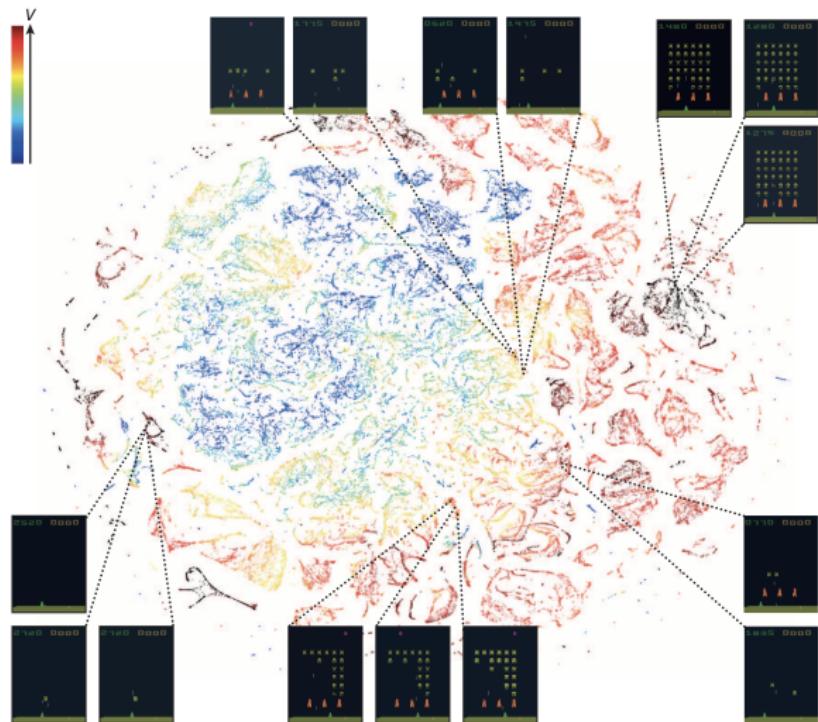


Second generation DQN



<https://storage.googleapis.com/deepmind-data/assets/papers/DeepMindNature14236Paper.pdf>

Second generation DQN: Interpretation



t-SNE representations of last layer for Space Invaders, color-coded for v_* .

DeepMind AI reduces energy used for cooling Google data centers by 40%

Jul 20, 2016 · 4 min read

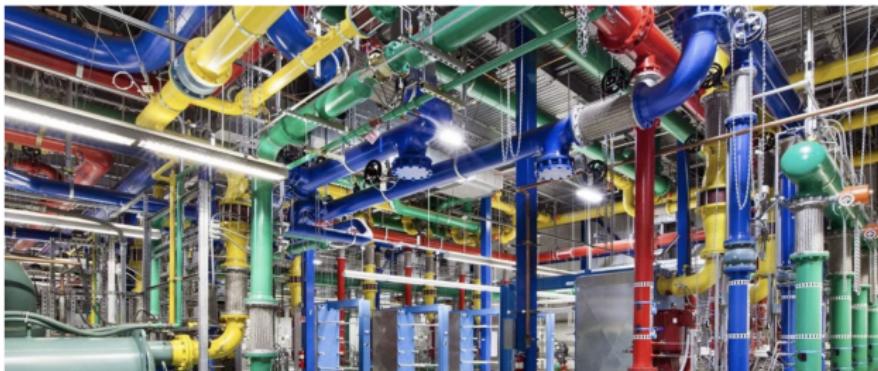


Rich Evans
Research Engineer,
DeepMind



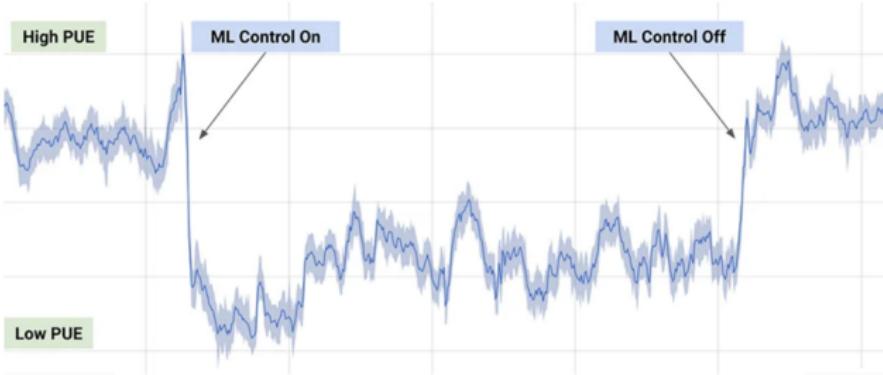
Jim Gao
Data Center Engineer,
Google

Share



The long term reward from Google's investment was very large 😊

We tested our model by deploying on a live data center. The graph below shows a typical day of testing, including when we turned the machine learning recommendations on, and when we turned them off.



Google DeepMind graph showing results of machine learning test on power usage effectiveness in Google data centers

The long term reward from Google's investment was very large ☺

A Neural Substrate of Prediction and Reward

Wolfram Schultz, Peter Dayan, P. Read Montague*

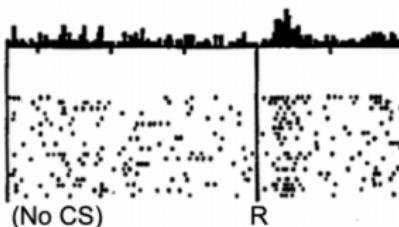
The capacity to predict future events permits a creature to detect, model, and manipulate the causal structure of its interactions with its environment. Behavioral experiments suggest that learning is driven by changes in the expectations about future salient events such as rewards and punishments. Physiological work has recently complemented these studies by identifying dopaminergic neurons in the primate whose fluctuating output apparently signals changes or errors in the predictions of future salient and rewarding events. Taken together, these findings can be understood through quantitative theories of adaptive optimizing control.

Science 1997

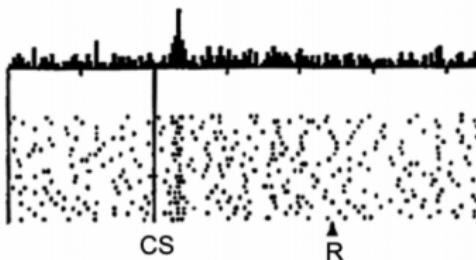
Neuroscience connection

Do dopamine neurons report an error
in the prediction of reward?

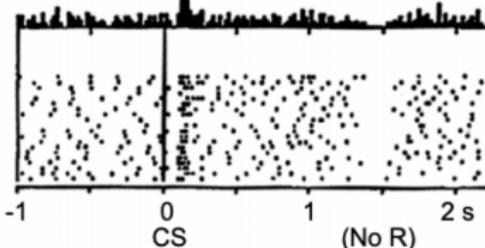
No prediction
Reward occurs



Reward predicted
Reward occurs



Reward predicted
No reward occurs



Dopamine

- Dopamine neurons fire when there is an error — expectations are violated
- Initially they fire when a reward is received when none was expected
- When a light is flashed before the reward (conditioning stimulus CS), dopamine firing moves to the stimulus, as the animal learns the association
- When the reward is removed, an error in expectation results; dopamine is again released

Summary

- For large state spaces, can't estimate Q -function directly
- Deep Q -learning: Using a (deep) neural network with state as input, value assigned to each possible action
- Success at increasing number of applications, outside games
- Close connection to neuroscience of behavior and reward