Applied Artificial Intelligence

Session 28: Reinforcement Learning II And Final Notes

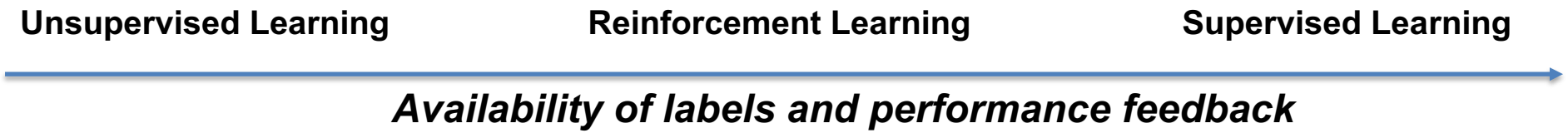
Fall 2018

NC State University
Lecturer: Dr. Behnam Kia

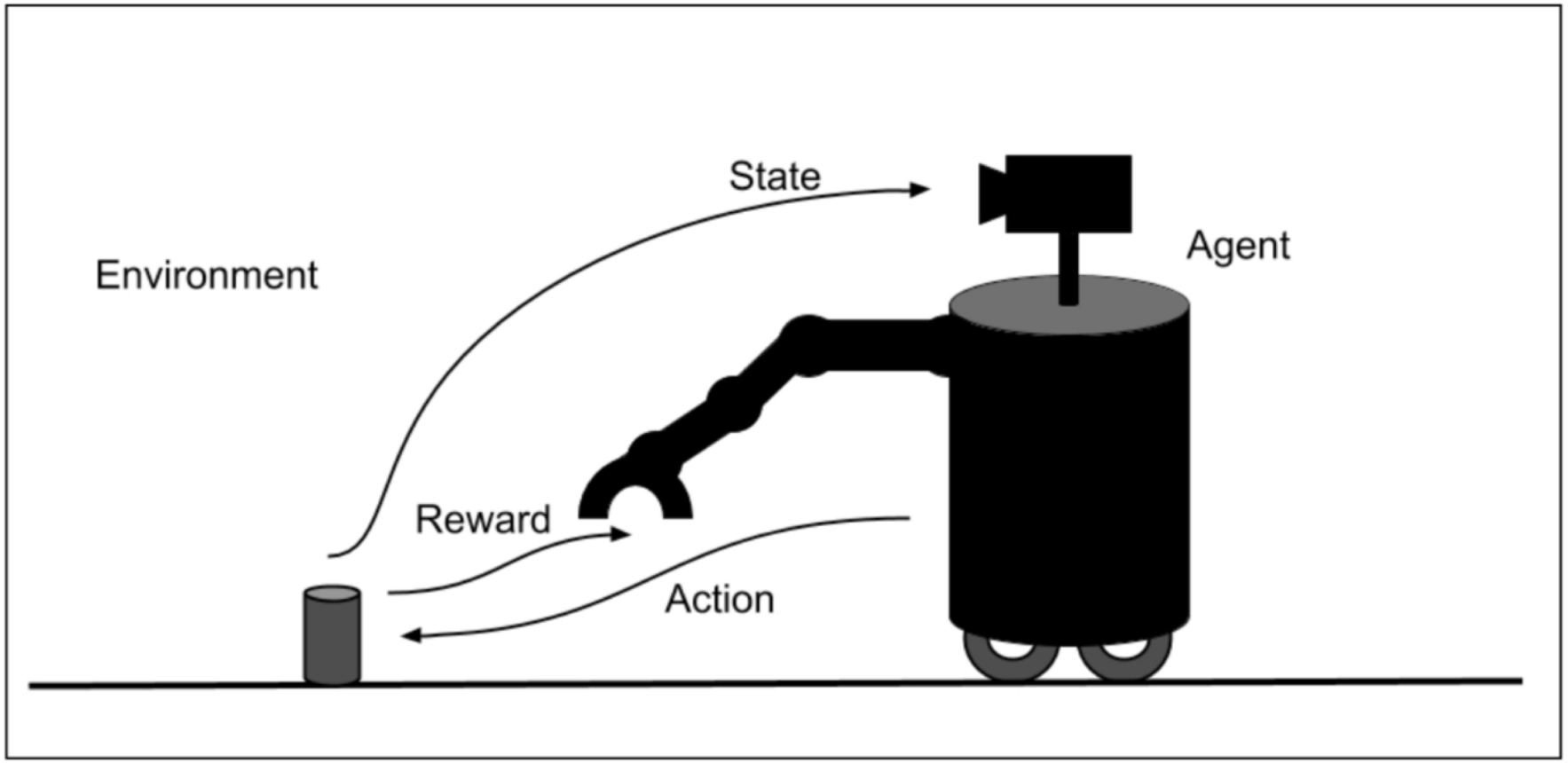
Course Website: https://appliedai.wordpress.ncsu.edu/

Reinforcement Learning

- Reinforcement Learning (RL): Learning Optimal Behavior in Complex Environment

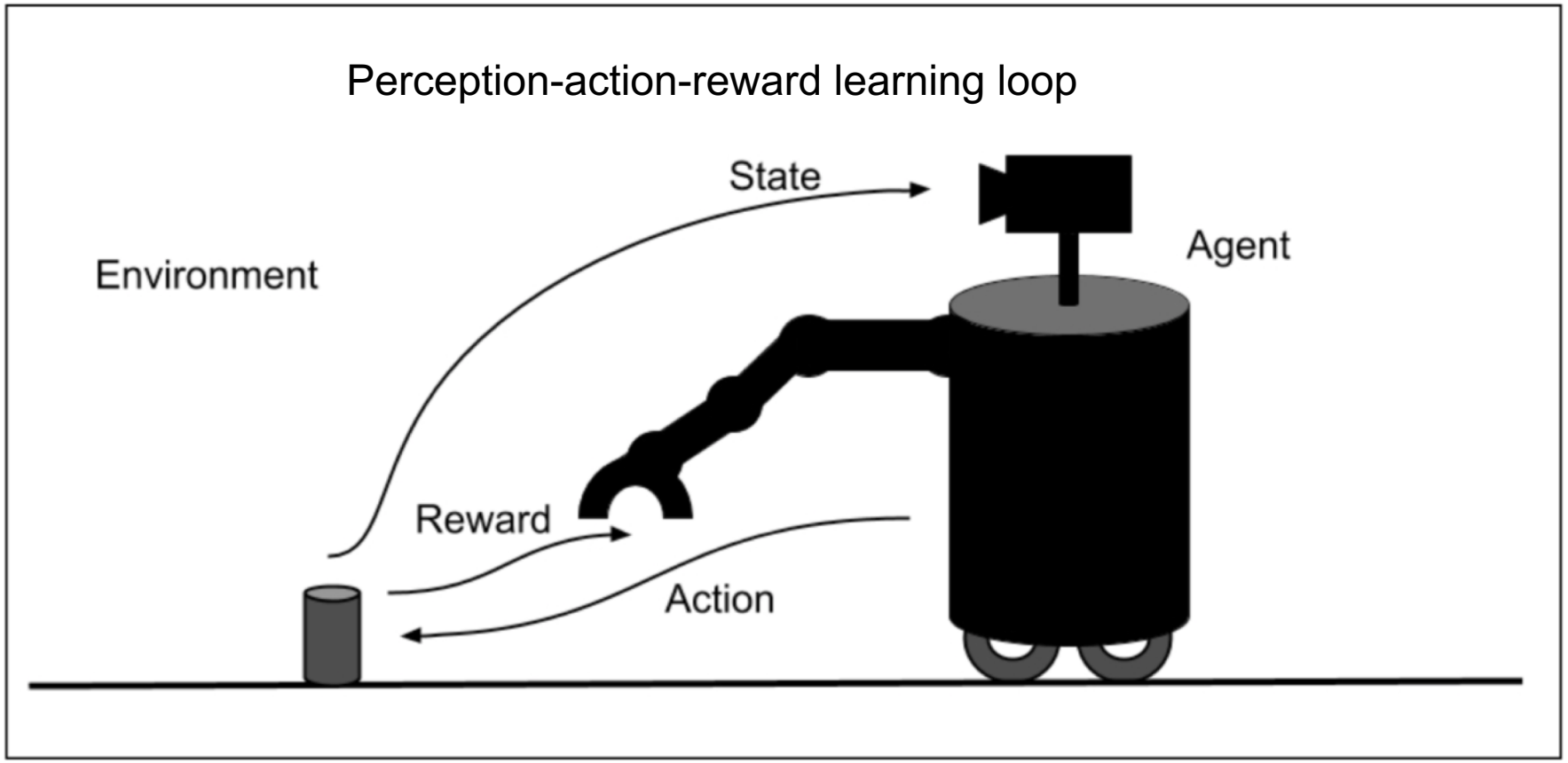


Reinforcement Learning



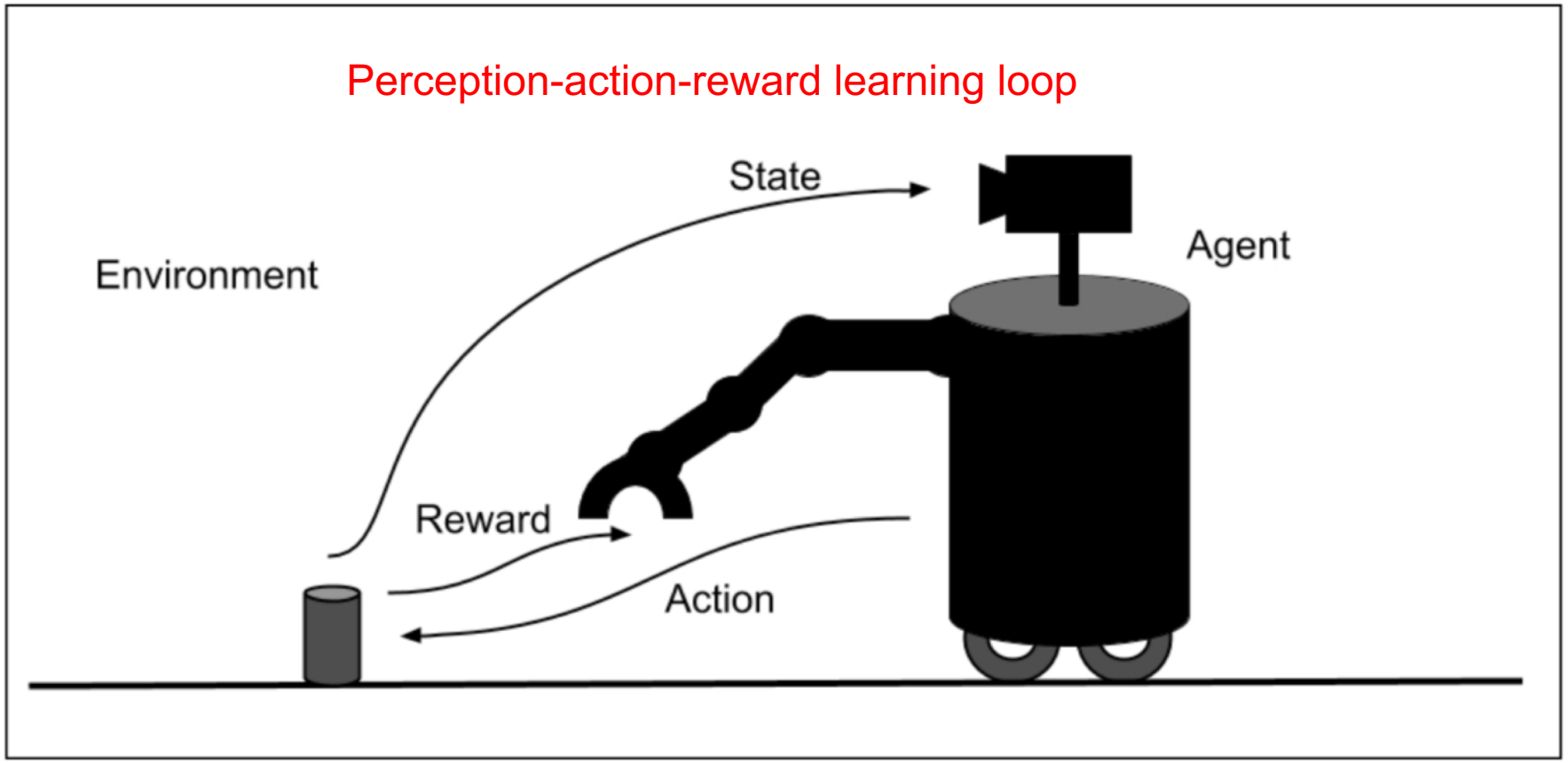
Picture from Advanced Deep Learning with Keras

Reinforcement Learning



Picture from Advanced Deep Learning with Keras

Reinforcement Learning

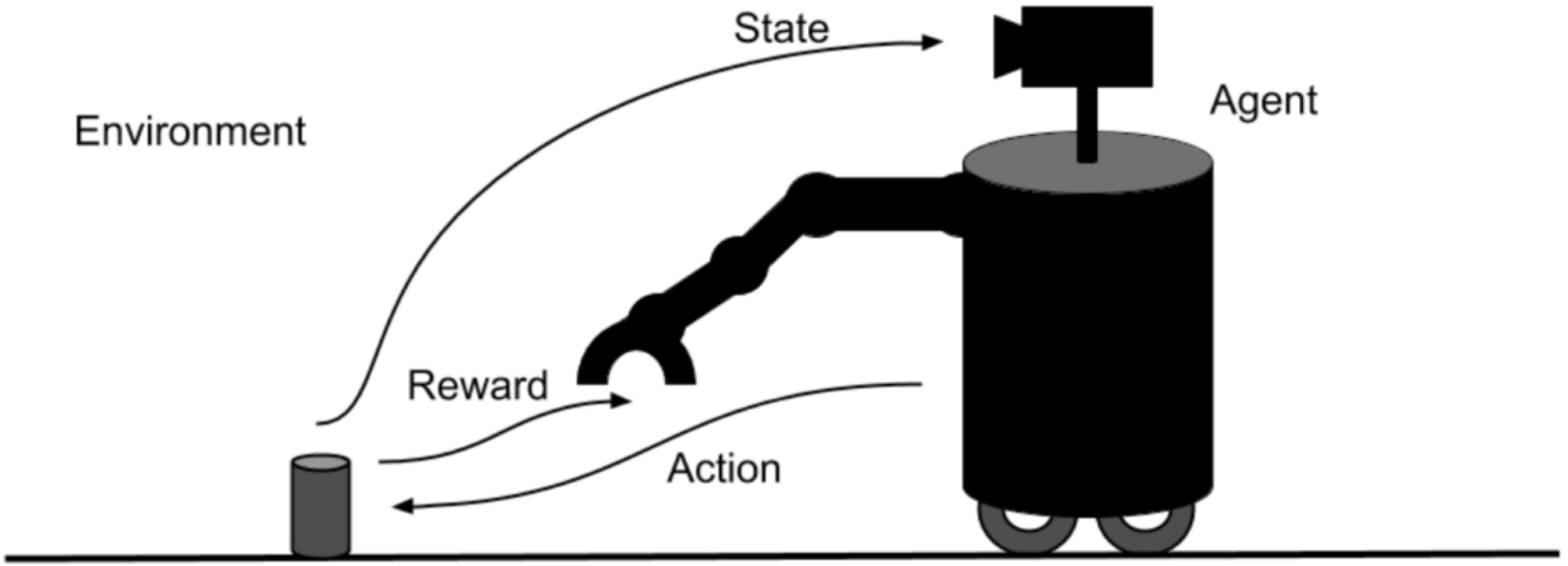


Picture from Advanced Deep Learning with Keras

Reinforcement Learning

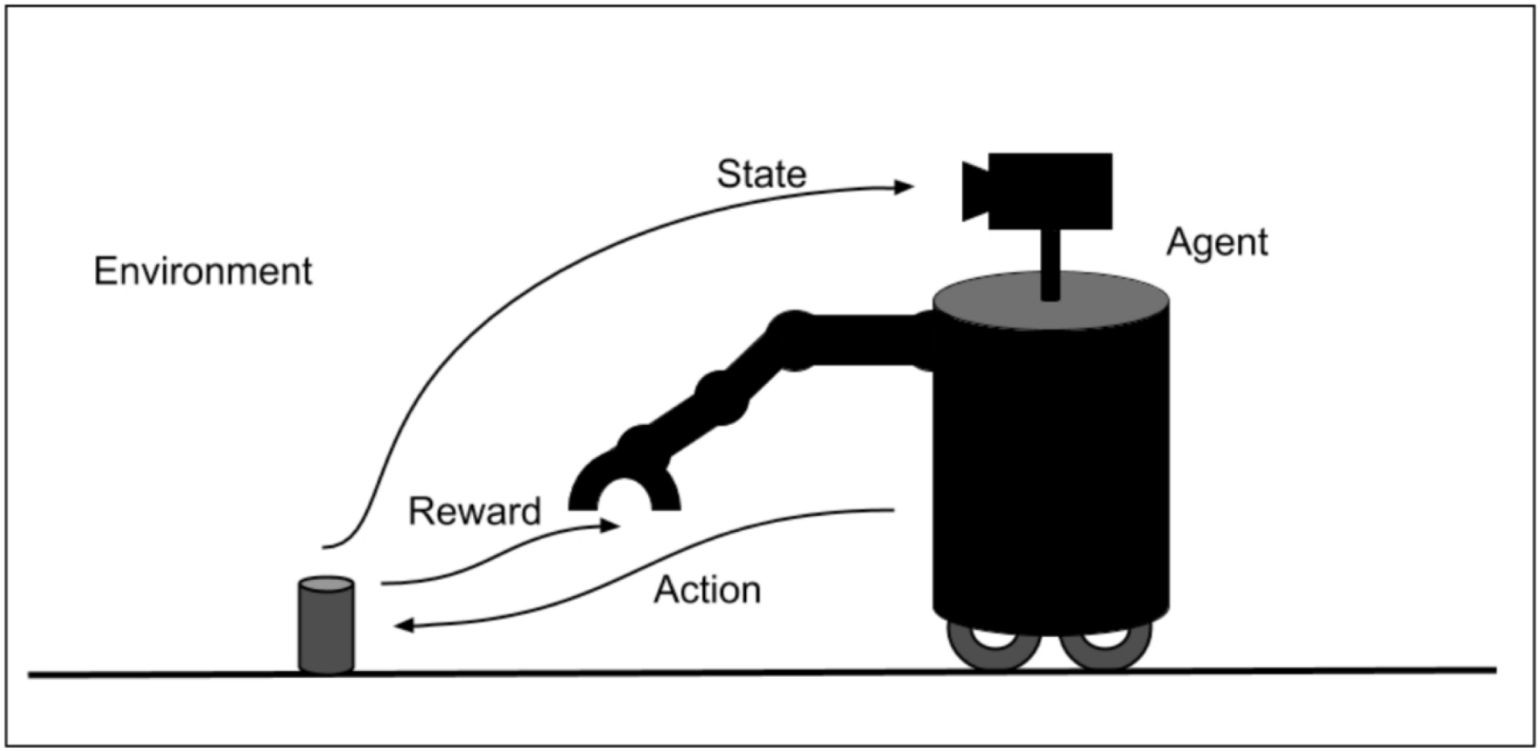
Perception-action-reward learning loop

Goal: To learn a policy to take actions that maximizes rewards



Picture from Advanced Deep Learning with Keras

- Policy: The algorithm used by an agent to determine its actions is called policy.



	a_0	a_1	a_2
s_0		✓	
s_1	✓		
s_2		✓	
s_3			✓
s_4	✓		
s_5		✓	
s_6		✓	

Problems

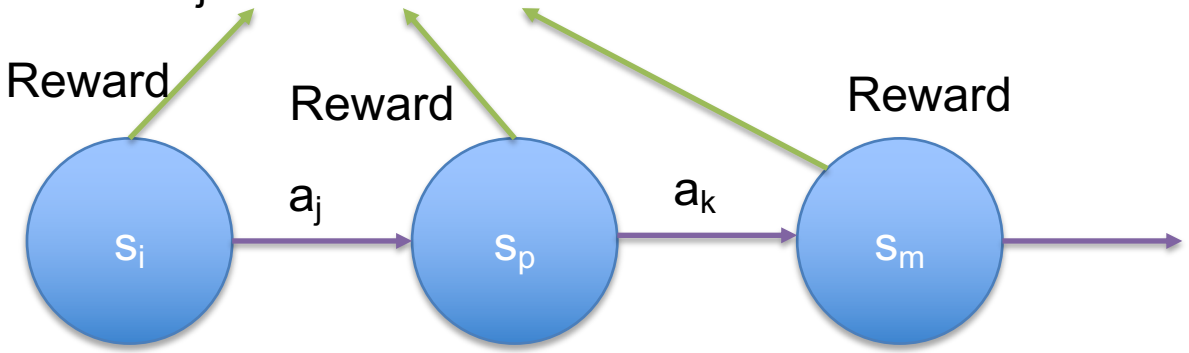
How about delayed reward?

The table can get very large. Imagine a chess game and the number of states!

Delayed reward

- We have to take into account the immediate reward, plus the reward we can take in the next step, and the next, and the next,… and pick an action that give us the maximum total award

- $Q(s_i,a_j)=r_0 + r_1 + r_2 + \ldots$



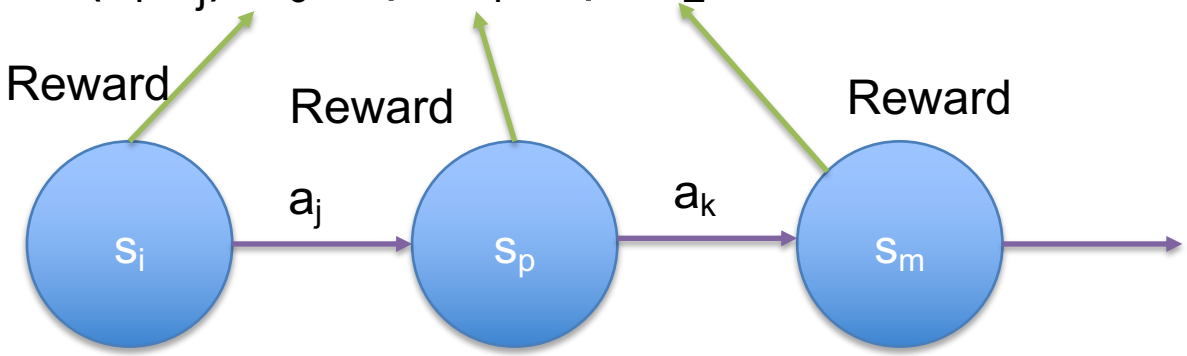
j, k, p, m are indexes that gives us the maximum total reward.

Delayed reward

- We have to take into account the immediate reward, plus the reward we can take in the next step, and the next, and the next,… and pick a sequence of actions that give us the maximum total award

- $Q(s_i,a_j)=r_0 + \gamma^1 r_1 + \gamma^2 r_2+\ldots$

γ *is discount factor*
$0 < \gamma < 1$



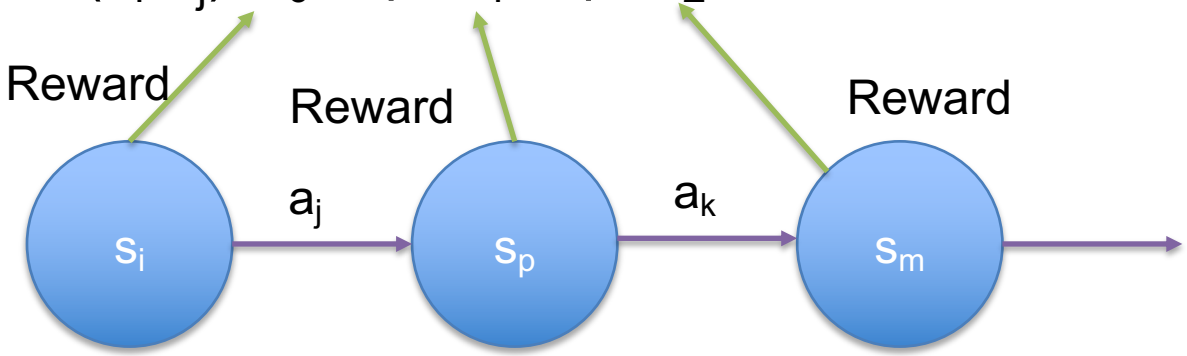
j, k, p, m are indexes that gives us the maximum total reward.

Delayed reward

- We have to take into account the immediate reward, plus the reward we can take in the next step, and the next, and the next,… and pick a sequence of actions that give us the maximum total award

- $Q(s_i,a_j)=r_0 + \gamma^1 r_1 + \gamma^2 r_2+\ldots$

γ *is discount factor*
$0 < \gamma < 1$



Not a tractable problem to solve!

j, k, p, m are indexes that gives us the maximum total reward.

- $Q(s_i,a_j)=r_0 + \gamma^1 r_1 + \gamma^2 r_2+\ldots$

- $Q(s,a)= r_0+ \gamma \max_{a'} Q(s',a')$

Action a takes us from state s to state s'.

- $Q(s,a) = r_0 + \gamma \max_{a'} Q(s',a')$

Action a takes us from state s to state s'.

$Q(s,a)_{k+1} = r_0 + \gamma \max_{a'} Q(s',a')_k$

We can solve this iteratively and it converges (if given enough time) to an optimal Q table that gives us the optimal policy.

- $Q(s,a) = r_0 + \gamma \max_{a'} Q(s',a')$

Action a takes us from state s to state s'.

$Q(s,a)_{k+1} = r_0 + \gamma \max_{a'} Q(s',a')_k$

	a_0	a_1	a_2
s_0	$Q(s_0,a_0)$	$Q(s_0,a_1)$	$Q(s_0,a_2)$
s_1	$Q(s_1,a_0)$	$Q(s_1,a_1)$	$Q(s_1,a_2)$
s_2	$Q(s_2,a_0)$	$Q(s_2,a_1)$	$Q(s_2,a_2)$
s_3	$Q(s_3,a_0)$	$Q(s_3,a_1)$	$Q(s_3,a_2)$
s_4	$Q(s_4,a_0)$	$Q(s_4,a_1)$	$Q(s_4,a_2)$
s_5	$Q(s_5,a_0)$	$Q(s_5,a_1)$	$Q(s_5,a_2)$
s_6	$Q(s_6,a_0)$	$Q(s_6,a_1)$	$Q(s_6,a_2)$

We can solve this iteratively and it converges (if given enough time) to an optimal Q table that gives us the optimal policy.

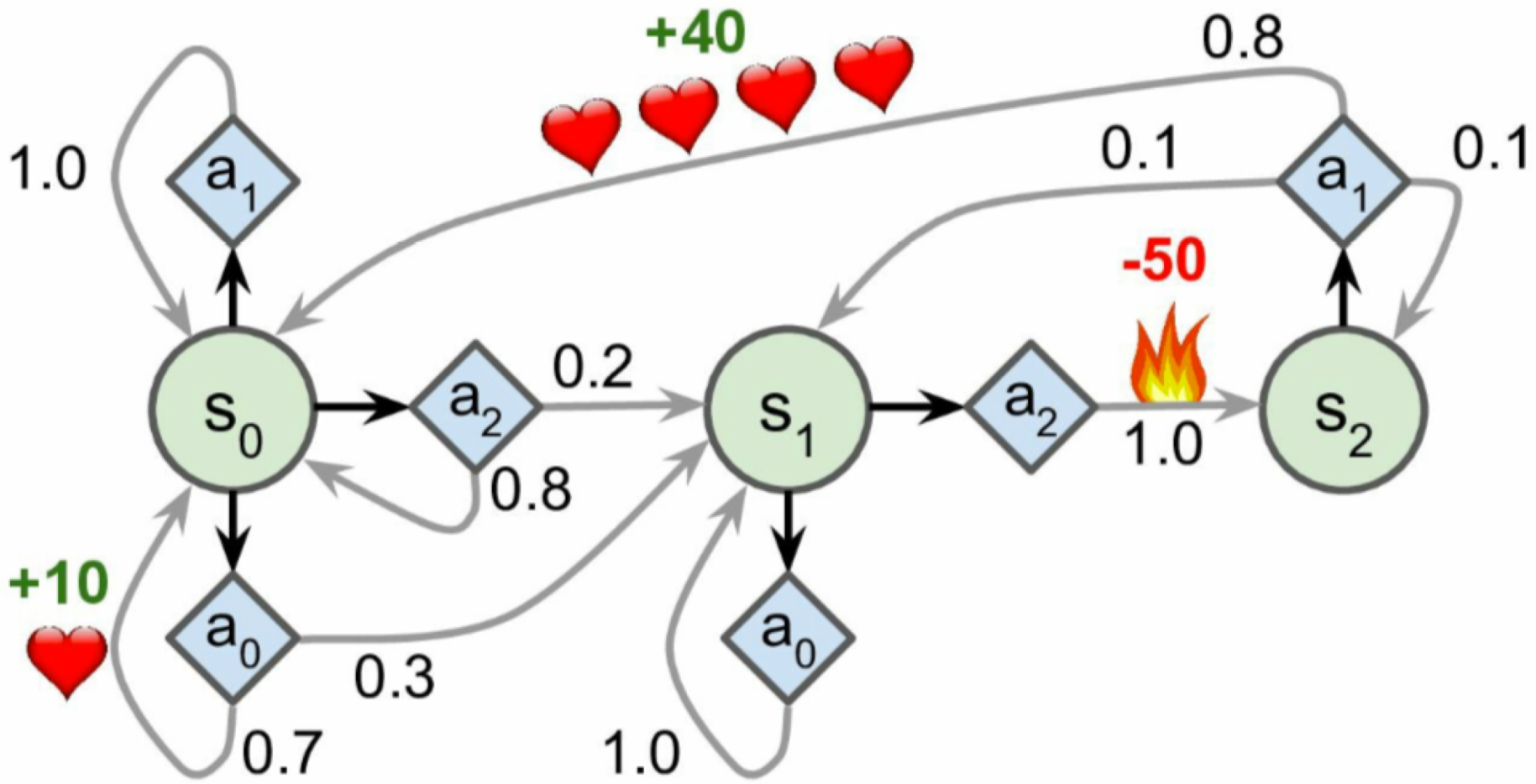
Let's add stochasticity

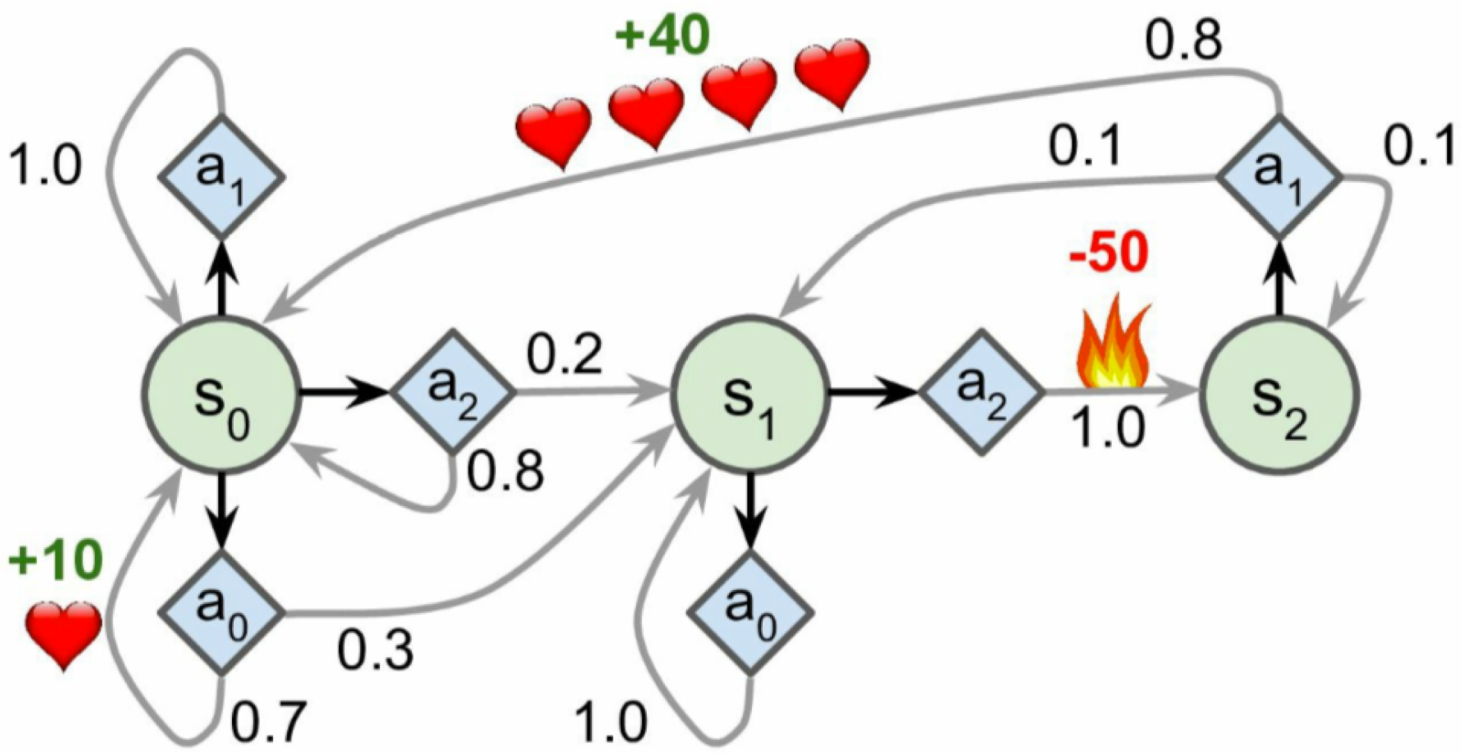
- When we are in state s, and take action a, we don't always end up in state s'. There can be some randomness involved.

Let's add stochasticity

- When we are in state s, and take action a, we don't always end up in state s'. There can be some randomness involved.

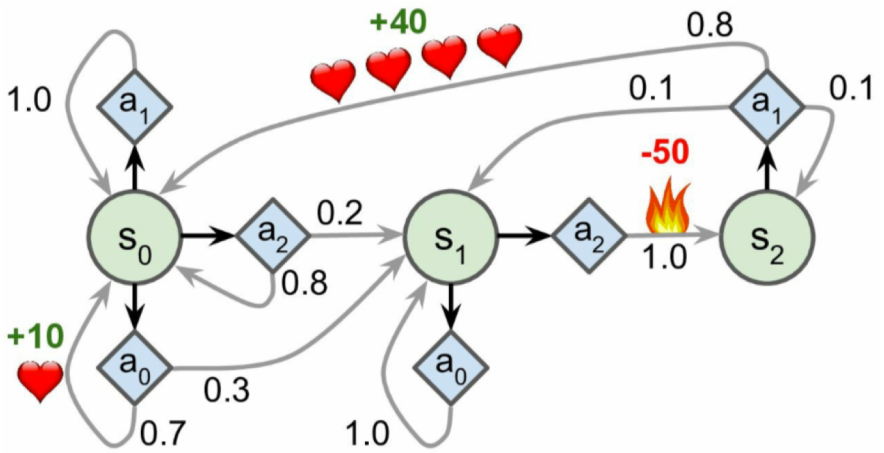
- Markov decision process models our problem (when actions are discrete).


+40
0.8
1.0
0.1
0.1
a_1
a_1
-50
s_0
a_2
0.2
s_1
a_2
s_2
1.0
0.8
+10
a_0
a_0
0.3
0.7
1.0



$Q(s,a)_{k+1} = r_0 + \gamma \max_{a'} Q(s',a')_k$

$Q(s,a)_{k+1} = \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma \max_{a'} Q(s',a')_k]$




```
nan=np.nan  # represents impossible actions
T = np.array([  # shape=[s, a, s']
        [[0.7, 0.3, 0.0], [1.0, 0.0, 0.0], [0.8, 0.2, 0.0]],
        [[0.0, 1.0, 0.0], [nan, nan, nan], [0.0, 0.0, 1.0]],
        [[nan, nan, nan], [0.8, 0.1, 0.1], [nan, nan, nan]],
    ])
R = np.array([  # shape=[s, a, s']
        [[10., 0.0, 0.0], [0.0, 0.0, 0.0], [0.0, 0.0, 0.0]],
        [[10., 0.0, 0.0], [nan, nan, nan], [0.0, 0.0, -50.]],
        [[nan, nan, nan], [40., 0.0, 0.0], [nan, nan, nan]],
    ])
possible_actions = [[0, 1, 2], [0, 2], [1]]
```

```
Q = np.full((3, 3), -np.inf)   # -inf for impossible actions
for state, actions in enumerate(possible_actions):
    Q[state, actions] = 0.0  # Initial value = 0.0, for all possible actions

learning_rate = 0.01
discount_rate = 0.95
n_iterations = 100

for iteration in range(n_iterations):
    Q_prev = Q.copy()
    for s in range(3):
        for a in possible_actions[s]:
            Q[s, a] = np.sum([
                T[s, a, sp] * (R[s, a, sp] + discount_rate * np.max(Q_prev[sp]))
                for sp in range(3)
            ])
```

- But do we always know those values?

Temporal Difference Learning

- Randomly explore them model (let the model run randomly) and estimate the *Q* values.

$$Q(s,a)_{k+1} = \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma \max_{a'} Q(s',a')_k]$$

$$Q(s,a)_{k+1} = (1-\alpha)Q(s,a)_k + \alpha[r + \gamma \max_{a'} Q(s',a')_k]$$

```
import numpy.random as rnd

learning_rate0 = 0.05
learning_rate_decay = 0.1
n_iterations = 20000

s = 0 # start in state 0

Q = np.full((3, 3), -np.inf)  # -inf for impossible actions
for state, actions in enumerate(possible_actions):
    Q[state, actions] = 0.0  # Initial value = 0.0, for all possible actions

for iteration in range(n_iterations):
    a = rnd.choice(possible_actions[s])  # choose an action (randomly)
    sp = rnd.choice(range(3), p=T[s, a]) # pick next state using T[s, a]
    reward = R[s, a, sp]
    learning_rate = learning_rate0 / (1 + iteration * learning_rate_decay)
```

```
    Q[s, a] = learning_rate * Q[s, a] + (1 - learning_rate) * (
            reward + discount_rate * np.max(Q[sp])
        )
    s = sp # move to next state
```

Q-table initialized to all zero at the beginning.

	a_0	a_1	a_2
s_0	0	0	0
s_1	0	0	0
s_2	0	0	0
s_3	0	0	0
s_4	0	0	0
s_5	0	0	0
s_6	0	0	0

Sometime in the middle

	a_0	a_1	a_2
s_0	23	0	0
s_1	0	4	0
s_2	0	0	0
s_3	5	0	0
s_4	0	6	0
s_5	0	0	0
s_6	0	0	0

Sometime in the middle

Should I take a good known action? (exploitation)

	a_0	a_1	a_2
s_0	23	0	0
s_1	0	4	0
s_2	0	0	0
s_3	5	0	0
s_4	0	6	0
s_5	0	0	0
s_6	0	0	0

Or take chances on unknown actions? (exploration)

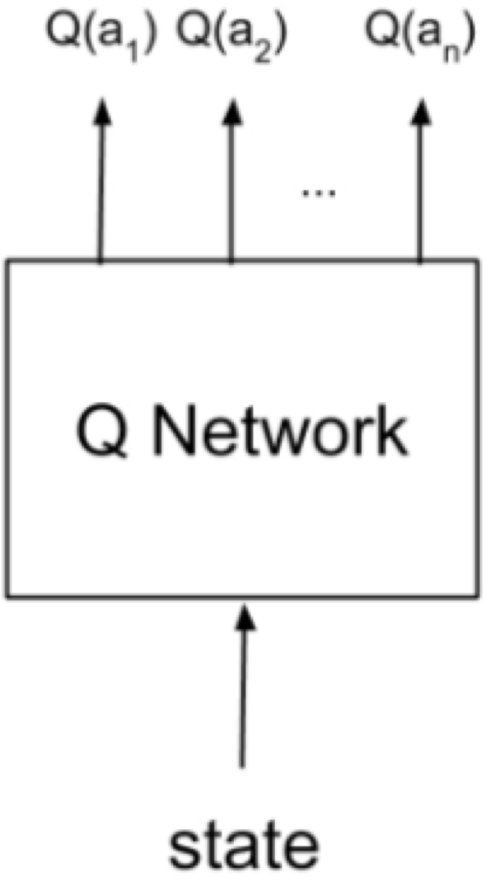
Exploration vs. Exploitation

ε-greedy method.

Deep Q-Learning

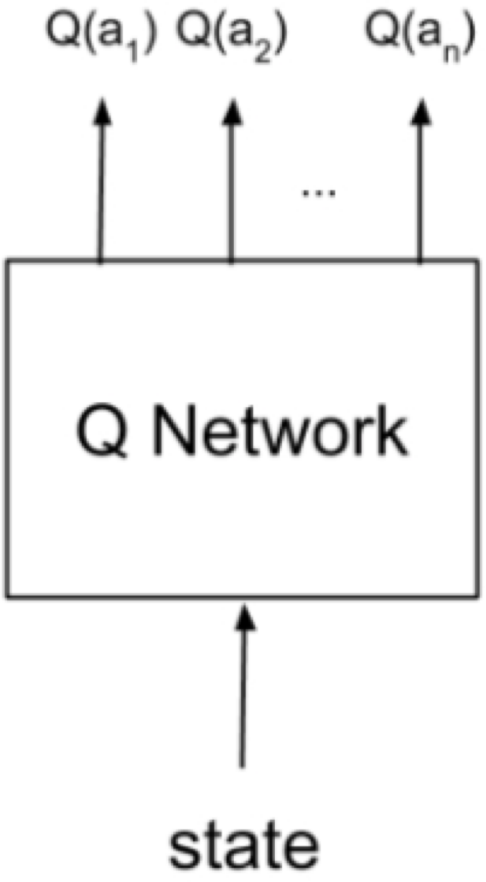
	a_0	a_1	a_2
s_0	$Q(s_0,a_0)$	$Q(s_0,a_1)$	$Q(s_0,a_2)$
s_1	$Q(s_1,a_0)$	$Q(s_1,a_1)$	$Q(s_1,a_2)$
s_2	$Q(s_2,a_0)$	$Q(s_2,a_1)$	$Q(s_2,a_2)$
s_3	$Q(s_3,a_0)$	$Q(s_3,a_1)$	$Q(s_3,a_2)$
s_4	$Q(s_4,a_0)$	$Q(s_4,a_1)$	$Q(s_4,a_2)$
s_5	$Q(s_5,a_0)$	$Q(s_5,a_1)$	$Q(s_5,a_2)$
s_6	$Q(s_6,a_0)$	$Q(s_6,a_1)$	$Q(s_6,a_2)$

Deep Q-Learning



	a_0	a_1	a_2
s_0	$Q(s_0,a_0)$	$Q(s_0,a_1)$	$Q(s_0,a_2)$
s_1	$Q(s_1,a_0)$	$Q(s_1,a_1)$	$Q(s_1,a_2)$
s_2	$Q(s_2,a_0)$	$Q(s_2,a_1)$	$Q(s_2,a_2)$
s_3	$Q(s_3,a_0)$	$Q(s_3,a_1)$	$Q(s_3,a_2)$
s_4	$Q(s_4,a_0)$	$Q(s_4,a_1)$	$Q(s_4,a_2)$
s_5	$Q(s_5,a_0)$	$Q(s_5,a_1)$	$Q(s_5,a_2)$
s_6	$Q(s_6,a_0)$	$Q(s_6,a_1)$	$Q(s_6,a_2)$

Deep Q-Learning



$$Q(s,a) = r + \gamma \max_{a'} Q(s',a')$$

End

- But is it?

- But is it?

NC STATE UNIVERSITY

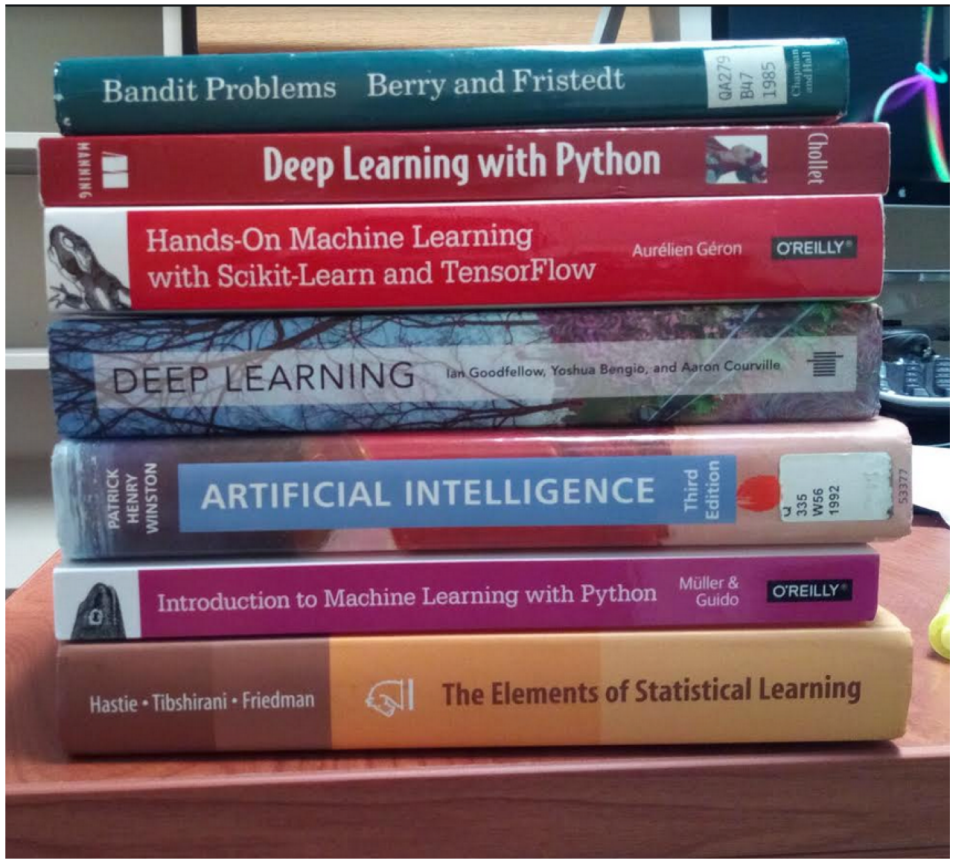
What you can expect from this course?

At the end of the semester you will:

- know what an AI problem is and what is not!
- learn the basic foundations of deep learning and how to apply it to AI problems.
- gain basic hands-on experience with AI development tools and software.
- get enough experience, knowledge, and confidence to pursue on your own and learn more advanced topics.


Python Machine Learning
Second Edition
Sebastian Raschka
Vahid Mirjalili
Duda
Hart
Stork
Second Edition
Pattern Classification
WILEY-INTERSCIENCE
FOURTH EDITION
DATA MINING
Witten
Frank
Hall
Pal
Advanced Deep Learning with Keras
Rowel Atienza
Python Data Science Handbook
VanderPlas
O'REILLY
THE DEEP LEARNING REVOLUTION
TERRENCE J. SEJNOWSKI
Deep Reinforcement Learning Hands-On
Maxim Lapan
Pearson
THIRD EDITION
Artificial Intelligence
Russell
Norvig


Python Machine Learning
Second Edition
Sebastian Raschka
Vahid Mirjalili
Duda Hart Stork
Second Edition
Pattern Classification
FOURTH EDITION
DATA MINING
Witten Frank Hall Pal
Advanced Deep Learning with Keras
Rowel Atienza
Python Data Science Handbook
VanderPlas
O'REILLY
THE DEEP LEARNING REVOLUTION
TERRENCE J. SEJNOWSKI
Deep Reinforcement Learning Hands-On
Maxim Lapan
Pearson
THIRD EDITION
Artificial Intelligence
Russell Norvig


Bandit Problems
Berry and Fristedt
Deep Learning with Python
Chollet
Hands-On Machine Learning
with Scikit-Learn and TensorFlow
Aurélien Géron
O'REILLY
DEEP LEARNING
Ian Goodfellow, Yoshua Bengio, and Aaron Courville
PATRICK HENRY WINSTON
ARTIFICIAL INTELLIGENCE
Third Edition
Introduction to Machine Learning with Python
Müller & Guido
O'REILLY
Hastie • Tibshirani • Friedman
The Elements of Statistical Learning

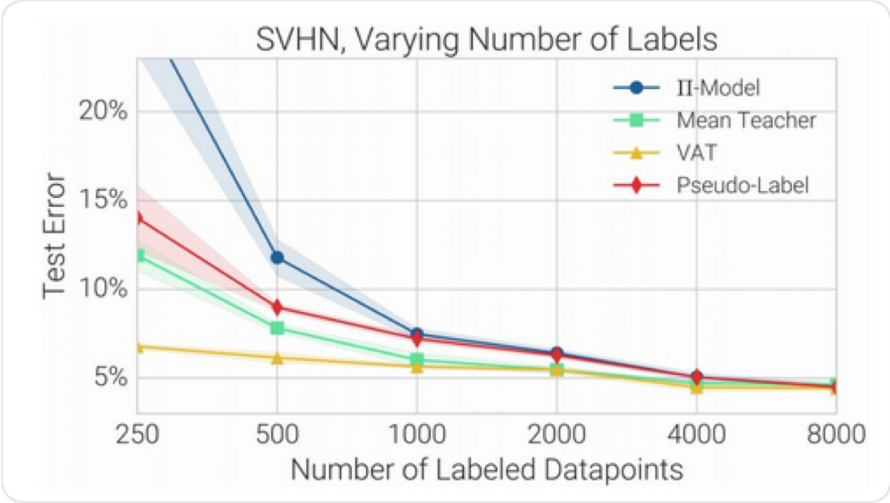
François Chollet @fchollet · 11m

Neat introduction to Reinforcement Learning with Keras, using Pong:
blog.floydhub.com/spinning-up-wi…

GIF

Ian Goodfellow @goodfellow_ian · 3h

#NeurIPS Check out our poster on realistic evaluation of semi-supervised learning, now until 12:45. nips.cc/Conferences/20…



Pinned Tweet

Andrew Ng @AndrewYNg · Nov 14

Announcing AI For Everyone, the newest @deeplearningai_ course! Whether you're a CEO, product manager, marketer, recruiter, designer, or financier, this non-technical course will help you understand how to apply AI in your company, and navigate AI's rise.

Announcing "AI for Everyone": a new course from deeplearning.ai

I am excited to announce the newest course from deeplearning.ai, "AI for Everyone." It will be available on Coursera in early 2019. AI is not only for...

medium.com

41 1.1K 3.1K