Reinforcement learning

Michal CHOVANEC, PhD
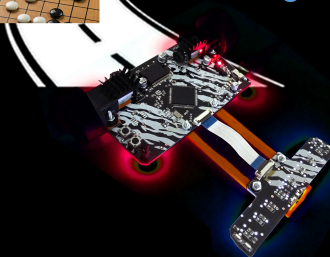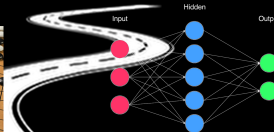
Reinforcement learning

- learn from punishment and rewards
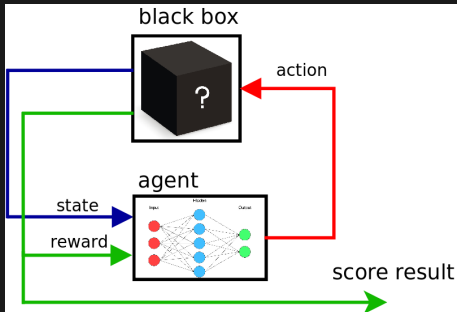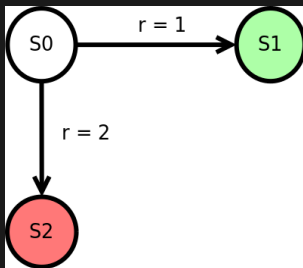- learn to play a game with unknow rules

Reinforcement learning

- obtain **state**
- choose **action**
- **execute** action
- obtain **reward**
- learn from **experiences**

Making decisions

two possible strategies

- strategy 1 : S0->S1, score = 1.0
- strategy 2 : S0->S2, score = 2.0

$$Q(s, a) = R(s, a)$$

where

s is state

a is action

Making decisions

two possible strategies, greedy = trap

- strategy 1 : S0->S1, score = 1.0
- strategy 2 : S0->S2->S3, score = 2.0 + (-20.0) = -18
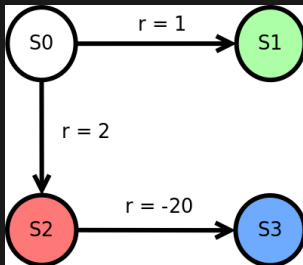
$$Q(s, a) = R(s, a) + FutureReward$$

Q learning

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

where
s is state
a is action
s' is next state
a' is best action in next state
$R(s, a)$ is reward
$\gamma \in \langle 0, 1 \rangle$ is discount factor

Q learning

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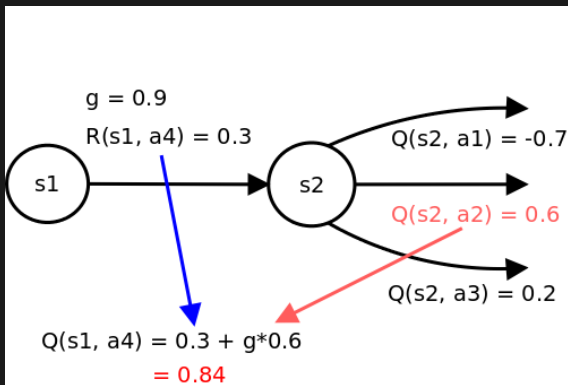
Q learning - stochastic

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

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

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

$$Q(s, a) = (1 - \alpha)Q(s, a) + \alpha(R(s, a) + \gamma \max_{a'} Q(s', a'))$$

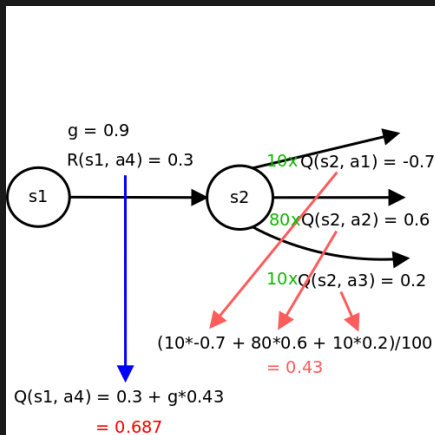# SARSA learning

$$\Delta Q(s,a) = \alpha(R(s,a) + \gamma Q(s',a') - Q(s,a))$$
$$Q(s,a) = (1-\alpha)Q(s,a) + \alpha(R(s,a) + \gamma Q(s',a'))$$

g = 0.9
R(s1, a4) = 0.3
10x Q(s2, a1) = -0.7
80x Q(s2, a2) = 0.6
10x Q(s2, a3) = 0.2

(10*-0.7 + 80*0.6 + 10*0.2)/100
= 0.43

Q(s1, a4) = 0.3 + g*0.43
= 0.687

TODO : in points, illustrate algorithm

Usefull links

- ImageNet Classification with Deep Convolutional Neural Networks `https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf`

- Alex Krizhevsky web, `https://www.cs.toronto.edu/~kriz/`

- Deep Belief Nets in C++ and CUDA C: Volume III `https://www.amazon.com/Deep-Belief-Nets-CUDA-Convolutional/dp/1530895189`

- Deep Learning (Adaptive Computation and Machine Learning `https://www.amazon.com/Deep-Learning-Adaptive-Computation-Machine/dp/0262035618`

- Densely Connected Convolutional Networks `https://arxiv.org/pdf/1608.06993.pdf`

- MNIST dataset `http://yann.lecun.com/exdb/mnist/`

- Digital signal processing for STM32 microcontrollers using CMSIS `https://www.st.com/resource/en/application_note/dm00273990.pdf`

- CMSIS-NN: Efficient Neural Network Kernels for Arm Cortex-M CPUs `https://arxiv.org/pdf/1801.06601.pdf`

michal chovanec (michal.nand@gmail.com)
`www.youtube.com/channel/UCzVvP2ou8v3afNiVrPAHQGg`