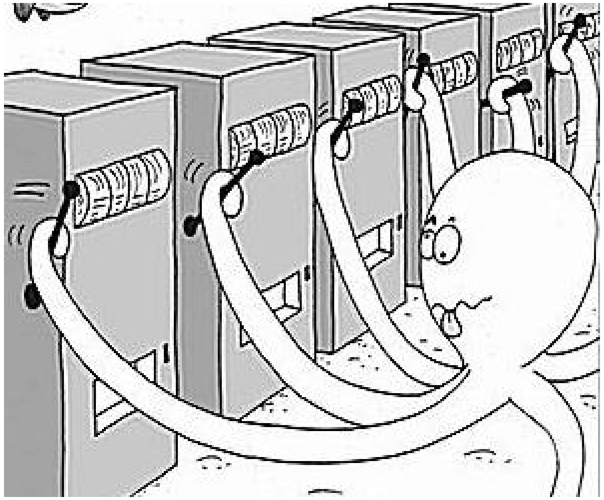
Week 8: Reinforcement Learning



http://stevehanov.ca/blog/index.php?id=132 https://www.youtube.com/watch?v=eG1Ed8PTJ18

Week 7 Review: Performance metrics

- What are metrics for regression?
  -
- Classification?
  -
- Clustering?
  -

Week 7 Review: Performance metrics

- What are metrics for regression?
  - MSE, MSLE, MAE, MAPE, R^2
- Classification?
  - Accuracy, precision, recall, F1, log loss
- Clustering?
  - WSS, silhoutte score, Calinski and Harabaz

Week 7 review quiz

- Create at least 2 models to compare on the breast cancer dataset (under week 8)
  - The V11 column is the target
  - 2 means no cancer (benign) 4 means cancer
  - Other columns are various bio-measurements
- Report at least 2 performance metrics, and choose the best model. Explain your choice and any caveats.

Types of reinforcement learning

- Multi-armed bandit
- Q-learning
- Deep q-learning
  - Policy gradient (PG)

Multi-armed bandit

- Exploration (try new things) vs exploitation (go with the best we know of at the time)
- Exploration is epsilon (ε)

https://jeongyoonlee.com/2017/08/09/transition-from-exploration-to-exploitation/

http://www.plexure.com/plexure-blog/2016/9/21/multiworld-testing

Multi-armed bandit

- Example: website with different designs, A, B, C
- Epsilon-greedy example:

```
def choose():
    if math.random() < 0.1:
        # exploration!
        # choose a random lever 10% of the time.
    else:
        # exploitation!
        # for each lever,
            # calculate the expectation of reward.
            # This is the number of trials of the lever divided by the total reward
            # given by that lever.
        # choose the lever with the greatest expectation of reward.
    # increment the number of times the chosen lever has been played.
    # store test data in redis, choice in session key, etc..
def reward(choice, amount):
    # add the reward to the total for the given lever.
```

Multi-armed bandit

- Example: website with different designs, A, B, C
- Epsilon-first example:

```
def choose():
    For n rounds:
        # choose a random lever
    if math.random() < 0.1:
        # exploration!
        # choose a random lever 10% of the time.
    else:
        # exploitation!
        # for each lever,
            # calculate the expectation of reward.
            # This is the number of trials of the lever divided by the total reward
            # given by that lever.
        # choose the lever with the greatest expectation of reward.
    # increment the number of times the chosen lever has been played.
    # store test data in redis, choice in session key, etc..
def reward(choice, amount):
    # add the reward to the total for the given lever.
```

Multi-armed bandit

- Example: website with different designs, A, B, C
- Epsilon-decreasing example:

```
def choose():
    epsilon = epsilon * 0.99  # choose whatever value you want to decrease epsilon by
    if math.random() < epsilon:
        # exploration!
        # choose a random lever epsilon% of the time.
    else:
        # exploitation!
        # for each lever,
            # calculate the expectation of reward.
            # This is the number of trials of the lever divided by the total reward
            # given by that lever.
        # choose the lever with the greatest expectation of reward.
    # increment the number of times the chosen lever has been played.
    # store test data in redis, choice in session key, etc..
def reward(choice, amount):
    # add the reward to the total for the given lever.
```

http://stevehanov.ca/blog/index.php?id=132

Multi-armed bandit pair programming exercise (50% of project grade this week)

- Example: website with different designs, A, B, C
- Use the distribution from the site (A = 2.8%, B = 3.2%, C = 2.6%) and write a function that is an epsilon-decreasing strategy

```
def choose():
    epsilon = epsilon * 0.99  # choose whatever value you want to decrease epsilon by
    if math.random() < epsilon:
        # exploration!
        # choose a random lever epsilon% of the time.
    else:
        # exploitation!
        # for each lever,
            # calculate the expectation of reward.
            # This is the number of trials of the lever divided by the total reward
            # given by that lever.
        # choose the lever with the greatest expectation of reward.
    # increment the number of times the chosen lever has been played.
    # store test data in redis, choice in session key, etc..
def reward(choice, amount):
    # add the reward to the total for the given lever.
```

http://stevehanov.ca/blog/index.php?id=132

Q-learning

We are seeking a policy, pi, state S leads to agent taking action A

$$\pi : S \rightarrow A$$

A policy has a value which is the sum of the reward at the current time, and the rewards in the future discounted by gamma:

$$V^{\pi} = r_t + \gamma r_{(t+1)} + \gamma^2 r_{(t+2)} \ldots = \sum_i \gamma^i r_{(t+i)}$$

Define function Q, which is value after executing action a at state s, plus the discounted value of the optimal policy thereafter

$$Q(s,a) = r(s,a) + \gamma V^o(s^{new})$$

V^o is the optimum value from an optimum policy π^o, s^{new} is the new state after action a

Our optimal policy at state s is the maximum Q-value over all possible actions

$$V^0(s) = \max_i (Q(s, a_i))$$

Q-learning

$$Q(s,a)=r(s,a)+\gamma V^o(s^{new})$$

$$V^o(s)=\max_i(Q(s,a_i))$$

Substitute max(Q) for V^o

$$Q(s,a)=r(s,a)+\gamma\max_i(Q(s,a_i))$$

If we have an estimate of Q, and choose action a, end up in s^{new}, we will have this result if our estimate is correct:

$$0=r(s,a)+\gamma\max_i(\hat{Q}(s^{new},a_i))-\hat{Q}(s,a)$$

But our estimate will probably not be correct, so we can update our estimate at iteration n by adding the difference to our current Q estimate:

$$\hat{Q}_{(n+1)}(s,a)=\hat{Q}_n(s,a)+\alpha[r(s,a)+\gamma\max_i(\hat{Q}_n(s^{new},a_i))-\hat{Q}_n(s,a)]$$

Q-learning

This is our Q-learning equation. We start at a state s, try an action, a, iteration n

$$\hat{Q}_{(n+1)}(s,a)=\hat{Q}_n(s,a)+\alpha\left[r(s,a)+\gamma \max_i\left(\hat{Q}_n(s^{new},a_i)\right)-\hat{Q}_n(s,a)\right]$$

$$\alpha=\frac{1}{1+visits_n(s,a)}$$

Example: tic-tac-toe

- Start with empty board
- Make a random move (all Q’s can be initialized randomly or to 0)
- No reward until we win (+1) or lose (-1)
- Epsilon-methods again, pick a random choice with probability epsilon, or choose action with highest Q-value
- One “episode” is when the game ends
- about 255k possible tic-tac-toe games, so need to train about that many times

https://www.cs.swarthmore.edu/~meeden/cs63/f11/ml-ch13.pdf

Tic-tac-toe example

- Say we're in this state, and it's x's turn
- X picks the right spot to win, r = 1
- r = 1, alpha = 1, new Q(s, a) is 1 (0 before)

x	x	
o	o	

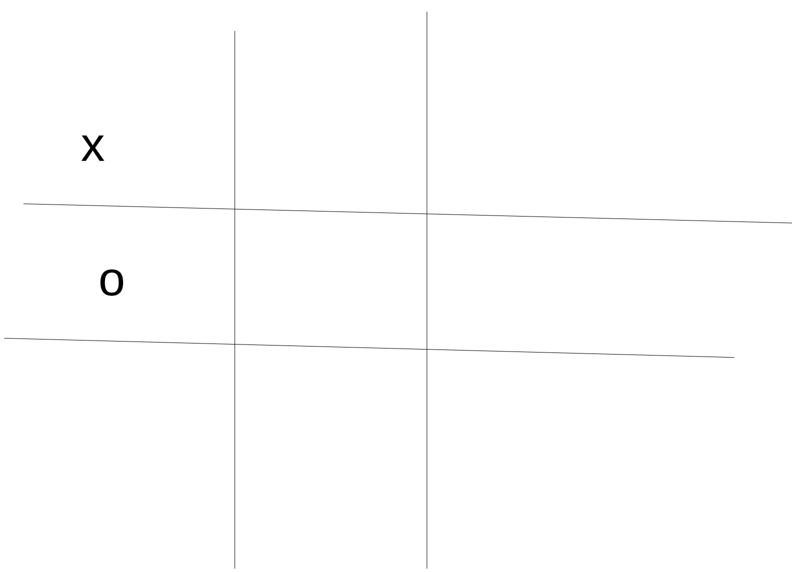
x	x	x
o	o	

$$\hat{Q}_{(n+1)}(s,a)=\hat{Q}_n(s,a)+\alpha[r(s,a)+\gamma \max_i(\hat{Q}_n(s^{new},a_i))-\hat{Q}_n(s,a)]$$

Q(state on left, x in upper right)$_{(n+1)}$ = 0 + 1/1[1 + 0.9*0 – 0] = 1

Tic-tac-toe example

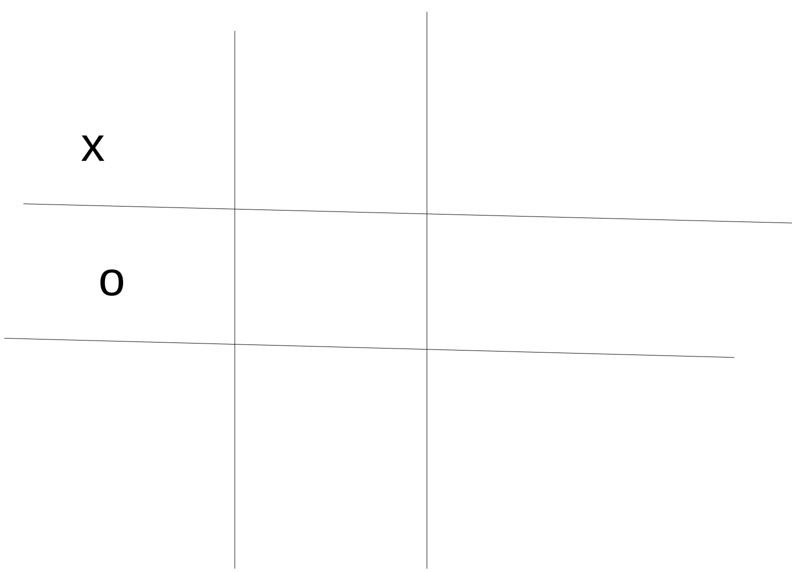
- Now we’re in this state, and it’s x’s turn
- Because we’ve been in this state before, it will update the Q-value for x in the top center, so that the action has a slightly higher Q-value



$$\hat{Q}_{(n+1)}(s,a)=\hat{Q}_n(s,a)+\alpha[r(s,a)+\gamma \max_{i}(\hat{Q}_n(s^{new},a_i))-\hat{Q}_n(s,a)]$$

Tic-tac-toe example

- This is the way Q-learning learns – backwards and slowly thought brute force.
- Can you think of an improvement to the algorithm?



$$\hat{Q}_{(n+1)}(s,a)=\hat{Q}_n(s,a)+\alpha[r(s,a)+\gamma \max_i(\hat{Q}_n(s^{new},a_i))-\hat{Q}_n(s,a)]$$

Tic-tac-toe example

- This is the way Q-learning learns – backwards and slowly thought brute force.
- Can you think of an improvement to the algorithm?
  - Propagate rewards back through Q-values after game ends

$$\hat{Q}_{(n+1)}(s,a)=\hat{Q}_n(s,a)+\alpha[r(s,a)+\gamma \max_i(\hat{Q}_n(s^{new},a_i))-\hat{Q}_n(s,a)]$$

Other Q-learning examples

- http://ai.berkeley.edu/reinforcement.html
  - https://github.com/gauthamvasan/Pacman-Reinforcement-Learning
- http://amunategui.github.io/reinforcement-learning/
- http://mnemstudio.org/path-finding-q-learning-tutorial.htm

How many states are there for an image?

Deep Q-learning

- ~255k possible tic-tac-toe combinations
- 256^{3nm} for an RGB image
- So if we are playing atari (160x192) we have 256^{92160} possible images (states)
  - This is a very low resolution, 0.03MP
  - Cell phone cameras are in the 1-20MP range
  - Even with 1080p (1920*1080), we are at $256^{6,220,800}$ possible states
- No way to explore (and save in memory) the full state-space with Q-learning
- Instead, use convolutional neural network and some other tricks (policy gradient)

Deep Q-learning (DQN) examples

- OpenAI gym
- Mario
- FlapPy bird (that one is python2 only)
- Python3 FlapPy bird
  - Try playing flapPy bird – it's really hard
- Space invaders
  - No shortage of atari DQN/DQL videos out there
- Pacman
- Starcraft II

Project: Go through tic-tac-toe Q-learning code, and...

- Code is here: https://github.com/khpeek/Q-learning-Tic-Tac-Toe
- Name the function where the Q-learning is actually happening (where the long Q-learning equation is being used).
- Show the two lines of code where the Q-learning update happens.
- Change the default Q-value from 1 to 0, and train for 10k episodes. Play the agents a few times. Are the behaviors of the agents different with 1 or 0 as the default Q?
  - Bonus: save each agent as a separate file, and play them against each other a number of times to see which one is smarter
  - Double bonus: use the Mann-Whitney U test or KS test (from MSDS660) to see if the result is significant or not
  - Triple Bonus: run the training through one step (make sure one of the agents has won), and print out the winning agent's Q-dictionary. Can you explain the board's notation, and the Q-values?

Woohoo! Done

Machine ethics

- Can machines be taught right and wrong (ethical behavior)?
- What sort of protections should we put in place to ensure ethical machine learning approaches?
- What problems should we especially watch out for with machine learning ethics?

- https://en.wikipedia.org/wiki/Machine_ethics
- https://www.wired.com/story/artificial-intelligence-seeks-an-ethical-conscience/
- https://theconversation.com/ethics-by-numbers-how-to-build-machine-learning-that-cares-85399
- https://arstechnica.com/information-technology/2016/02/the-nsas-skynet-program-may-be-killing-thousands-of-innocent-people/