Reinforcement learning

Joseph Leslie Hsueh 薛世恩

Main Component of RL

- Environmemt
  States, which transfer when you make an action.

- Action
  The decision you make.

- Reward(or Penalty)
  The feedback after your decision.

Methods

- Q learning https://github.com/joseph1027/reinforcement_learning/blob/master/q_learning_find_treasure_with_barrier.py

- DQN https://github.com/joseph1027/reinforcement_learning/blob/master/DQN.py

Q learning

- We use Q table to store the value that help us do the decision. (Limited by memory capacity, we could not have unlimited states and actions)

- The model always look at the Q table before making action. It choose the action with max value(max future reward).

- Get a reward which comes from the environment after our action. And up date the value in Q table with target value which come from bellman equation and predict value.

Q Table Update

	Action1 (up)	Action2 (down)	Action3 (right)	Action4 (left)
State1	0.1	0.2	**0.6**	0.2
State2	0.1	**0.2**	-0.5	-0.1
State3	…	…	…	…

Assume GAMMA=1,ALPHA=1

	Action1 (up)	Action2 (down)	Action3 (right)	Action4 (left)
State1	0.1	0.2	**0.3**	0.2
State2	0.1	**0.2**	-0.5	-0.1
State3	…	…	…	…

```
q_predict = q_table.loc[S, A]
q_target = R + GAMMA * q_table.iloc[S_, :].max()
q_table.loc[S, A] += ALPHA * (q_target - q_predict)
```

State = State1
Action = ‘right’
Reward=0.1
State’ = State2

```
q_predict = 0.6
q_target = 0.1 + 1*(0.2) = 0.3
0.6 + 1*(0.3-0.6) = 0.3
```

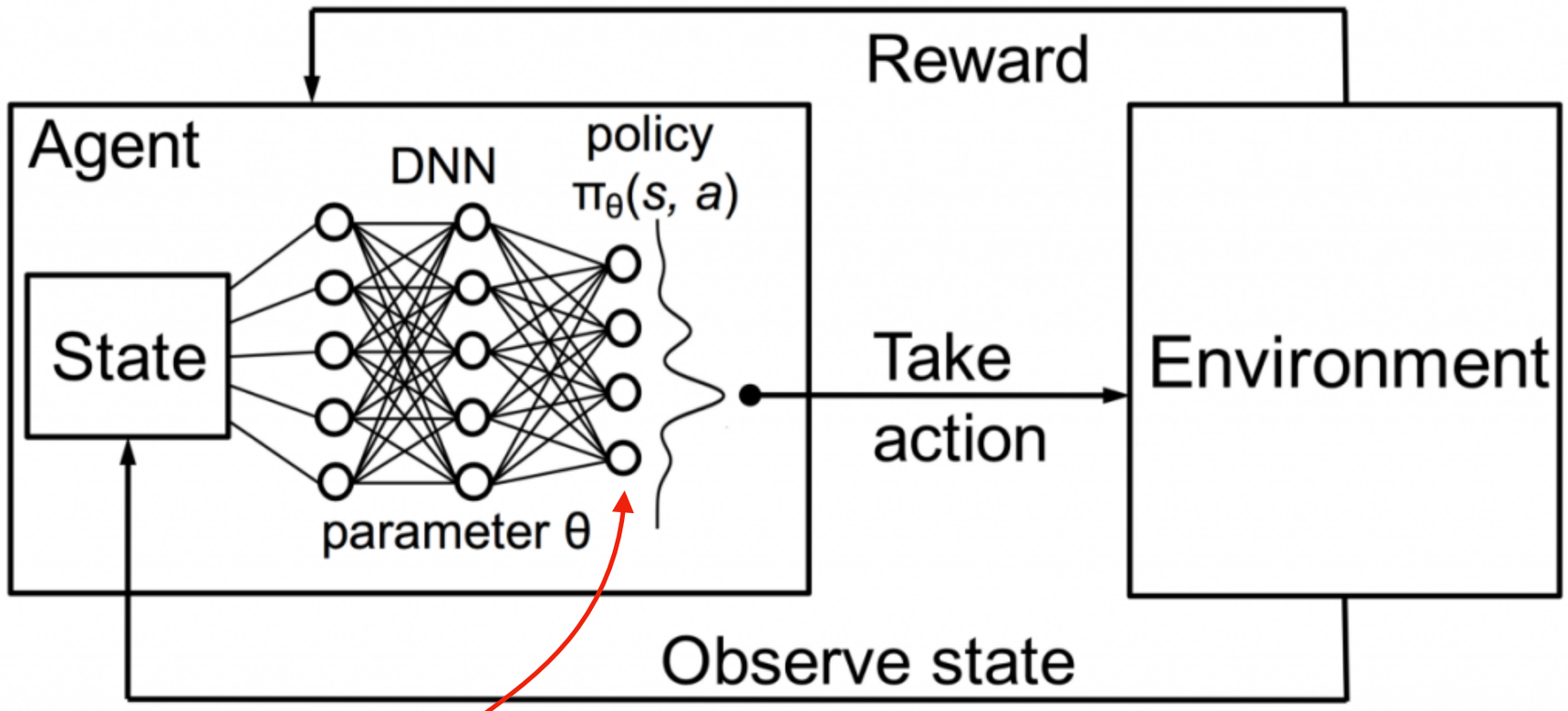
DQN

- Comparing to Q learning, it replace Q table with a neural network which solves the problem of infinite states and actions.

- **Fix Q-Target** : We have to train and maintain a second neural network whose structure is exactly the same with the first one but contains previous parameters.(If we use the same parameters, then our target keep changing when we update the model.) We update the target after a period of time.

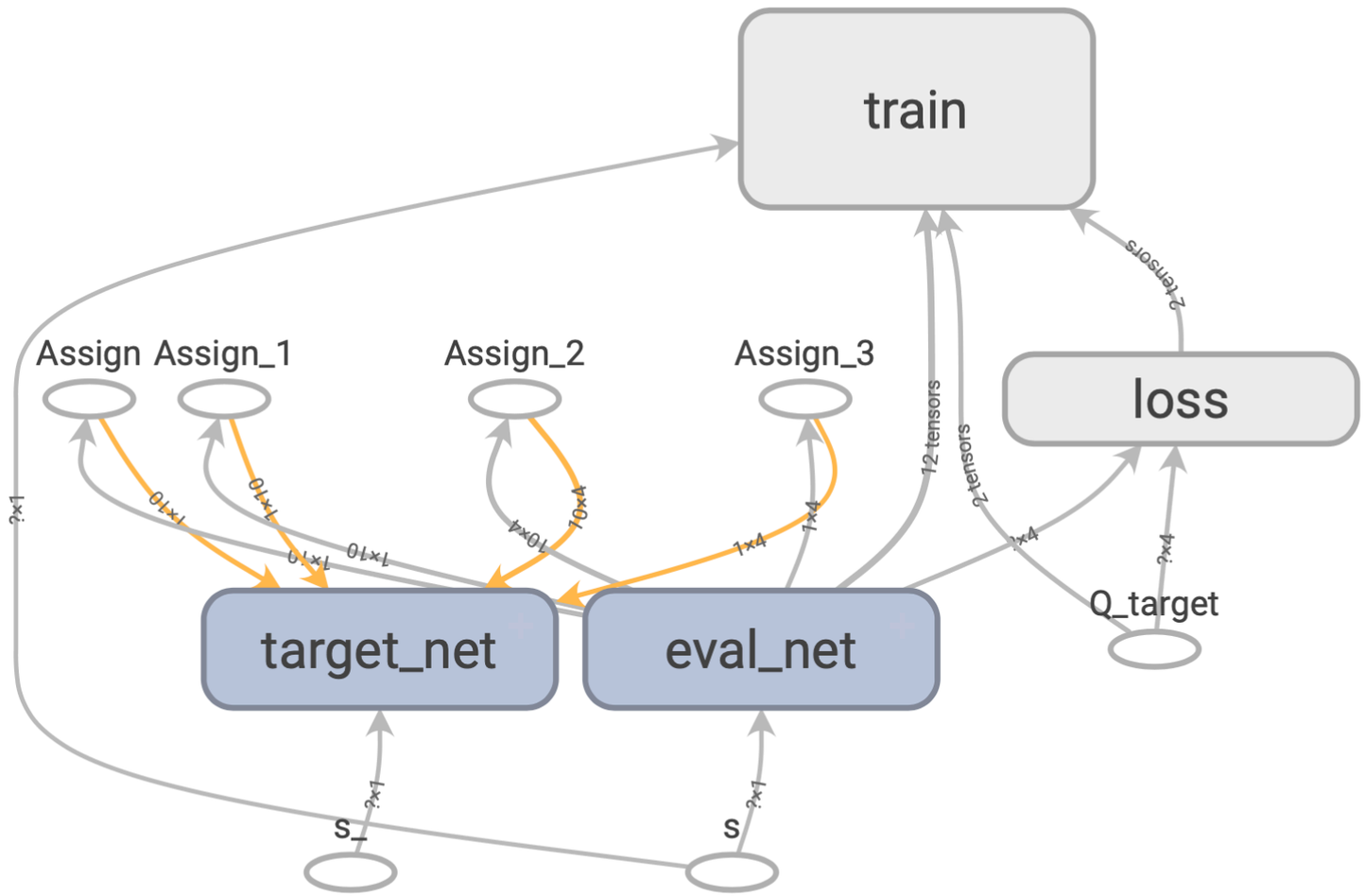
https://medium.freecodecamp.org/improvements-in-deep-q-learning-dueling-double-dqn-prioritized-experience-replay-and-fixed-58b130cc5682

DQN Structure



We use DNN to replace Q table.
The output layer of DNN is the value of each action.
We could decide out action according to those values.

Fixed Q-Target



Use tf assign to update the parameters of target model.

Hints for updating

- q_target and q_eval contain value of actions. However, the only value that we need is the value of chosen the action. All the other should be 0 when we do back propagation.
- For example we want to do…
  q_target - q_eval = [1,0,0] - [-1,0,0] = [2,0,0]
  q_eval = [-1,0,0] means we choose action 0 and Q(s,a0) = -1 with action0.
  However, Q(s,a1),Q(s,a2) = 0.
  q_target = [1,0,0] means r+gamma*maxQ(s_) = 1.
  In addition, no matter which action we choose on s_,
  we have to make our value align the position of action on q_eval.
  Thus, we put 1 on the position of action0 => [1,0,0].