法律声明

- 本课件包括：演示文稿，示例，代码，题库，视频和声音等，小象学院拥有完全知识产权的权利；只限于善意学习者在本课程使用，不得在课程范围外向任何第三方散播。任何其他人或机构不得盗版、复制、仿造其中的创意，我们将保留一切通过法律手段追究违反者的权利。

- 课程详情请咨询
  - 微信公众号：大数据分析挖掘
  - 新浪微博：ChinaHadoop

增强学习

主讲人：李伟

纽约城市大学博士

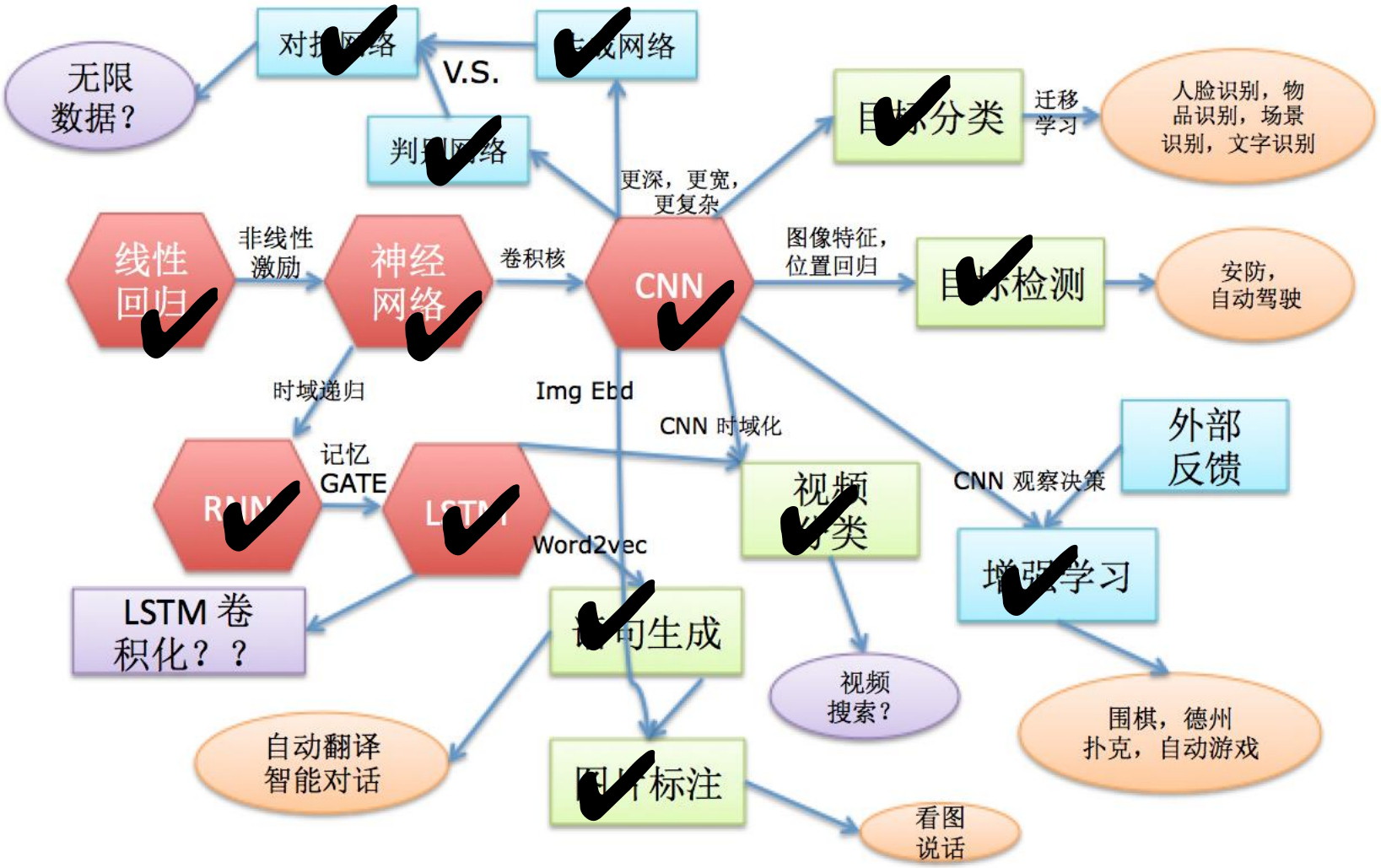
主要研究深度学习，计算机视觉，人脸计算

多篇重要研究文章作者，重要会议期刊审稿人

微博ID：weightlee03（相关资料分享）

GitHub ID：wiibrew（课程代码发布）

结构


无限数据？
对抗网络
V.S.
判别网络
生成网络
更深，更宽，更复杂
目标分类
迁移学习
人脸识别，物品识别，场景识别，文字识别
线性回归
非线性激励
神经网络
卷积核
CNN
图像特征，位置回归
目标检测
安防，自动驾驶
时域递归
Img Ebd
CNN 时域化
记忆
GATE
RNN
LSTM
视频分类
CNN 观察决策
外部反馈
增强学习
LSTM 卷积化？？
Word2vec
语句生成
视频搜索？
围棋，德州扑克，自动游戏
自动翻译
智能对话
图片标注
看图说话

提纲

- 1. 增强学习基础
- 2. DQN 深度增强学习
- 3. DQN 改进模型
- 4. A3C 模型
- 5. 实例学习，自动游戏机器人

期待目标

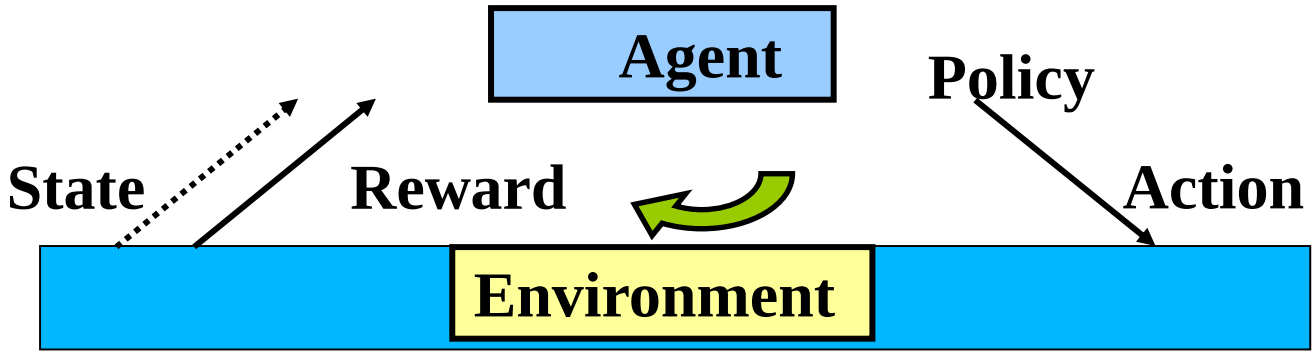
- 1. 了解增强学习的基本组成
- 2. 简单任务中Q – learning如何实现最优策略
- 3. DQN工作原理，训练过程用到的调整策略，优化方式，特殊设计的用途
- 4.基本DQN存在的问题，

提纲

- 1. 增强学习基础
- 2. DQN 深度增强学习
- 3. DQN 改进模型
- 4. A3C 模型
- 5. 实例学习，自动游戏机器人

增强学习基础

什么是增强学习



Agent：要学习的智能程序

Policy：程序知道所处某状态后采取行为的策略(复杂情况DL，简单情况lookup table)

Environment: 智能程序交互的空间，接受action产生状态变化，返回reward，可以是真实世界，游戏模拟器，棋牌等

增强学习基础

- 什么是增强学习
- 激励方程 – –reward function

  某次行为结果产生的激励；

  定义增强学习的目的
- 价值方程 – –value function

  agent能够带来的长期累计回报

  激励方程短期，价值方程长期
- 环境模型 – –Model of the environment

  环境模拟模型，模拟action作用之后环境返回的价值／激励以及状态变化，模拟器中不需要，真实世界中有指导训练意义

增强学习基础

- 增强学习

通过训练学到最优的state－action映射关系的过程，使agent得到最好的value／reward

- MDPs(Markov Decision Process)

马尔可夫决策过程

状态 state

行为 action

状态转化 transition

回报 reward

增强学习基础

增强学习中的问题

1. 策略学习 Policy learning

   policy 梯度学习

2. 价值／回报迭代学习

   确定方法： Q－learning；DQN，DQN扩展

3. 环境模型学习 Environment modeling

   知道此刻的状态以及行为(s, a) 预测下一刻的状态以及回报，模拟真是的环境反馈

增强学习基础

Q-learning

$$Q(s_t, a_t) = max\, R_{t+1}$$

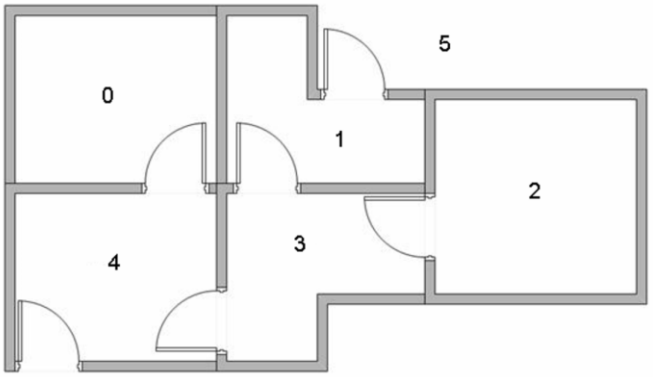
$$\pi(s) = argmax_a\, Q(s,a)$$

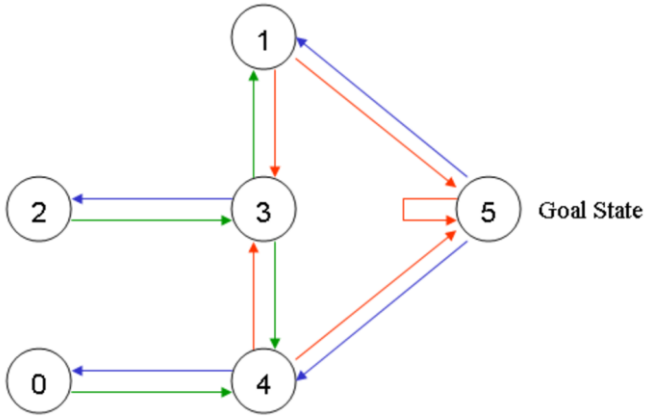
$$Q(s,a) = r + \gamma max_{a'} Q(s',a')$$

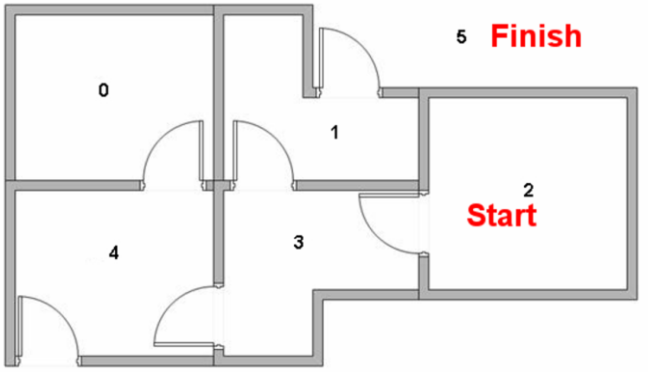
```
initialize Q[num_states,num_actions] arbitrarily
observe initial state s
repeat
      select and carry out an action a
      observe reward r and new state s'
      Q[s,a] = Q[s,a] + α(r + γ max_a' Q[s',a'] - Q[s,a])
      s = s'
until terminated
```

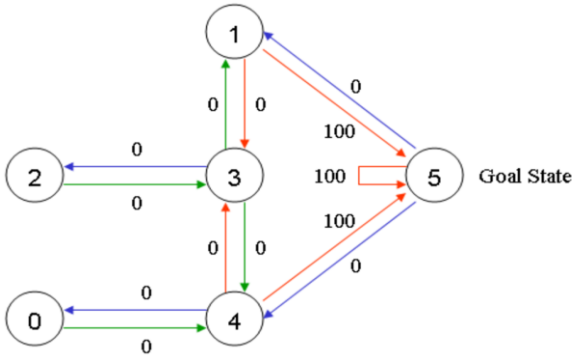
增强学习基础

Q-learning 实例 http://mnemstudio.org/path-finding-q-learning-tutorial.htm









增强学习基础

- Q-learning 实例 http://mnemstudio.org/path-finding-q-learning-tutorial.htm

构建 R，Q table，policy instruction guide，策略的参考

$$R = \begin{array}{c} \\ \text{State} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \text{Action} \\ \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix} \end{array}$$

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

增强学习基础

Q-learning 实例 http://mnemstudio.org/path-finding-q-learning-tutorial.htm

假设： 当前s: 1， action to s:5

Q(state, action) = R(state, action) + Gamma * Max[Q(next state, all actions)]

Q(1, 5) = R(1, 5) + 0.8 * Max[Q(5, 1), Q(5, 4), Q(5, 5)] = 100 + 0.8 * 0 = 100

Action

State

$$R = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix} \end{array} \qquad Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

增强学习基础

Q-learning 实例 http://mnemstudio.org/path-finding-q-learning-tutorial.htm

假设：当前s：3，下一步应选择1

Q(state, action) = R(state, action) + Gamma * Max[Q(next state, all actions)]

Q(3,1)=R(3,1)+ 0.8 * Max[Q(1, 2), Q(1, 5)] = 0 + 0.8 * Max(0, 100) = 80

Action

$$R = \begin{array}{c} \text{State} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix} \end{array} \qquad Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 80 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

增强学习基础

- Q-learning 实例 http://mnemstudio.org/path-finding-q-learning-tutorial.htm

R每次保持不变，policy，Q具有可以积累

多次迭代计算机后Q趋向稳定

Action

State

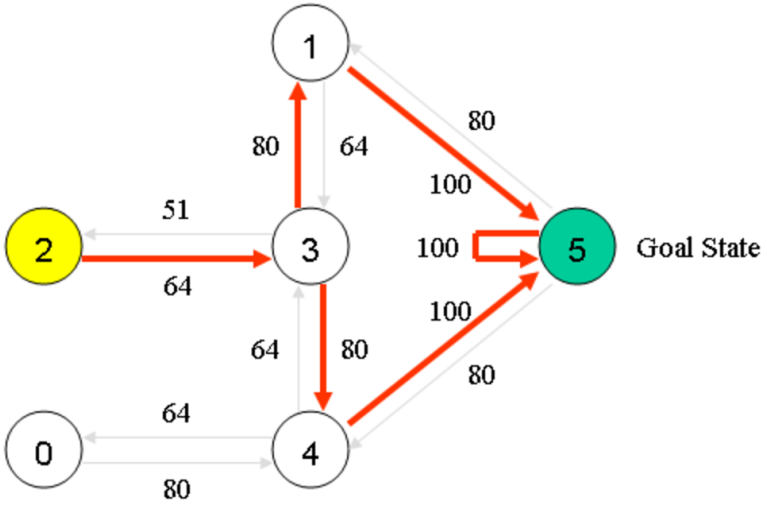
$$R=\begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array}\begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix} \end{array} \qquad Q=\begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array}\begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 80 & 0 \\ 0 & 0 & 0 & 64 & 0 & 100 \\ 0 & 0 & 0 & 64 & 0 & 0 \\ 0 & 80 & 51 & 0 & 80 & 0 \\ 64 & 0 & 0 & 64 & 0 & 100 \\ 0 & 80 & 0 & 0 & 80 & 100 \end{bmatrix} \end{array}$$

增强学习基础

Q-learning 实例 http://mnemstudio.org/path-finding-q-learning-tutorial.htm

Q 完成学习之后，可用于直接寻找最优解

$$R = \begin{array}{c} \\ \text{State} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \text{Action} \\ \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix} \end{array}$$



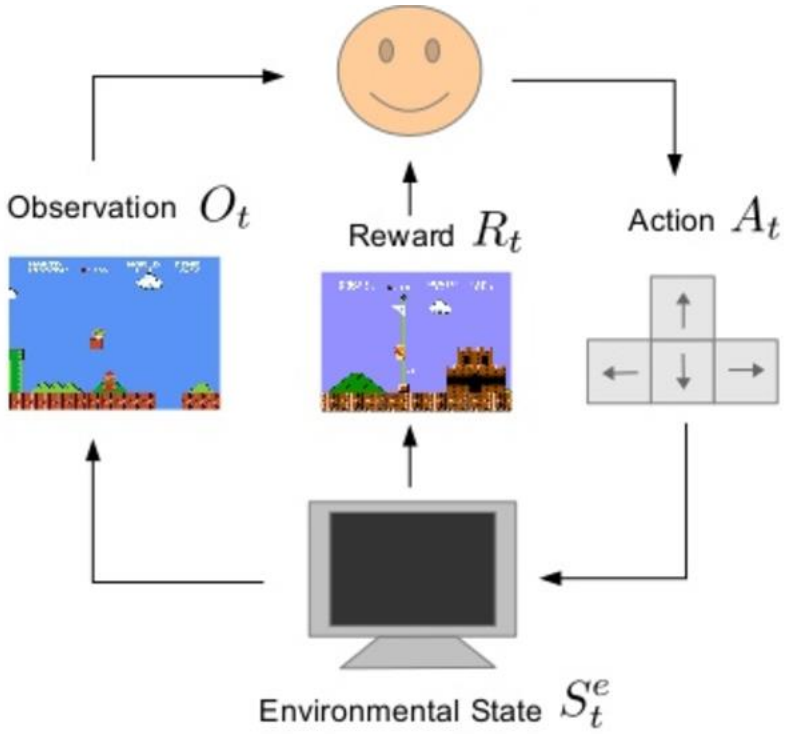
提纲

- 1. 增强学习基础
- 2. DQN 深度增强学习
- 3. DQN 改进模型
- 4. A3C 模型
- 5. 实例学习，自动游戏机器人

DQN 深度增强学习

- DQN (Deep Q-Network)

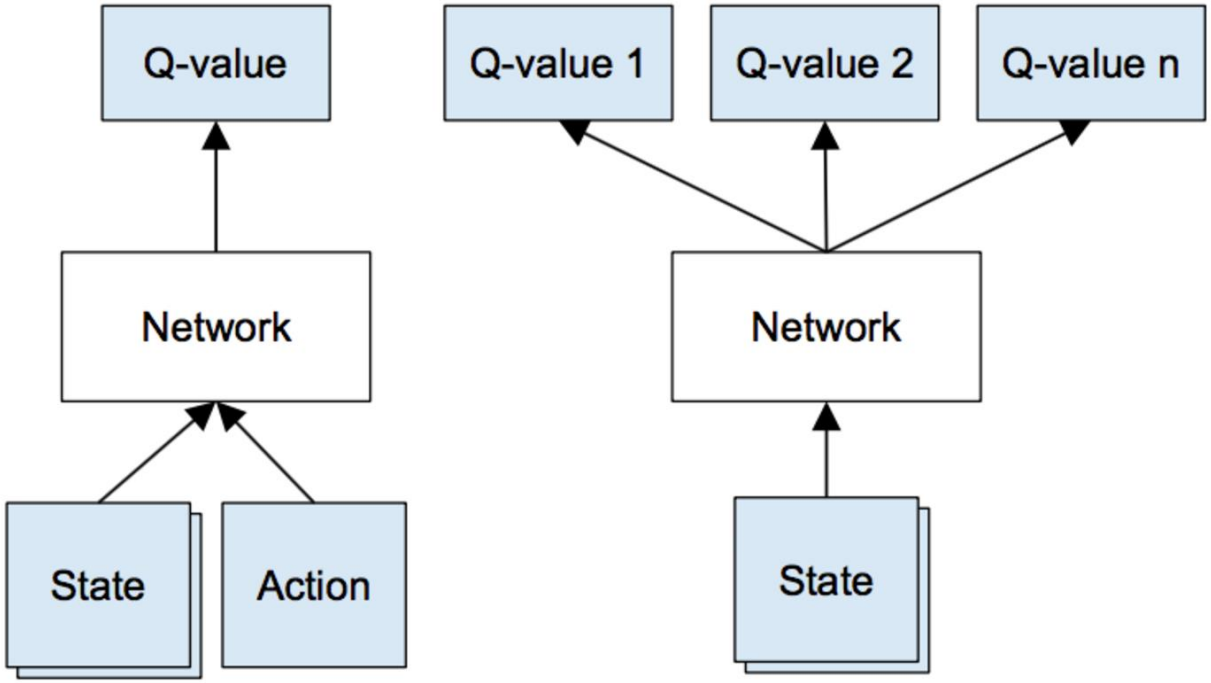
1. 2015 Nature，用于Atari 2600游戏的自动学习
2. 同Q-learning的过程类似，迭代优化
3. Action-Reward不是简单的table，是深度模型要学习的目标
4. 添加了经验回放(experience reply)，用于模型训练



DQN 深度增强学习

DQN (Deep Q-Network) vs Q

1. Q matrix 无法描述复杂问题
2. 神经网络作为Q 函数
3. 优化输入输出直接生成Q + a



DQN 深度增强学习

- DQN (Deep Q-Network) 基本结构

Layer	Input	Filter size	Stride	Num filters	Activation	Output
conv1	84x84x4	8x8	4	32	ReLU	20x20x32
conv2	20x20x32	4x4	2	64	ReLU	9x9x64
conv3	9x9x64	3x3	1	64	ReLU	7x7x64
fc4	7x7x64			512	ReLU	512
fc5	512			18	Linear	18

没有pooling 层：位置信息很重要，不希望被弱化

DQN 深度增强学习

- DQN (Deep Q-Network) loss以及优化

- Loss $$L = \frac{1}{2}[\underbrace{r + max_{a'}Q(s', a')}_{\text{target}} - \underbrace{Q(s, a)}_{\text{prediction}}]^2$$

- 训练过程：给定一个transition<s, a, r, s'>

1. 前向计算当前状态s，得到action-value列表
2. 前向计算下一状态s‘，得到$max_{a'}(s', a')$
3. 设置target值，2中action对应Q更新，其他同Q(s,a)一致
4. 反向计算梯度

DQN 深度增强学习

- DQN 经验回放

<s,a,r,s'> 存储

只用最近变化，数据少，容易收敛到局部极值

使训练更像监督学习

中间过程可以反复使用帮助训练

很像针对性练习

DQN 深度增强学习

- DQN：exploration-exploitation

深度探索－随机尝试

深度探索一直在优化自己最初的方向，但有可能是错的，贪婪算法

随机尝试按照一定概率允许尝试不同的可能性

ε-exploration greedy

按照一定概率进行随机／最优action尝试

ε按照递减顺序进行

DQN 深度增强学习

DQN算法

```
initialize replay memory D
initialize action-value function Q with random weights
observe initial state s
repeat
      select an action a
            with probability ε select a random action
            otherwise select a = argmax_a' Q(s,a')
      carry out action a
      observe reward r and new state s'
      store experience <s, a, r, s'> in replay memory D

      sample random transitions <ss, aa, rr, ss'> from replay memory D
      calculate target for each minibatch transition
            if ss' is terminal state then tt = rr
            otherwise tt = rr + γmax_a' Q(ss', aa')
      train the Q network using (tt - Q(ss, aa))^2 as loss

      s = s'
until terminated
```

提纲

- 1. 增强学习基础
- 2. DQN 深度增强学习
- 3. DQN 改进模型
- 4. A3C 模型
- 5. 实例学习，自动游戏机器人

DQN 改进模型

1. Double DQN

DQN可能问题：在目标max(Q(s',a'))确定过程中，选用的方法是直接找到value-action列表的最大值，由于模型的不稳定性，最大值不一定是最优值。

改进方法：训练两个独立的模型Q_1，Q_2

$$Q_1(s,a) \rightarrow r + \gamma Q_2(s', argmax_a Q_1(s',a)) \qquad Q_2(s,a) \rightarrow r + \gamma Q_1(s', argmax_a Q_2(s',a))$$

Target 与prediction分别用不同的模型生成

DQN 改进模型

1. Double DQN – 优先回放 PER(Prioritized Experience Replay)

训练过程中，每个batch的reply对训练的贡献并不相同，target和prediction预测差别大的有更大的贡献

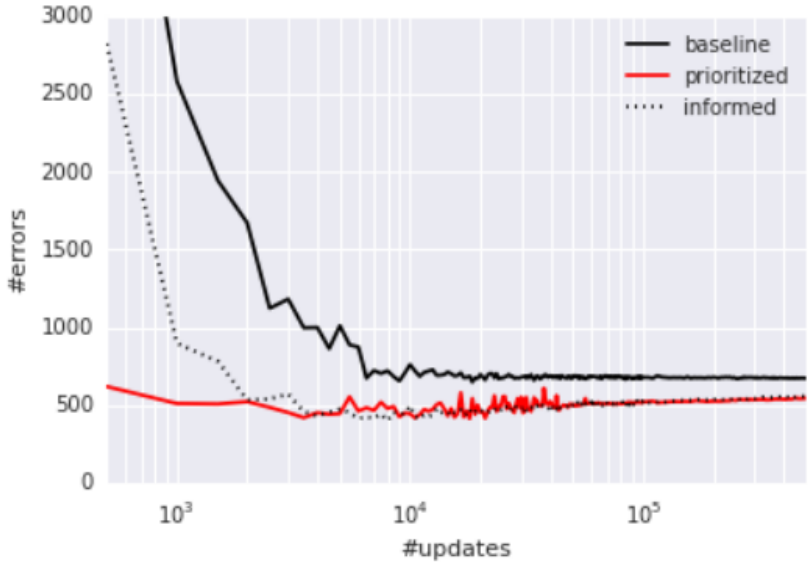
$$error = |Q(s,a) - T(S)|$$

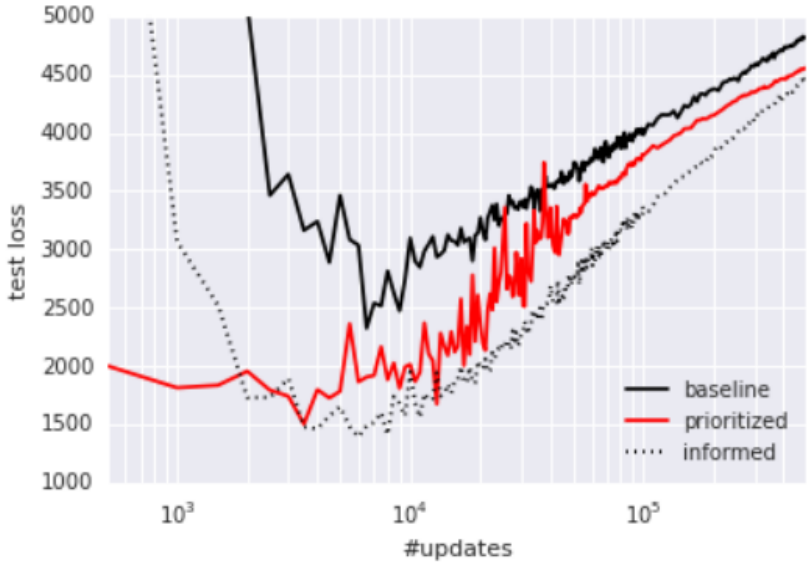
$$p = (error + \epsilon)^{\alpha}$$

$$P_i = \frac{p_i}{\sum_k p_k}$$

DQN 改进模型

1. Double DQN改进效果





DQN 改进模型

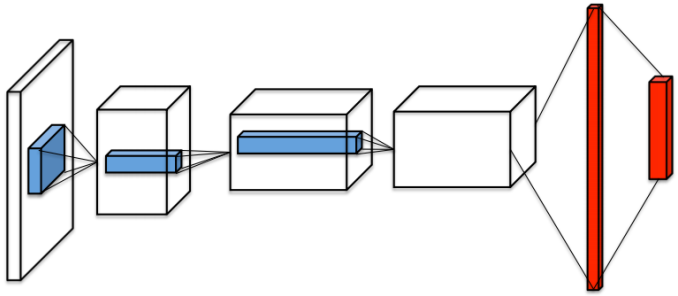
2. Dueling DQN

原理

最终的一个value loss

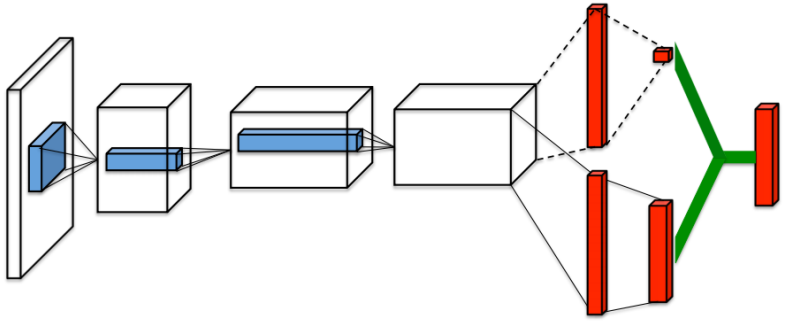
没有学习过程的某些中

间量没有引导作用



中间层分出value，advantage

层 Q(s, a)=V(s)+A



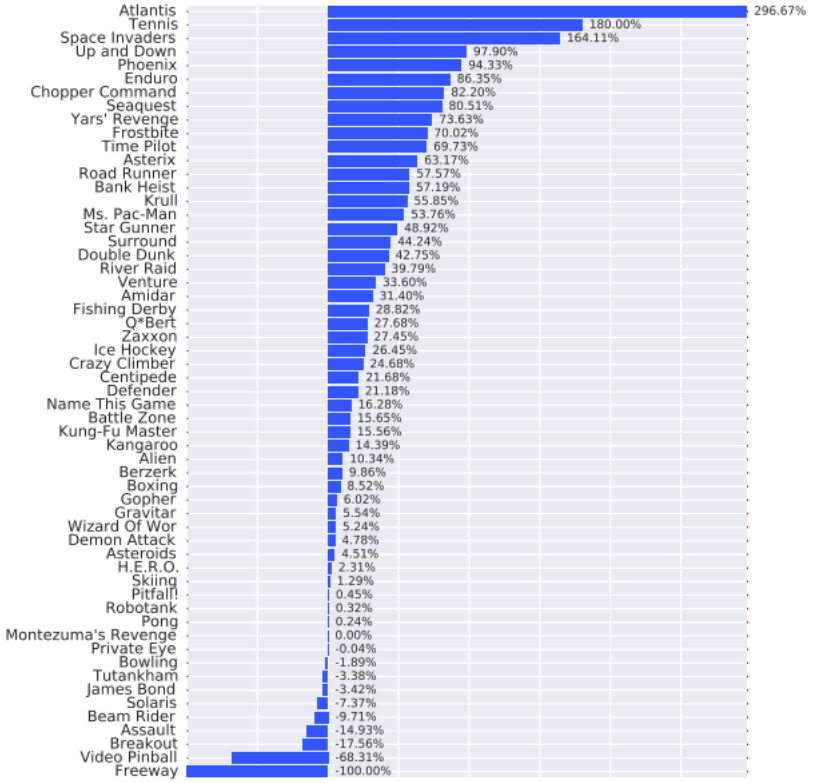
最终输出均为action – value

DQN 改进模型

2. Dueling DQN

Dueling结构特点：训练中侧重于有用信息，

效果对比：



提纲

- 1. 增强学习基础
- 2. DQN 深度增强学习
- 3. DQN 改进模型
- 4. A3C 模型
- 5. 实例学习，自动游戏机器人

A3C 模型

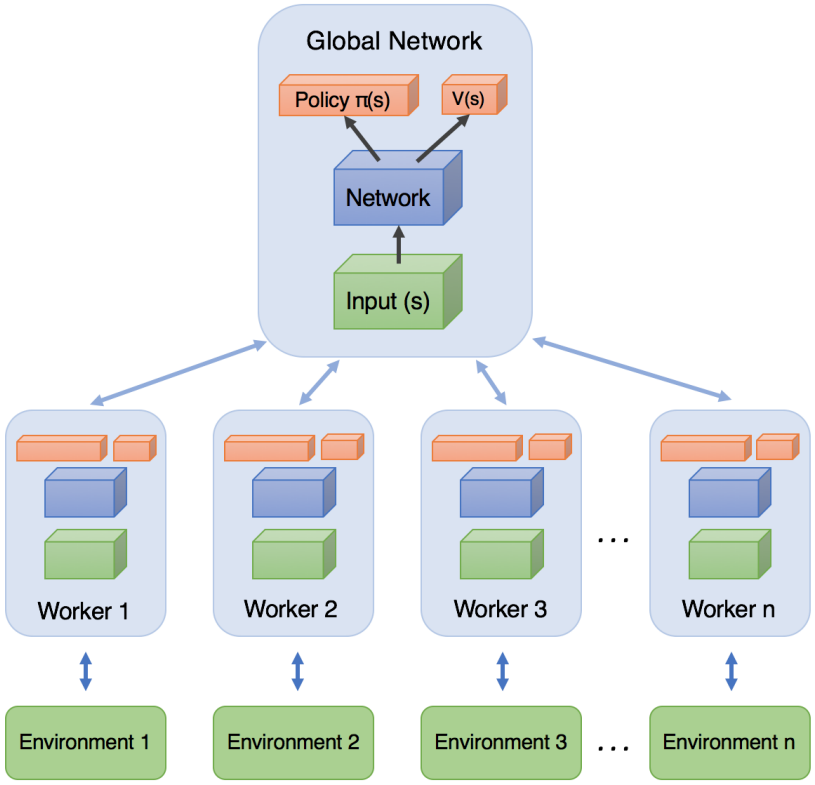
- A3C – Asynchronous Advantage Actor-Critic

1. Asynchronous 异步

2. Advantage 优势比较

3. Actor-Critic 回报反馈

   决策（批评－演员）



A3C 模型

- A3C – Asynchronous Advantage Actor-Critic

1. Asynchronous 异步

DQN： 单个agent，单个神经网络，一个环境

A3C： 一个全局神经网络，多个worker agent，每个agent复制一份神经网络，一个环境，单独进行优化

通过独立的进行多个worker agent训练，增加训练的多样性

A3C 模型

- A3C – Asynchronous Advantage Actor-Critic

2. Actor-Critic 演员 – 评论

演员：Policy，根据神经网络推出state下应有的行为的概率分布

评论：Value，不同行为能够得到的回报

Value和policy结合，通过fc layer 生成

A3C 模型

- A3C – Asynchronous Advantage Actor-Critic

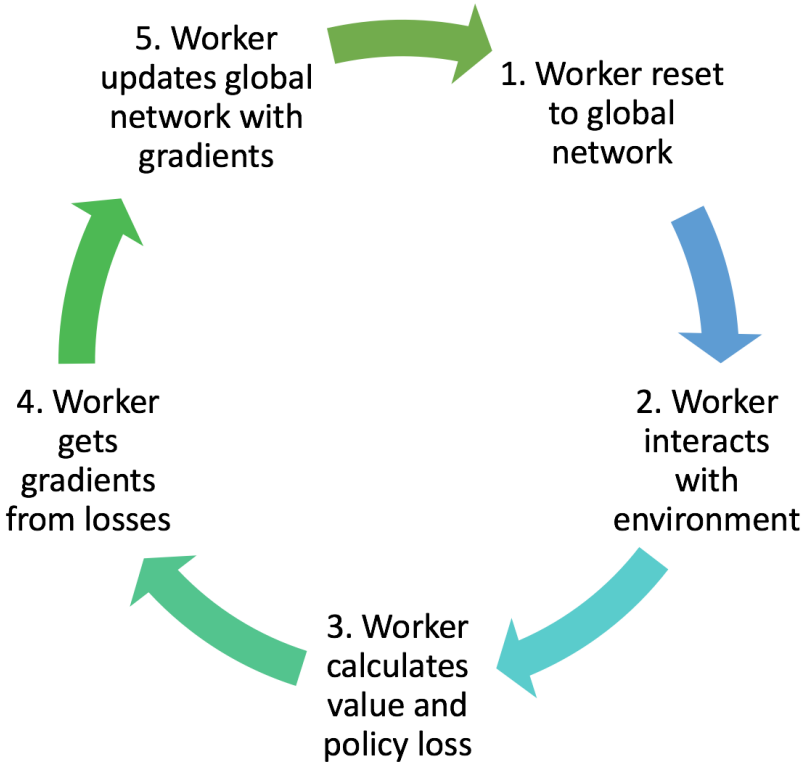
3. Advantage 不仅考虑模型的回报，还考虑某一个具体的行为带来的贡献有多大

A=R-V(s) A: advantage R: reward V(s): Value 方程， advantage 用于 value loss 的形成

A3C 模型

A3C – Asynchronous Advantage Actor-Critic

3个A 结合



$$Value\ Loss: L = \Sigma(R - V(s))^2$$

$$Policy\ Loss: L = log(\pi(s)) * A(s) + \beta * H(\pi)$$

https://medium.com/emergent-future/sim
tensorflow-part-8-asynchronous-actor-critic-agents-a3c-c88f72a5e9f2

提纲

- 1. 增强学习基础
- 2. DQN 深度增强学习
- 3. DQN 改进模型
- 4. A3C 模型
- 5. 实例学习，自动游戏机器人

实例学习

- DQN + GYM + Tensorflow

总结

- 有问题请到课后交流区
  - 问题答疑： http://www.xxwenda.com/
    - 可邀请老师或者其他人回答问题
- 课堂QQ群438285995，微信群
- 讲师微博：weightlee03，每周不定期分享DL资料
- GitHub ID：wiibrew （课程代码发布）

https://github.com/wiibrew/DeepLearningCourseCodes