增强学习（Reinforcement Learning）简介

Jan 4, 2017 • Snowkylin

Categories: RL

增强学习（Reinforcement Learning）是介于有监督学习和无监督学习之间的一种方式。在有监督学习中对于每个训练样本都有对应的标签，无监督学习完全没有标签，而增强学习则有稀疏且时延的标签（即奖励）。智能体基于奖励在环境中进行动作。

本部分介绍主要参考[1]（部分为翻译，强烈建议阅读原文，以及这里有一个深度增强学习的资料整理）。

为了形式化增强学习这一过程，首先介绍马尔科夫决策过程（Markov Decision Process）。假设你是一个智能体（Agent），在一个环境的特定状态（State）中，你可以做出特定动作（Action），而这些动作有时会带来奖励（Reward）。每当智能体做出一个动作，就会进入一个新的状态。如何选择进行什么动作则叫做策略（Policy）。状态和动作的集合以及从一个状态到另一个状态的转移规则构成了一个马尔科夫决策过程。这个过程的一个片段由一个定长的状态-动作-奖励序列所构成，如：

$$s_0, a_0, r_1, s_1, a_1, r_2, s_2, \ldots, s_{n-1}, a_{n-1}, r_n, s_n$$

其中s_i代表状态，a_i代表动作，r_{i+1}代表进行动作后获得的奖励，s_n是终止状态。一个马尔科夫决策过程建立在马尔科夫假设上，即下一时刻的状态s_{i+1}只和当前状态s_i和动作a_i有关，和之前的状态及动作无关。

为了能够在长期也表现良好，我们不能只考虑即时的奖励，还要考虑我们未来能获得的奖励。对于马尔科夫决策过程，我们可以将某个时间t的总未来奖励（total future reward）定义为：

$$R_t = r_t + r_{t+1} + r_{t+2} + \ldots + r_n$$

但是环境是随机性的，我们不能确定我们下次做相同动作时会不会得到相同的回报，越远的将来随机性越大。所以我们引入有折扣的未来奖励（discounted future reward）为：

$$R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots + \gamma^{n-t} r_n$$

其中γ是0到1之间的折扣因子。如果$\gamma = 0$，我们将变得“短视”，只关心即时收益。

对于一个智能体而言，一个好的策略是：**总是选择能够将（有折扣的）未来奖励最大化的动作**。

于是，我们引入Q-learning的概念，定义函数$Q(s,a)$表示我们在状态s进行动作a时的最大折扣未来奖励，即：

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

你可以简单地把$Q(s,a)$理解为“当在状态s进行状态a时能获得的最佳可能分数”。因为$Q(s,a)$表示给定状态下进行特定动作的“质量”（Quality），所以我们称之为“Q函数”。

于是，我们之前提到的“好的策略”可以表示为：

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

这里π表示策略，规定每种状态下我们应当怎样行动。

现在，如何计算Q函数呢？我们来关注一个转移$\langle s,a,r,s^{'}\rangle$。我们可以将Q在状态s和动作a时的值用下一时刻的状态$s^{'}$表示如下：

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

这被称为贝尔曼公式。其实很好理解——对于当前状态s和动作a的最大未来奖励等于动作a后的立即奖励加上下一时刻状态s在所有可能动作下能获得的的最大未来奖励。

而Q-learning的主要想法是，我们可以用贝尔曼公式迭代近似Q函数。我们考虑Q函数是一个以状态为行，动作为列的表格，则Q-learning的算法如下所示。其中，α是学习率（如果$\alpha=1$则就是贝尔曼公式）。

Algorithm 5 Q-learning 迭代算法

随机初始化 $Q[num\_states, num\_actions]$
获得初始状态 s
repeat
  选择并执行一个动作 a
  获得奖励 r 和新状态 $s^{'}$
  $Q[s,a]=Q[s,a]+\alpha(r+\gamma\max_{a'}Q[s^{'},a^{'}]-Q[s,a])$
  $s=s^{'}$
until 终止

看上去已经挺不错了，然而还有一个问题亟待解决。在很多问题中，状态往往多到难以计数（比如屏幕像素点的状态可以是$256^{84\times84\times4}$种）。如果我们还把Q函数当做一个表格，算法将难以操作。此时，我们需要引入深度学习，用神经网络来表示Q函数。有两种方法，一种是把状态s和动作t都输入神经网络，输出$Q(s,t)$，另一种是输入状态s，对于所有可能的动作a_i（一般动作种类不会太多，比如弹球游戏就三种：左、右、不动）都输出$Q(s,a_i)$。方便起见往往使用后者。

损失函数如下：

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

给定一个转移$\langle s, a, r, s' \rangle$，Q表格的更新步骤需要换为以下步骤：

- 将当前状态s输入神经网络，得到对于所有动作的Q-value预测值（prediction）
- 将下一时刻的状态s'输入神经网络，求其输出的动作的Q-value预测值中最大的动作$a'_{max} = \max_{a'} Q(s', a')$
- 将动作a'_{max}的target设为$r + \max_{a'} Q(s', a')$（第二步中的值），其余动作的target还是第一步中的值（从而损失为0）
- 反向传播更新权值

参考

1. Guest Post (Part I): Demystifying Deep Reinforcement Learning https://www.nervanasys.com/demystifying-deep-reinforcement-learning/ ↩

雪麒的百草园 - Snowkylin's Blog

Snowkylin Lazarus (Xihan Li)
xihanli316 AT gmail dot com
(replace AT by @, dot by .)

snowkylin
snowkylin

Seeking the Truth and Pioneering New Trails.