**实验报告**

**杨春陶 2017202130**

**Cartpole 环境介绍**

C*art Pole*即车杆游戏，游戏模型如下图所示。游戏里面有一个小车，上有竖着一根杆子，每次重置后的初始状态会有所不同。小车需要左右移动来保持杆子竖直，为了保证游戏继续进行需要满足以下两个条件：

1.杆子倾斜的角度θ不能大于15°

2.小车移动的位置xxx需保持在一定范围（中间到两边各2.4个单位长度）

动作（action）：

1.左移（0）

2.右移（1）

状态变量（state variables）：

1.x：小车在轨道上的位置（position of the cart on the track）

2.θ：杆子与竖直方向的夹角（angle of the pole with the vertical）

3.：小车速度（cart velocity）

4.：角度变化率（rate of change of the angle）

import gym

env = gym.make('CartPole-v0')

observation = env.reset()

print(observation)

#> [-0.00478028 -0.02917182  0.00313288  0.03160127]

以上代码显示了初始状态下的取值，每次调用env.reset( )将重新产生一个初始状态。打印出的observation的四个元素分别表示了小车位置、小车速度、杆子夹角及角变化率。

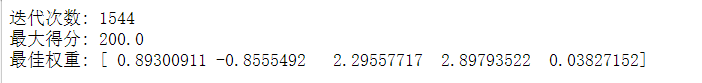
游戏奖励（reward）：

在gym的Cart Pole环境（env）里面，左移或者右移小车的action之后，env会返回一个+1的reward。其中CartPole-v0中到达200个reward之后，游戏也会结束。

1. **爬山算法优化cartpole决策**

爬山算法的基本思路是每次迭代时给当前取得的最优权重加上一组随机值，如果加上这组值使得有效控制倒立摆的持续时间变长了那么就更新它为最优权重，如果没有得到改善就保持原来的值不变，直到迭代结束。在迭代过程中，模型的参数不断得到优化，最终得到一组最优的权值作为控制模型的解。

在爬山算法中，简单来说就是用当前给的observation这个四维向量，通过建立一个简单的线性模型，来预测下一步正确的action。因为每次更新权值都是基于一轮n个动作以后的reward值更新，不能像求导一样定向更新权值。



按照爬山算法的测试结果并不理想，因为是无方向的调整权值，导致和随机数再一次生成权值相比，效果虽提升了，但平均也需要迭代1700多次才能找到最优。

1. **监督式学习**

实际应用中的机器学习在大部分情况下我们都会使用监督式学习。

监督式学习指的是你拥有一个输入变量和一个输出变量，使用某种算法去学习从输入到输出的映射函数

这种学习方式就称之为监督式学习，因为算法学习从训练数据集学习的过程可以被看成类似于一名教师在监督学习学习的过程。我们已经知道了正确的答案，而算法不断迭代来对训练数据做出预测同时不断被一名教师修正。当算法达到一个可接受程度的表现时学习过程停止。

监督式学习问题可以进一步被分为回归和分类问题

* 分类:分类问题指的是当输出变量属于一个范畴，比如“红色”和“蓝色”或者“生病”和“未生病”。
* 回归:回归问题指的是输出变量是一个实值，比如“价格”和“重量”

邻近算法，或者说K最近邻(kNN，k-NearestNeighbor)分类算法是数据挖掘分类技术中最简单的方法之一。所谓K最近邻，就是k个最近的邻居的意思，说的是每个样本都可以用它最接近的k个邻居来代表。Cover和Hart在1968年提出了最初的邻近算法。KNN是一种分类(classification)算法，它输入基于实例的学习（instance-based learning），属于懒惰学习（lazy learning）即KNN没有显式的学习过程，也就是说没有训练阶段，数据集事先已有了分类和特征值，待收到新样本后直接进行处理。与急切学习（eager learning）相对应。



决策树（Decision Tree）是一种非参数的有监督学习方法，它能够从一系列有特征和标签的数据中总结出决策规则，并用树状图的结构来呈现这些规则，以解决分类和回归问题。

优点：简单易懂，容易解释，可视化，适用性广

缺点：容易过拟合，

　　 数据中的小变化会影响结果，不稳定，

　　 每一个节点的选择都是贪婪算法，不能保证全局最优解。



随机森林就是通过集成学习的思想将多棵树集成的一种算法，它的基本单元是决策树，而它的本质属于机器学习的一大分支——集成学习（Ensemble Learning）方法。随机森林的名称中有两个关键词，一个是“随机”，一个就是“森林”。“森林”我们很好理解，一棵叫做树，那么成百上千棵就可以叫做森林了，这样的比喻还是很贴切的，其实这也是随机森林的主要思想--集成思想的体现。



logistic回归是一种广义线性回归（generalized linear model），因此与多重线性回归分析有很多相同之处。它们的模型形式基本上相同，都具有 w‘x+b，其中w和b是待求参数，其区别在于他们的[因变量](https://baike.baidu.com/item/%E5%9B%A0%E5%8F%98%E9%87%8F)不同，多重线性回归直接将w‘x+b作为因变量，即y =w‘x+b，而logistic回归则通过函数L将w‘x+b对应一个隐状态p，p =L(w‘x+b),然后根据p 与1-p的大小决定因变量的值。如果L是logistic函数，就是logistic回归，如果L是多项式函数就是多项式回归。



支持向量机（Support Vector Machine, SVM）的基本模型是在特征空间上找到最佳的分离超平面使得训练集上正负样本间隔最大。SVM是用来解决二分类问题的有监督学习算法，在引入了核方法之后SVM也可以用来解决非线性问题。

线性核函数



多项式核函数



监督式学习的好坏一部分取决于样本的好坏，一部分取决于模型的选择。

一开始我提供的样本是在随机决策下提供的action选择，所有的action只取决于这个动作是否能让木棒不倒，忽略了这个动作对下一个动作的影响，因此最后训练出来的结果只有70%的准确率，依靠这样的模型只能跑20分，后来我用模拟退火算法下跑出来的数据作为样本训练，准确率提高到了95%以上。

另外一个模型的选择。分类问题一般都源于线性分类问题，尤其是二分类问题。Cartpole游戏决策是一个典型的二分类问题，因此，大部分模型都表现良好。而对于解决非线性问题模型，比如SVM中的多项式核函数，准确率就大大折扣。由于cartpole模型较为简单，因此复杂的随机森林等模型就显得有点大材小用。在检验中，在小车游戏模型中使用逻辑分类法，小车模型的每一次运行均能达到最高分200。

1. **DQN神经网络**

简单的神经网络类似，比如单层隐藏层以relu为激活函数，其实就类似于模拟退火算法，并且依赖于样本提供，不能自动学习。

书中提到的Q网络，Q-learning是通过不停地探索和更新Q表中的Q值从而计算出智能体行动的最佳路径的，公式为

Q(s0,a)新=Q(s0,a) 旧 + α\* [Q(s0,a)目标 - Q(s0,a)旧]  
Q(s0,a)目标 =R(s1) + γ\*max Q(s1,a)

[深度学习](https://www.jianshu.com/p/4f6eb4b7c22f)就是用神经网络来学习数据，常见的深度学习网络如全连接的，CNN，RNN等等。

DQN不用Q表记录Q值，而是用神经网络来预测Q值，并通过不断更新神经网络从而学习到最优的行动路径。

DQN中有两个神经网络(NN)一个参数相对固定的网络，我们叫做target-net，用来获取Q-目标(Q-target)的数值, 另外一个叫做eval\_net用来获取Q-评估(Q-eval)的数值。

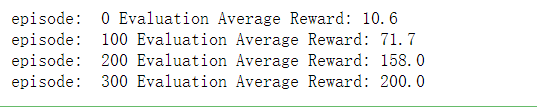
反向传播真正训练的网络是只有一个，就是eval\_net。target\_net 只做正向传播得到q\_target (q\_target = r +γ\*max Q(s,a)). 其中 Q(s,a)是若干个经过target-net正向传播的结果。

训练的数据是从记忆库中随机提取的，记忆库记录着每一个状态下的行动，奖励，和下一个状态的结果(s, a, r, s')。记忆库的大小有限，当记录满了数据之后，下一个数据会覆盖记忆库中的第一个数据，记忆库就是这样覆盖更新的。

q\_target的网络target\_net也会定期更新一下参数，由于target\_net和eval\_net的结构是一样的。更新q\_target网络的参数就是直接将q\_eval 的参数复制过来就行了。

greedygreedy策略，顾名思义，是一种贪婪策略，它每次都选择使得值函数最大的action，即at=maxaQ(St,a;ω)at=maxaQ(St,a;ω)。但是这种方式有问题，就是对于采样中没有出现过的(state, action) 对，由于没有评估，没有Q值，之后也不会再被采到。

DQN网络中



最后，在小车游戏模型中，进行多次效果检测：



在检验中，小棒模型的每一次运行均能达到最高分200。

可以看到策略网络中，需要250左右每个iteration，每个iteration有10个episode，而DQN修改参数后能在400个episode内达到最优。