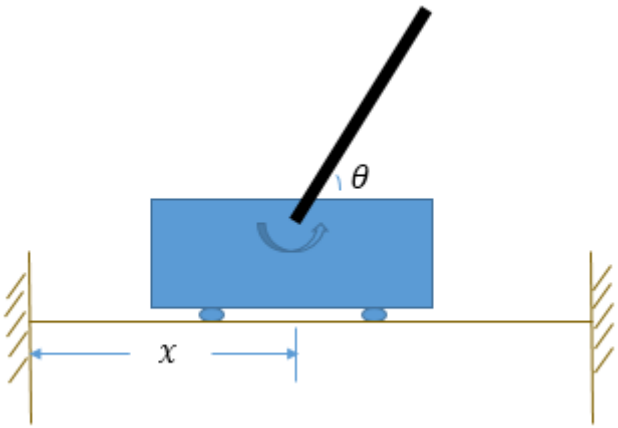
Machine Learning 17 ---MDP(2)

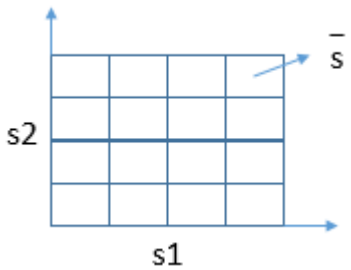
Andrew Ng 老师在课堂上就上一节课同学们不太明白的地方做了很详细的回答，具体的不重复看视频的前一段吧。即使是 Stanford 的学生学习这一方面的知识都是不容易的，我们更需要加油！

上一节课所讲到的 MDP 相关知识都是针对离散状态的：11 个格子表示 11 个状态。现在，让我们尝试将上一节课学到的众多知识点推广到更大的范围：在连续型状态下或者状态的数量是无限的情况下，我们如何解决？

举例来说，假如你想要控制无人机在三维空间中飞行，那么状态将会由方位坐标 x, y, z 以及姿态信息 ϕ, θ, ψ （航向角，分别表示横滚（roll）、俯仰（pitch）和偏仰（yaw）），更多情况下还会包括以上信息的导数 $\dot{x}, \dot{y}, \dot{z}, \dot{\phi}, \dot{\theta}, \dot{\psi}$ 也作为状态量。这里给出的 12 中状态都是连续型的。另外以一个倒立摆为例（接下来的文章会以该例子为主），其中的四个状态 $x, \theta, \dot{x}, \dot{\theta}$ 都是连续型状态。倒立摆模型如下图：



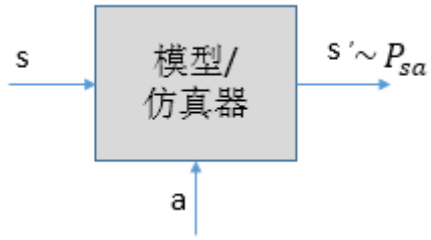
那么，我们如何在连续型状态下应用上一节课的两个算法 value iteration 与 policy iteration 呢？一种经常想到的方法就是将遇到的问题模型转化为已经得到解决的问题模型，即化连续为离散。假设有一个二维的连续状态空间（最简单的一种模型），那么我们可以尝试将它切分为若干个分离的单元，如下图 $\bar{s}$ 所示：



经过离散化处理之后，就能够像上节课所讲述的一样，找到 MDP 的五元组，运用算法来求解对应的各值。但这样的处理方式并不能取得好的效果。原因有：a.精度问题；b.维数祸根问题（用来描述当（数学）空间维度增加时，体积指数增加的难题）。

那么我们怎么做呢？为了引出解决办法，我们先来对问题进行恰当的描述：

假设状态 S 是连续的，行为集 A 是离散的。这当然是简单处理，以方便问题的解决。很多实际问题都是状态的个数要远远大于行为的个数，将行为集看成是离散的有一定的可操作性。比如倒立摆例子中状态个数为 4，而行为个数为 2（向左走、向右走）。假设现在有一个关于 MDP 的模型（或者称为仿真器），其本质上是一个关于如何表示状态转移概率 P_{sa} 的模型。让我们从下图来看如何表述该模型：它能够输入状态 s 和行为 a，然后能够输出 $s' \sim P_{sa}$ 服从状态转移分布。



现在，我们需要构建这样的一个模型，只有这样我才可以着手解决状态连续的问题。构建模型的方法有很多，其典型代表有：内部机理法建模、外部观察法建模（让我想起了本科专业中关于控制系统的建模方式，原来是相通的~）。

1.机理建模：即通过力学原理以及系统间各部分相关关系来构建模型。在倒立摆例子中，令

$$s=\begin{pmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{pmatrix}，那么\ \dot{s}=\begin{pmatrix} \dot{x} \\ \alpha \\ \dot{\theta} \\ \beta \end{pmatrix},其中\begin{cases} \alpha=\cdots \\ \beta=\cdots \end{cases}具体表达式不复述$$

那么我们就可以得到：$s_{t+1}=s_t+(\Delta t)\dot{s},假设\Delta t=0.1\sec$。

这样就建好了关于 MDP 的模型，只要输入 s 和 a，那么就可以得到下一个状态 S'。不过值得注意的是，该模型中的状态输出是确定性的；当然也可以领状态输出 S' 服从一定概率分布.

2.观察（学习）法建模：即通过外部观察，不再研究机理。具体做法如下：随机初始化 s_0，有意执行某个策略 π 或者随机执行，不断记录下一个状态，一直记录 T 个状态，即：

$$M次\begin{cases} s_0^{(1)} \xrightarrow{a_0^{(1)}} s_1^{(1)} \xrightarrow{a_1^{(1)}} s_2^{(1)} \xrightarrow{a_2^{(1)}} \cdots s_T^{(1)} \\ s_0^{(2)} \xrightarrow{a_0^{(2)}} s_1^{(2)} \xrightarrow{a_1^{(2)}} s_2^{(2)} \xrightarrow{a_2^{(2)}} \cdots s_T^{(2)} \\ \vdots \end{cases}$$

重复了 M 次操作之后（每一次的策略可以不同），我们就有了一个包含足够多数据的训练集。接下来我们就可以使用相关的学习型算法（前期我们学过的监督学习算法），将 s_t, a_t 作为参数，将 s_{t+1} 看成是标签（待预测量）。比如线性回归算法：$s_{t+1}=As_t+Ba_t$。在倒立摆

问题中，可知道待求量 $A \in \mathbb{R}^{4x4}, B \in \mathbb{R}^4$，目标函数可以使用平方误差回归来表示：

$$\min_{A,B} \frac{1}{2} \sum_{i=1}^{M} \sum_{t=0}^{T-1} || s_{t+1}^{(i)} - (As_t^{(i)} + Ba_t^{(i)}) ||^2$$

求得满足上述条件的矩阵 A, B 之后，我们就可以根据给出的 s_t, a_t 得到预测的 s_{t+1}。这个过程对于一路学习这门课程学到了第 17 课的同学来讲应该不难理解。实践表明，局部加权回归算法是一种较为有效的方法（用于构建 MDP 模型）。

通过观察法得到的模型可以表示为：

$$s_{t+1} = As_t + Ba_t \quad \text{（确定性模型）}$$

$$s_{t+1} = As_t + Ba_t + \zeta_t, \quad \zeta_t \sim normal(0, \zeta) \quad \text{（随机性模型）}$$

在随机性模型中，状态输出量 s_{t+1} 的随机性由 ζ_t 来提供，这样就可以实现状态连续型的 MDP 的模型建立了。不过需要记住的是，MDP 的模型构建不仅可以使用线性模型，也可以使用非线性模型，需要针对不同的问题加以选择（有一种学以致用的感觉，哈哈）。

千里之行的第一步已经踏出去了，接下来的路还很长。接着我们需要讲到的是：在 MDP 模型确定下来的基础上，我们如何求状态连续情况下的最优值函数 V^*？

我们现在介绍”fitted value iteration”算法来近似求解状态连续情况下的最优值函数 V^*。在求解该问题上，和之前的假设一样，我们仍然假设的是：**状态集 S 是连续的，行为集 A 是离散的。同时假设我们已经确定下来了 MDP 模型。**

让我们先来回顾一下离散情况下的 value iteration 算法的核心步骤：

重复以下步骤直至收敛 **{**

对每个状态 s，更新 $V(s) := R(s) + \max_{a \in A} \gamma \sum_{s'} P_{sa}(s') V(s')$

}

在状态连续的情况下，以上步骤将会改写为：

对每个状态 s，更新 $V(s) := R(s) + \gamma \max_a \int_{s'} P_{sa}(s') V(s') ds'$。可以看到，连续状态下的更新步骤的第二项已经变成了积分项，而不是离散情况下的求和项。”fitted value iteration”算法的核心思想是通过对连续状态进行采样（sample）得到一系列状态值 $s^{(1)}, \cdots, s^{(m)}$，进而估算积分项的值。特别的，我们将会使用一种监督学习算法来近似值函数：

如在线性回归算法中，我们将值函数看成是状态 s 的线性或非线性函数：$V(s) = \theta^T \phi(s)$。请联想一下第二节课讲到的房价问题，我们可以使用一维特征（面积 x）来估计价格 y，也

可以用三维特征$\begin{bmatrix} x \\ x^2 \\ x^3 \end{bmatrix}$来估计价格$y$。同样的，在近似值函数$V(s)$时，我们选择状态$s$的相关组合构成特征$\phi(s)$（可一维，亦可多维）；如在倒立摆的问题上，可以令$\phi(s)=\begin{bmatrix} 1 \\ x \\ x^2 \\ x\cdot\theta \\ \theta^2 \end{bmatrix}$。

值得注意的是，$\phi(s)$的选择很有技巧，选的不好会导致值函数的估计不够准确。

”fitted value iteration”算法的具体步骤如下所示：

这里的 m 个采样状态相当于 m 个训练样本

1. Randomly sample m states $s^{(1)}, s^{(2)}, \ldots s^{(m)} \in S$.
2. Initialize $\theta := 0$.
3. Repeat {

给定状态 s（i）,使用 MDP 模型估计不同动作 a 带来的可能的下一个状态，**使用离散采样的均值来估计连续**

这里的 y(i)相当于第 i 个训练样本的标签

```
    For i = 1, ..., m {
        For each action a ∈ A {
            Sample s'_1, ..., s'_k ~ P_{s(i)a} (using a model of the MDP).
            Set q(a) = (1/k) Σ_{j=1}^{k} R(s(i)) + γ V(s'_j)
                // Hence, q(a) is an estimate of R(s(i)) + γ E_{s'~P_{s(i)a}}[V(s')].
        }
        Set y(i) = max_a q(a).
            // Hence, y(i) is an estimate of R(s(i)) + γ max_a E_{s'~P_{s(i)a}}[V(s')].
    }
    // In the original value iteration algorithm (over discrete states)
    // we updated the value function according to V(s(i)) := y(i).
    // In this algorithm, we want V(s(i)) ≈ y(i), which we'll achieve
    // using supervised learning (linear regression).
    Set θ := arg min_θ (1/2) Σ_{i=1}^{m} (θ^T φ(s(i)) − y(i))^2
}
```

上面的做法和监督学习中训练过程很像，不是么？在该算法中，涉及到奖励函数 R（s），实际上奖励函数可以由设计者来设置，在倒立摆例子中，我们可以设置奖励函数如下：

$$R(s)=\begin{cases} -1, \text{如果}|\theta|>30 \\ 0, \text{其他情况} \end{cases}$$

当然，我认为这里的关键点倒不是算法的过程，而应该是：**1.** $V(s)=\theta^T\phi(s)$是唯一选择？ **2.** 采取离散采样的均值来估计连续，这样的做法是否能够使得估计出来的值函数收敛？

就疑问 1，我们很清楚其他的回归算法（局部加权线性回归算法等）都是可行的。就疑问 2，http://www.cs.utah.edu/~piyush/teaching/continuous-mdp.pdf 中明确指出，使用"fitted value iteration"算法得到的结果并不能保证总是收敛的。实际上，对于大多数问题该算法是收敛的，并且可以得到挺好效果。

好的，进入我们的最终环节：有了前面的基础（估算得到了 MDP 模型$\{P_{sa}\}$、V^*），我们如何求取连续状态下的最佳策略π^*？毕竟最佳策略π^*才是我们的目标啊。

在离散的情况下，我们知道：

$$\pi^*(s) = \arg\max_{a \in A} E_{s' \sim P_{sa}}[V^*(s')]$$

由于状态 S 是连续的，那么不可能在每一个状态时都采取相应的行动 a，因为状态 S 连续且无穷多。我们需要做的是，只在某些特定的状态时，才采取行动（如危险状态下）；这些都是依赖于设计者自身的设计。具体的实现过程与"fitted value iteration"算法的内循环类似，实现过程虽然繁琐，但是计算结果可靠。

下面的几种情况可以视为特殊情况：

（1） 当 MDP 模型为确定性模型时，由于均值就是本身，可得：

$s' = f(s,a)$

$$\pi^*(s) = \arg\max_{a \in A} E_{s' \sim P_{sa}}[V^*(s')] = \arg\max_{a \in A} V^*(f(s,a))$$

（2） 当 MDP 模型为随机模型，但不确定量服从均值为零的高斯分布时，可以看成是确定性模型来近似表示。