# 期末提纲

对比期末复习建议制作



Supervised Learning

• To perform the desired output given the data and labels p(y|x)

Unsupervised Learning

• To analyze and make use of the underlying data patterns/structures p(x)

Reinforcement Learning

• To learn a policy of taking actions in a dynamic environment and acquire rewards $\pi(a|x)$

## SVM

重点是算法原理和原始优化问题的推导过程。

**算法原理**

支持向量机（support vector machines, SVM）是一种二分类模型，它的基本模型是定义在特征空间上的**间隔最大的线性分类器**，间隔最大使它有别于感知机。

> **对比感知机：**
>
> 感知机的目标: 找到一个超平面使其能正确地将每个样本正确分类。感知机使用误分类最小的方法求得超平面，不过此时解有无穷多个(例如图1.1的H2和H3以及它俩的任

意线性组合)。而线性可分支持向量机利用间隔最大化求最优分离超平面,这时解是唯一的。

对于支持向量机来说，数据点若是p维向量，我们用p-1维的超平面来分开这些点。

# NLP

## 深度学习VS传统学习

1. 数据依赖

   深度学习和传统机器学习最重要的区别在于，随着数据规模的增加，深度学习的性能会有所提高。当数据很小的时候，深度学习算法的表现并不好。这是因为深度学习算法需要大量的数据才能完全理解它。另一方面，传统的机器学习算法及其手工制定的规则在这种情况下很受欢迎。

2. 硬件依赖性
   深度学习算法严重依赖高端机器，而传统的机器学习算法可以在低端机器上工作。这是因为深度学习算法的要求包括了gpu, gpu是其工作的重要组成部分。深度学习算法本质上要做大量的矩阵乘法运算。使用GPU可以有效地优化这些操作，因为GPU就是为此而构建的。

3. **特征处理**

特征处理是将领域知识放入特征提取器里面来减少数据的复杂度并生成使学习算法工作的更好的模式的过程。特征处理过程很耗时而且需要专业知识。深度学习尝试从数据中直接获取高等级的特征，这是深度学习与传统机器学习算法的主要的不同。基于此，深度学习削减了对每一个问题设计特征提取器的工作。

4. 问题解决

   应用传统机器学习算法解决问题的时候，传统机器学习通常会将问题分解为多个子问题并逐个子问题解决最后结合所有子问题的结果获得最终结果。相反，深度学习提倡直接的端到端的解决问题。

5. 训练时间和测试时间

   通常，深度学习算法需要很长时间来训练。这是因为深度学习算法中有太多的参数，训练这些参数需要比平时更长的时间。最先进的深度学习算法ResNet需要大约两周

的时间从头开始训练。而机器学习的训练时间相对要少得多，从几秒钟到几个小时不等。

然而在测试时间上深度学习一般都比机器学习快，

6. 可解释性

   深度学习的可解释性不如传统机器学习。假设我们使用深度学习来自动给论文打分。它在评分方面的表现相当出色，接近人类的表现。但有一个问题。它没有透露为什么会给这个分数。实际上，你可以从数学上找出深层神经网络的哪个节点被激活了，但我们不知道这些神经元应该建模什么，以及这些神经元层集体在做什么。所以我们无法解释结果。

# CNN

**CNN 和DNN的区别**

1. 部分连接，参数大大减少
2. 在图像处理中，相邻像素的关联性大，CNN可以不用看整张图就可以分类。DNN必须扫完全部的图。

## Batch Normalization

对多个变量进行归一化，不能使得某个变量特别大。这样可以加快收敛。BN放在hidden layer之间。

**Advantages of BN**

1. It allows thee model to converge faster and speed up training.
2. BN lets you use highter learning rates.
3. BN also reduces the dependence of gradients on the initial weight values.

**BN不适用的场景：**

BN doesn’t work well with smaller batch sizes.

BN is not used with recurrent network.(RNN不可以用BN！序列化的东西怎么可以)

**CNN上的反向传播：**

在全连接层和MLP反向传播一样。

区别1：在pooling层要双采样：maxpoling 只要管最大值，average pooling都要算。

区别2：计算梯度的时候要旋转180度， 因为和卷积的定义有关，本来就是对矩阵做反转变换。

**CNN的缺点：**

1. 很难对抗攻击，容易收到噪声干扰
2. 对空间的联系性差

- 问题定义。该问题是序列建模问题，假设我们要做的是用户的next-item prediction即预测下一个可能去的店铺。
- 数据处理。训练集、验证集和测试集的划分。在这个问题中，假设我们将所有用户序列中的最后一个店铺作为测试集，序列中倒数第二个店铺作为验证集，其余的作为训练集。
- 特征嵌入。不同类型的特征如何嵌入。比如numerical feature 、categorical feature、图片、文本等各类特征。
- 模型选择。考虑需要使用的预测模型，比如RNN、Transformer等适合序列建模的模型。对模型的结构进行必要的说明和解释。可以包含具体的模块设计，重要参数的设置等。
- 模型训练。选用什么样的Loss和优化目标，比如这里我们会选用最小化cross entropy loss；选用什么样的优化器和优化方法；是否做模型检验等。
- 模型评估。比如在这里我们选取recall和precision作为评价指标，对top 5的预测输出结果在测试集上进行评估，希望取得尽可能高的recall和precision。

# 经典案例

1. 使用CNN进行图像分类

**问题定义：** 输入各种带有标签的图像，使得训练出来的模型可以预测没有见过的图像的类别。

**数据处理：**

数据样本：样本都来自日常穿着的衣裤鞋包，每个都是28×28的**灰度**图像，其中总共有10类标签，每张图像都有各自的标签。可以对图像进行处理，比如旋转，亮度调整等增加样本的数量。

划分为训练集，验证集，测试集，（具体多少划分XX）

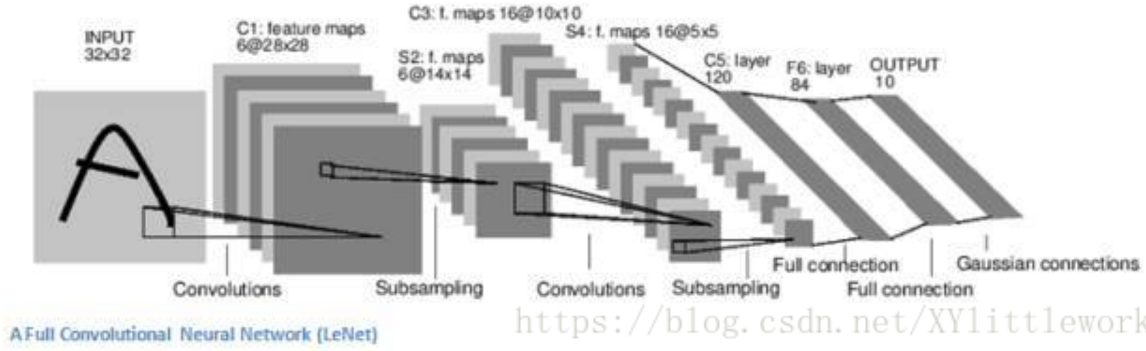
**特诊嵌入：**

图片就是变成N*N*3的vector

**模型选择：**

典型的CNN模型：

输入-卷积-ReLU-卷积-ReLU-池化-ReLU-卷积-ReLU-池化-全连接




A Full Convolutional Neural Network (LeNet)



**卷积层：**

定义卷积核大小，padding， 步长。进行卷积运算。

***非线性激活层：***

卷积层对原图运算多个卷积产生一组线性激活响应，而非线性激活层是对之前的结果进行一个非线性的激活响应。

在神经网络中用到最多的非线性激活函数是Relu函数，它的公式定义如下：f(x)=max(0,x)，为什么要这么做呢？上面说到，卷积后产生的特征图中的值，越靠近1表示与该特征越关联，越靠近-1表示越不关联，而我们进行特征提取时，为了使得数据更少，操作更方便，就直接舍弃掉那些不相关联的数据。

**pooling池化层：**

Max Pooling 最大池化、Average Pooling平均池化。顾名思义，最大池化就是取最大值，平均池化就是取平均值。

因为最大池化保留了每一个小块内的最大值，所以它相当于保留了这一块最佳匹配结果（因为值越接近1表示匹配越好）。这也就意味着它不会具体关注窗口内到底是哪一个地方匹配了，而只关注是不是有某个地方匹配上了。这也就能够看出，CNN能够发现图像中是否具有某种特征，而不用在意到底在哪里具有这种特征。这也就能够帮助解决之前提到的计算机逐一像素匹配的死板做法。但是在做图像压缩的时候，也有可能使得图像失真。

**全连接层**：这一层处理输入内容（该输入可能是卷积层、ReLU 层或是池化层的输出）后会输出一个 N 维向量，N 是该程序必须选择的分类数量。例如，如果你想得到一个数字分类程序，如果有 10 个数字，N 就等于 10。这个 N 维向量中的每一数字都代表某一特定类别的概率。例如，如果某一数字分类程序的结果矢量是 [0 .1 .1 .75 0 0 0 0 0 .05]，则代表该图片有 10% 的概率是 1、10% 的概率是 2、75% 的概率是 3、还有 5% 的概率是 9（注：还有其他表现输出的方式，这里只展示了 softmax 的方法）。完全连接层观察上一层的输出（其表示了更高级特征的激活映射）并确定这些特征与哪一分类最为吻合。

**模型训练：**

1. 使用反向传播来进行训练，为了防止过拟合，采用drop out或者batch normalization的策略。
2. Dropout 层将「丢弃（drop out）」该层中一个随机的激活参数集，即在前向通过（forward pass）中将这些激活参数集设置为 0。Dropout 层只能在训练中使用，而不能用于测试过程。
3. 使用MSE（最小均方误差）为损失函数。

$$E_{total} = \sum \frac{1}{2}(target - output)^2$$



前向传导、损失函数、后向传导、以及参数更新被称为一个学习周期。对每一训练图片，程序将重复固定数目的周期过程。一旦完成了最后训练样本上的参数更新，网络有望得到足够好的训练，以便层级中的权重得到正确调整。

**模型测试：**

我们准备不同的另一组图片与标记集（不能在训练和测试中使用相同的）并让它们通过这个 CNN。我们将输出与实际情况（ground truth ）相比较，看看网络是否有效并，输出正确率，recall rate。

如果使用dropout, 在权重还要乘（1-dropout rate）,这个dropout rate是超参数， 要在训练的时候人为调整。

| | | 预测类别 | | |
|---|---|---|---|---|
| | | Yes | No | 总计 |
| 实际类别 | Yes | TP | FN | P（实际为Yes） |
| | No | FP | TN | N（实际为No） |
| | 总计 | P'（被分为Yes） | N'（被分为No） | P+N |

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

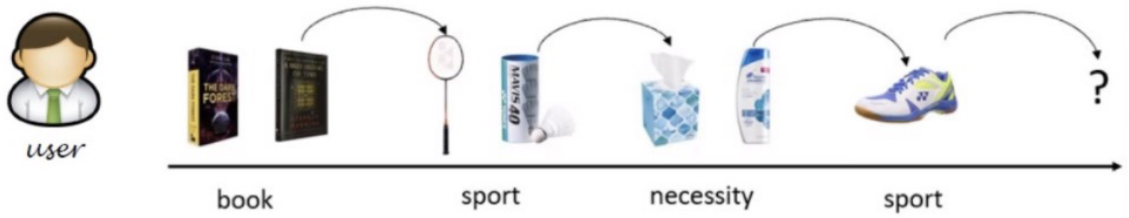
recall=TP/(TP+FN)=TP/P=sensitive

# 推荐系统

# 讨论并设计预测模型

- 假设我们拿到了10000个不同用户的Yelp点评数据，经过处理后发现，这些点评记录的时间跨度为2年，每个用户都有50条店铺消费记录，依次按照时间顺序排列，每笔消费都写了评论并上传了图片。

- 问题：预测用户之后会去哪家店铺消费并为每个用户给出推荐店铺。




定义问题：输入50个data, 预测nextk

数据处理：**数据集的划分**

拿到数据有50个， 可以前49做训练，最后一个做测试，next k,就一个一个向下到k个。如果要切分验证集就48+1+1；

**特征嵌入：**

特征需要筛选：比如需要在同一个城市等，要说出哪些数据是什么信息，用什么方法来做特诊嵌入。

数值型（归一化）

类别型（onehot编码之后embedding：case a: 001, caseB:010,然后乘以一个dense的矩阵，得到一个dense vector,需要这么做的原因是，如果类别很多10000，那只要10000的vector变成一个64*64的矩阵）

(embedding 就是压缩vector把高维的数据降低纬度到低维的表示)

文字：

简单的处理方式：word2Vec, 把所有的文字的每个单词都用CBOW或者skip-gram或者生成一个64维vector, 然后这些vector做平均，就表示这一段话的vector.

或者使用预训练的模型eg. BERT, gpt得到64维的向量。

图片：输入CNN，若干层之后有feature的输出层.得到向量。（可以适当描述一下CNN）

最后所有特征都要嵌入到一个item里面，无论多少所有的特征都排好，只要过一遍MLP，总是能够达到64维（这个人为规定）。（编码常见的做法）

**预测模型:**

是一个序列建模问题，可以使用RNN， transformer, encoder-decoder, LSTM等。 用transformer只用decoder或者encoder就可以。我们把前面所有的数据打包作为一个block, 然后一个一个按照时间顺序输入，这样就得到了一个序列。我们要在encoder里面加position embedding 位置变量，【简要描述transformer的几层，这里只要encoder】， 再过一个MLP 得到vector之后用softmax算概率，挑出最大概率的即可。如果是next k, 可以再把预测出的数据作为新的block 送给原来结构的模型再算一次（可以使用decoder得到next k吗？）

**模型训练：**

使用cross-entropy 梯度下降

**评估：**

accuracy, recall rate, hit rate

# 强化学习

**问题定义：**

输入：the observation of machine represented as a vector or a matrix.用向量或矩阵表示的模型对于环境的观察 ，这里是图像

输出：each action corresponds to a neeuron in output layer.

每个动作对应输出层的一个神经元。

Action=$\pi(observation)$

寻找一个最优的策略或者值函数。要学内容给定一个observation ，输出action ，学习的内容就是pi函数，其实就是学习这个神经网络，使得累积reward最高。

**数据处理：**

整个画面就是observation. 初始画面是初始state.

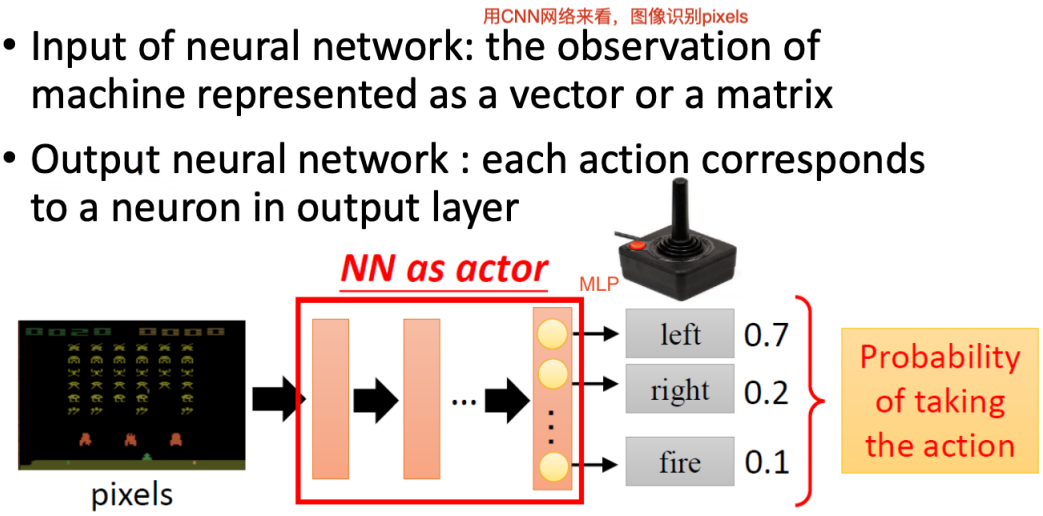
确定我的action：出招表

确定reward: 整个episode的总分（累计的而不是每一步），因为有局部战略和全局战略最优。

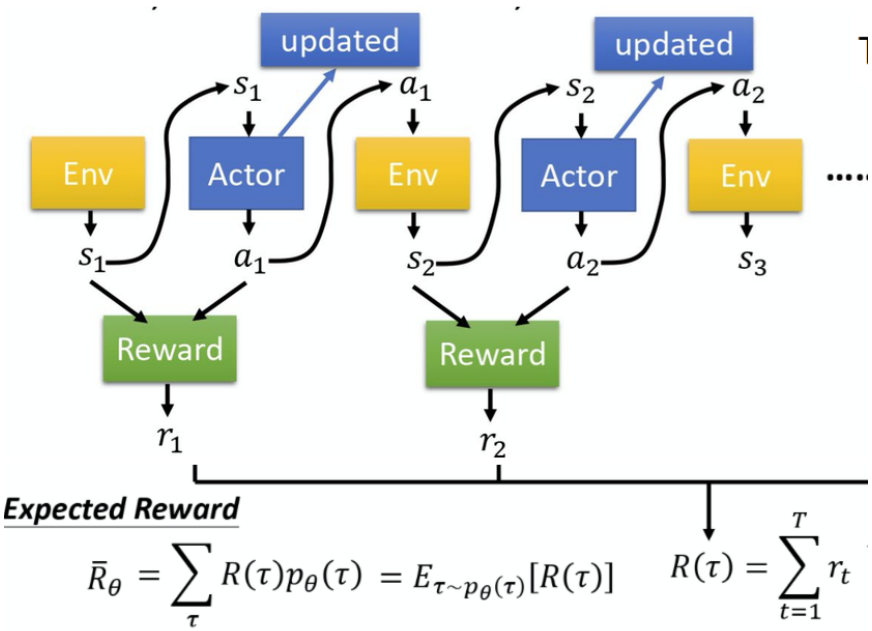
**特征嵌入：**

获取当前state的图像。

使用CNN神经网络得到一个vector, 再把这个vector丢给MLP后softmax，得到出招的概率分布.

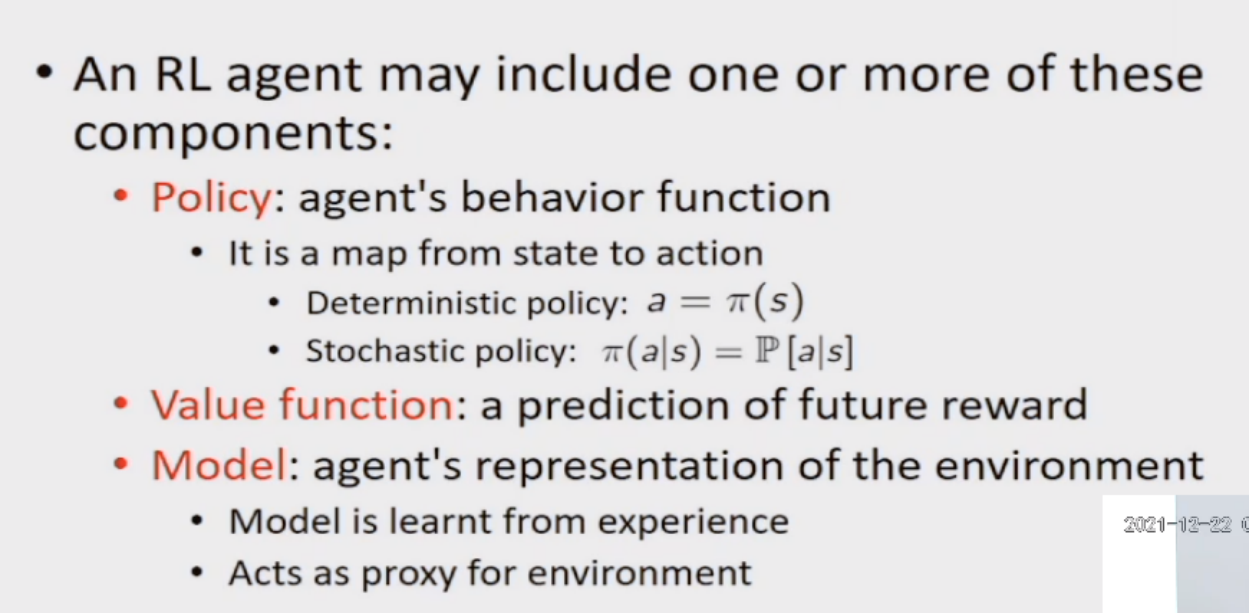



之后根据这个概率分布进行一个episode的游戏，在此期间我们获取的是一个序列，即<s1,a1,r1><s2,a2,r2>,... <sn,an,rn>, 对这个序列我们先对每个数值进行归一化之后输入到transformer的encoder里面，输出一个vector, 这个就是模型的参数$\theta$




模型选择：

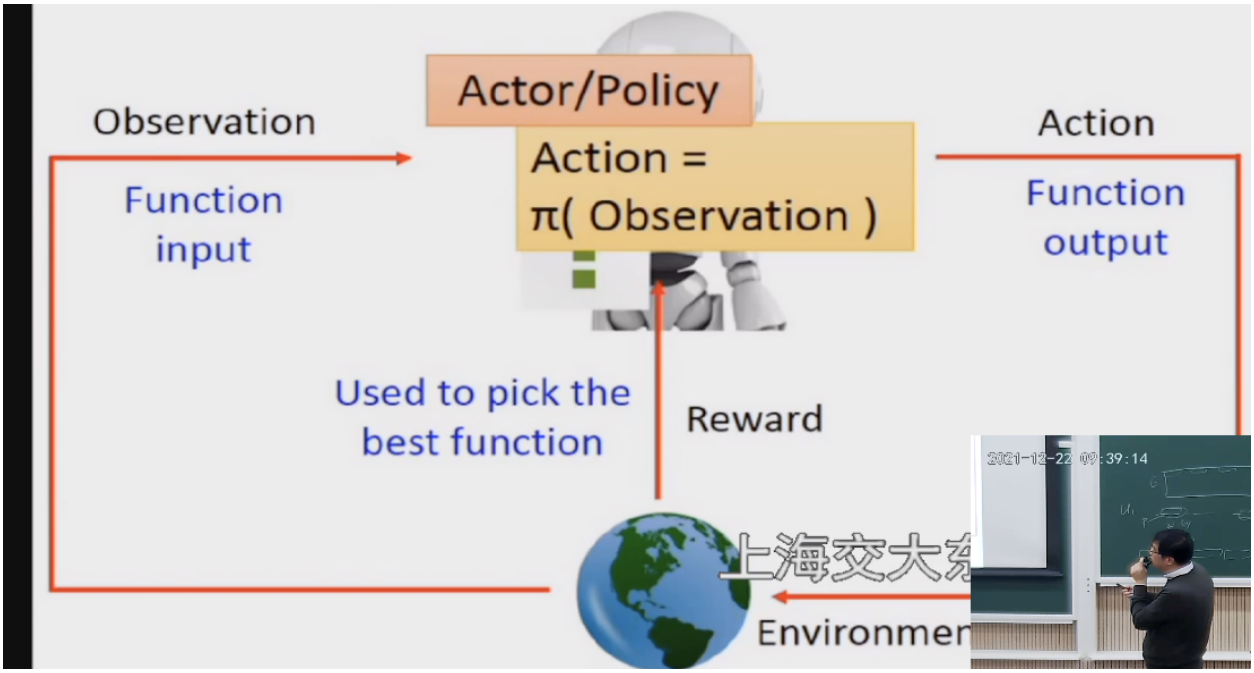
Model-free appraoch和model-based approach.前者 有policy-based 和value based,这里选用前者的policy-based 。




我们学习的是随机的policy. 模型流程如下：

- RL is a general-purpose framework for decision-making
  - RL is for an agent with the capacity to act
  - Each action influences the agent's future state
  - Success is measured by a scalar reward signal
  - Goal: select actions to maximize future reward

**模型训练:**




model 观察环境得到observation, 然后根据之前的pi 函数得到action, 这个action对环境产生新的影响，于是又得到新的reward， model就根据这个reward调整pi函数，生成新的action.

和传统的监督学习是强化学习是一个连续的过程。

我们需要累积的reward, 把所有的reward加起来，让total reward最大。但是由于游戏有随机性，用期望来表示total reward, 但是由于不可能遍历所有的可能性，我们就采样N个来完成。

- An episode is considered as a trajectory $\tau$
  - $\tau = \{s_1, a_1, r_1, s_2, a_2, r_2, \cdots, s_T, a_T, r_T\}$
  - $R(\tau) = \sum_{t=1}^{T} r_t$

采样，不可能得到所有的t

$$\bar{R}_\theta = \sum_\tau R(\tau) P(\tau|\theta) \approx \frac{1}{N} \sum_{n=1}^{N} R(\tau^n)$$

Sum over all possible trajectory

我们的目标就是找到最大的R(\theta bar)对应的theta, 我们通过**策略梯度**来更新我们的模型参数$\theta$。

- Problem statement

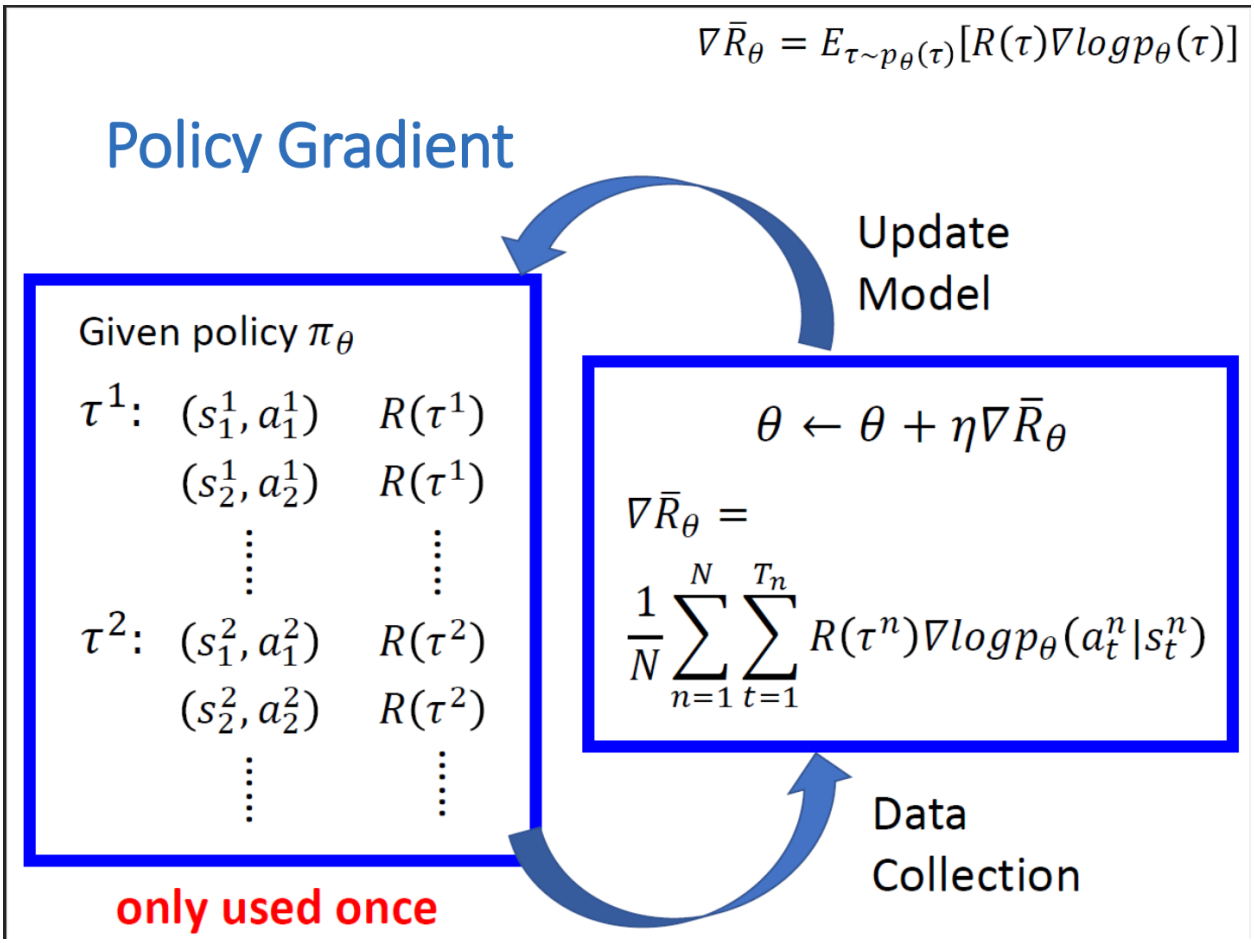
$$\theta^* = arg \max_\theta \bar{R}_\theta \qquad \bar{R}_\theta = \sum_\tau R(\tau) P(\tau|\theta)$$

游戏给分，黑箱

- Gradient ascent 梯度上升
  - Start with $\theta^0$
  - $\theta^1 \leftarrow \theta^0 + \eta \nabla \bar{R}_{\theta^0}$
  - $\theta^2 \leftarrow \theta^1 + \eta \nabla \bar{R}_{\theta^1}$
  - .....

$$\theta = \{w_1, w_2, \cdots, b_1, \cdots\}$$

$$\nabla \bar{R}_\theta = \begin{bmatrix} \partial \bar{R}_\theta / \partial w_1 \\ \partial \bar{R}_\theta / \partial w_2 \\ \vdots \\ \partial \bar{R}_\theta / \partial b_1 \\ \vdots \end{bmatrix}$$




值得注意的是：

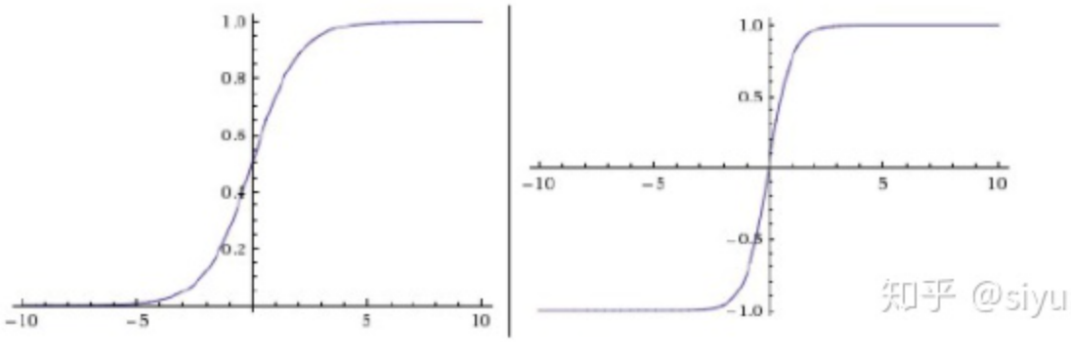
当模型的参数被更新了之后，需要全部重新采样。然后输入到策略梯度里面更新参数。

模型评估：

直接用reward大小就可以。

## 常见激活函数

**饱和激活函数(sigmoid、tanh)：**




用Seq2Seq做时间序列预测

1、 keras-seq-2-seq-signal-prediction 2、 seq2seq-signal-prediction 3、 ts_seq2seq_intro/ 4、 https://github.com/Arturus/kaggle-web-traffic 5、 Sequence to

https://www.dazhuanlan.com/alexmajy/topics/1658633