## 一、算法原理

### 策略梯度

给定一组状态-动作序列

则有表示序列τ的回报。表示序列τ出现的概率，则策略梯度的目标函数可以表示为：

策略梯度的目标就是找到最优参数θ，使得J(θ)最大。因此策略梯度是一个优化问题，最简单的就是用梯度上升法来求解：



现在我们来对目标函数求导：











因此最终的策略梯度就变成求的期望了，这样当采样m条样本序列时，就可以利用m条序列的均值逼近策略梯度的期望：



从上面的式子可以看出第一项是轨迹τ 的概率随参数θ变化最陡的方向。第二项R(τ)控制了参数更新的方向和步长。

对于每一条序列τ的概率P(τ;θ)都可以表示成下面的形式：



对上面的式子代入到梯度的式子中，通过对数展开可以得到：



但是上面的式子有一个问题，就是后面的是对整条序列的回报，但是在t时刻的策略对应的回报不应该和t时刻之前的状态-动作价值无关。因此对这个式子进行修改可以得到：



在上面的式子中是不会改变的，这个一般称为分值函数，但后面的是可能发生改变的。

### REINFORCE算法

在蒙特卡洛策略梯度reinforce算法，我们用价值函数v(s) 近似的替代上面的目标函数中的Q(s,a)。则整个流程如下：

假设迭代轮数为EPISODES，采样的序列最大长度为L，学习速率为α，状态集为S，动作集为A。

1）for episode in range(EPISODES):   # 开始迭代

2）初始化状态s，在这里s为状态向量

3）for step in range(T):  # 序列采样

　　　a) 将状态向量s输入到策略函数中，我们可以得到softmax之后每个动作的概率，根据概率去选择动作（增加了随机性，而不是每次选择概率最大的动作）；

　　　b) 在状态s下执行当前动作a，获得下一状态s′，当前奖励R，是否终止状态is\_end；

　　　c) 将当前的状态s，动作a，奖励R分别存储在一个列表中；

　　　d) 更新状态，s=s′；

　　　e) 判断是否是最终状态，如果是则将这条采样得到的序列用来更新策略函数中的参数；否则继续循环采样。

### 优势函数

1. 一般意义上的优势函数

优势函数表达在状态s下，某动作a相对于平均而言的优势；从数量关系来看，就是随机变量相对均值的偏差。使用优势函数是深度强化学习极其重要的一种策略，尤其对于基于policy的学习。

优势函数将Q-Value“归一化”到Value baseline上，这样有助于提高学习效率，同时使学习更加稳定；同时经验表明，优势函数也有助于减小方差，而方差过大是导致过拟合的重要因素。

1. REINFORCE with baseline

baseline 指的是一个函数 b ，它不依赖于动作 A ，可以得到公式：



有Baseline的REINFORCE其实就是优势函数在REINFORCE中的实现：使用状态价值V 函数作为Baseline，也就是当前状态下Reward的总期望，优势函数 A = Q - V，使用A 替换掉Loss 中的价值函数（Reward）：



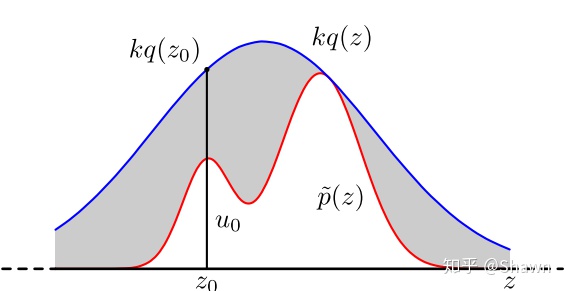


### 重要性采样

1. **蒙特卡罗采样**

蒙特卡洛方法是一种近似推断的方法，通过采样大量粒子的方法来求解期望、均值、面积、积分等问题，蒙特卡洛对某一种分布的采样方法有直接采样、接受拒绝采样与重要性采样三种，直接采样最简单，但是需要已知累积分布的形式。接受拒绝采样与重要性采样适用于原分布未知的情况，这两种方法都是给出一个提议分布，不同的是接受拒绝采样对不满足原分布的粒子予以拒绝，而重要性采样则是给予每个粒子不同的权重。

1. 直接采样: 直接采样的方法是根据概率分布进行采样。对一个已知概率密度函数与累积概率密度函数的概率分布，可以直接从累积分布函数（cdf）进行采样，在其值域[0, 1]上均匀采样，然后通过cdf的反函数获取x
2. 接受-拒绝采样: p(z)是希望采样的分布，q(z)是提议的分布(proposal distribution)，令kq(z)>p(z)，首先在kq(z)中按照直接采样的方法采样粒子，接下来判断这个粒子落在途中什么区域，对于落在灰色区域的粒子予以拒绝，落在红线下的粒子接受，最终得到符合p(z)的N个粒子



1. 重要性采样(important sampling): 给予每个粒子不同的权重，使用加权平均的方法来计算期望
2. **什么时候会用到重要性采样**

强化学习算法一般分为两种：

1. On-policy: 学习到的agent以及和环境进行互动的agent是同一个agent
2. Off-policy: 学习到的agent以及和环境进行互动的agent是不同的agent

强化学习算法有两个策略：一个用于决策的 behavior policy μ，一个用于更新的 target policy π，当μ和π一样时就是On-Policy，不一样时就是Off-Policy算法。

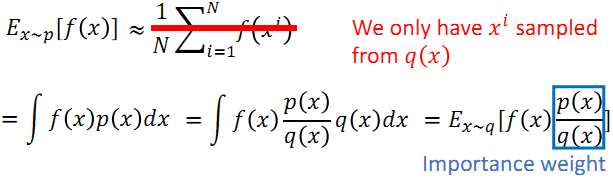
在贝尔曼方程中需要计算的下一个状态的Q函数是基于概率分布π的，而Off-Policy使用μ进行决策(为了进行探索)，两种策略不一致时才需要重要性采样。

如果我们使用 πθ 来收集数据，那么参数 θ 被更新后，我们需要重新对训练数据进行采样，这样会造成巨大的时间消耗。

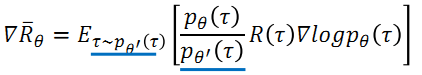
目标：利用 πθ′ 来进行采样，将采集的样本拿来训练 θ， θ′ 是固定的，采集的样本可以被重复使用。

1. **重要性采样原理**

当我们只有通过另外一个分布得到的样本时，期望值可以做出以下更改，更换分布之后，需要使用重要性权重 p(x)/q(x) 来修正 f(x) ，这样就实现了使用q分布来计算p分布期望值。



从上述思想出发，可以达到上文所述的目标，即“利用 πθ′ 来进行采样，将采集的样本拿来训练θ，θ′是固定的，采集的样本可以被重复使用”表示为：



### 交叉熵

用来衡量在给定的真实分布下，使用非真实分布所指定的策略消除系统的不确定性所需要付出的努力的大小

equation

其中，p 表示真实分布(目标策略), q 表示非真实分布(采取的策略)。在机器学习中的分类算法中，我们总是最小化交叉熵，因为交叉熵越低，就证明由算法所产生的策略最接近最优策略，也间接证明我们算法所算出的非真实分布越接近真实分布。

在代码中，相关部分是这样实现的:

entropy = - (probs\*probs.log()).sum()

这个公式是信息熵的计算方法。

**信息熵** - 信息熵代表的是随机变量或整个系统的不确定性，熵越大，随机变量或系统的不确定性就越大。

也就是说，在代码中加入了一项信息熵，而不是交叉熵，而且该信息熵在REINFORCE的原始公式中应该是没有对应项的；属于神经网络训练技巧。

### 神经网络归一化 - Normalization

深度学习中的数据分布偏移：深度神经网络涉及到很多层的叠加，而每一层的参数更新会导致上层的输入数据分布发生变化，通过层层叠加，高层的输入分布变化会非常剧烈。虽然神经网络的各层的输入信号分布不同，但最终“指向“的样本标记是不变的，即边缘概率不同而条件概率一致。

为了降低分布变化的影响，可使用归一化策略Normalization，把数据分布映射到一个确定的区间。神经网络中，常用的归一化策略有BN(Batch Normalization)， WN(Weight Normalization)， LN(Layer Normalization)， IN(Instance Normalization).

BN是针对单个神经元进行归一化操作，多用于batch size大的CNN网络。使用batch size数量的样本的均值和方差，近似整体样本的均值和方差，独立地规范每一个输入维度x。也有人说，BN输出的是标准正态分布（不明白，大量样本的分布是正态分布？）。BN使用的这样均值、方差近似，给神经网络引入了噪声，提高了泛化性。BN对batch size和样本的随机要求相对严苛。

WN是对神经网络的权重进行归一化，不直接依赖于输入样本。虽然是对权重w进行限定，但限定的结果还是反应到y=w＊x+b的y中。

LN是对当前神经网络层单个训练样本的所有神经元的输入进行计算，利用统一的均值和方差，对数据进行归一化。但是，对不同类别的特征使用同样的限定，会降低网络性能。

IN是对单个图像进行的归一化操作。常用于风格迁移等。

GN是对通道进行分组，每个组做归一化操作。GN可以看作是LN与IN之间的一种折中方案。把当前层的所有通道都作为一个组的时候，GN结果就是LN。如果把所有通道N分为N个组里，GN结果就是IN。

本次试验中使用了WN归一化操作，用于优化神经网络训练。

### 正则化 - L2范数

神经网络中损失函数后一般会加一个额外的正则项L1或L2,也成为L1范数和L2范数。正则项可以看做是损失函数的惩罚项，用来对损失函数中的系数做一些限制。

L1正则化是指权值向量w中各个元素的绝对值之和;

L2正则化是指权值向量w中各个元素的平方和然后再求平方根;

一般都会在正则化项之前添加一个系数，这个系数需要用户设定，系数越大，正则化作用越明显。

L1正则化可以产生稀疏权值矩阵，即产生一个稀疏模型，可以用于特征选择，一定程度上，L1也可以防止过拟合;

L2正则化可以防止模型过拟合（overfitting）;

本次实验使用了L2正则化作为对照项。

## 二、实验分析

### 实验数据

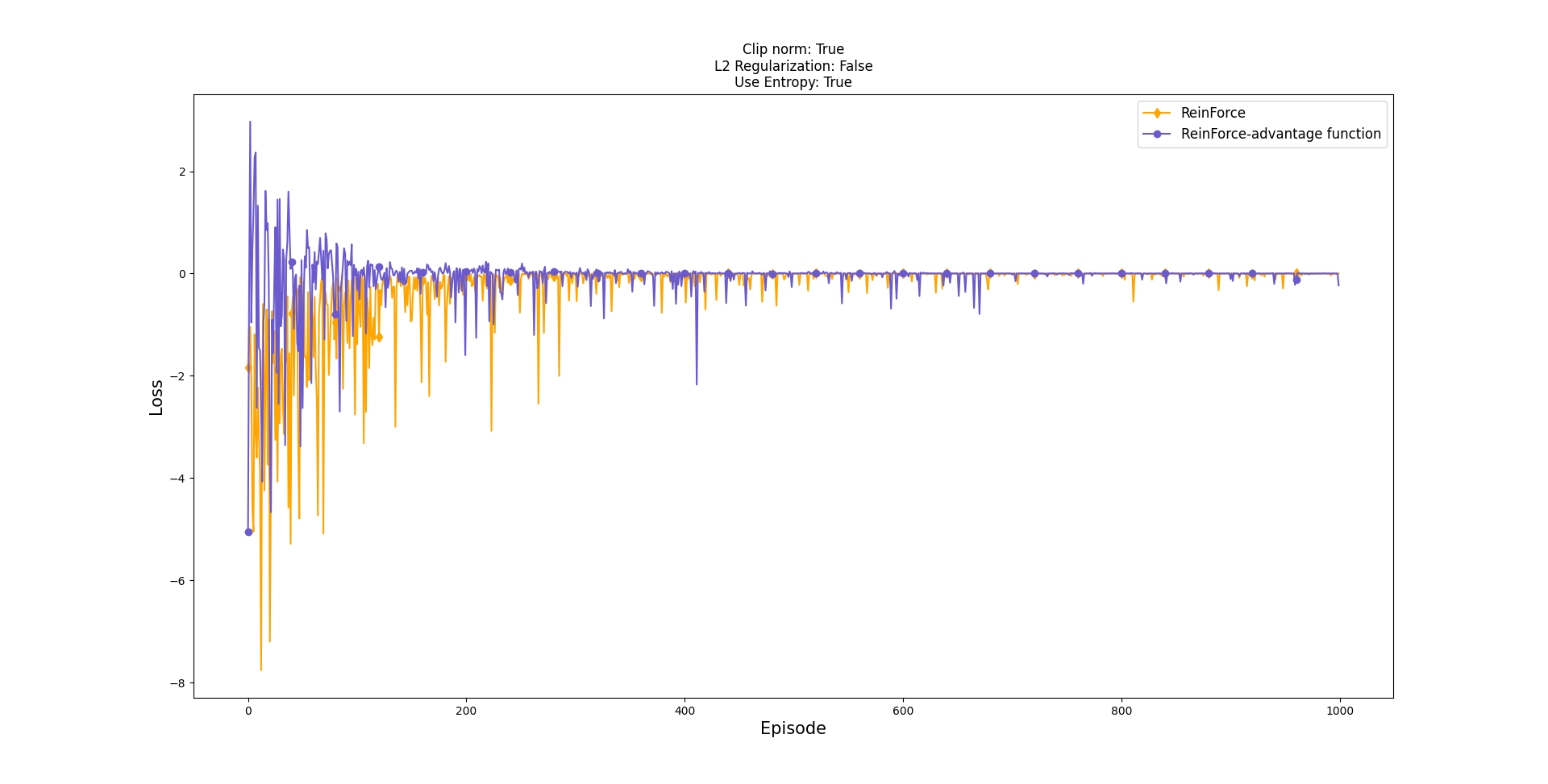
本次实验数据集为通过仿真得到的数据计算与传输中，采取不同的切片位置时，网络云/边缘云的时延与能耗数据，切片位置共有23个，时延数据共有4\*5=20组，能耗数据共有5组。自变量包括：是否使用GPU(0/1)、数据切片位置(0-22)；因变量包括：能耗、时延。

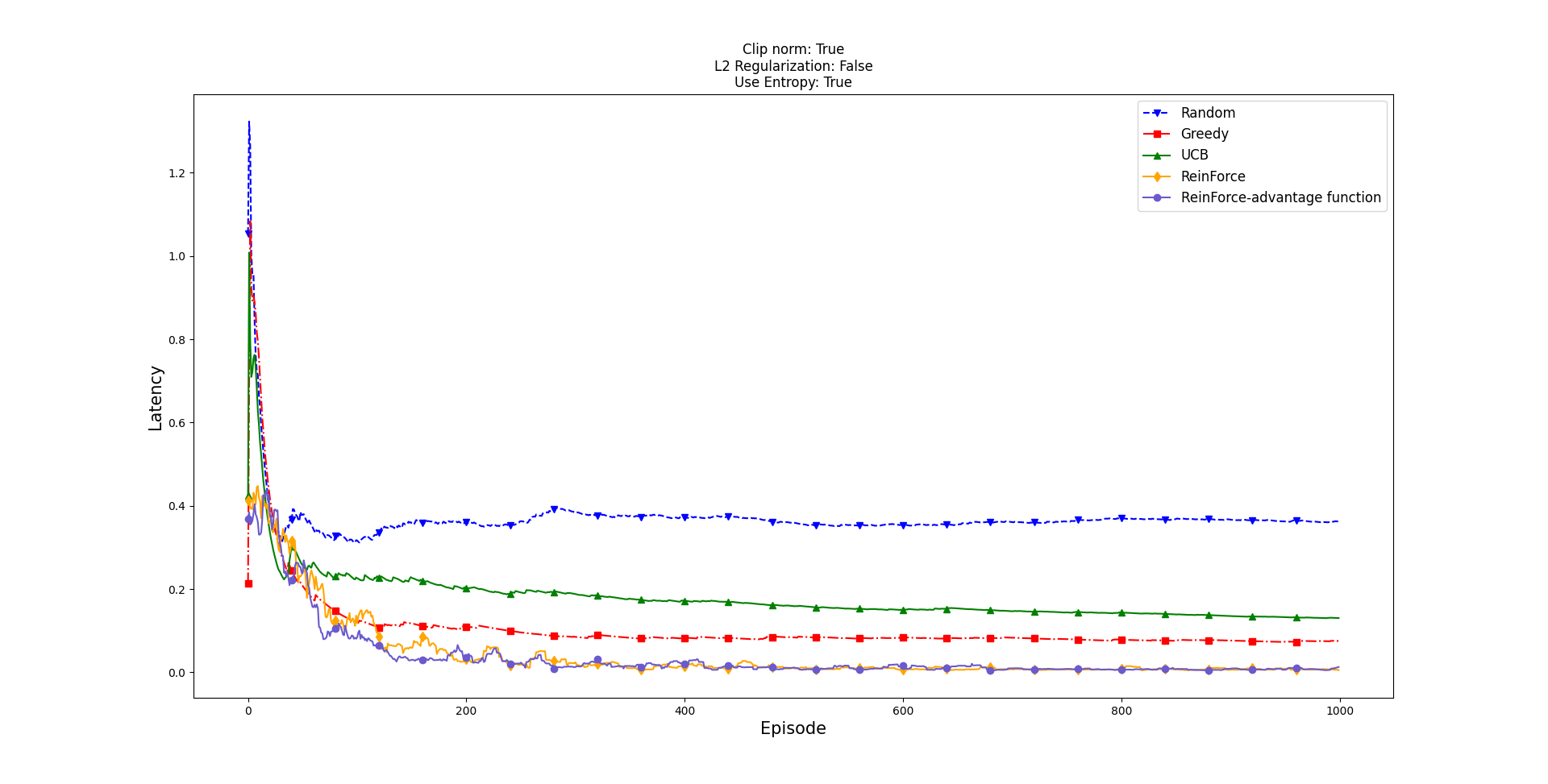
### 实验步骤

分别采用 ϵ - Greedy、UCB、Random、REINFORCE、REINFORCE with Baseline算法，使用数据集进行训练，目标是采取最优的切片位置，使标准化处理之后的时延与能耗数据之和最小。

其中，REINFORCE、REINFORCE with Baseline采用神经网络实现，并且采取了多个变量进行对照实验：是否在Loss中加入L2范数、是否在Loss中加入信息熵、是否在训练时采取Normalization

### 实验结果





经过分析对比，得到以下结论：

1. L2范数 负作用，影响正常收敛，向0收敛，但波动很大
2. Entropy信息熵 正作用，尤其对不加优势函数的Reinforce效果更好，加快收敛
3. Normalization 看不出明显影响
4. REINFORCE with Baseline 效果比REINFORCE 明显要好

因此最好的策略是：加入熵、不加L2范数，Normalization加不加都可，此时REINFORCE with Baseline算法结果为最优

其他影响因素：UCB等算法的数据取值具有随机性，能耗与时延的综合奖励公式为标准化后相加