关于排队系统结构和参数的实验和调研

（一）首先对‘判别器网络‘和从 ciw 模拟出的数据提取 feature 需要注意的事项做一个总结

在围绕郑教授上次提到的几点做实验的过程中一个之前存在的问题被放大了：有时候 wgan 网络得到的 loss 从开始就为 0，并且这时候 bp 到 model 参数的梯度也全为 0，之前这种情况偶尔出现，一般只需要重新进行训练基本不会再出现 loss=0。但这次的基本所有实验中都频繁出现 loss=0，且无法消除。此外得到的 loss 数值异常小，在 10^-5 量级，导致参数更新慢。

调试过后发现这是因为我对判别器网络结构做的一个修改导致的，并且也让我发现了一些之前不太明显但的确较重要的点，对于结构和参数分布复杂的排队系统它们对于训练结构起决定性作用。

首先给出结论：

- 判别器网络激活函数使用 leakyrelu，不使用 relu（relu 会使训练不稳定），网络结构为 k＊（conv1d+leakyrelu）+ linear （k<=3）
- 针对 ciw 模拟的结果，分别提取出各个节点的 arrival time sequence 和 exit sequence，然后依序把 k 个 node 的 sequence 连接成一个一维 tensor 用于训练

具体细节如下

1）判别器网络中每层输入的 feature 卷积后的结果紧接着通过一个 nn.ReLu 层，但是各个卷积层输出的结构中负数都占了很大一部分，所以 relu 相当于消除了卷积层输出的绝大部分信息，参考了 github 上其他 wgan 的项目，发现它们设计的判别器网络几乎都是用的 nn.LeakyRelu 作为激活函数，猜测在它们的项目应该中也存在卷积层输出的 feature map 很大比例为负的这个问题。

2）之前提到过对于从 ciw 模拟结果提取 feature 我的想法是对每个节点分别取出经过它的所以顾客的到达时间序列和离开时间序列，再把所有节点的数据整合。开始时我的想法是各个节点的数据向量通过相同的判别器网络但相互之间没有联系（代码中则体现为用 conv2d 对输入的二维 feature 处理，kernel size=（4*1），stride=（1，2），），由于这次实验我选的四节点和五节点网络比较复杂，发现训练效果很差，一个有意思的现象是对于其中一个 4 节点对称网络（node1 与 node2，关于 node3 和 node4 结构上是对称的，反之也是）某次训练得到的参数是 ground truth 的对称结果（即训练后 node1（2）的参数接近真实网络中 node2（1）的参数。之后我对从 ciw 模拟得到的数据提取训练的 feature tensor 的各种形式和判别器网络的结构做了对比实验，结果是：判别器最好的结构如下

```
self.main_module = nn.Sequential(
          nn.Conv1d
          nn.LeakyReLU(0.2, inplace=True),
          nn.Conv1d
         nn.LeakyReLU(0.2, inplace=True),
          nn.Linear
)
```

而输入判别器的 feature tensor 为：f = n1_f+n2_f+… 时训练的效果最好（各个 node 的数据在判别器的 linear 层进行整合）。

（二）对于系统输入的顾客服从复杂泊松过程的测试

problem formulation：

- 排队系统中有 k>=1 个 node 接受来自外界的顾客
- 顾客的 arrival rate 服从 peicewise constant NHPP 过程，具体有多少段未知，且这 k 个 node 各自的 NHPP 的参数不一定相同
- 如何学习这样的 arrival distribution

上次我提出的方法是假设 NHPP 的阶段数为 n（n 的值假设的不要太小，即不小于 k，也不要太大，保证每段内到达的顾客数目至少在 30 左右），然后分阶段训练：

假设总的模拟时间为 T，第 k 个阶段对应的时间段为[T*（k-1）/n，T * k / n)

1.)第 k 个阶段训练第 k 个阶段的 arrival distribution 的参数，之前训练好的 arrival distribution 参数固定（或者使用更新步长很小（比当前训练的 arrival parameter 对应的更新步长要小一到两个数量级）的优化器进行更新)。

2.）service distribution 一直参与训练

3.）第 k 个阶段的训练只使用前 k 个阶段的数据

Details and performance：

这种方法减少每次训练的难度，主要的点在于：

1.）当前面的参数训练好的前提下当前的参数理论上是可以训练好的，而且每个阶段训练至收敛需要的迭代数目是差不多的。

2.）对于复杂 NHPP 过程，如果一股脑对所有参数进行训练主要的难点在于 wgan 的 loss 的 global optimum 相对周围的 local optimum 并没有小太多，或者说 global optimum 周围的梯度不是很陡，所以从随机参数开始训练要越过一个一个 local optimum 到达 global optimum 很难（从训练中可以看到当判别器基本训练好了的时候它 bp 到 model 的梯度很小的，所以训练的迭代次数较大时会自动收敛或者在某个不太好的解的附近震荡，无论参数训练的好不好)。分阶段可以减少各个阶段局部极值的数目，使收敛到 ground truth 的时间和概率增加

3.）使用分阶段训练可行的一个必要条件是对于输入为泊松过程的情况，也就是第一阶段，我们可以很好的训练出接近 ground truth 的结果。

在实验中发现，最关键的是第一阶段，而这一阶段的 arrival distribution 可以很准确的收敛，service distribution 则需要更多的迭代才能收敛，于是可以选择早停，在之后的阶段的训练中 service distribution 的参数基本都是处于微调状态，最后可以较好的收敛到 ground truth。

一个加快训练的技巧是第 k 个阶段训练时，arrival distribution 的参数直接初始化为第 k-1 个阶段训练好的参数，这样可以大大减少训练需要的迭代数。

（三）服务时间服从 lognormal 和 gamma 分布时排队系统的训练

这两个分布都有两个参数，因此训练的难度都较大。

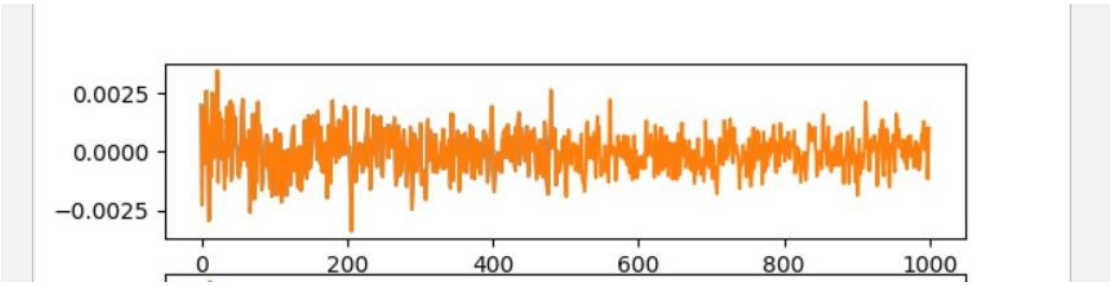
开始时对于复杂网络训练的效果不太好所以我直接用 one-node queueing system 来测试我们目前的方法对于它们的训练时的表现

首先训练时一个明显的地方在于训练时长显著增加，所需的迭代次数也显著增加，即这两种分布的采样的计算相比其他分布来说要花更多时间，另外一点是即使是最简单的 one-node queueing system，对这两种参数的训练也不是很轻松，具体来说，这两种分布更新时没有明确的规律可循，虽然最后都可以变化到 ground truth 附近，但是一直在小范围震荡，不会稳定收敛，不过观察可以发现，那些离 ground truth 近的点对应的 w-distance（d-loss）比其他点小，所以训练时只需要选择最后稳定在一个水平时最小 d-loss 对应的参数作为最终的结果即可。有趣的一点是这两种参数的学习都是开始阶段是微调，之后快速收敛到 ground truth，但 loss 的变化并不能指示这一过程。

这是 lognormal 的（由于在 950 次迭代左右才收敛，所以可能再训练 200 个阶段 loss 曲线会相对给出更明确的信息）

Ground truth：（0.8，0.4）

参数初始化为（0.2751, 0.9136）训练收敛结果（0.8 ± 0.04，0.4 ± 0.02）

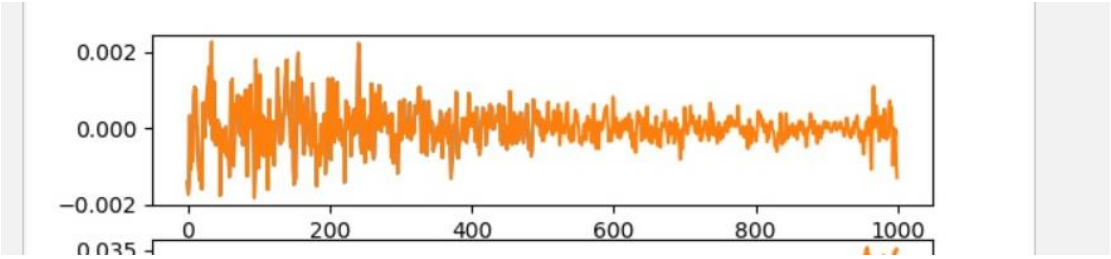


这是 gamma 的（相对来说 loss 曲线有些意义，在第 900 次迭代左右参数基本收敛到 ground truth，这一阶段的 loss 的震荡明显减小）

Ground truth：（0.8，0.4）

参数初始化为（0.6197, 1.0266），

第 900 次迭代过程中 loss 最小，对应的结果为（0.8 ± 0.01，0.4 ± 0.01），



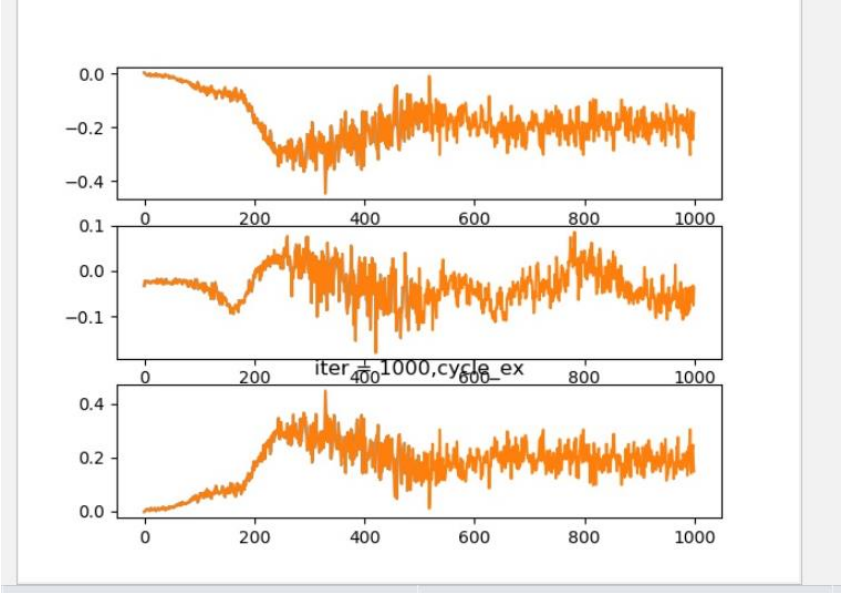
对于复杂网络，当其中某些 node 的服务时间服从 lognormal 或者 gamma 分布时，训练最后的结构的好坏与训练的阶段数密切相关，有时候可能可以很快训练好，有时候则需要几千个阶段才能得到较好的解。但是对于 gamma 分布的参数的学习相对来说效果不那么好，并且若系统中各个节点服从的是不同种类的分布，即同时有 exponential，lognormal，normal 时训练起来难度也很大，或者可能就训练不好。当 server 服从的分布相对一致，例如均为 exponential 或均为 lognormal 时则训练效果要好很多

（四）有反馈或者说存在环状结构的排队系统的测试

之前研究的排队系统都是流水线一般的，顾客从进入到出去不断经过新的 node 而没有返回之前到过的 node，但对于医院等实际情况，顾客可能会频繁往返于几个 node 直接，对于这种情况，我设计了两种种结构的四节点排队系统（环形（其中一个节点供进出），线形（后面的节点的顾客以一定概率返回前面的某些节点），），经过测试，都可以像其他系统一样良

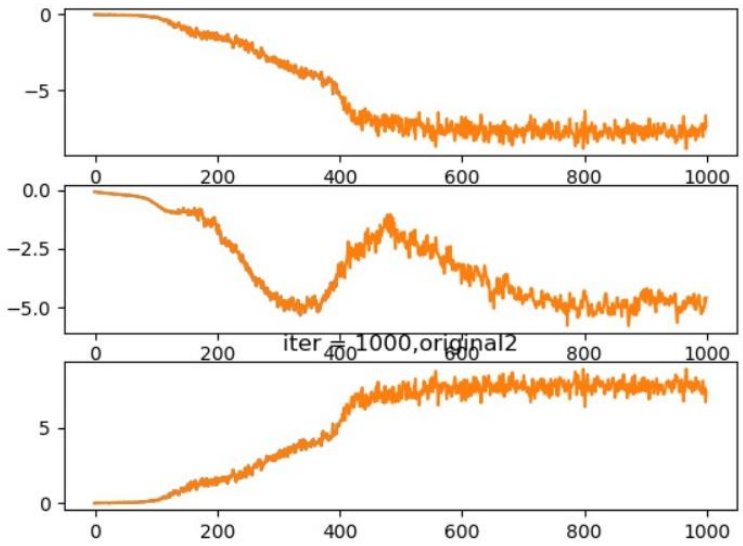
好收敛，输出的数据和真实数据相似。

对于以上所有情况进行的实验，在参数都随机初始化时，loss 的变化曲线基本都是如下趋势：



开始阶段对应判别器网络的学习，此时返回的梯度信息很多，model 更新快，loss 也快速变大（即判别器可以明确指出模拟数据和真实数据的分布差很多），之后 model 的学习进入微调阶段，当 loss 稳定在一个数量级且参数每次迭代的变化不超 0.001 倍时即可停止训练，然后选最小 loss 对应的参数作为最后结国。
一个训练的建议是使用两个优化器来训练 model，（判别器网络用一个优化器训练就行）分别对应两个步长λ，一个较大（0.005），一个较小(0.001)。小步长优化器为主，大步长优化器是为了使训练过程中有一定的可能性跳出局部极值，加快训练速度。

另一点需要注意的是：并不能保证每次训练都可以成功（即 loss 最终降下来并维持在某个水平震荡），若出现下面情况则有极大可能训练的阶段数不够或者参数陷入一个很差的 local optimum，此时需要重新训练直至出现上面走势的 loss。



（五）：排队系统的结构如何学习

参数的训练基本没有问题，现在的主要问题是排队系统的结构方面的三个问题：

1）如何确定各个 node 直接的转移概率矩阵（这个矩阵其实也完全定义了排队系统的结构）

2）如何确定各个 node 的 servers 的数量。这些数据无法用梯度下降类算法来训练（没有对于它们的梯度）。

3）对于 arrival distribution 和 service distribution 的具体是那种分布应该按什么标准来设定

上面的三个部分目前是假定都知道，上面的所有的训练都是以此为基础的。

（六）：排队系统一般关注问题以及（五）中提到问题的调研

在医院中 patient flow 是研究的热点之一，如何建立一个尽可能简单的模型用于模拟实际系统，如何得到 arrival rate 的分布以及如何控制 model 中的参数（与医院的各种规则相对应）来减小系统的平均阻塞等是关注比较多的问题。不过那些成熟的数据集中记录了每个病人在医院的流通过程，因此通过数据可以直接得到各个节点之间的概率转移矩阵。在大部分论文中研究着重点在于如何建立合适的模型，如何得到模型的参数以良好的近似真实系统，并借助模型来研究其他下游问题，而模型的具体结构往往是已知的。即（五）中的问题 1 其实没有那么重要。我们着重点在于学习到网络内在的参数，而此时为了使模型有实际用处，需要考虑的问题有：如何给定关于模型好坏的一个度量准则（例如 congestion，server occupation rate，delays）；我们训练得到的模型可以用于哪些下游任务（指导实际系统改进系统结构以及合理分配 server，或者做决策）；以及和传统统计方法比我们的模型的表现怎样。

医院是复杂网络（多节点，复杂 patient flow，到达时间和服务时间的复杂分布）的一个典型代表，而实际生活中的 calling center 和 limit order book 等则对应着简单网络（one node 或者多个互相没有交叉的 node），它们关注点更多是在有些资源情况下如何对资源进行分配以获得更好的 performance，而对这种情况下我们的方法可能需要做一定的调整，即如何把资源分配这一行为变得可求导，以通过梯度下降来学习，或者一个替代的方法是建立一个 metric，通过对改 metric 的参数更新（借助梯度下降）来间接指导资源的更新。