首先考虑的是连续变化参数的训练，训练数据来源：
Ground truth： ciw 模拟生成，参数为确定的 tensor
Simu data：ciw 生成，参数为随机初始的 tensor

1. 如何从 ciw 返回的 records 提取 feature 用于训练

ciw 得到的数据较为杂乱，而且由于模拟中存在多个随机因素导致每次得到的 record 的结构和数据分布都相差较大，为了使训练能够稳定且能够收敛到目标参数附近必须对每次返回的数据进行处理。

一个猜想:当模拟的时间（或顾客数量）达到某个阈值后，他们的数据即可表示总的网络的分布，也就是提取 feature 时并不需要用所有数据，但总顾客数量达到一定值即可。经过观察，各个 node 的 service time，arrival time 等数据的分布在 customer number 达到 30 多时基本稳定，后面的实验结果也验证了这点。这点也是为运用 wgan loss 以及其他 loss 做准备，用于训练的数据必须是确定 size 的。

直接用数据的统计量（均值，方差，极值）做 feature 向量结果并不理想，因此还是得用原来的数据。**处理方法为**：

对每个节点分别获取经过其的所有 individual 的数据，按照 arrival time 由小到大进行排序，再分别取其 data 中各类数据（arr_t，ser_t，wait_t，ser_start_t，blocked_t）构成对应的 feature 向量。由于 arrival time，service start time，exit time 这类增长数据在不同模拟中初值可能差较多，由此导致 loss 会差较大，但这不应该影响 loss，所以对其 feature 向量的每个分量均减去第一个分量来消除初始值的不必要影响。对于 waiting time，service time 则先按原始的量构成 feature 向量实验，发现 loss 不稳定，对应的策略是将其 feature 向量分段（比如每段长 15），然后用每段的均值代替原有的值，效果良好。

2. loss 的收敛情况和参数的收敛情况

训练过程的 loss 下降很快，大概 20~40epoch 即可下降到较低的水平，之后 loss 小幅度的震荡，最终趋于稳定。

训练参数中两类重要参数：arrival interval time 和 service time 服从的分布的参数，其中 arrival interval time 对应的参数收敛很快，而且能精确收敛到 ground truth 附近，service time 的训练则很难，多次实验发现其对应的参数在训练中往往表现为：先随着其他参数变化到某个数附近，然后朝着 ground truth 方向变化，但是震荡得厉害，且最终往往不能收敛到 ground truth，具体收敛到，哪每次训练都不同。

分析器原因我认为是复杂网络的对应的 loss 函数局部极值太多，训练过程很容易陷入其中，当步长较小时则无法跳出局部极值，表现为参数不断震荡，并最终在朝 ground truth 逼近的过程中收敛于其他值。因此我选择将训练分为两个阶段，第一阶段步长 learning_rate=0.005，保持较小，使参数稳定变化到 ground truth 附近，第二阶段，使用交替步长，即每 5 个 epoch 中将 4 个 epoch 的 learning_rate=0.002，一个 epoch 的 learning_rate=0.007。训练时果然起到预期的效果，参数通过定期步长的变大可以跳出局部极值。不过一个副作用是对于已良好收敛的参数（arrival interval time 相关的参数），这个变化可能会导致其偏离原来的收敛值，因此在第二阶段，需要固定已训练好的参数。

3. 训练结果

网络为：

```
N = ciw.create_network(
    arrival_distributions=[ciw.dists.Exponential(self.param[0]),
                           ciw.dists.Exponential(self.param[1]),
                           ciw.dists.NoArrivals()],
    service_distributions=[ciw.dists.Exponential(self.param[2]),
                           ciw.dists.Exponential(self.param[3]),
                           ciw.dists.Exponential(self.param[4])],
    routing=[[0.0, 0.3, 0.7],
             [0.0, 0.0, 1.0],
             [0.0, 0.0, 0.0]],
    number_of_servers=[1, 2, 2]
)
```

Experiment result example 1:

Ground truth 取 Param = [0.4,0.3,0.7,0.4,0.6]
训练 300 epoch
Loss：初始为 40，最终在 3~9 波动
参数初始化为：[0.9,0.9,0.9,0.9,0.9]
第一阶段（100 epoch）训练后稳定收敛到[0.41, 0.21, 0.55, 0.56, 0.52]
固定 param[0], param[1](这类参数总是能快速准确收敛，因此可以在阶段一结束后固定)进入第二阶段
第二阶段结束后参数最终收敛到[0.41，0.31，0.61，0.42，0.55]

Example 2:

Ground truth 取[0.3,0.15,0.9,0.4,0.6]
模型参数初始化为[0.9,0.9,0.9,0.9,0.9] loss=35

阶段一结束，参数初步收敛到：[0.31,0.16,0.48,0.50,0.55] loss=11
同样固定第一类参数：0.31，0.16，进入阶段二
阶段二结束，loss=7，参数收敛到：[0.31,0.16,0.62,0.48,0.6]

Example3：
Ground truth 取[0.3,0.15,0.3,0.4,0.2]
阶段一结束，参数收敛到[0.3，0.17，0.31，0.47，0.45]
阶段二结束，参数收敛到[0.3，0.17，0.33，0.40，0.23]

一个奇怪的现象是 arrival distribution 的参数总能快速准确收敛，但 service distribution 则总是容易陷入局部极值，或者说很难准确收敛到真实值，而会和 ground truth 差一些，不过差距可以容忍。另外一个现象就是对于较小的参数 mu，对应随机变量的期望为 1/mu 则较大，此时训练效果较好，对于较大的参数 mu，训练需要的 epoch 则较大，且最终收敛的值和 ground truth 的误差在±0.1 左右。