Ciw 模块模拟排队系统考虑的因素有：
arrival distribution,
service distribution,
batching distribution（每次到达某个 node 的 customer 数量服从的分布）
queue capacity of each node
customer class and class_change_matrix for each **node** ,
routing_matrix for each **customer class**,
baulking function (probs about whether to join a queue when there are many waiting customers)
schedule for servers (server num of each node in different time of a time cycle) and pre-empt

其中的参数可以分为这三类：
一：continuous，如 arrival distribution 的参数，可以通过重参数化使其加入到 network 的计算中并在 back prop 中更新，已修改好了相关的部分，目前 ciw 中这类参数我都确保了其参数可以求梯度。
二：integer：如 queue capacity，batching distribution 的参。Gumbel trick 是相应处理方法，但是考虑到实际情况中他们的值可能取得很大，所以在没有关于其的先验知识的情况下不太好处理，但是当给定其上界时可以训练，**此时一个主要问题是如何取初始分布，例如均匀分布，泊松分布，或是其他更好的，当 integer 可取的范围很大时均匀分布效果异常差，基本不能用。**
另外在接触到的介绍 gumbel relaxation 的资料包括原始 paper 中都**没有具体说明如何处理 softmax 输出的概率向量**，也就是为何可以解决 argmax 不可导的问题。(一个粗糙的说明：a natural way to extend (or 'relax') a discrete random variables is by allowing it to take values in the probability simplex）之前说到的用 softmax 向量和 categorical 向量相乘作为 sample 其实还是局限性太大，比如一个 categorical 分布可能取值为 (0.3, 1.6, 4.1)，把它和 softmax 向量相乘得到的结果不属于这三个可能 value 的任何一个，也不想整数变量可以简单的通过取整来处理。由于原始论文没有贴出代码我也没法看它们如何处理的，相关介绍资料基本都是简单复制，没讲到关键，而讨论区大家也似乎都默认 softmax 后就是可导的，但没人说过原因，很诡异。
三：categorical and Boolean：如 class change matrix ，routing matrix，它们在网络中不会直接参与计算，而是在某些时刻引导网络计算流（和第二类不太一样，第二类利用 gumgel trick 之后，参数会加入到网络的计算并体现在最终的 loss 中）。Ciw 中自己定义了 choice 采样按概率函数，当 customer 需要做诸如改变 class 或者决定下一个 node 时，便根据 choice 采样的结果来决定。而像 baulking，pre-empt 这类 boolean 变量，取值为 true 或 false，也对应在网络计算某些步骤决定某些操作是否会发生。这类变量无法直接通过梯度更新。
这类参数的更新应该如何设置？是否应在每次 backprop 时都同其他普通的参数一同进行更新？（查阅资料后我觉得这时的情况和强化学习中策略梯度有些类似，都是对决策的概率分布进行更新，但是和策略梯度更新有点不同的是这里没有明确的奖励来直接指导更新。）
目前我的想法是对这类变量采用枚举方法，也就是每个 batch 训练时先训练其他两类参数，然后对所有第三类参数的每个可能的取法组合分别多次 sample 模拟，取 loss 最小的那个组合，还需要修改代码跑看效果怎么样，但这种方法还是太笨重了。

关于 loss：
我们的目标是希望从实际排队系统的数据（各类时间）通过和模拟出的数据进行对比并不断更新参数减少两者差距来推测真实数据。一个选择是 wasserstein gan loss，pytorch 目前还没有内置实现该 loss 的函数，看了一下 github 的 wgan 项目和 wgan 算法。

Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

Require: : α, the learning rate. c, the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic per generator iteration.
Require: : w_0, initial critic parameters. θ_0, initial generator's parameters.
while θ has not converged **do**
  for $t = 0, ..., n_{\text{critic}}$ **do**
    Sample $\{x^{(i)}\}_{i=1}^{m} \sim \mathbb{P}_r$ a batch from the real data.
    Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
    $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))\right]$
    $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$
    $w \leftarrow \text{clip}(w, -c, c)$
  end for
  Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
  $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))$
  $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$
end while

上面的 loss 中 g 代表 generator，f 代表 discriminator，f 输出的是标量，即判断结果，loss= $\left[\frac{1}{m}\sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))\right]$ 。loss 求的是期望。看了 wgan 的代码的 loss 部分，我觉得不适用我们的网络的训练。**因为我们并没有定义判别器这么个网络，也就是 wganloss 中的 f，这样的话 loss 的表达式就无法表示**，gan 的任务和我们的任务的确很相似，但是 gan 里面需要训练的是 d，g 对应的卷积网络的参数，而我们则是有点回归训练的意思，所以我不太确定 wgan 的 loss 是否有用，但是首先要解决如何对 simulation 这个问题定义 f，然后才能用以上 loss，代码层面倒是不难。

我们的是一系列统计的时间相关的数据，希望的是这个数据可以和模拟的数据的分布近似，进而近似真实的参数，目前认为 hingeembeddingloss 稍微合适一些，所以实验代码暂时是按这个 loss 跑的，然后考虑了 wgan 的建议，选用了 RMSprop 优化，对模型参数进行 clip 夹逼。
不过在跑代码时发现了一个问题：**之前提到了网络有一类参数：转移矩阵。每个 customer 在每个 node 的结束 service 后向下一下 node 转移时选择的 node 都是不确定的，即使是相同参数的同一个网络，在两次不同模拟中得到的结果由于个体 node 选择的不同最终得到的数据分布也极可能差别很大**。（比如同样是 100 个 customer，按照 0.3，0.7 的转移概率，有 30 个到了 A node，但是可能是前 30 个，也可能是随意顺序的 30 个）考虑这点，对于复杂网络，像 exit_date 这类增长类型的数据就不能直接用了，因为每次 simulate 出来肯定相差很多。这点在代码里面我还没想好这么修改。
转移矩阵带来的不确定性随着网络复杂性提高（node 数量变多，node 之间转移的可选对象的增加，customer class num 的增加），而体现在 loss 里面的应该是每类数据直接用则没有意义了，这样训练的 loss 肯定是震荡的。需要用多次模拟的平均值，但是取平均值也还存在问题。

几个问题需要确认：
关于训练，对于不同的 queue network 或是相同的 network 结合不同的 distributoin 类型，整个 simulator 模拟的过程都是不同的，即使是所有的条件都一样，两次不同的模拟对应的 computation graph 都是不一样的，所以
1.**我们训练参数时必须对网络定义良好，也就是网络结构必须是已知的，只有具体的参数是未知的，这点是否可以得到保证？**（如果结构和 ground truth 不一样比如 node 的数量少了或者某个分布的类型和 ground truth 不同，那训练可能无法收敛，但是我们也不知道具体是哪里出了问题，这样训练也是没有意义了）
2.**我们要做的似乎限于：寻找适合这个任务的 loss 以及找出针对不同网络的训练 trick，使训练结果可以快速收敛到 ground truth?**
3.**开始列举的这些参数是否都要考虑训练？**我们的问题考虑的模型需要有多复杂。
考虑了之前的所有问题后我打算统计数据时以 node 为对象，考虑每个 node 的各类数据，并分别体现在 loss 里面，而不是像之前把所有数据全放在一起。统计的是各个 node 的 avg_waiting_time，server_utilization，arrival_time_interval…等数据，也就是不统计增长的数据，而是统计增长的间隔，比如 arrivaltime=[1,3,7,13]，训练时不是用 1，3，7，13 来计算 loss，而时考虑 node 的 service_start_time，但同样也是取其间隔并计算均值。目的也是尽可能减少网络的不缺定性导致 arrivaltime，exit_date 这类数据变化太大造成的 loss 的不稳定。

以上是目前我的考虑，需要解决的部分用粗体标出来了。