上面我们用模型输出结果和已有的解的交叉熵作为损失函数，属于有监督学习，模型求解的质量理论上受限于训练数据集的质量，是在模仿之前的策略。而 tsp 问题目前并没有确定性的寻找最优解算法，因此我们也希望通过训练找到更优的策略，而不是模仿已有的策略。

考虑使用强化学习算法，下面给出强化学习的场景的符号描述：

定义状态集 S，状态 $s\in S$，这里 S 为 $\left([0,1]\times[0,1]\right)^n\times\left(\{1,2,\cdots,n\}\right)^2$ 上的集合，表示在 n 个点的图中的哪个位置，走到了第几步。

定义状态序列 $\{s_0,s_1,\cdots s_t\}$，和动作序列 $\{a_0,a_1,\cdots a_t\}$，其中 s_0 为初始状态，s_t 为在时刻 t 处于的状态，a_t 为在时刻 t 采取的行动，且满足 $a_t\sim\pi\left(s_t\right)$，意为在状态 s_t 时，按照策略 π 的概率分布采取行动 a_t，这里 a_t 即为在当前点前往下一个点的动作

定义每一步行动的回报 $R\left(s_t,a_t\right)$，则这里一步的回报为第 t 个点到第 $t+1$ 个点的距离。

目标函数有多种定义方式，如初始状态到终止状态的回报和，或是动作序列每个状态到终止状态的回报和平均值等，这里我们采用初始状态到终止状态的回报和来定义，即

$$J=E\left(\sum_t\gamma^tR\left(s_t,a_t|\pi\right)\right)$$

.

其中 γ 是衰减因子，这里我们取 $\gamma=1$，则事实上目标函数可写成 $J\left(s\right)=E_{a\sim\pi(s)}\left(L\left(a|s\right)\right)$，其中 s 仅表示图中点的信息，a 为按照策略 π 选择的排列，$L\left(a|s\right)$ 即为按这个排列走出的路径长。

我们的目标是最小化目标函数。

我们依然采用 Pointer-Network 来给出策略 π（即在每个时刻选择下一个点的概率），令 θ 为网络参数，则有

$$J\left(\theta|s\right)=E_{a\sim\pi_\theta(s)}\left(L\left(a|s\right)\right)$$

.

则根据策略梯度定理，有

这里 $\pi_\theta\left(a|s\right)$ 为参数 θ 下按策略 π_θ 在图 s 得到排列 a 的概率。

采用蒙特卡洛方法对图分布 $s\sim S$ 进行采样（就是在 $[0,1]\times[0,1]$ 上随机生成图），并在选择输出排列时按概率分布 p_θ 进行采样，则可得到对 $J\left(\theta\right)$ 的无偏估计

$$\nabla J\left(\theta\right)\approx\frac{1}{B}\sum_{i=1}^{B}L\left(a_i|s_i\right)\nabla_\theta\log\pi_\theta\left(a_i|s_i\right).$$

如果样本对期望的方差过大，由于我们训练时每步都更新参数，可能导致模型难以收敛到最优区域。考虑加入 baseline function $b\left(s\right)$，用来减小更新参数时梯度的方差，则式子改写为：

$$\nabla J\left(\theta\right)\approx\frac{1}{B}\sum_{i=1}^{B}\left(L\left(a_i|s_i\right)-b\left(s_i\right)\right)\nabla_\theta\log\pi_\theta\left(a_i|s_i\right).$$

加入 $b(s)$ 对梯度的期望没有影响。这是因为

$$\begin{aligned}E_{a\sim\pi_\theta(s)}b(s)\nabla_\theta\log\pi_\theta(a|s) &= \sum_{a\in\pi_\theta(s)}\pi_\theta(a|s)\,b(s)\,\nabla_\theta\log\pi_\theta(a|s)\\ &= b(s)\sum_{a\in\pi_\theta(s)}\pi_\theta(a|s)\frac{\nabla_\theta\pi_\theta(a|s)}{\pi_\theta(a|s)}\\ &= b(s)\sum_{a\in\pi_\theta(s)}\nabla\pi_\theta(a|s)\\ &= b(s)\nabla 1 = 0.\end{aligned}$$

考虑选取怎样的 $b(s)$ 能使方差更小。考虑 $b(s)$ 对方差的影响:

$$\begin{aligned}&D_{a\sim\pi_\theta(s)}\left(\sum_{i=1}^{B}\left(L(a_i|s_i)-b(s_i)\right)\nabla_\theta\log\pi_\theta(a_i|s_i)\right)\\ =&E_{a\sim\pi_\theta(s)}\left(\left(\sum_{i=1}^{B}\left(L(a_i|s_i)-b(s_i)\right)\nabla_\theta\log\pi_\theta(a_i|s_i)\right)^2\right)-E^2_{a\sim\pi_\theta(s)}\left(\sum_{i=1}^{B}\left(L(a_i|s_i)-b(s_i)\right)\nabla_\theta\log\pi_\theta(a_i|s_i)\right)\\ \approx&E_{a\sim\pi_\theta(s)}\left(\left(\sum_{i=1}^{B}\left(L(a_i|s_i)-b(s_i)\right)\nabla_\theta\log\pi_\theta(a_i|s_i)\right)^2\right)\quad(1)\\ \approx&\sum_{i=1}^{B}E_{a\sim\pi_\theta(s)}\left(\left(L(a_i|s_i)-b(s_i)\right)^2\right)\sum_{i=1}^{B}E_{a\sim\pi_\theta(s)}\left(\nabla_\theta\log\pi_\theta(a_i|s_i)^2\right)\quad(2)\\ \approx&\sum_{i=1}^{B}E_{a\sim\pi_\theta(s)}\left(\left(L(a_i|s_i)-b(s_i)\right)^2\right).\end{aligned}$$

这里 (1) 步用到之前证明的 $b(s)$ 对期望无影响，故我们可以不考虑期望的平方项；(2) 步假设各样本之间、几个参数之间都具有独立性；(3) 步去掉了与 $b(s)$ 无关的项。

很明显，$b(s)$ 对方差的影响项 $\sum_{i=1}^{B}E_{a\sim\pi_\theta(s)}\left(\left(L(a_i|s_i)-b(s_i)\right)^2\right)$ 是一个均方误差的形式，故我们选择 $b(s)=E_{a\sim\pi_\theta(s)}L(a_i|s_i)$ 时能将这一项的值降到最小。

下面考虑估计 $E_{a\sim\pi_\theta(s)}L(a_i|s_i)$.

一种方法较为简单，采用指数移动平均的形式，即把每个样本 s_i 在经过网络后得到的结果 $L(a_i|s_i)$ 进行加权平均作为估计，早期的结果权重会指数衰减，总的估计可以采用以下式子计算:

$$EMA_t=\begin{cases}L(a_t|s_t), & t=1\\ (1-\beta)L(a_t|s_t)+\beta EMA_{t-1}, & t>1\end{cases}$$

其中 β 为衰减因子，衡量衰减速度的快慢。

另一种方式是用一个另外的网络对 $E_{a\sim\pi_\theta(s)}L(a_i|s_i)$ 进行估计，这个网络称为 critic network，之前的 pointer network 称为 actor network，它的结构如下:

1. 首先是一个与 actor network 中 encoder 类似的 LSTM RNN
2. 接另一个 LSTM 单元，将 encoder 的最后一个隐层向量作为初始隐状态，每次令其经过单元并计算之前 attention 机制所述的 context，将 context 作为下一步的输入。这个过程重复 P 次
3. 让最后的隐层向量经过 $d*d$ 全连接——ReLU——$d*1$ 全连接层，得到最终的估计值。d 为隐层向量维数

损失函数选用估计值与当前的解的均方误差 MSEloss.

则整个训练的过程为:

1. 获取数据
2. 数据经过 actor network 得到当前解

   解的选取方式：不使用贪心策略，根据当前的概率分布选择，即采样（可以用 pytorch 中的 multinomial 函数实现），并且选择过程中对选项加 mask（之前提到的第二种写法），从而保证解是合法的
3. 用 critic network（或者指数移动平均）对解的期望长度进行估计，得到 baseline function
4. 计算 actor network 的 loss 并 bp.
5. 计算 critic network 的 loss 并 bp（若使用指数移动平均则没有这一步）.