图示

描述已自动生成

1. Action space, obs space, reward, state space

Action: 多维

- space boom

- 维度有关联，相关性

1. Work Mode:Active SM1 SM2 SM3
2. Cover Size: SMALL, MIDDLE, LARGE
3. User Connections Matrix:

U0 U1 U2 U3 U4

Bs0 Y N Na

Bs1

Bs2

#注意，这个是一种action情况，因为BS0，BS1，BS2是同时选择的，没有先后顺序

[ .... ]

- a user connection [......]

- work mode, cover size \* user connection matrix

State:

- Work Mode State: Active SM1 SM2 SM2

-

- Waiting Packet Size:

PENDING U0 U1 U2 U3 U4

BS0 100 10 20 - - -

BS1 200 - - 10 - 30

BS2 300 - - - 30 -

- N: f() -> [random()， ...]

-

- LSTM(两个数据): 预测用户下一个数据的大小。()  
LSTM(u0, t0) - > u0的t1

ENV -> s0 -> AGENT -> action next\_state

真的有值 真的有值

Reward:

waitpacketsize0+waitpacketsize1+waitpacketsize2 == 0：

reward =50

when waitpacketsize0+waitpacketsize1+waitpacketsize2 ！= 0：

reward =-waitpacketsize0+waitpacketsize1+waitpacketsize2 /1000

Transform Path:

S0 -> A0 -> ENV -> S1,R0

BS1state (x) (S0.A0)

S0,A0 —————|——————— 1ms 100 ———————————— 1ms 100

SM1-activate activate(处理数据) activate

0.4us． 100\*0.6 = 60 pending ：40。 下-时刻40+U0+U3

prolog

flatmap

Dim： [\_\_\_\_ , \_\_\_, \_\_\_, ]

Action的详细过程:

1.Action: 多维

- space boom

- 维度有关联，相关性

1. Work Mode: Active SM1 SM2 SM3
2. Cover Size: SMALL, MIDDLE, LARGE

(红色圆表示SMALL，黑色圆表示MIDDLE，绿色圆表示LARGE)

1. User Connections Matrix:

U0 U1 U2 U3 U4

Bs0 Y N Na

Bs1

Bs2

首先，action的三个维度没有先后顺序，三个维度的action是同时的。

1. **维度1（work mode）与维度2(cover size）：**

有4个work model，分别为Active SM1 SM2 SM3，有3个cover size，分别为SMALL, MIDDLE, LARGE。

（a）当基站处于Active的工作模式时，有三类cover size的情况，分别为：

* Active-SMALL
* Active- MIDDLE
* Active- LARGE

（b）当基站处于SM1的工作模式时，有一类cover size的情况，为：

* SM1- MIDDLE

（c）当基站处于SM2的工作模式时，有一类cover size的情况，为：

* SM2- MIDDLE

（d）当基站处于SM3的工作模式时，有一类cover size的情况，为：

* SM3- MIDDLE

维度1（work mode）与维度2(cover size）的关系图如下：

图示

描述已自动生成

1. **维度1（work mode）与维度3(User Connections Matrix）：**

有4个work model，分别为Active SM1 SM2 SM3，有5个user，分别为：user1, user2, user3, user4, user5。

当基站处于任何工作模式时，都可以连接任何用户。

（a）当基站处于Active工作模式时，可连接的用户为：

* Active- user1, user2, user3, user4, user5

（b）当基站处SM1工作模式时，可连接的用户为：

* SM1- user1, user2, user3, user4, user5

（c）当基站处SM2工作模式时，可连接的用户为：

* SM2- user1, user2, user3, user4, user5

（d）当基站处SM3工作模式时，可连接的用户为：

* SM3- user1, user2, user3, user4, user5

维度1（work mode）与维度3(User Connections Matrix）的关系图如下：

图示

描述已自动生成

1. **维度2（Cover size）与维度3(User Connections Matrix）：**

有3个cover size，分别为SMALL, MIDDLE, LARGE。，有5个user，分别为：user1, user2, user3, user4, user5。

这两维度的关系取决于用户的位置是否在所属的圆圈范围内，根据下述图，可以直观的看出用户位置是否在cover size范围内，如果用户位置在圆圈范围内，则属于合法情况，如果不在圆圈范围内，则属于不合法情况

图示

描述已自动生成

（a）对于用户1位于：

* User1- BS1红圆，BS1黑圆，BS1绿圆，BS2绿圆

（b）对于用户5位于：

* User2- BS0红圆，BS0黑圆，BS0绿圆，BS1绿圆

（c）对于用户3位于：

* User3- BS0绿圆，BS2红圆，BS2黑圆

（d）对于用户4位于：

* User4- BS2红圆

（e）对于用户5位于：

* User5- BS0绿圆，BS1绿圆，BS2绿圆

维度2（Cover size）与维度3(User Connections Matrix）的关系图如下：

图示

描述已自动生成

1. **维度1（work mode）与维度2(cover size）与维度3(User Connections Matrix）：**

图示

描述已自动生成

图示

描述已自动生成

Work model and

Cover Size

And

User Connect

图示

描述已自动生成

State:

- Work Mode State: Active SM1 SM2 SM2

-

- Waiting Packet Size:

PENDING U0 U1 U2 U3 U4

BS0 100 10 20 - - -

BS1 200 - - 10 - 30

BS2 300 - - - 30 -

- N: f() -> [random()， ...]

-

- LSTM(两个数据): 预测用户下一个数据的大小。()  
LSTM(u0, t0) - > u0的t1

ENV -> s0 -> AGENT -> action next\_state

真的有值 真的有值

State的详细过程:

State\_space = （time，BS0Model，BS1Model，BS2Model，waitpacketSizes0，waitpacketSizes1，waitpacketSizes2，N1-15，LSTM\_user1, LSTM\_user2, LSTM\_user3, LSTM\_user4, LSTM\_user5）

疑问：

1. state中是否还需要加上当前用户的连接情况
2. 对于Work model的转换，由于SM1到SM2再到SM3之间的转换时间为小于 1ms，然而一个循环是1ms，是否一个循环只能执行一次action的变化，如果是的话，那当基站向activate转换时，由SM3向activate转换至少需要3ms，latency太大
3. 基站的work model转换，因为要用PPO探索action，如果按照我画的图，那不就又是按照我们的逻辑来选择action了吗，没有PPO探索的过程了，是否只是需要判断waitpacketsize的大小，如果大于0，则在action\_space中删除不合法的情况（既基站SM1，SM2，SM3的action），如果等于0，则可以让基站随机探索
4. State由13个元素组成，是否可以认为是13维还是5维（1.Time 2.Work Model 3.WaitpacketSize 4.N 5.LSTM）

**一、State的组成介绍**

1. State 由13个元素组成，可将这13个元素按照类型分为5组，分别为：

* Time: 时间，单位为1ms
* Work model: BS0 model, BS1 model, BS2 model
* WaitpacketSizes: WaitpacketSizes0, WaitpacketSizes1, WaitpacketSizes2,分别表示当时刻，3个基站上的数据分别是多少
* N1-15:15个符合正态分布的随机数
* LSTM：LSTM\_user2, LSTM\_user3, LSTM\_user4, LSTM\_user5，分别表示使用LSTM分别预测5个用户下一时刻的数据大小，用于决策BS是否从当前时刻开始由睡眠模式（SM1，SM2，SM3）转为工作模式（activate）

1. State 如何根据action 变化为 next\_state:

* Time: 时间,循环一次的时间为1ms，因此每循环一次，time+1
* Work model:
  + 根据当前时刻的work model 、当前时刻的WaitpacketSizes大小和LSTM的预测情况

流程图如下

图示

描述已自动生成

* WaitpacketSizes：
  + 根据上一时刻剩余的流量情况+当前连接的用户情况以及连接用户携带的数据+基站对数据的处理能力+是否执行了work model的转换

流程图如下：

图示

描述已自动生成

* N1-15：15个符合正态分布的数字
* LSTM：根据5个user的现有数据，预测5个user分别下一时刻（下一个1ms）的数据大小

**二、State各维度间的关系**

1. 与Work Model相关的维度：

图示

描述已自动生成

2. 与WaitpacketSize相关的维度：

图表

描述已自动生成

1. N： N的计算与stata的维度无关，是15个符合正态分布的随机数
2. LSTM：LSTM与其他数据无关，根据5个user的现有数据预测user的未来数据大小

Reward:

waitpacketsize0+waitpacketsize1+waitpacketsize2 == 0：

reward =50

when waitpacketsize0+waitpacketsize1+waitpacketsize2 ！= 0：

reward =-waitpacketsize0+waitpacketsize1+waitpacketsize2 /1000

Transform Path:

S0 -> A0 -> ENV -> S1,R0

BS1state (x) (S0.A0)

S0,A0 —————|——————— 1ms 100 ———————————— 1ms 100

SM1-activate activate(处理数据) activate

0.4us． 100\*0.6 = 60 pending ：40。 下-时刻40+U0+U3

prolog

flatmap

Dim： [\_\_\_\_ , \_\_\_, \_\_\_, ]

LSTM的过程:

将LSTM分为三个部分，分别为：

* LSTM\_UserDataCOmbination.py
* LSTM\_DataProcess.py
* LSTM.py

1. **LSTM\_UserDataCOmbination.py**

每个user的原始数据各有两个，分别为generationTime.mat和packetSizes.mat文件，每个文件各有一列数据，generationTime.mat表示数据生成时间，packetSizes.mat表示对应时间的数据大小，格式如下：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| generationTime.mat(ms) | 0 | 0 | 0 | 49 | 49 | 135 | 135 |
| packetSizes.mat. | 100 | 100 | 100 | 300 | 500 | 700 | 900 |

LSTM主要用户预测下一时刻data的大小，用于生成next\_state, 然而目前的数据在时间上是不连续的，且生成时间和数据大小位于两个mat文件中，**LSTM\_UserDataCOmbination.py**文件将这两个mat文件合并为一个csv文件，新生成的csv文件第一列为generationTime，第二列为packetSizes

1. **LSTM\_DataProcess.py**

* 在LSTM\_UserDataCOmbination.py将文件合并为一个csv文件后，由于LSTM主要预测data（packetSizes）的大小，而这列数据在时间上是不连续的，因此，对其两步处理：
* 将在相同时间产生的数据进行求和，如在1中用于举例的表格，在第0ms有3个数据，分别大小为100，100，100，将其进行求和操作，最终数据变为

|  |  |  |  |
| --- | --- | --- | --- |
| generationTime.mat(ms) | 0 | 49 | 135 |
| packetSizes.mat. | 300 | 800 | 1600 |

* 生成时间上连续的数据，在NAN处用0补齐，如下：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| generationTime.mat(ms) | 0 | 1 | 2. | …… | 49 | …… | 135 |
| packetSizes.mat. | 300 | 0 | 0 | 0 | 800 | 0 | 1600 |

1. **LSTM.py**

标准的LSTM 模型

PPO过程:

疑问：

1. PPO.py中的**take\_action**函数

由于PPO是基于策略的强化学习，action的概率都是有actor网络根据给定的state输入来输出的，所以对action的概率是直接由agent做出的决策，而在DQN中，由于DQN是基于值函数的强化学习，因此对action的选择是直接在action\_space中随机选择的，因此在DQN中需要人为的加上对action的某些条件限制，如：

文本

描述已自动生成

而在PPO中，就不需要人为加入这些条件了，可以有agent自己决策

1. PPO中的actor网络输出已经是个概率分布了，为何在**take\_action**函数中还要加入**Categorical**分布
2. 在base\_station\_env.py中，class BaseStationEnv()类的构造函数的time, BS model, waitpacketSize, N1-15, LSTM等需要初始化给他一个值吗，如time = 0。

图形用户界面, 文本

描述已自动生成

1. 在‘for episode in range(1000):’循环中，是循环一次，agnet与env交互一次，还是循环一次，一个回合结束，循环1000次，循环了1000次循环，（while一次交互一次）

6.

文本

描述已自动生成

**我做的工作：**

1. 创建Vars.py文件，定义了一些会用到的全局变量
2. 创建了function.py文件，用于base\_station\_env.py文件中的step()函数直接调用
3. 在main.py中导入了user1-user5的数据集（处理好的将generation.mat和packetsize.mat文件合并的数据集）
4. 在main.py文件中加入：（1）读取5个user的数据 （2）创建3个arrival\_data\_matrix

（3）

|  |  |  |
| --- | --- | --- |
| **User\_data** | **User\_data 到达时间** | **User\_data处理完的时间** |
| Numpy数组 |  |  |

1. 在base\_station.py的step函数中，写一个while循环，实现agent与env的交互