Для использования алгоритмов обучения с подкреплением:

1. Скачайте пакет Python “Open\_VO” с репозитория на гитхабе используя “python/build/Open\_VO.cp312-win\_amd64.pyd” или “python/build/open\_vo-1.0.0-py3-none-any.whl”.
2. Создайте проект в PyCharm (или другой IDE) и создайте \*.py файл. Установите “Open\_VO”:
   1. “.pyd”: положите пакет “Open\_VO.cp312-win\_amd64.pyd” в ваш проект и импортируйте библиотеку “import Open\_VO”.
   2. “.whl”: установите пакет “pip install python/build/open\_vo-1.0.0-py3-none-any.whl” и импортируйте библиотеку “import Open\_VO”.

3. На примере алгоритма SARSA добавьте следующие модули:

import torch  
import torch.optim as optim  
  
from Open\_VO import BaseAgentSARSA, BaseModelSARSA, BaseTrainerSARSA, MemoryCell

4. После чего используйте следующую конструкцию:

4.1. Инициализируйте необходимые переменные:

maxMemory = 100000  
batchSize = 1000  
randCoef = 60  
randRange = 200  
inputSize = 4  
hiddenSizes = [32, 16]  
numClasses = 4  
lr = 0.001  
gamma = 0.9  
device = torch.device('cpu')

4.2. Получите состояния среды для вашей задачи:

oldState = [1, 0, 0, 1]  
newState = [0, 1, 0, 1]

\* Для описания среды, вы можете использовать базовый класс “BaseEnvironment”.

4.3. Предскажите следующее действие вашего агента:

finalMove = agent.act(oldState)  
newMove = agent.act(newState)  
print(finalMove, newMove)

4.4. Либо опишите процесс обучения агента:

cell = MemoryCell(state=oldState, action=finalMove, next\_state=newState, next\_action=newMove, reward=2.3, done=False)  
  
agent.train\_short\_memory(cell)  
agent.remember(cell)  
agent.remember(cell)  
agent.remember(cell)  
agent.train\_long\_memory()

Важно учитывать, что разные алгоритмы используют разные пайплайны обучения и оперируют разными переменными. Модуль обучения с подкреплением включает 5 алгоритмов:

* DQN
* SARSA
* PG
* A2C
* PPO

Для лучшего понимания, следует изучить теорию в области интересующего вас алгоритма.