교육 코드공유 토론 More Search NN weonwo…

# 블럭 장난감 제조 공정 최적화 AI경진대회

산업 | LG | 인공지능 AI 활용 제조 공정 최적화 알고리즘 | 강화학습

상금 : 총 600만원

2020.06.01 ~ 2020.06.30 17:59 + Google Calendar

547팀 D-14

참여중

대회안내 데이터 코드 공유 토론 리더보드 팀 제출

## ▲ 4 PPO를 활용한 베이스라인

2020.06.15 22:10 13 views Python by 나단단단단단단단단단단단단단단단단단단단단 댓글 0

discrete action space ppo 모델 두개를 사용하는 방법입니다.

수렴 가능성은 그다지 높진 않다고 생각됩니다. 참고용으로 봐주시면 감사하겠습니다.

저는 이 코드를 활용해서 87점까지 학습이 가능했습니다. 아마 더 좋은 방법도 있을거라 생각됩니다.

다른 문제가 발생하거나 궁금한점이 있으시면 댓글 남겨주시길 바랍니다.

다운로드 북마크 0

코드

```
import gym
import numpy as np
import pandas as pd
import math

from simulator import Simulator

class FactoryEnv(gym.Env):
    def __init__(self, is_train):
        self.is_train = is_train
        self.simulator = Simulator()

        self.order_data = pd.read_csv("data/order.csv")
        for i in range(40):
            self.order_data.loc[91+i,:] = ['0000-00-00', 0, 0, 0, 0]

        self.submission = pd.read_csv("data/sample_submission.csv")

        self.work_time = [28, 98] * 17 + [42]
        self.action_plus = [(0.0, 0.0), (5.8, 0.0), (0.0, 5.8), (5.8, 5.8)]

        self.MOL_queue = np.zeros([49, 4])

    def save_csv(self):
        PRTs = self.submission[["PRT_1", "PRT_2", "PRT_3", "PRT_4"]].values
        PRTs = (PRTs[:-1]-PRTs[1:])[24*23:]
        PRTs[-1] = [0., 0., 0., 0.]
        PRTs = np.ceil(PRTs * 1.1)+1
        PAD = np.zeros((24*23+1, 4))
        PRTs = np.append(PRTs, PAD, axis=0).astype(int)
        self.submission.loc[:, "PRT_1":"PRT_4"] = PRTs

        self.submission.to_csv("test.csv", index=False)

    def reset(self):
        self.now_stock = np.array(pd.read_csv("data/stock.csv"), dtype=np.float32)[0]

        self.step_count = 0
        self.work_index = 0
        self.remain_time = 0

        self.line_A_yield = 0.0
        self.line_B_yield = 0.0

        self.line_A_MOL = []
        self.line_B_MOL = []

        state = np.concatenate([[self.step_count], [0]*20, self.now_stock[8:]])/1000000

        return state

    def step1(self, action):
        action_list = [(1, 1), (2, 2), (3, 3), (4, 4), (1, 2), (1, 3), (1, 4), (2, 3), (2, 4), (3, 4)]

        self.line_A_MOL.append(action_list[action][0])
        self.line_B_MOL.append(action_list[action][1])
    def step2(self, action):
        if self.remain_time == 0:
            self.remain_time = self.work_time[self.work_index] - 1
            self.work_index += 1
        else:
            self.remain_time -= 1

        if self.step_count == 552:
            self.line_A_yield = 3.2
            self.line_B_yield = 3.2

        def process():
            self.now_stock[4:8] += self.MOL_queue[0]
            if self.step_count > 551:
                self.MOL_queue[-1][self.line_A_MOL[math.floor((self.work_index-1)/2)]-1] = self.line_A_yiel
                self.MOL_queue[-1][self.line_B_MOL[math.floor((self.work_index-1)/2)]-1] = self.line_B_yiel

            self.MOL_queue[:-1] = self.MOL_queue[1:]
            self.MOL_queue[-1] = [0, 0, 0, 0]

            if self.step_count > 551:
                self.now_stock[self.line_A_MOL[math.floor((self.work_index-1)/2)]-1] -= self.line_A_yield
                self.now_stock[self.line_B_MOL[math.floor((self.work_index-1)/2)]-1] -= self.line_B_yield

        if self.work_index % 2 == 0:
            self.line_A_yield = self.action_plus[action][0]
            self.line_B_yield = self.action_plus[action][1]

            process()

        self.submission.loc[self.step_count, "PRT_1":"PRT_4"] = self.now_stock[:4]

        # done, reward
        if self.step_count == 2183:
            done = True
            score, _ = self.simulator.get_score(self.submission)
            reward = (20000000 - score) / 20000000
            print(f"reward : {reward}")
        else:
            reward = 0
            done = False

        # write
        if self.work_index % 2 != 0:
            self.submission.loc[self.step_count, "Event_A"] = f"CHECK_{self.line_A_MOL[math.floor((self.wor
            self.submission.loc[self.step_count, "MOL_A"] = 0.0
            self.submission.loc[self.step_count, "Event_B"] = f"CHECK_{self.line_B_MOL[math.floor((self.wor
            self.submission.loc[self.step_count, "MOL_B"] = 0.0
        else:
            self.submission.loc[self.step_count, "Event_A"] = "PROCESS"
            self.submission.loc[self.step_count, "Event_B"] = "PROCESS"
            if self.step_count > 551:
                self.submission.loc[self.step_count, "MOL_A"] = round(self.line_A_yield, 1)
                self.submission.loc[self.step_count, "MOL_B"] = round(self.line_B_yield, 1)
            else:
                self.submission.loc[self.step_count, "MOL_A"] = 0.
                self.submission.loc[self.step_count, "MOL_B"] = 0.
        # state t+1
        self.step_count += 1
        state = np.concatenate([[self.step_count], np.array(self.order_data.loc[self.step_count//24:(self.s

        info = {}
        return state, reward, done, info
```

```
import os
import signal
import itertools

import numpy as np
import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
from torch.distributions import Categorical


clip_range = 0.2
gamma = 0.99
lam = 0.95
learning_rate = 0.001

hidden_size = 256

HORIZON = 2184
train_iter = 2

save_interval = 5

# True -> train
# False -> inference
is_train = True


class GracefulKiller:
    def __init__(self):
        self.kill_now = False
        signal.signal(signal.SIGINT, self.exit_gracefully)
        signal.signal(signal.SIGTERM, self.exit_gracefully)

    def exit_gracefully(self, signum, frame):
        self.kill_now = True


class PPO(nn.Module):
    def __init__(self, output_shape):
        super(PPO, self).__init__()
        self.buffer = []
        input_shape = 1 + 20 + 4

        self.fc1 = nn.Linear(input_shape, hidden_size)
        self.fc2 = nn.Linear(hidden_size, hidden_size)
        self.logits_net = nn.Linear(hidden_size, output_shape)
        self.v_net  = nn.Linear(hidden_size, 1)
        self.optimizer = optim.Adam(self.parameters(), lr=learning_rate)

    def pi(self, x):
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.logits_net(x)
        return Categorical(logits=x)

    def v(self, x):
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        v = self.v_net(x)
        return v

    def store(self, transition):
        self.buffer.append(transition)

    def update(self):
        states = torch.tensor([e[0] for e in self.buffer], dtype=torch.float)
        actions = torch.tensor([[e[1]] for e in self.buffer])
        rewards = torch.tensor([[e[2]] for e in self.buffer], dtype=torch.float)
        next_states = torch.tensor([e[3] for e in self.buffer], dtype=torch.float)
        probs = torch.tensor([[e[4]] for e in self.buffer], dtype=torch.float)
        dones = torch.tensor([[1-e[5]] for e in self.buffer])
        self.buffer = []

        for _ in range(train_iter):
            td_target = rewards + gamma * self.v(next_states) * dones
            delta = td_target - self.v(states)
            delta = delta.detach().numpy()

            advantages = []
            advantage = 0.0
            for delta_t in delta[::-1]:
                advantage = gamma * lam * advantage + delta_t[0]
                advantages.append([advantage])
            advantages.reverse()
            advs = torch.tensor(advantages, dtype=torch.float)

            pi = self.pi(states)
            ratio = torch.exp(pi.log_prob(actions) - torch.log(probs))
            cilp_ratio = torch.clamp(ratio, 1-clip_range, 1+clip_range)

            pi_loss = -torch.mean(torch.min(ratio*advs, cilp_ratio*advs))
            vf_loss = torch.mean(torch.pow(self.v(states) - td_target.detach(), 2))

            loss = pi_loss + vf_loss

            self.optimizer.zero_grad()
            loss.mean().backward()
            self.optimizer.step()


def main():
    env = FactoryEnv(is_train)
    killer = GracefulKiller()
    model1 = PPO(10)
    model2 = PPO(4)
    if os.path.exists("save.pt"):
        print("model loaded!")
        checkpoint = torch.load("save.pt")
        model1.load_state_dict(checkpoint["model1"])
        model2.load_state_dict(checkpoint["model2"])

    if not is_train:
        model1.eval()
        model2.eval()

    for i in itertools.count():
        s = env.reset()
        done = False
        while not done:
            for t in range(HORIZON):
                def get_action(model):
                    pi = model.pi(torch.from_numpy(s).float())
#                     if is_train:
                    a = pi.sample()
#                     else:
#                         a = torch.argmax(pi.probs)
                    return a.item(), pi.probs[a].item()

                a1, prob1 = get_action(model1)
                a2, prob2 = get_action(model2)

                if t % 126 == 0:
                    env.step1(a1)
                next_s, r, done, info = env.step2(a2)
                if t % 126 == 0:
                    model1.store((s, a1, r, next_s, prob1, done))
                model2.store((s, a2, r, next_s, prob2, done))

                s = next_s

                if done:
                    break

            model1.update()
            model2.update()

        if not is_train:
            env.save_csv()
            break

        if i%save_interval==0 and i!=0:
            torch.save({"model1": model1.state_dict(),
                        "model2": model2.state_dict()}, f"save_{i}.pt")
            if killer.kill_now:
                if input('Terminate training (y/[n])? ') == 'y':
                    env.save_csv()
                    break
                killer.kill_now = False


if __name__ == '__main__':
    main()
    main()
```

# 설명

- 실행을 위해서는 simulator(baseline)와 data 파일들을 경로에 위치시켜주는것이 필요합니다.
- 중지를 원하실때는 중지를 누르시고 5step을 기다리셔야 합니다. y를 입력하시면 중지됩니다.
- inference를 위해선 저장된 파일의 이름을 save.pt로 바꾸시고 is_train을 False로 변경하시고 실행하시면 됩니다.

댓글 0개 ▲ 4

NN weonwon123

댓글 올리기

목록으로

| | | | | | |
|---|---|---|---|---|---|
| 이전 글 | **블럭 장난감 제조 공정 최적화 AI 경진대회 일자별 순위 변동 그래프(~6/14)**<br>대회 - 블럭 장난감 제조 공정 최적화 AI경진대회 | 0 vote | 26 views | 댓글 0 | 11시간 전 |
| 현재 글 | **PPO를 활용한 베이스라인**<br>대회 - 블럭 장난감 제조 공정 최적화 AI경진대회 | 4 vote | 13 views | 댓글 0 | 30분 전 |
| 다음 글 | 다음 글이 존재하지 않습니다. | | | | |

개인정보보호정책 이용약관

DACON Inc., 919, 33, Gukjegeumyung-ro 6-gil, Yeongdeungpo-gu, Seoul, KOREA