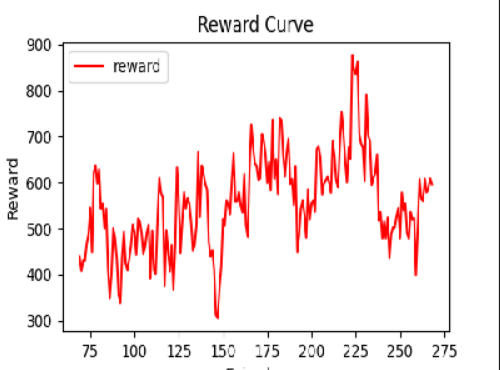
A2C:

gamma=0.99, ent\_coef=0.01,lr\_actor=3e-4,lr\_critic=1e-3

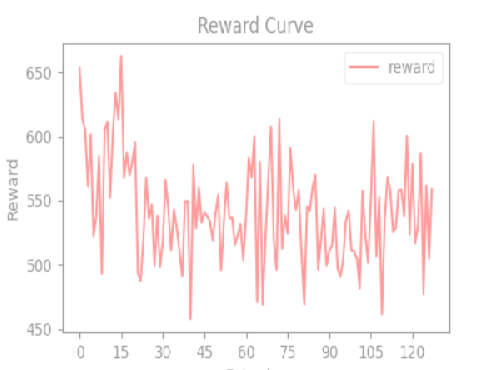
train\_env=StockTradingWrapper(e\_train\_gym)

train\_env=NormalizeObsSingle(train\_env)



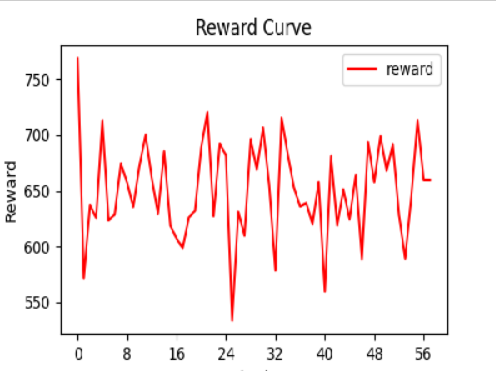
gamma=0.99, ent\_coef=0.01,lr\_actor=3e-4,lr\_critic=1e-3

train\_env=StockTradingWrapper(e\_train\_gym)



gamma=0.99, ent\_coef=0.01,lr\_actor=3e-4,lr\_critic=1e-3

train\_env=e\_train\_gym



gamma=0.99, ent\_coef=0.01,lr\_actor=3e-4,lr\_critic=1e-3

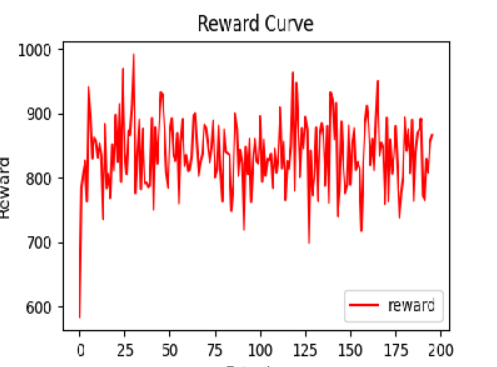
train\_env=e\_train\_gym

self.actor\_optimizer = optim.Adam(self.actor\_net.parameters(), lr=lr\_actor)

self.critic\_optimizer = optim.Adam(self.critic\_net.parameters(),lr=lr\_critic)

self.actor\_scheduler = CosineAnnealingLR(self.actor\_optimizer, T\_max=200, eta\_min=1e-6)

self.critic\_scheduler =CosineAnnealingLR(self.critic\_optimizer, T\_max=200, eta\_min=1e-6)



NAF算法

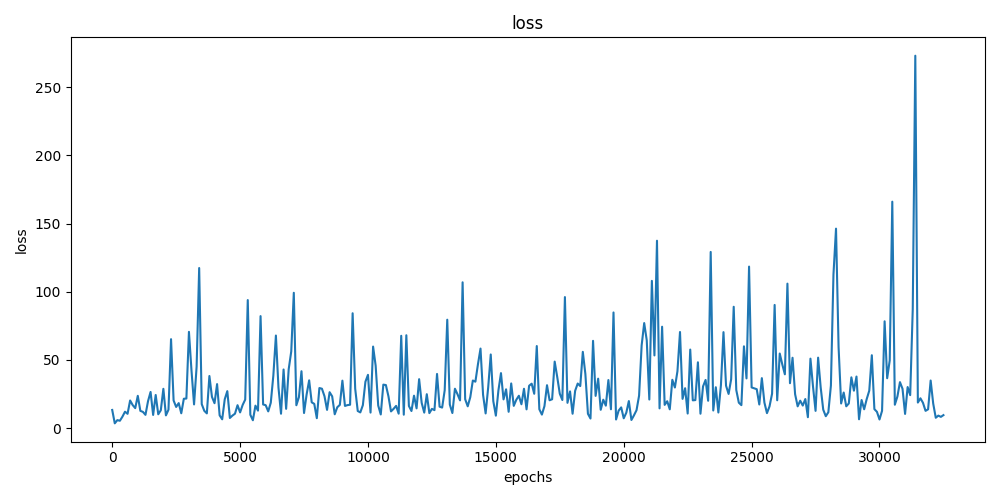
N\_steps=5

self.optimizer = optim.Adam(self.policy\_net.parameters(), lr=2e-4) self.scheduler = CosineAnnealingLR(self.optimizer, T\_max=200, eta\_min=1e-6) target更新是tau=0.001的软更新，已经使用了 ObservationWrapper 在线做 Z-score，Reward Clipping=1e-4

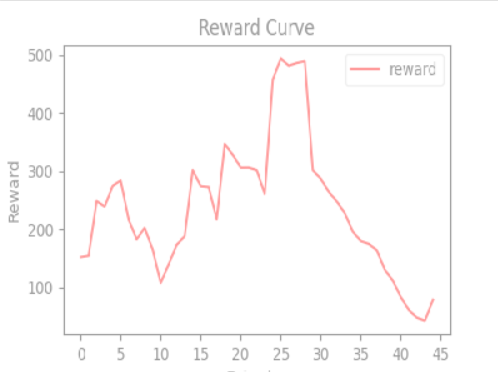
经验回放区:2w

探索T=20w

loss曲线:



训练集收益:



NAF算法

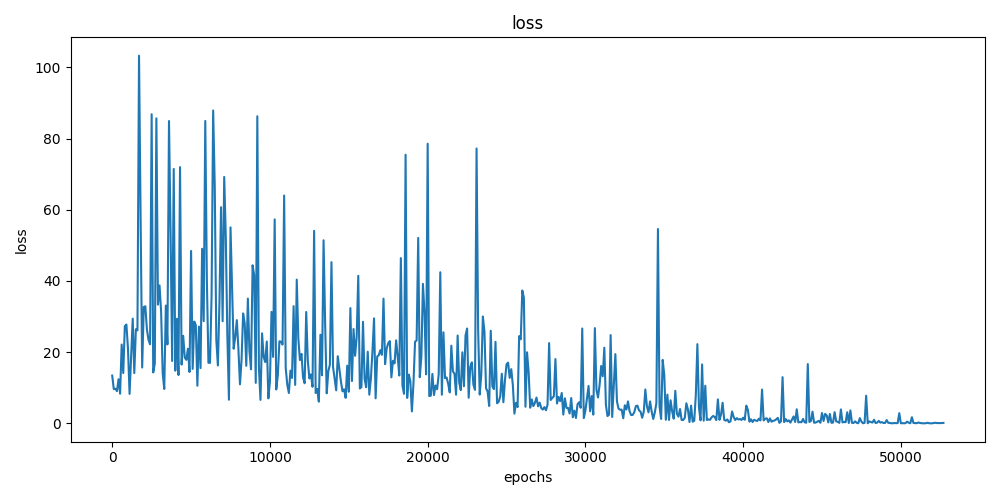
N\_steps=5

self.optimizer = optim.Adam(self.policy\_net.parameters(), lr=1e-4) self.scheduler = CosineAnnealingLR(self.optimizer, T\_max=400000, eta\_min=1e-6) target更新是tau=1e-4的软更新，已经使用了 ObservationWrapper 在线做 Z-score，Reward Clipping=1e-4

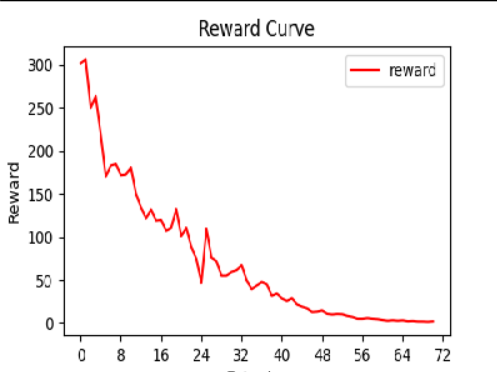
经验回放区:2w

探索T=20w

loss曲线



训练集收益：



NAF算法

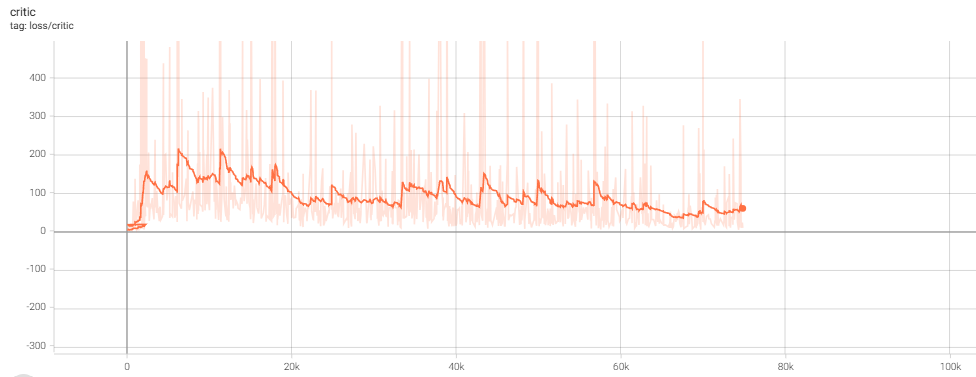
N\_steps=5

self.optimizer = optim.Adam(self.policy\_net.parameters(), lr=1e-4) self.scheduler = CosineAnnealingLR(self.optimizer, T\_max=400000, eta\_min=1e-6) target更新是tau=1e-4的软更新，已经使用了 ObservationWrapper 在线做 Z-score，Reward Clipping=1e-4

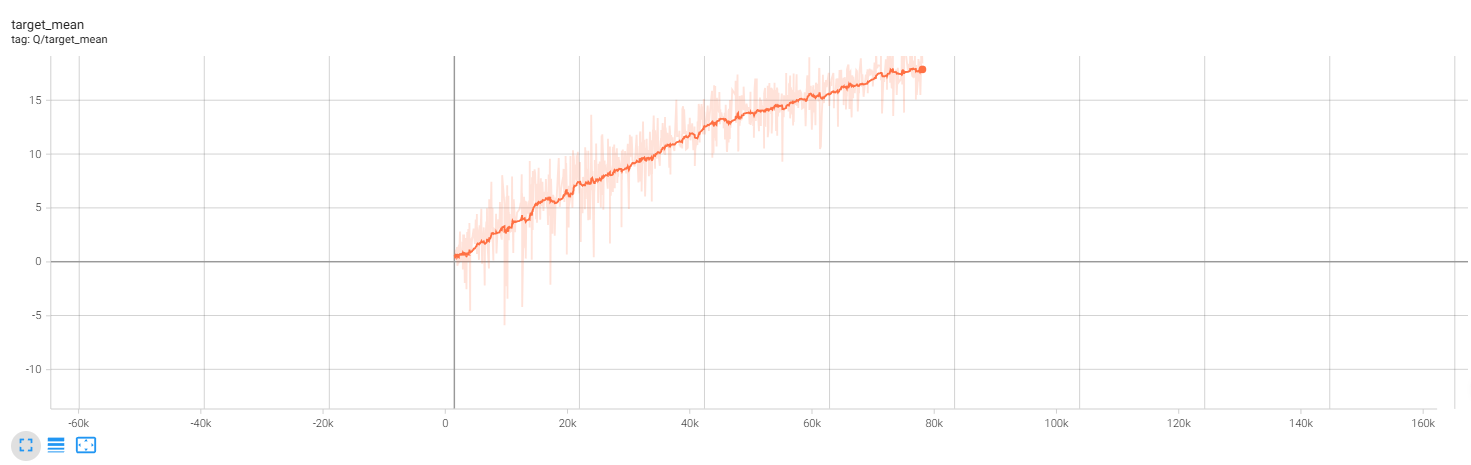
经验回放区:6w

探索T=100w

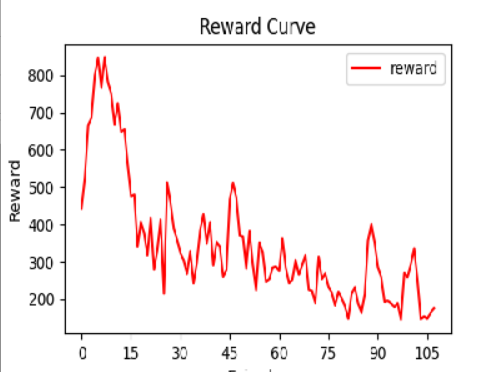
loss曲线:



target平均评估曲线：



Reward曲线：



NAF算法

N\_steps=5

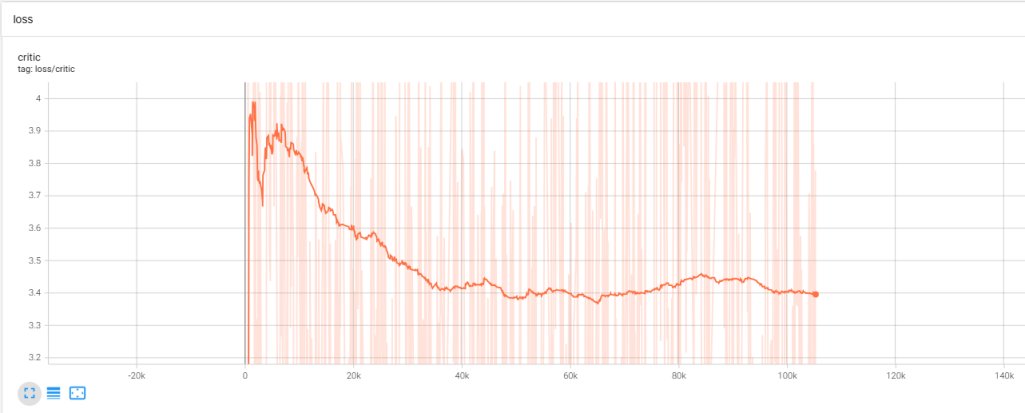
self.optimizer = optim.Adam(self.policy\_net.parameters(), lr=1e-4) self.scheduler = CosineAnnealingLR(self.optimizer, T\_max=400000, eta\_min=1e-6) target更新是tau=1e-4的软更新，已经使用了 ObservationWrapper 在线做 Z-score，Reward Clipping=1e-4

经验回放区:6w

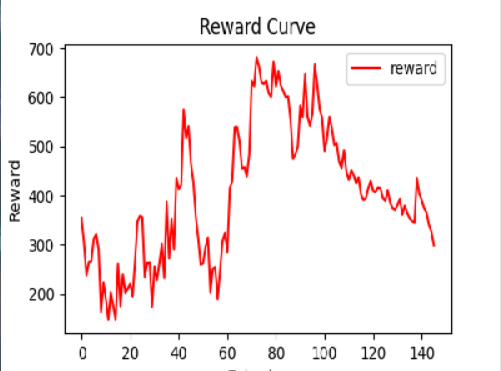
探索T=100w

使用SmoothL1Loss、双q网络

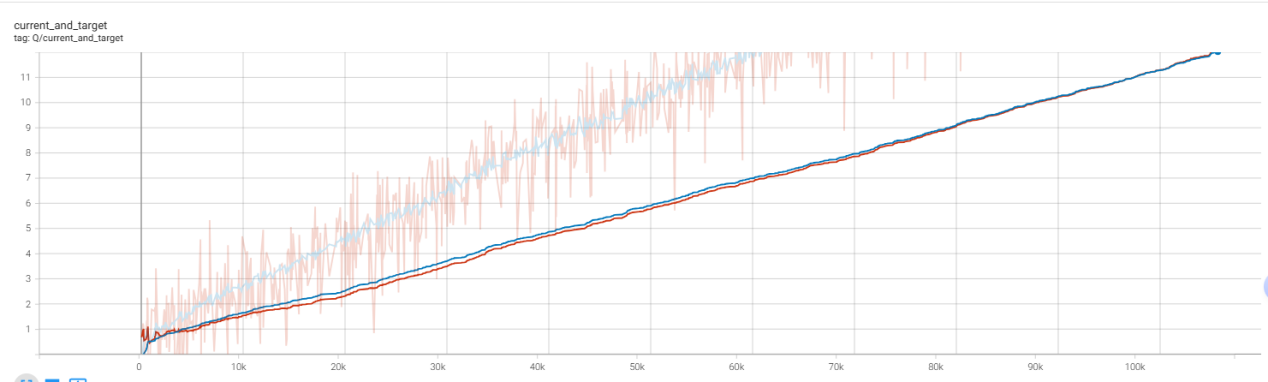
loss曲线:



reward曲线



Current&target(比起之前相同迭代次数确实减少，但是上升趋势没变，越过估计reward就越小）



NAF算法

N\_steps=5

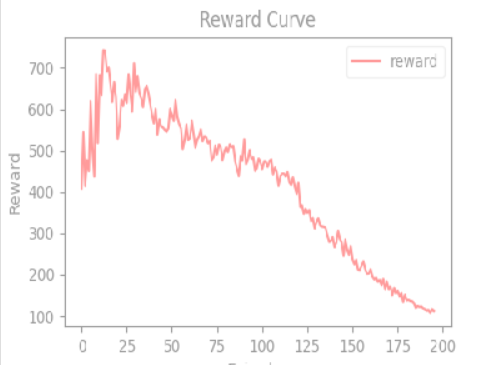
self.optimizer = optim.Adam(self.policy\_net.parameters(), lr=1e-4) self.scheduler = CosineAnnealingLR(self.optimizer, T\_max=400000, eta\_min=1e-6) target更新是tau=1e-4的软更新，已经使用了 ObservationWrapper 在线做 Z-score，Reward Clipping=1e-4

经验回放区:6w

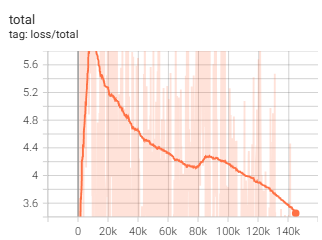
探索T=100w

使用SmoothL1Loss、双q网络+popart

Reward曲线：

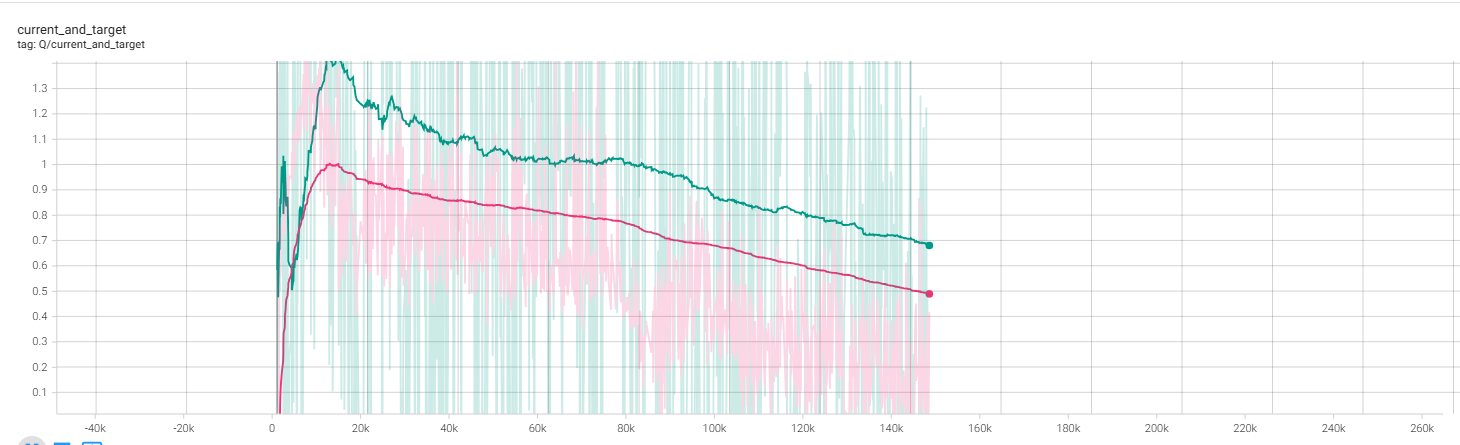


loss:



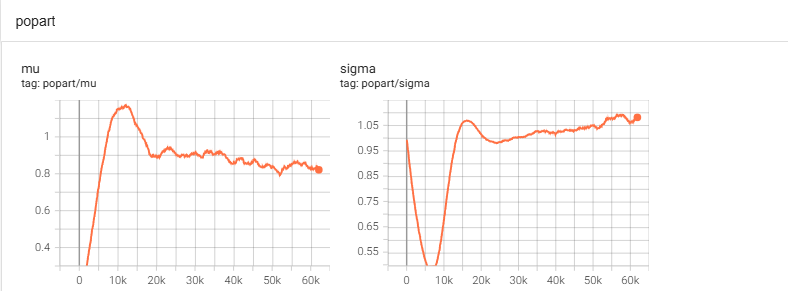
可以看出loss在下降reward也在下降，显然是学到了错误的东西

Current\_and\_target



问题：1.预估值太小了 2.两条曲线间隔太大（需要调整tau）

Popart:



NAF算法

N\_steps=5

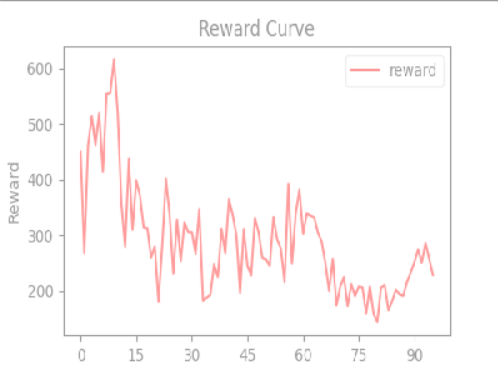
self.optimizer = optim.Adam(self.policy\_net.parameters(), lr=1e-4) self.scheduler = CosineAnnealingLR(self.optimizer, T\_max=400000, eta\_min=1e-6) target更新是每隔1000步同步target网络，已经使用了 ObservationWrapper 在线做 Z-score，Reward Clipping=1e-4

经验回放区:6w

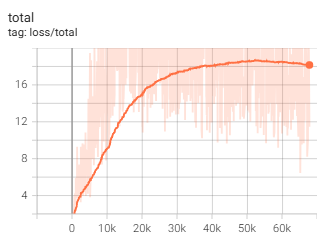
探索T=100w

使用SmoothL1Loss、双q网络+popart

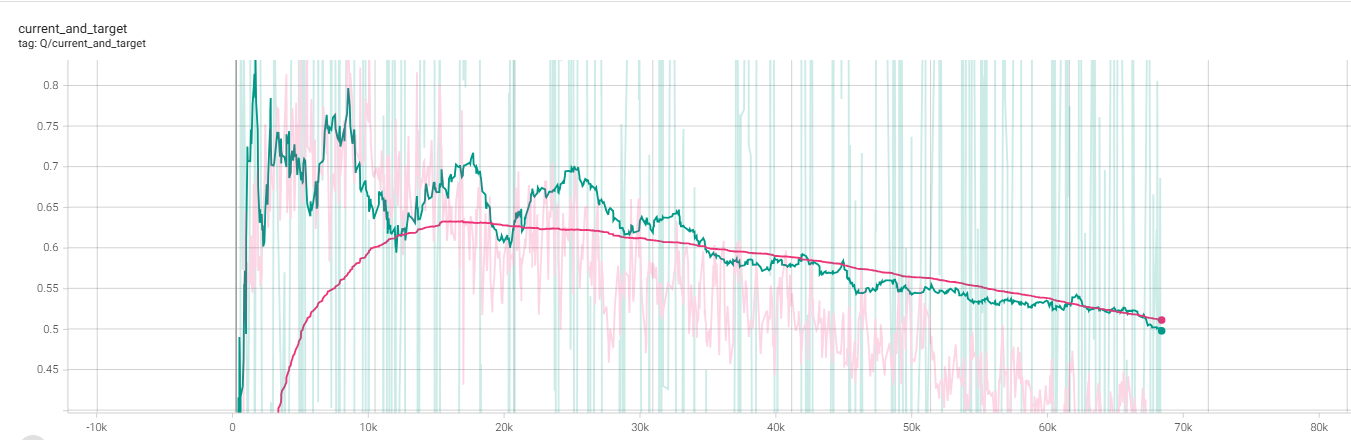
Reward曲线:



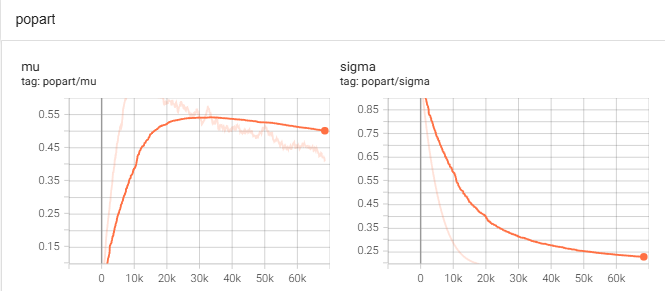
loss:



Current\_and\_target



Popart:



NAF算法

N\_steps=5

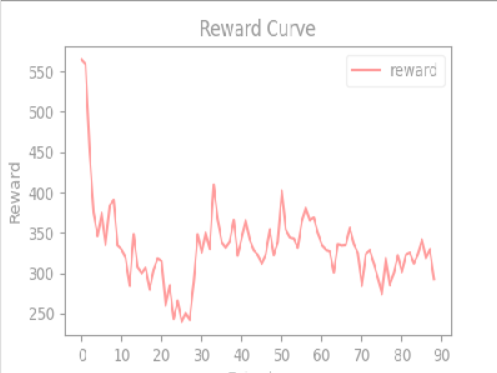
self.optimizer = optim.Adam(self.policy\_net.parameters(), lr=1e-4) self.scheduler = CosineAnnealingLR(self.optimizer, T\_max=400000, eta\_min=1e-6) target更新是tau=1e-3的软更新，已经使用了 ObservationWrapper 在线做 Z-score，Reward Clipping=1e-4

经验回放区:6w

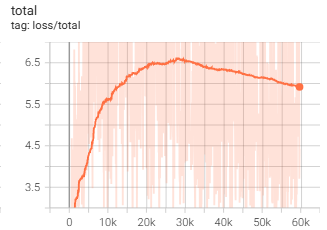
探索T=100w

使用SmoothL1Loss、双q网络+popart

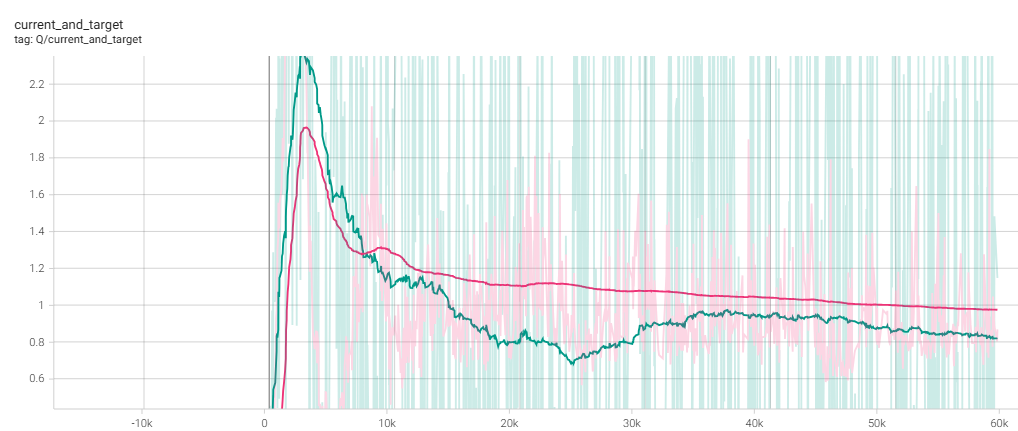
reward曲线:



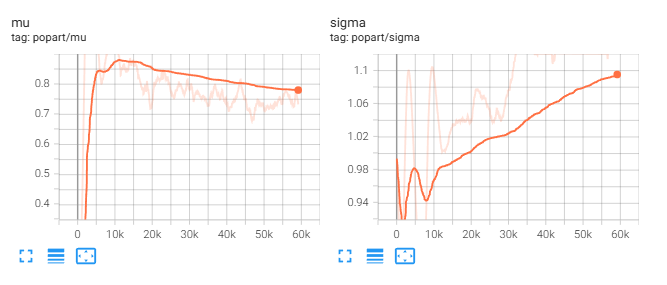
Loss:



Current\_and\_target:



Popart:



NAF算法

N\_steps=5

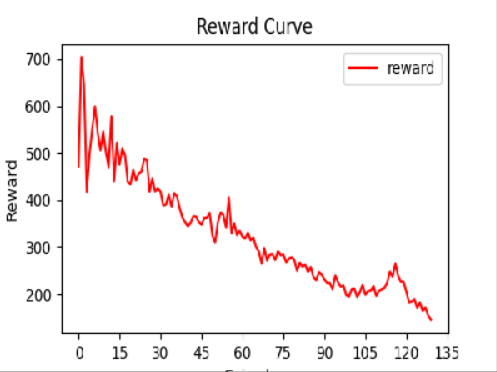
self.optimizer = optim.Adam(self.policy\_net.parameters(), lr=1e-4) self.scheduler = CosineAnnealingLR(self.optimizer, T\_max=400000, eta\_min=1e-6) target更新是tau=2e-4的软更新，已经使用了 ObservationWrapper 在线做 Z-score，Reward Clipping=1e-4

经验回放区:6w

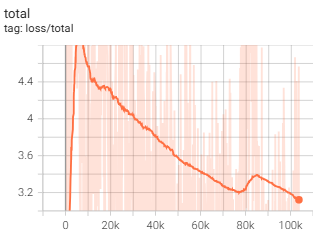
探索T=100w

使用SmoothL1Loss、双q网络+popart

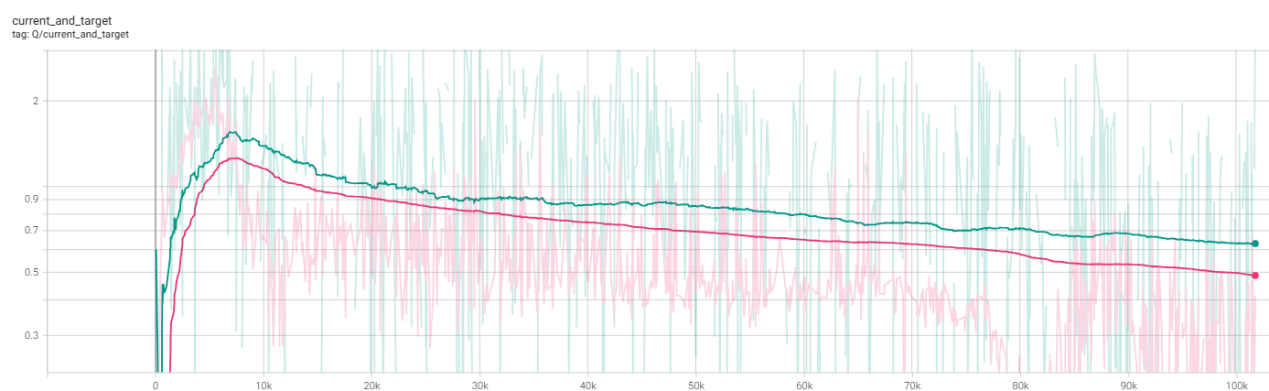
reward曲线：



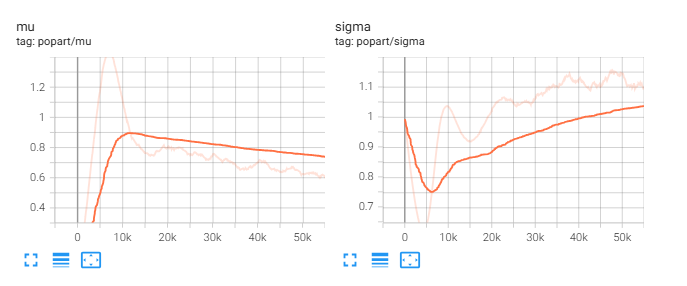
Loss:



Current\_and\_target:



Popart:



NAF算法

N\_steps=5

self.optimizer = optim.Adam(self.policy\_net.parameters(), lr=1e-4) self.scheduler = CosineAnnealingLR(self.optimizer, T\_max=400000, eta\_min=1e-6) target更新是tau=1e-4的软更新，已经使用了 ObservationWrapper 在线做 Z-score，Reward Clipping=1e-4

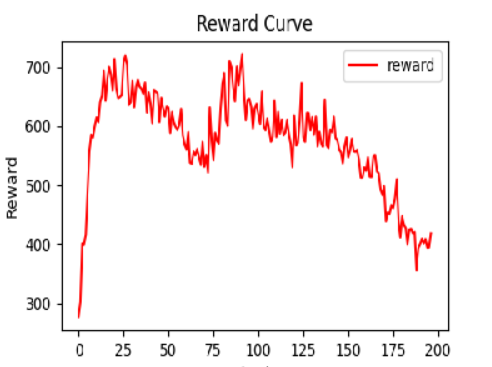
无CQL\_LOSS，双网络分开，Actor网络使用loss=-Q.mean() Critic网络使用原不变。

经验回放区:6w

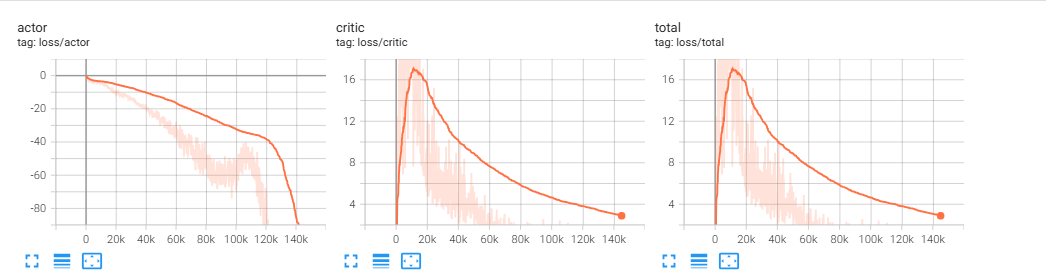
探索T=100w

使用SmoothL1Loss、双q网络+popart

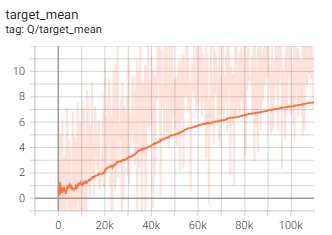
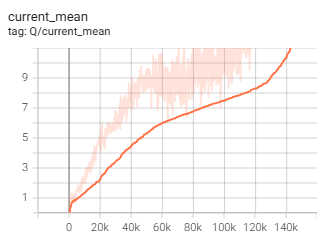
reward曲线：



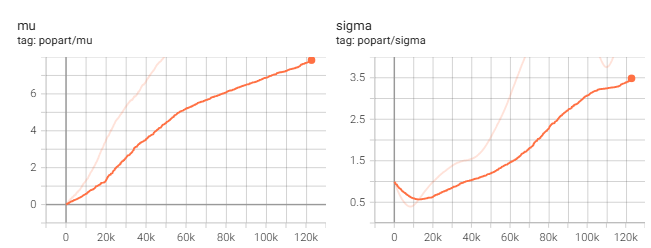
Loss:



Current\_and\_target:



Popart:



NAF算法

N\_steps=5

self.optimizer = optim.Adam(self.policy\_net.parameters(), lr=1e-4) self.scheduler = CosineAnnealingLR(self.optimizer, T\_max=400000, eta\_min=1e-6) target更新是tau=1e-4的软更新，已经使用了 ObservationWrapper 在线做 Z-score，Reward Clipping=1e-4

双网络分开，Actor网络使用loss=-Q.mean() Critic网络使用原不变。

经验回放区:6w

探索T=100w

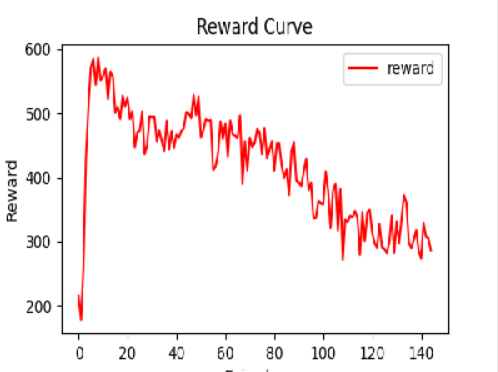
使用SmoothL1Loss、双q网络+popart

cql\_loss:scaler = 1.0 / (self.ema\_max + 1e-6)

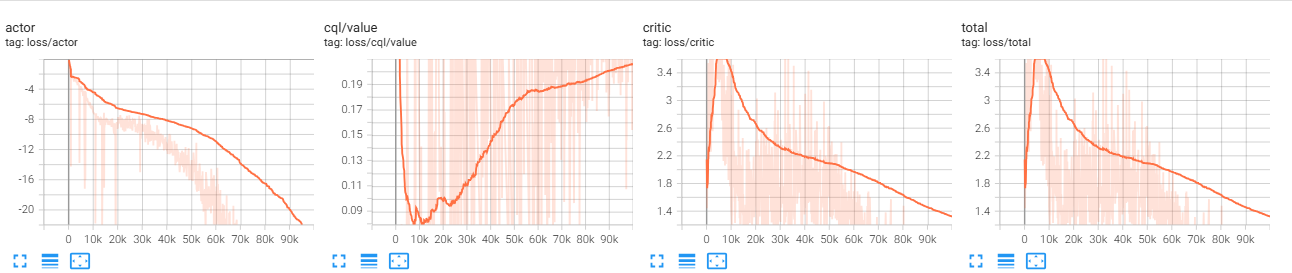
q\_norm = torch.tanh(Q2 \* scaler)

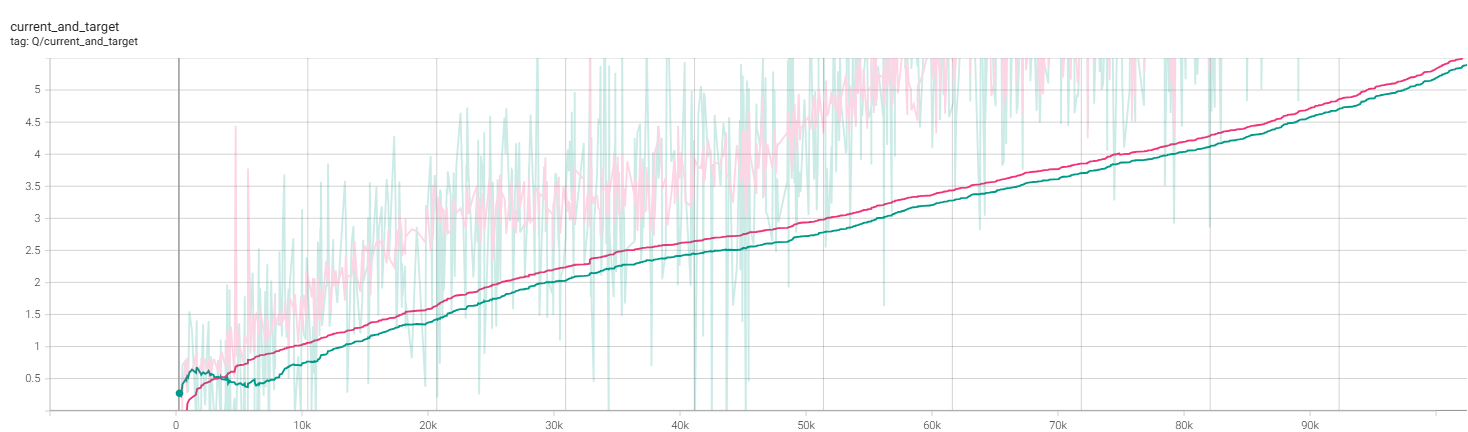
cql\_loss = q\_norm.mean()

Reward曲线:



Loss：



Current\_and\_target:

Popart:

