Urna Semper

Instructor's Name

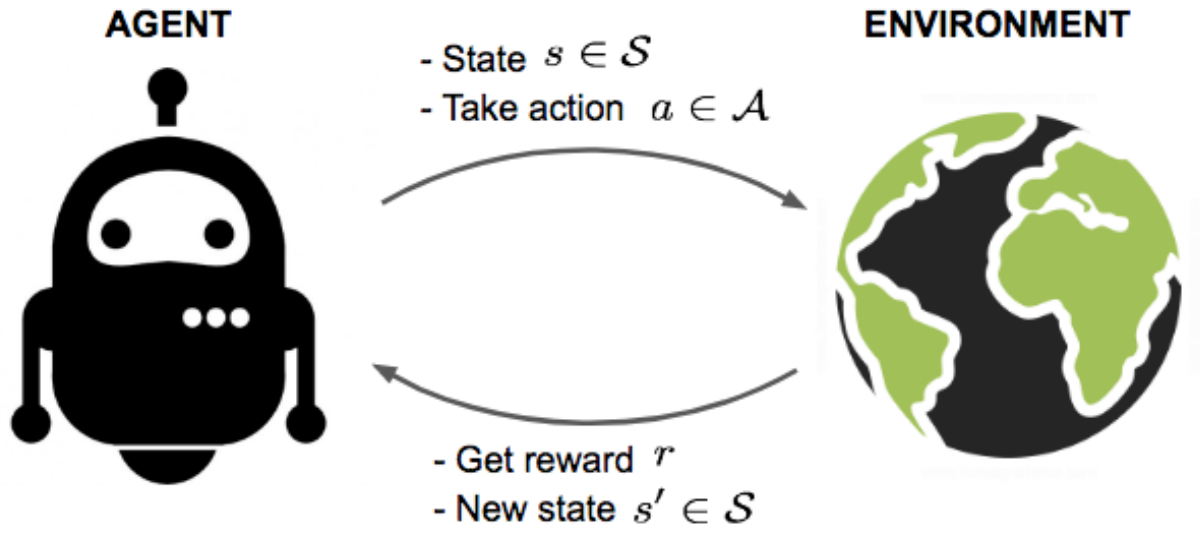
Course Title

November 1, 2020

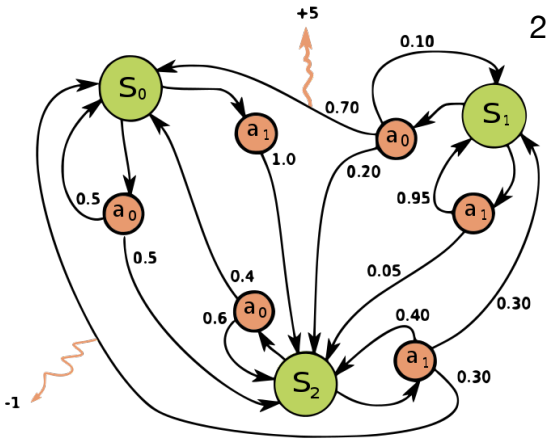
강화학습을 이용한 가상화폐 자동매매 시스템 만들기

Motivation

1. 강화학습이란



강화학습은 에이전트와 환경이 상호작용하는 과정이다. 에이전트는 환경으로부터 상태를 받고 행동을 취한다. 취한 행동을 다시 환경으로부터 보상을 통해 평가받는다. 이 보상을 최대화 시키는 것이 강화학습의 관건이다. 트레이딩에서는 명확한 보상이 주어진다. 바로 "수익" 이다.



2. 강화학습으로 풀기 좋은 문제들

MDP(Markov Decision Process) 는 상태집합, 행동집합, 상태전환확률, 보상함수로 이루어진 Discrete-Time Sequential Decision Process이다. 이산적인 시간에 따른,

순차적인 행동 과정을 모델링하는 방법이다. 대부분의 강화학습 알고리즘이 MDP 혹은 그 변형으로 모델링된 문제를 풀기 위해 만들어졌고 존재한다. 금융에서 트레이딩 의사결정 과정은 MDP로 모델링하기 적합하다.

2. 알고리즘 트레이딩 전성시대

주식시장, 암호화폐시장, 파생상품시장 할 것 없이 트레이딩 봇이 판을 치는 시대이다. 월스트리트의 전문 투자은행인 골드만삭스에는 트레이더 대신 IT인력만이 남아 있다. 앞으로 트레이딩봇에 대한 수요는 더 많아질 것으로 예상한다.

Goal

1. 강화학습을 이용해 전략 개발, 전략 검증, 실거래가 가능한 시스템을 만들고 사용한다.
2. 기타 AI 기술이 트레이딩에 어떻게 적용되는지 배운다.
3. 가능하다면, 빗썸 거래소를 이용해 소액 넣고 트레이딩을 해본다.

Project Structure

1. Code Structure

callbacks.py: 트레이닝 시 성능 향상 기록을 위한 콜백

config.py: 하이퍼파라미터 딕셔너리들이 담겨 있는 파일

env.py: 환경과 시뮬레이터 클래스

evaluate.py: 학습된 모델을 과거 데이터에 대해 테스트

imports.py: 라이브러리 빠른 임포트

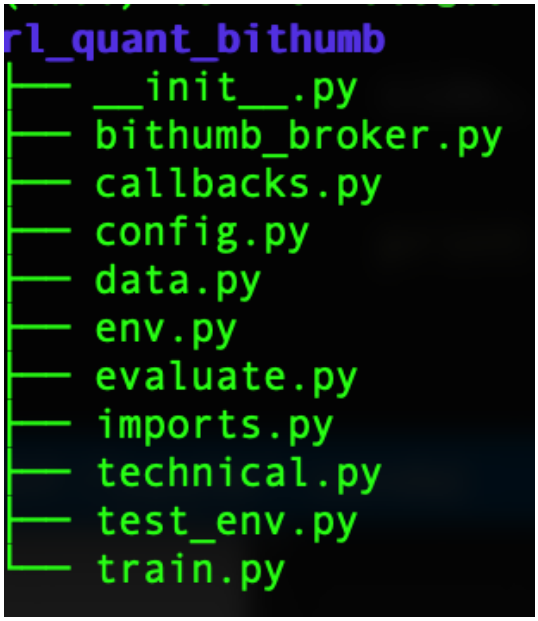
test_env.py: 환경 테스트(바이앤홀드)

train.py: 훈련

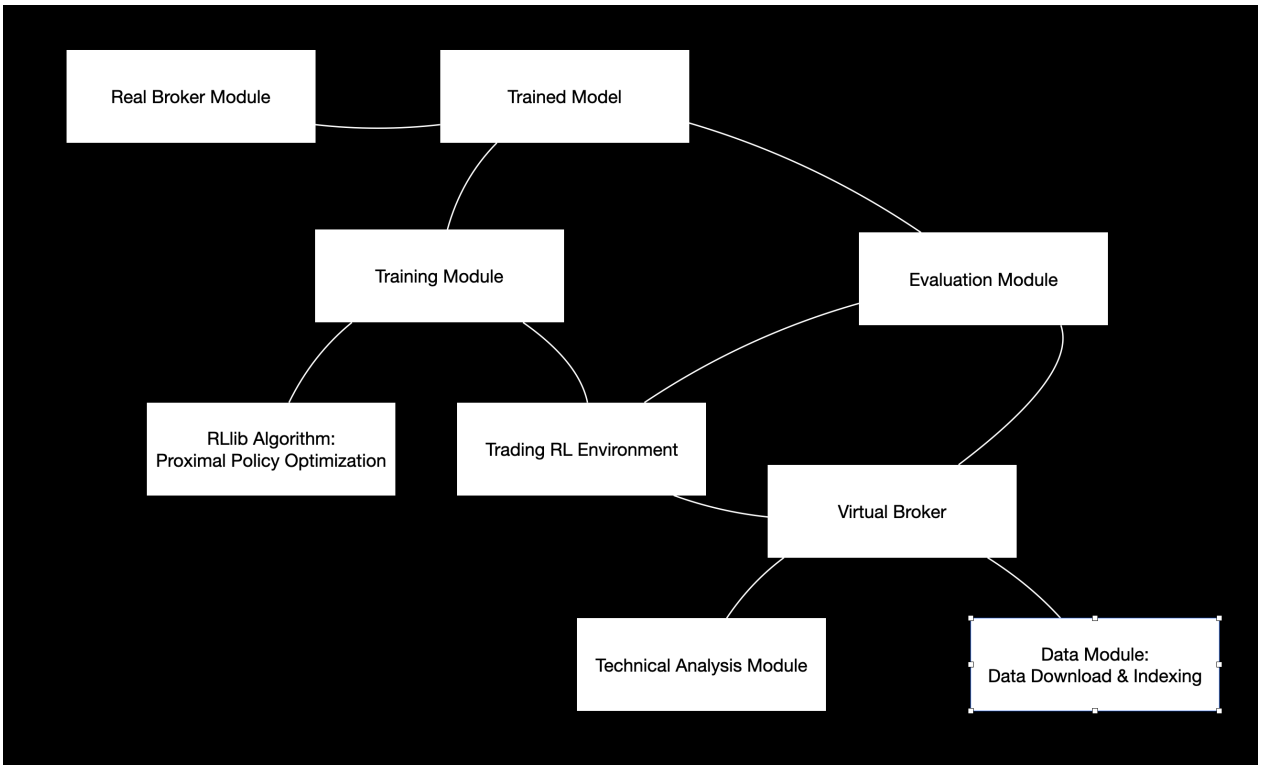
bithumb_broker.py: 빗썸 거래소 통해 거래하는 모듈

technical.py: 기술적 지표 구현체

data.py: 금융 데이터 인덱싱/자료구조 재편



2. Project Diagram



The Problem Formulation

0. Formulation

POMDP(Partially Observable Markov Decision Process)를 가정하고 모델링한다.

1. Define Observation

딥 러닝 모델은 raw 데이터 속에서도 일정한 패턴을 찾아내는 능력이 있지만, 모든 데이터에서 그렇지 않고, 찾는다고 하더라도 매우 느리다. 사람이 도메인에 대한 지식으로 feature들을 뽑아내서 줄 필요가 있다. 따라서 이 프로젝트에서도 raw candlestick data를 그대로 사용하는 것이 아닌 Technical Indicators(기술적 지표)를 입력으로 줄 것이다.

RSI 지표

$$RSI = 100 - \left(\frac{100}{1 + \left(\frac{\overline{A}}{\overline{B}} \right)} \right)$$

$\overline{A}$ = Average Gain

$\overline{B}$ = Average Loss

RSI(Relative Strength Index)는 주식이 얼마나 과매도/과매수되었는지를 나타내어 주는 오실레이터 지표이다. 0~100 사이의 값으로 나타난다. 값이 너무 크면 딥러닝 모델에서 오버플로 문제가 생길 수 있으므로 100을 나누어 0~1 사이의 값으로 만든 후 넣어준다.

ROC 지표

(종가 - n일전종가) / 종가로 나타내는 인디케이터이다. 단순히 현재 종가가. n일 전에 비해 얼마나 상승했는지를 나타내어 준다.

MA 지표

n일간 종가의 평균으로 나타내어지는 비교적 간단한 값이지만 이 값을 돌파, 지지하는 등 많은 현상들을 이끌어 낼 수 있다. 한번 더 정제하여 현재 종가와 MA간의 괴리율, 즉 (Close - MA(20))/Close을 상테에 넣어준다.

RETSTD 지표

전일종가대비 현재종가 (%) 의 n일간 표준편차를 나타내는 지표이다. 주식의 최근 변동성을 나타내어 익절/손절의 라인을 유연하게 잡는 데 쓰이기도 한다.

최종 상태는 다음과 같은 4차원 벡터가 된다:

[(Close - MA(20))/Close, RSI(14) / 100, RETSTD(20), ROC(20)]

2. Define The Reward Function

포트폴리오의 변동(%)를 리워드로 준다. 즉, (현재 평가액 - 전일 평가액) / 현재 평가액 으로 구한다. 가장 직관적인 리워드이다. 강화학습에서는 “수익” 을 극대화하는 것이 목적이기 때문이다.

강화학습에는 global optima와 local minima가 있다. 전역 최적점과 국소 최적해라는 뜻이다. 우리는 global optima 혹은 global optima에 가까운 local minima에 도달하는 것이 학습의 목표이다. Local minima에 빠지는 예시로는 이런 것이 있다. 자동차 등이 경로를 찾아가는 강화학습 문제에서 경로를 벗어났을 때 너무 과도한 패널티를 주게 되면, expected reward가 너무 낮아져 그냥 장애물을 받아버리고 에피소드를 끝내는 local minima로 수렴하는 것을 볼 수 있다.

만약 이렇게 구한 리워드가 0 이하라면 소량의 default reward를 빼준다. 만약 default reward가 없는 상황을 생각해보자. 만약 트레이딩을 하지 않는다면 아무 보상도 벌도 받지 않으므로 계속 0의 리워드를 받을것이다. 하락장에는 exploration을 위해 랜덤으로 액션을 취했을 때 음의 리워드를 받을 가능성이 더 높다. 그럴 때 에이전트는 아무 액션도 하지 않는 local minima로 빠지게 된다. 그러므로 이것을 방지하기 위해 음의 default reward를 넣어준다.

Training The Model

1. command line

```
python train.py \
      –logdir LOGDIR \
      –ticker TICKER \
      –total_episodes TOTAL_EPISODES \
      –checkpoint_freq CHECKPOINT_FREQ \
      –restore RESTORE_CKPT
```

LOGDIR: 학습 정보/텐서보드를 저장할 디렉토리

예) ~/results/

TICKER: 학습 데이터로 사용할 코인 명

예) BTC

TOTAL_EPISODES: 학습에 사용할 에피소드 수

예) 20000

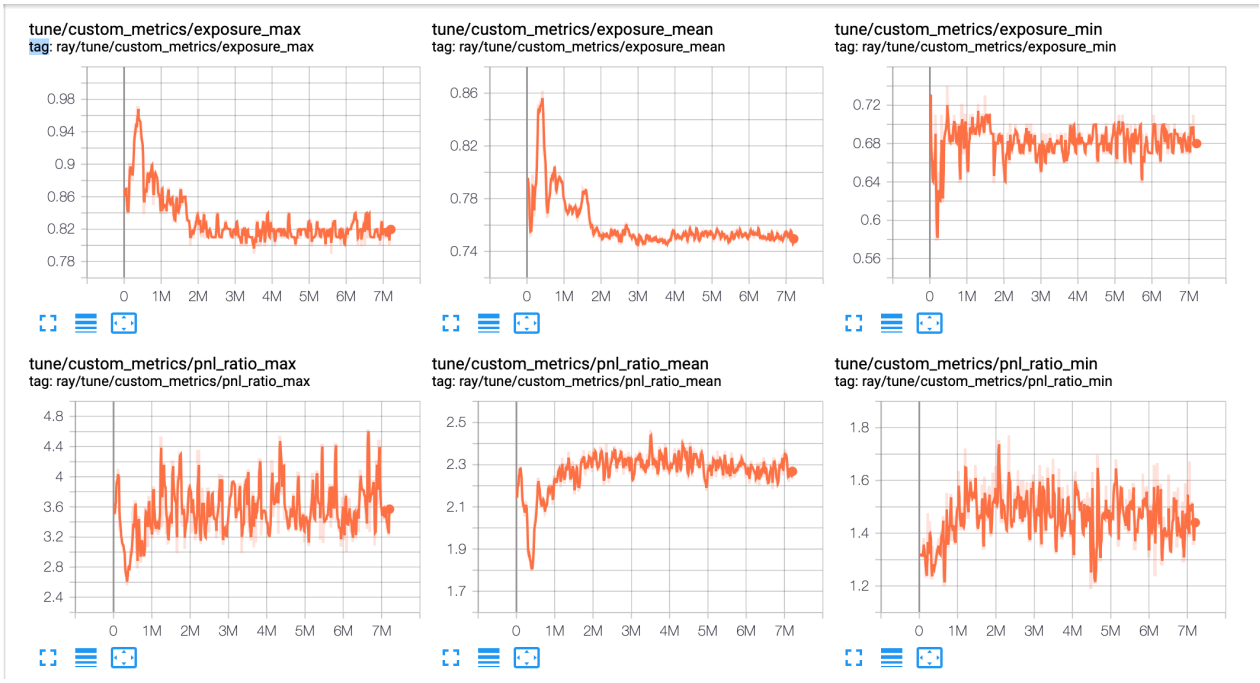
CHECKPOINT_FREQ: 체크포인트 빈도

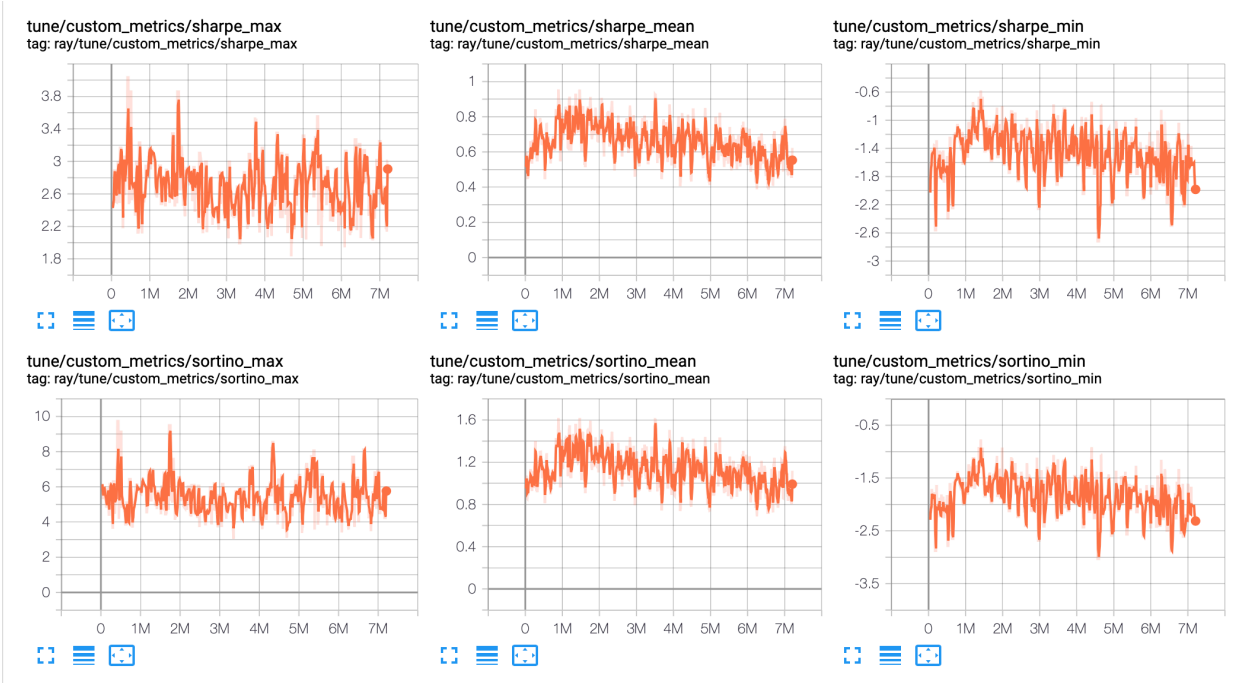
예) 100

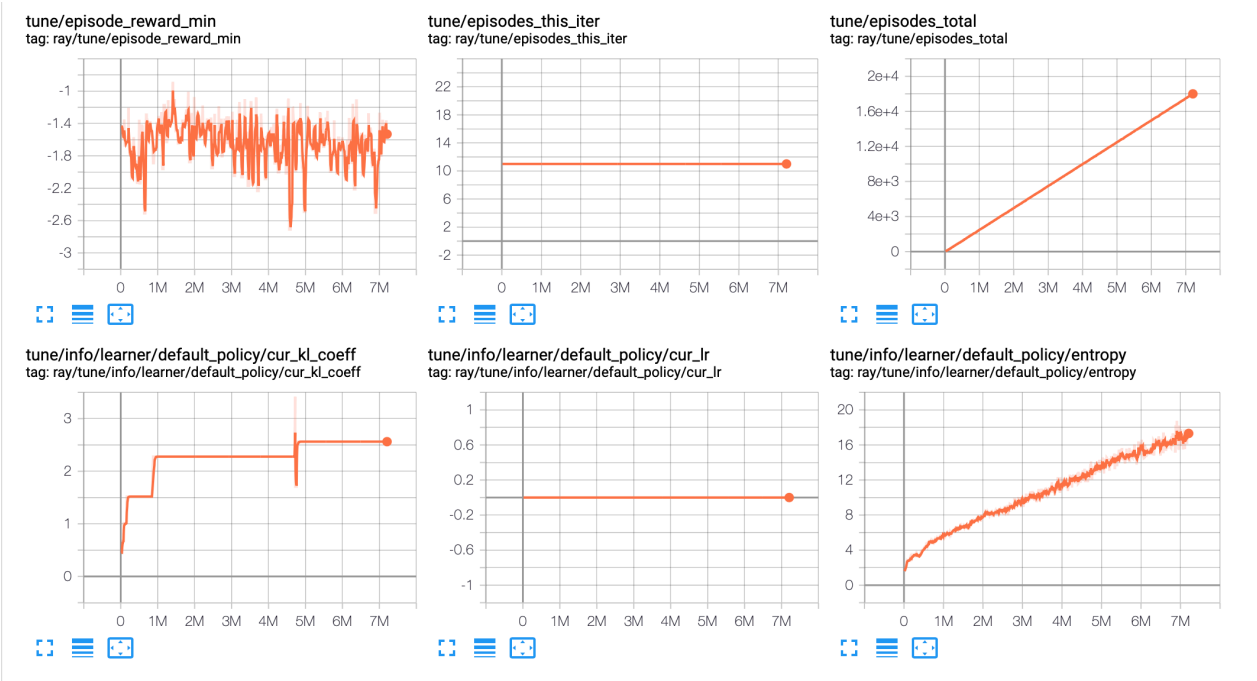
RESTORE_CKPT: 이전 학습 결과 체크포인트

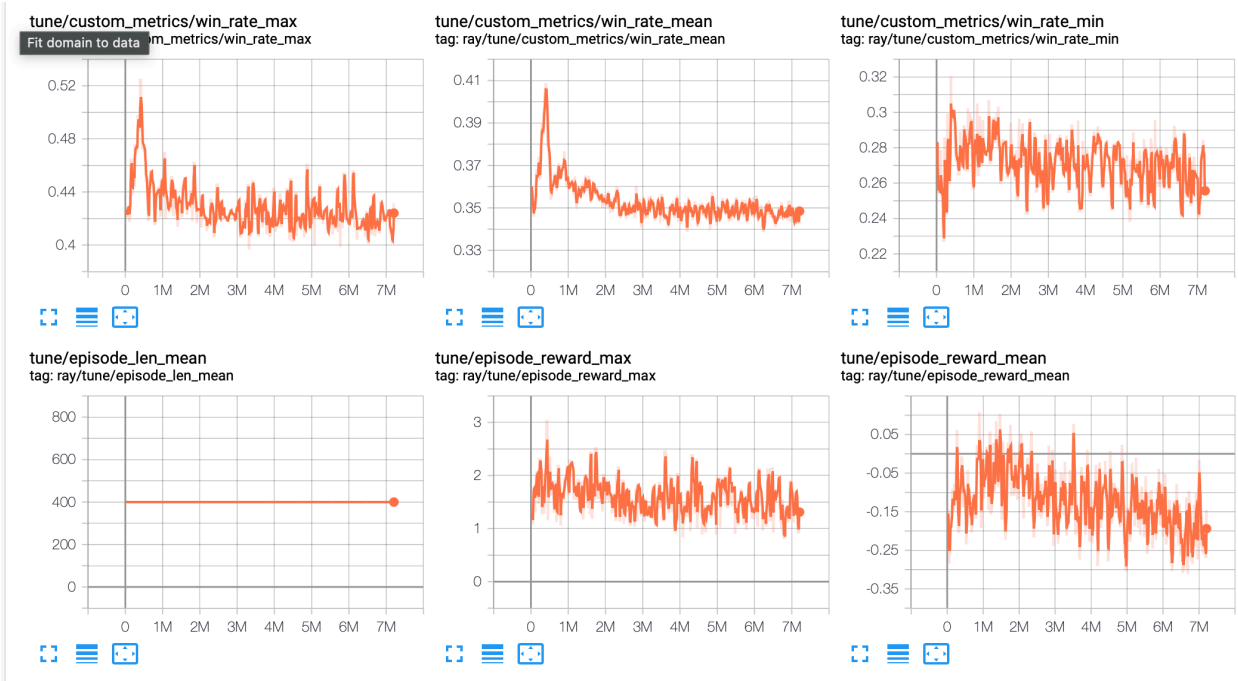
예) ray_results/PPO/PPO_SimpleTradingEnv_4093f_00000_0_2020-11-01_03-02-34/checkpoint_1600/checkpoint-1600

2. Training Result









손익비, 승률, 소르티노/샤프 지수 등 금융 메트릭과, 엔트로피/오차함수 값 등 여러 가지 학습 정보들을 텐서보드에서 살펴볼 수 있다.

Evaluating The Model

1. Command line

```
python evaluate.py \
        –render False \
        –ticker TICKER \
        –restore RESTORE_CKPT
```

TICKER: 평가 데이터로 사용할 코인 명

예) BTC

RESTORE_CKPT: 이전 학습 결과 체크포인트

예) ray_results/PPO/PPO_SimpleTradingEnv_4093f_00000_0_2020-11-01_03-02-34/checkpoint_1600/checkpoint-1600

RENDER: 실시간 렌더링 여부

예) False

2.Result

실행 결과 html 파일이 나온다.

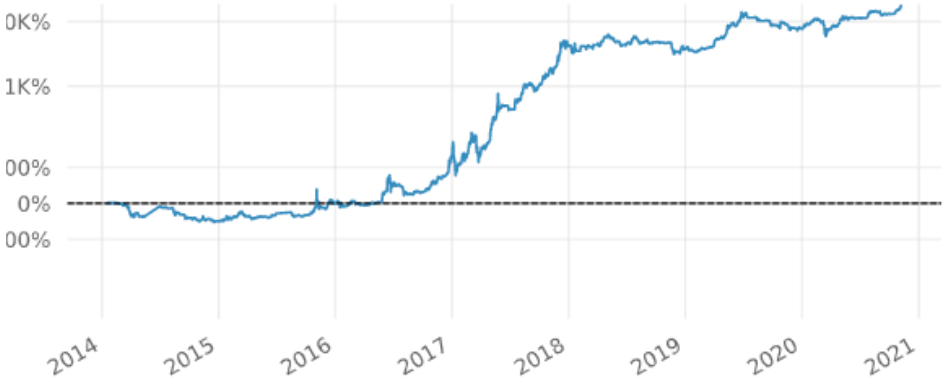
학습한 코인에 대해 한 번, 학습하지 않은 코인에 대해 한 번 평가해준다. 아래 예시에서는 BTC에 대해 학습하고, ETH에 대해 평가한다. 바로 과적합을 방지하기 위해서다. 인공 신경망의 단점 중 하나는 신경망의 층이 깊어 일반화 성능을 빠르게 높일 수 있지만, 동시에 과적합될 확률도 올라간다는 것이다. 그러므로 우리는 에이전트가 정말 일반적인 트레이딩의 룰을 학습했는지 학습에 사용되지 않은 데이터를 이용해 평가할 필요가 있다.

만약 학습한 코인에 대해서만 결과가 잘 나오고 다른 코인에서는 그렇지 않다면, 모델이 과적합되었다고 평가할 수 있다. 모델이 과적합되었다면 이전 체크포인트를 사용해 다시 평가해보는 방법이 있다. 아래 예시는 두 모델 모두 Buy&Hold에 비해 실적이 잘 나왔으니 과적합되지 않았다고 평가할 수 있다.

<BTC 학습, BTC 평가>

Cumulative Returns

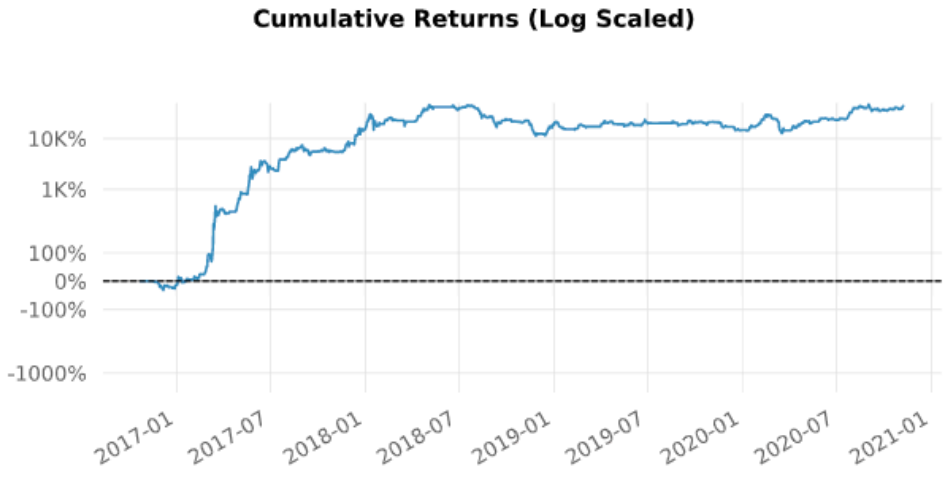
Cumulative Returns (Log Scaled)



Key Performance Metrics

Metric	Strategy
Risk-Free Rate	0.0%
Time in Market	87.0%
Cumulative Return	17,609.81%
CAGR%	113.9%
Sharpe	1.26
Sortino	2.05
Max Drawdown	-57.16%
Longest DD Days	623
Volatility (ann.)	55.07%
Calmar	1.99
Skew	1.54
Kurtosis	22.83
Expected Daily %	0.22%
Expected Monthly %	6.6%
Expected Yearly %	109.49%
Kelly Criterion	13.43%
Risk of Ruin	0.0%
Daily Value-at-Risk	-5.43%
Expected Shortfall (cVaR)	-5.43%
Payoff Ratio	1.19
Profit Factor	1.34
Common Sense Ratio	1.6
CPC Index	0.84
Tail Ratio	1.19
Outlier Win Ratio	6.54
Outlier Loss Ratio	4.61

<BTC 학습, ETH 평가>



Key Performance Metrics

Metric	Strategy
Risk-Free Rate	0.0%
Time in Market	79.0%
Cumulative Return	43,774.56%
CAGR%	351.11%
Sharpe	1.66
Sortino	2.91
Max Drawdown	-75.23%
Longest DD Days	849
Volatility (ann.)	82.7%
Calmar	4.67
Skew	1.39
Kurtosis	12.92
Expected Daily %	0.41%
Expected Monthly %	12.94%
Expected Yearly %	237.63%
Kelly Criterion	16.83%
Risk of Ruin	0.0%
Daily Value-at-Risk	-8.03%
Expected Shortfall (cVaR)	-8.03%
Payoff Ratio	1.47
Profit Factor	1.5
Common Sense Ratio	2.1
CPC Index	1.11
Tail Ratio	1.4
Outlier Win Ratio	7.31
Outlier Loss Ratio	4.3

Trade With Real Money

0. Warning

Bithumb API를 이용해 실제 돈으로 트레이딩을 해본다. 절대 권장하지 않지만 추후 트레이딩 모델을 실제 시장에 적용하는 방법을 알기 위해 코드를 만들었다. 현재 학습한 모델은 end-to-end 모델로써 해석력이 떨어진다. 벌어도 왜 벌었는지 모르고, 잃어도 왜 잃었는지 모른다. 한마디로 리스크관리가 안된다. 이를 해결하기 위해 주로 meta-labeling이라는 방법을 실제 모델에서는 쓰게 된다. 그리고 모델에 들어가는 정보가 매우 간단화되어 있다. 실제 트레이딩 모델은 더 많은 정보를 더 정교하게 받아야 할 것이며, 더더욱 엄밀한 백테스팅과 검증 과정을 거쳐야 한다.

1. command line

```
python real_trade.py \
--log_interval INTERVAL \
--conkey CONKEY \
--seckey SECKEY \
--target_currency CURRENCY
```

INTERVAL: 평가액 로깅 빈도(초)

CONKEY: 빗썸 API CONKEY

SECKEY: 빗썸 API SECKEY

CURRENCY: 거래대상자산

Conclusion & FurtherMore

비교적 간단한 강화학습 모델을 이용해 시장을 이기는 에이전트를 만들 수 있었다. 그러나 아직 넘을 산이 많다. 첫 번째로는 정교한 feature extraction이 필요할 것이다. 인간이 가장 많이 고민해야 하는 것 중 하나이다. 강화학습 에이전트에게 어떤 시각으로 시장을 보아야 하는지 알려주는 지표이다. 두 번째는 explainability이다. 수익률만 잘 나오면 그만이라고 생각할 수도 있지만, 그렇지 않다. 효율적 주문집행, 상/하방 리스크 관리, 자산군 선정 등 금융시장은 매우 많은 것이 고려되어야 하는 복잡계이다.

이 프로젝트에는 몇 가지 개선할 점이 있다. 첫 번째는 explainability 문제이다. 에이전트가 특정한 자리에서 왜 팔았는지, 왜 샀는지 알려주지 않는다. Layer-Wise Propagation 같은 feature importance analysis를 수행하거나, bet-sizing 등의 부분적 문제만 에이전트에게 풀게 하는 방법이 있다. 또한 over-fitting 문제를 해결하기 위해 직접 학습된 모델을 적용해보며 해결하기보다, Cross-Validation이나 각종 regularization 방법론을 적용할 수 있다. 마지막으로 모델이 학습에 따라 너무 다른 양상을 보이는 경향을 해결하기 위해 ensemble 방법론들을 적용할 수 있겠다.

앞으로 강화학습을 금융시장에 적용하는 방법도, 효과적인 방법도 많을 것이다. 이 프로젝트는 수많은 방법들 중 가장 간단하고 직관적인 방법을 보여준다. 이 프로젝트를 기반으로 강화학습을 금융시장에 적용하는 새로운 방법론들을 많이 찾아내고 개발할 수 있을 것이다.