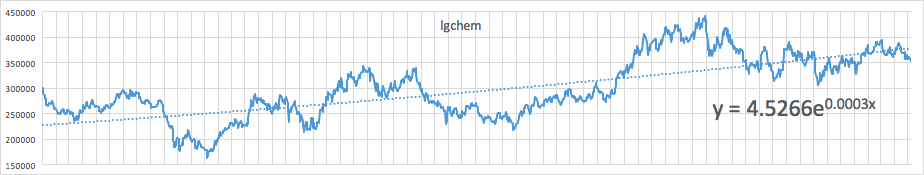
**데이터기반학습 Project 4 보고서**

5조 정유민, 문현지, 한태구

**1. Data Analysis / Background Research**

주어진 약 10년의 데이터 중 과거의 데이터는 거래에 미치는 영향이 적을 것이라는 가정 하에 2014년 1월 1일부터 약 5년의 데이터를 사용하기로 했다. 주어진 10개의 기업에 대하여 아래와 같이 추세 그래프를 그려본 결과, 4개의 기업(신한, 현대모비스, 현대자동차, 포스코)의 이자율은 현저히 낮았으며, 나머지 6개는 약 0.0003 ~ 0.001의 이자율을 보였다. 따라서 해당 6개 기업(셀트리온, LGH&H, LG케미컬, SK, 삼성전자1, 삼성전자2)을 선택하기로 하였다. 또한, 추후 사용할 모델의 *gamma*를 1/(1+0.0004)로 설정하기로 했다.



본 과제에서 사용되는 DQN 관련 논문들 중, 2017년에 발표된 “Rainbow: Combining Improvements in Deep Reinforcement Learning”에서는 기존에 발표된 6가지 DQN 알고리즘을 모두 적용하여 state-of-art를 보였다. rainbow 모형은 기존 6개의 dqn을 혼합하여 사용하는 모형으로 본 과제에서는 이 중 Double DQN, Dueling network architecture, Multi-step bootstrap targets, Noisy DQN을 이용했다. 각 모형에 대한 간단한 설명은 다음과 같다.

Double DQN은 Q값을 target과 evaluation을 분리하여 학습하고, Dueling network은 Q를 value와 advantage 2개 성분으로 나누어 학습하며, multi step learning은 하나의 reward를 받은 후 bootstrap을 위해 다음 단계에서 greed action을 하는 Q-learning과 다르게 향후 n-step 의 reward를 bootstrap하는 방식, distributional RL은 q를 분포로 학습하고 다음 step의 분포를 바탕으로 current guess에 활용하는 방식이다. Noise net은 epsilon에 noise를 주며 학습하는 방식이다.

**2. Feature Engineering**

**(1) Data preprocessing**

기본적인 Open이나 Close 값 이외에도 다음과 같이 가공된 feature를 생성하여 사용하였다. Open을 사용한 지표의 경우 Open을 사용했다는 말이 생략되어 있다. 또한 MA, EMA, SMA, MACD의 경우 하락세와 상승세를 discrete하게 변환한 지표도 같이 후보로 고려하였다.

|  |  |  |  |
| --- | --- | --- | --- |
| MA, EMA | 10일 간 이동평균, 지수이동평균 | BNF | 이동평균선 괴리율 |
| MA5 (Close, Volume) | 5일 간의 Close, Volume 이동평균 | MACD | (12일 이동평균) - (26일 이동평균) |
| H/C, L/C ratio | (당일 High, Low) / (당일 Close) | O/last\_C ratio | (당일 Open) / (전날의 Close) |
| Volume ratio | (당일 Volume) / (전날 Volume) | Momentum Oscillator | (당일 Open) / (n일 전 Open) |

**(2) Normalization**

위의 feature들에 대하여, 이전 input\_days일 만큼의 데이터를 당일의 features로 사용하게 된다. 이 때, 비율이 아닌 feature들은, 아래와 같이 마지막 날의 값으로 정규화하여 사용하기로 했다.

*normalized-feature* t = ( *feature* t-input\_days : t-2 – *feature* t-1 ) / *feature* t-1

**(3) Data cleaning**

누락된 데이터가 거의 없었기에, 누락된 데이터에 대하여 dropna()를 수행하였다.

**3. Modeling**

**(1) Q-Network**

“Playing Atari with Deep Reinforcement Learning”에 따르면 기존 DQN은 두 가지 문제점을 가진다. 첫째는 experience sample 간의 상관성이 크다는 점으로, 이는 experience = (current state, action, reward, next state)를 replay memory에 저장한 뒤 batch\_size 만큼을 임의로 선택하여 학습에 이용하는 방식으로 해결할 수 있다. 둘째, 최종 학습목표인 Q network 가 고정돼 있지 않다는(nonstationary target) 문제이다. 이는 target-Q를 따로 만들어 주기적으로 업데이트 하는 Double DQN (DDQN)으로 해결 가능하다.

여러 DQN 중 최종선택된 DQN은 **Distributional Dueling DDQN**으로, 이에 대한 구현은 Background Research에서와 같다. 모형 선택 과정에서 특기할 만한 사항은 Distributional Network의 형태에 따른 모수, 즉 atom이다. 기존에 (action의 수)였던 output의 차원이, 분포 형태가 됨에 따라 (atom의 수) X (action의 수)로 증가한다. 즉, 어떤 action이 어떤 q-value를 가질 확률을 uniform 분포를 따르는 atom에 저장하고, 그 기대값을 최종 q-value으로 쓰는 것이다.

**(2) State**

기본적으로, DataGenerator를 통해 만들어진 Features가 State이다. 여기에, t시점의 open\_price나 budget, 혹은 num\_stocks를 추가할 수 있다. Budget과 num\_stocks는 같은 맥락에 있으므로, 다음의 네 가지 경우로 나누어 볼 수 있다. 이 중 테스트를 통해 case0을 사용하였다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Case | 0 | 1 | 2 | 3 |
| State | Features | + open\_pice | + budget + num\_stocks | + open\_pice + budget + num\_stocks |

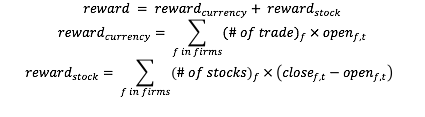
**(3) Action**

*action* & *do\_action()* : 선택한 6개의 기업에 대하여, 6개의 DQN모델을 따로 만들어 학습하는 방식과 하나의 모델로 동시에 학습하는 방식이 존재하는데 우리는 후자의 방식으로 모델링을 하였다. 본 과제의 목표는 10일 동안의 단기 수익을 내는 것이므로, Buy / Hold / Sell 중 Sell은 Buy 다음 날 무조건 일어난다는 전제 하에 Sell을 없앴다. 또, *action*의 수를 줄이기 위해 하루에 한 종목의 주식만 사도록 하였다. 최종적으로, [ 각 기업의 Buy, 아무 것도 사지 않음(Hold) ]이 최종 *action*으로 선택되었다. 따라서 *action*의 총 개수는 ( 기업의 수 + 1 )이다.

*select\_action()* : *epsilon*의 비율로 랜덤하게 선택되고, 나머지 비율로 DQN에서 action이 선택된다. 랜덤하게 선택될 때 ⅓ 의 비율로 Hold를, ⅔ \* (기업의 수) 비율로 각 기업의 Buy를 선택하도록 한다.

(4) Reward

reward는, 현금자산에 대한 부분과 주식의 가치에 대한 부분을 합하여 계산하였는데, 상세한 식은 아래와 같다.



**4. Feature selection/Parameter tuning**

**(1) Validation**

time series에 많이 사용되는 walk-forward 방식을 사용하였으며, time step은 test기간이 2주이므로 각 validation set이 겹치지 않도록 똑같이 2주, k는 3으로 설정하였다.

**(2) Feature selection**

대부분의 관련 논문에서 사용되는 MA와 EMA를 기본으로 사용하였고, 나머지 feature들은 Company를 [ Celltrion / SamsungElectronics / Skhynix ]로 설정한 상태로 grid search를 통해 선정하였다. 유사한 성격을 지니는 BNF와 Momentum Oscillator는 둘 다 사용하는 것보다는 Momentum Oscillator만을 사용하는 것이 나은 것으로 확인되었고, discrete 변수들이나 volume ratio는 모델 학습이 이루어지지 않아 제외했다. 여러 조합에 대해 실험한 결과, [ Close / EMA / SMA / MA5\_Close / Momentum Oscillator /KOSPI]가 최종 선정되었다.

**(3) Parameter tuning**

일종의 할인률인 *gamma*는, Data Analysis에서 설명한 바와 같이 약 0.9996으로 설정하였다. *Memory size*의 경우 사용하는 컴퓨터의 한계로 설정하였다. *Distributional min/max*는 위에서 정의한 reward의 0.1과 0.9 분위수 정도에 해당하는 ± 2\*107로 설정되었는데, 이는 기존 연구들에서 보아온 reward 범주보다 넓다(가령, Atari에서는 -10 ~ 30). 따라서, 기존 51개보다 큰 1001개를 atom으로 설정했다. 기업의 경우, 두 개의 삼성 주식은 상관성이 매우 높아 2만 사용하기로 했고, LGH&H도 없을 때의 성능이 나아 총 4개의 기업을 후보군으로 설정했다. 나머지 변수들의 경우 일반적으로 많이 쓰이는 범위에서 grid search를 통해 선택되었고, 최종 하이퍼파라미터는 아래와 같다.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Total Epoch** | 1000 | **Q learning epoch** | 20+20n | **Target Q Learning Epoch** | 20+100n |
| **Epsilon\_start** | 1.0 | **Epsilon decay rate** | 0.999 | **Epsilon min** | 0.1 |
| **Gamma** | 0.9996 | **Memory size** | 100000 | **Batch size** | 200000 |
| **Multi step** | 3 | **Distributional atoms** | 1001 | **Distributional min/max** | ± 2\*107 |
| **Learning rate** | 0.0001 | **Input days** | 5 |  |  |
| **Company** | Celltrion / LGChemical / SamsungElectronics2 / Skhynix | | | | |

**5. Result**

위의 하이퍼-파라미터에 대하여, 총 1000번의 epoch 중 포트폴리오의 중간값이 가장 높았던 147과 550 epoch의 모델을 checkpoint로 저장하여 테스트에 사용하였는데 서로 경향이 달라 두 모델을 표준편차(risk)의 역수만큼 비중을 두어budget을 분배하는 방식으로 앙상블하였다. 테스트 기간 동안의 action의 변화, 그리고 최종 포트폴리오 값은 각 fold에서 99409250, 100424600, 101110600.000이었다. 실제 주식거래내역을 확인한 결과, 147epoch의 경우 변동이 큰 셀트리온을 통해 이득을 많이 얻으려는 공격적인 모델, 550epoch의 경우 셀트리온을 제외하고 나머지 세 종목을 활발히 거래하는 일종의 보수적인 모델이 선정된 것으로 보였다.