Dokumentacja do projektu z USD

Roman Moskalenko Pavel Tarashkevich

1 Treść zadania

LL1.AnyTrading Stocks

Zapoznaj się z gym-anytrading oraz zbiorem danych tam dostępnym:
LL1 - STOCKS_GOOGL

Przygotuj agenta podejmującego akcje na rynku i uczącego się ze wzmocnieniem oraz porównaj jego skuteczność ze strategią losową. Przetestuj co najmniej 2 różne algorytmy uczenia ze wzmocnieniem. Przygotuj skrypt umożliwiający wizualizację działań wytrenowanego agenta.

2 Projekt wstępny

2.1 Założenia ogólne

Dla zaimplementowania agenta zajmującego się handlowaniem na rynku mamy posłużyć algorytmem uczenia się ze wzmocnieniem (dalej RL). Jako założenie przyjmujemy, że środowisko agenta, którym posłużymy się, będzie tylko częściowo odwzorowywać rzeczywiste środowisko rynku. Będzie ono zawierało minimalny zestaw atrybutów, niezbędnych do nauczenia agenta oraz najprostsze akcje (*kup*/*sprzedaj*) do dyspozycji agenta.

Zakładamy, że algorytmy RL sprawdzą się lepiej, niż strategia losowa.

2.2 Krótki opis przydzielonego środowiska

W ramach projektu będziemy używać środowiska do symulacji handlowania na rynku akcji podane w treści zadania: gym-anytrading. Dane środowisko jest pochodną popularnego środowiska do trenowania agentów RL gym i zawiera ono podstawowe jego funkcje. Zawiera ono także minimalny zestaw atrybutów opisujących rynek akcji. Ono zawiera specjalną instancję *StockEnv*, która naszym zdaniem najlepiej odwzorowuje środowisko dla naszego zadania.

2.3 Wybrane algorytmy

Do realizacji zadania wybraliśmy algorytmy A2C, PPO. Literatura dotycząca stosowania algorytmów RL dla trenowania agentów do handlowania na rynku akcji wskazuje, że te dwa algorytmy dobrze się sprawdzają [1].

Algorytm A2C (*Advantage Actor-Critic*) bazuje na pomyśle wykorzystania sieci neuronowych do aproksymacji polityki agenta (aktor) oraz do aproksymacji funkcji wartości (krytyk).

Algorytm PPO (*Proximal Policy Optimization*) rozwija algorytm A2C, wykorzystując specjalne metody do optymalizacji polityki agenta.

Porównamy je z algorytmem losowym.

2.4 Biblioteki wybrane do realizacji projektu

- **gym-anytrading** - środowisko do trenowania agentów RL do handlowania na rynku akcji.
- **Stock-Trading-Visualization** - skrypt do wizualizacji działań agenta oparty o bibliotekę *matplotlib*.
- **stable-baselines** - biblioteka zawierająca gotowe implementację algorytmów RL.

2.5 Propozycja eksperymentów do przeprowadzenia

Zamierzamy przeprowadzić trenowanie wybranych modeli stosując różne podejścia wyboru podzbioru danych uczących - danych z różnych okresów czasowych.

Nie jest oczywiste jak powinna wyglądać strategia nagradzania agenta: warto uwzględnić ten fakt, że w handlowaniu na rynku może być większa preferencja natychmiastowego zysku, aby móc go potem wykorzystać w późniejszych akcjach agenta. Uwzględnimy to, dostosowując parametr dyskontowania nagród.

W celu ewaluacji działania algorytmów zastosujemy następujące metryki m.i. wykorzystane w [1]: *Cumulative return*, *Annualized return*, *Annualized volatility*, *Sharpe ratio*, *Max drawdown*.

3 Projekt końcowy

3.1 Uzupełnienie projektu wstępnego

Niestety postanowiliśmy zrezygnować z korzystania ze skryptów **Stock-Trading-Visualization**, ponieważ opierają się na innej bibliotece do handlowania na rynku przez agentów RL i jest niekompatybilny z **gym-anytrading**.

W przykładach użycia biblioteki **gym-anytrading** zauważyliśmy użycie biblioteki **quantstats** do analizy operacji finansowych. Postanowiliśmy ją wykorzystać, gdyż pozwala wyznaczyć metryki wspomniane przez nas w rozdziale 2.5.

3.2 Opis środowiska gym-anytrading

Aby mieć możliwość poprawnie interpretować wyniki uczenia i działania modeli na danym środowisku musieliśmy zagłębić się w jego strukturę. Chcielibyśmy wymienić kilka najważniejszych cech, które zwróciły naszą uwagę:

- **pozycje *Long* i *Short*** - W zarysie: w pozycji *Long* agent chcę kupować akcje, kiedy ceny na nie są niskie, żeby w przyszłości zyskać na wzroście cen, a w pozycji *Short* agent chcę sprzedawać akcje, kiedy ceny są wysokie, aby potem na uzyskane środki kupić akcje kiedy ceny zmaleją, zyskując na różnicy.

- **nagrody (reward)** - różnica pomiędzy ceną zamknięcia akcji w danej chwili a ceną zamknięcia akcji w chwili ostatniej transakcji (transakcja jest wykonywana w sytuacji kiedy agent miał pozycję *Long* oraz wybrał akcję *Sell*).

- **zysk (profit)** - jest aktualizowany wraz z każdą transakcją, a jest równy liczbie akcji, które posiadaliśmy podczas ostatniej transakcji, razy aktualna cena zamknięcia akcji.

3.3 Uczenie agentów

Dane dostępne z biblioteki **gym-anytrading**, które zostały wykorzystane w ramach projektu podzieliliśmy na dwa podzbiory: dane uczące i dane testowe. Podział został dokonany w taki sposób, że pierwsze 80% danych tworzą zbiór uczący, a reszta - zbiór testowy.

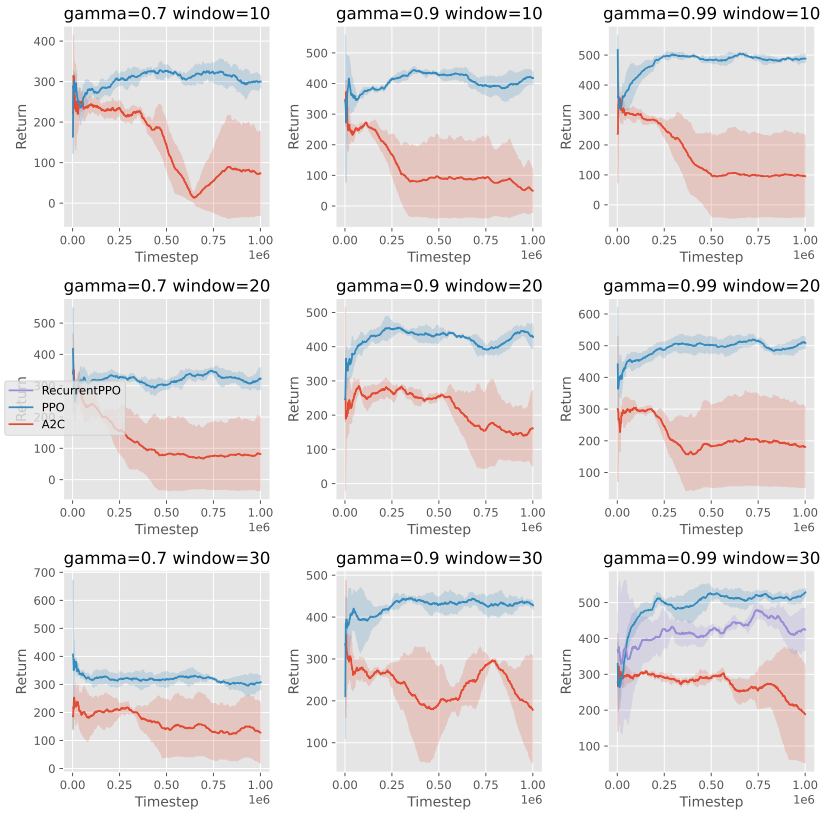
Z powodów oczywistych agent wykorzystujący politykę losową nie był uczony.

Wybraliśmy domyślne konfiguracje algorytmów A2C i PPO. Ich uczenie dokonywano w ciągu 1,000,000 kroków, gdzie za krok uznajemy pojedynczą akcją dokonaną przez agenta na danym środowisku. Uczenie przeprowadziliśmy dla różnych zestawów parametrów *rozmiaru okna* - zdarzeń historycznych dostępnych agentowi w każdym kroku działania (10, 20, 30) oraz *dyskonta* (0.7, 0.9, 0.99).

Po przetestowaniu nauczonych agentów uznaliśmy, że ich sprawność nie jest satysfakcjonująca, dlatego też użyliśmy rekurencyjnej wersji algorytmu PPO (dla *rozmiaru okna* 30 oraz *dyskonta* 0.99), którą pobraliśmy z repozytorium stable-baselines3-contrib (ze względu na brak oficjalnej implementacji w bibliotece stable-baselines3).

Uczenie jest dokonane łącznie 3 razy dla każdego algorytmu w celu zmniejszenia wpływu losowości na uzyskane wyniki.

Na rysunku 1 pokazano przebieg uczenia agentów. Jak widać sumy nagród uzyskiwane przez agentów dla parametru dyskonta równego 0.7 są wyraźnie mniejsze w porównaniu z 0.9 i 0.99. Także widać, że rozmiar okna czasowego nie wiele zmienia pod względem efektywności uczenia. Charakterystycznym jest ten fakt, że algorytm PPO pokazuje lepszą wydajność, niż A2C, czego można było po nim się spodziewać. Nieco dziwnym jest ten fakt, że sumy nagród uzyskiwane przez algorytm rekurencyjny PPO są mniejsze od sum nagród uzyskiwanych przez algorytm domyślny.



Rysunek 1: Przebieg uczenia agentów w przeciągu 1,000,000 kroków. Na wykresie pokazano zakres sum nagród, uzyskany przez średnią sumę nagród oraz odchylenia standardowego.

3.4 Ewaluacja działania agentów

Do ewaluacji działania nauczonych agentów każdy z nich przetestowaliśmy na danych testowych, wykorzystane metryki z rodziału 2.5 uzyskaliśmy za pomocą raportów Quantstats. Wyniki wybranego zbioru nauczonych agentów podajemy w tabeli 1.

Krótki opis tych metryk podany poniżej:

- **Cumulative return** - zwrot inwestycji na całym okresie czasowym.
- **Annualized return** - zwrot inwestycji w okresie jednego roku.
- **Annualized volatility** - miara rozrzutu zwrotów uzyskanych w okresie jednego roku, najczęściej liczona jako odchylenie standardowe zwrotów.

- **Sharpe ratio** - stosunek różnicy średniego zwrotu i stopy wolnej od ryzyka (w naszym projekcie przyjęliśmy ją równą 0) do wartości *volatility*. *Sharpe ratio* większe od 1 zwykle uznawane jest za dobre [2].
- **Max drawdown** - największa wartość różnicy między szczytem a kolejnym dołkiem wyrażonej w procentach.

	Cum. return	Ann. return	Ann. volatility	Sharpe ratio	Max drawdown
Random	-71.67%	-49.87%	15.0%	-4.79	-72.23%
A2C	-68.82%	-45.36%	13.14%	-5.06	-68.96%
PPO	0.74%	-0.75%	14.71%	0.1	-17.72%
RecPPO	14.86%	-0.42%	19.33%	0.51	-23.42%

Tabela 1: Wartości metryk sprawności agentów dla danych testowych

Wyniki te w postaci bardziej szczegółowej można znaleźć w dodatku A.

Jak można było się spodziewać agent losowy zachowuje się najgorzej i nie przynosi żadnego zysku przez cały okres testowy. Algorytm A2C sprawuje się podobnie do algorytmu losowego, jego działanie nie możemy uznać za sensowne. Algorytmy z rodziny PPO pokazują o wiele lepsze wyniki, przy czym modyfikacja o rekurencyjną politykę wyraźnie wzmocniła algorytm i zwiększyła jego efektywność. Jest to naturalne, gdyż rozważany w ramach projektu problem jest w uproszczeniu działaniem na szeregach czasowych, gdzie modele rekurencyjne generalnie dobrze się sprawują.

Wyniki te są mało satysfakcjonujące, gdyż generalnie algorytmy częściej tracą, niż zyskują, pomijając fakt, że prawdziwe środowiska do tradingu są zdecydowanie bardziej złożone. Stosowanie rekurencyjnej polityki jest mocno zalecane.

Referencje

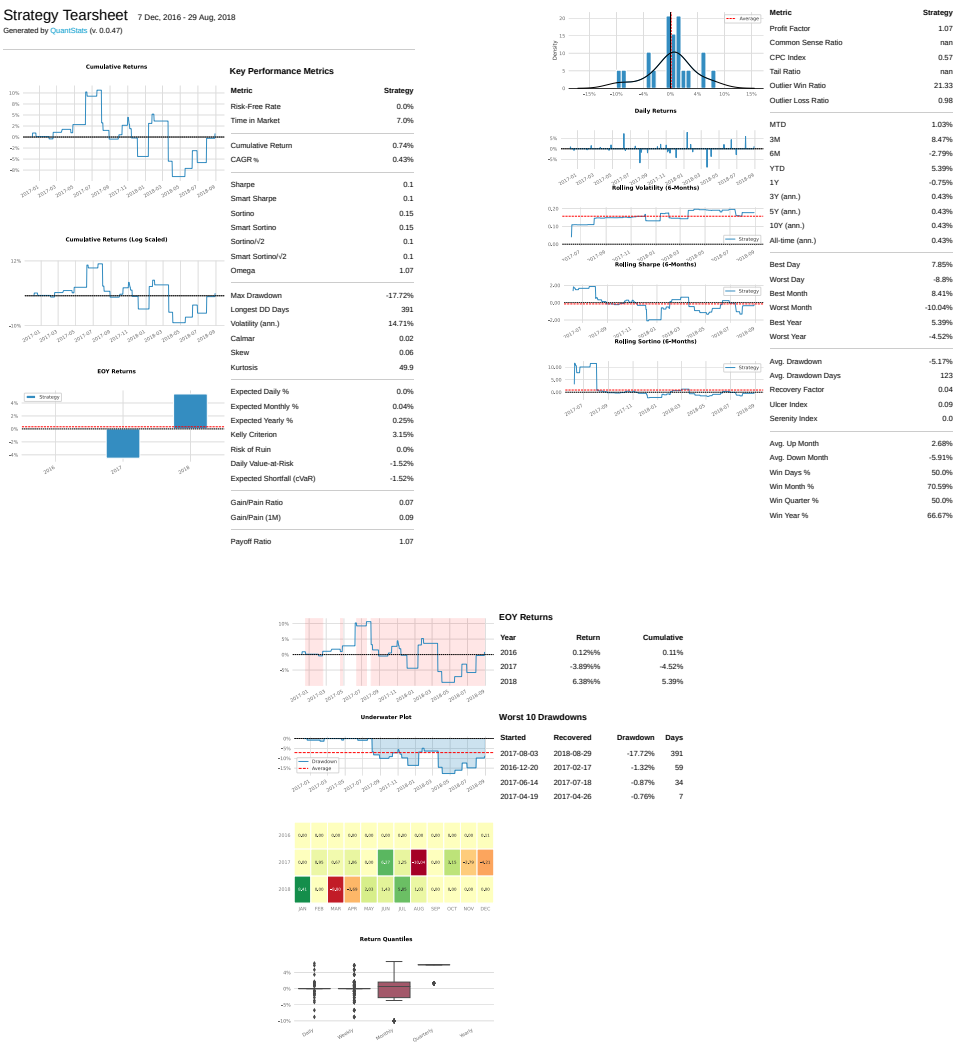
[1] Yang, Hongyang and Liu, Xiao-Yang and Zhong, Shan and Walid, Anwar, Deep Reinforcement Learning for Automated Stock Trading: An Ensemble Strategy (September 11, 2020). Available at SSRN: https://ssrn.com/abstract=3690996 or http://dx.doi.org/10.2139/ssrn.3690996

[2] `https://www.investopedia.com/terms/s/sharperatio.asp`

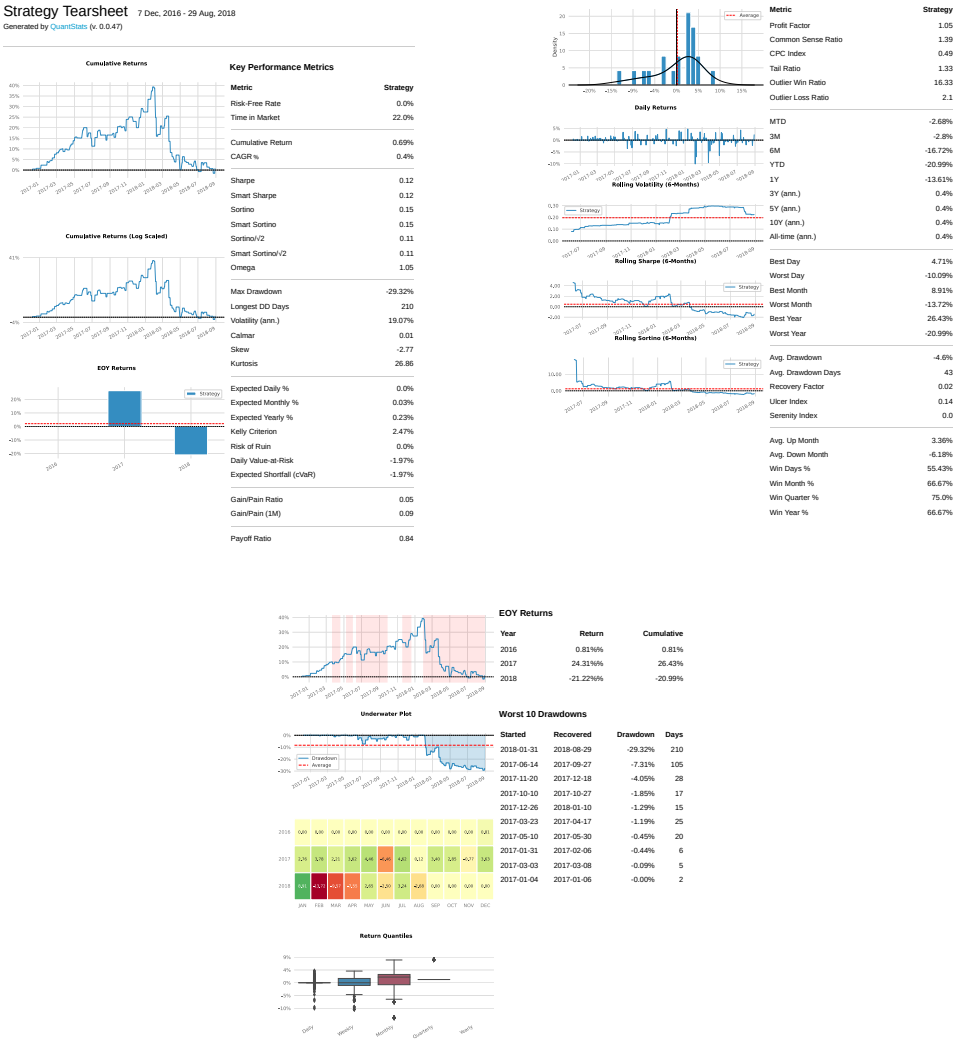
A Raporty Quantstats dla rozmiaru okna 30 oraz dyskonta 0.99

Rysunek 2: Raport Quantstats dla algorytmu losowego na danych testowych

Rysunek 3: Raport Quantstats dla algorytmu A2C na danych testowych



Rysunek 4: Raport Quantstats dla algorytmu PPO na danych testowych



Rysunek 5: Raport Quantstats dla algorytmu RecurrentPPO na danych testowych