Введение

Торговля на финансовых рынках представляет собой сложную задачу, требующую не только понимания рыночных механизмов, но и эффективного анализа большого объема данных в реальном времени. В последние годы методы машинного обучения, особенно обучение с подкреплением (RL), стали важным инструментом для разработки торговых стратегий. Обучение с подкреплением — это метод машинного обучения, при котором агент обучается принимать решения, максимизируя совокупное вознаграждение, взаимодействуя с окружающей средой. В контексте финансовых рынков агент может быть представлен как торговый бот, который принимает решения о покупке или продаже активов на основе сигнала о состоянии рынка.

Одним из подходов в области обучения с подкреплением, который получил широкое применение в финансовой торговле, является метод многоруких бандитов.

Многорукий бандит — это алгоритм принятия решений, суть которого заключается в обеспечении баланса между исследованием и использованием какого-то из рассматриваемых вариантов решения с целью максимизации целевой метрики и минимизации убытков.

Этот термин описывает ситуацию, в которой агент сталкивается с несколькими альтернативами (или "руками игровых автоматов"), каждая из которых имеет свою скрытую вероятность выигрыша. Метод многоруких бандитов подходит для задач, связанных с принятием решений в условиях неопределенности. В рамках этого подхода можно разработать торговую стратегию, которая будет адаптироваться и улучшаться на основе собранного опыта и данных о рынке.

Цели и задачи проекта:

Цели

- Разработка торгового бота

- Оптимизация торговой стратегии

- Подключение к реальным данным

Задачи

- Изучение и анализ стратегий

- Сбор и обработка данных

- Проектирование и реализация торговой стратегии:

- Тестирование

Теоретическая основа

Мы рассматриваем задачу многоруких бандитов (MAB): нам дан игровой автомат с N рукоятками; на каждом временном шаге (t = 1, 2, 3, …) необходимо выбрать одну из N рукояток для игры. Каждая рукоятка i при игре, приносит случайное вознаграждение, которое распределено по некоторому фиксированному (неизвестному) распределению с поддержкой на интервале [0, 1]. Случайные вознаграждения, получаемые при многократной игре на одной рукоятке, являются независимыми и идентично распределёнными (i.i.d.) и не зависят от игр на других рукоятках. Вознаграждение наблюдается немедленно после игры на рукоятке.

Алгоритм для задачи MAB должен решать, какую рукоятку играть на каждом временном шаге t, исходя из результатов предыдущих t-1 игр. Пусть μi обозначает (неизвестное) ожидаемое вознаграждение для рукоятки i. Популярной целью является максимизация ожидаемого общего вознаграждения за время T, т.е. \(E\left[\sum_{t=1}^{T} \mu_{i(t)}\right] \), где \(i(t) \) — это рукоятка, сыгранная на шаге \(t \), и ожидание берётся по случайным выборам \(i(t) \), сделанным алгоритмом. Удобнее работать с эквивалентной мерой ожидаемого общего сожаления: суммарная потеря из-за несыгрывания оптимальной рукоятки на каждом шаге. Чтобы формально определить сожаление, введём некоторую нотацию. Обозначим \(\mu^* := \max_i \mu_i \), и \(R_i := \mu^* - \mu_i \). Также пусть \(k_i(t) \) обозначает количество раз, когда рукоятка \(i \) была сыграна до шага \(t-1 \). Тогда ожидаемое общее сожаление за время \(T \) выражается как:

\[

E[R(T)] = E\left[\sum_{t=1}^{T} R_{i(t)}\right] = \sum_{i} R_i E[k_i(T)]

\]

Основными алгоритмами классических многоруких бандитов являются:

- ε-greedy;
- UCB (upper confidence bound);
- Thompson sampling.

1) Суть алгоритма ε-greedy: выбираем стратегию с максимальной средней наградой (средним значением метрики, которую мы оптимизируем) и иногда с определённой заранее вероятностью выбираем случайную стратегию для исследования.
Стратегия имеет единственный параметр: ε - вероятность выбора не лучшей "ручки", а случайной для исследования среды. Можно уменьшать эту вероятность со временем (ε-decreasing).

2) UCB проводит своё исследование не случайно, а на основе растущей со временем неопределённости у стратегий. В начале работы алгоритм случайно задействует все стратегии, после чего рассчитывается средняя награда каждой. Далее после каждой итерации обновляются средние награды стратегий. С течением времени чем реже выбиралась та или иная стратегия,

тем больше будет у неё неопределённость. Окончательный выбор стратегии — это максимальная сумма средней награды и неопределённости среди всех стратегий.
Если представить алгоритм в математическом виде, то получится следующее выражение:

$$A_t = argmax\left[Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}}\right]$$

где:

- *Qt(a)* — среднее значение награды стратегии *a*;
- *t* — общее количество наблюдений;
- *Nt(a)* — количество раз, когда была выбрана стратегия *a;*
- *c* — коэффициент бустинга исследования. Чем он больше, тем больше алгоритм направлен на исследование. (Вообще этот коэффициент может быть как статичным, так и динамичным. Так как награда может меняться со временем, лучше сделать его зависимым от её значения.)

3) Thompson sampling основан на байесовском подходе, поэтому с ним неразрывно связано два термина:

- Априорное распределение - распределение, которое выражает предположения до учета экспериментальных данных.
- Апостериорное распределение - распределение, которое получено после учёта экспериментальных данных.

У каждого варианта перед запуском бандита есть априорное распределение его награды, которое по мере поступления новых данных становится апостериорным. Сэмплирование Томпсона берет рандомные значения из этих распределений, сравнивает их и выбирает вариант с максимальным значением.

Давайте рассмотрим реализацию Thompson sampling для биномиальной метрики. В качестве априорного распределения возьмём бета-распределение с

параметрами α и β:

$$\rho(\theta) = \frac{\Gamma(\alpha_k + \beta_k)}{\Gamma(\alpha_k)\Gamma(\beta_k)} \theta_k^{\alpha_k - 1} (1 - \theta_k)^{\beta_k - 1}$$

где α_k и β_k являются по своей сути кумулятивной суммой количества удачных и неудачных исходов:

$$\alpha_k \leftarrow \alpha_k + R_t$$

$$\beta_k \leftarrow \beta_k - R_t$$

Выбор стратегии в данном случае — максимальное значение theta, полученное из апостериорных распределений наших стратегий.

Используемый датасет

Набор данных о ценах 249 российских акций. Датасет содержит исторические цены открытия, максимума, минимума, закрытия и объема акций, торгуемых на финансовых рынках Российской Биржи. Взяты почасовые данные в период с `2023-06-01` по **2024-08-27**

	datetime	open	high	low	close	volume
0	2009-12-11 10:00:00	9.87	11.95	9.75	9.79	605
1	2009-12-11 11:00:00	9.79	9.91	9.71	9.87	231
2	2009-12-11 12:00:00	9.87	9.87	9.79	9.87	52
3	2009-12-11 13:00:00	9.89	9.95	9.87	9.95	69
4	2009-12-11 14:00:00	9.95	10.87	9.91	10.47	623
...	...	...	...	...	...	...
26461	2024-08-27 19:00:00	79.36	79.36	78.50	78.86	657
26462	2024-08-27 20:00:00	79.00	79.04	78.90	78.92	93
26463	2024-08-27 21:00:00	78.90	79.14	78.90	78.92	169
26464	2024-08-27 22:00:00	79.12	79.12	78.98	79.12	37
26465	2024-08-27 23:00:00	79.12	80.16	79.00	79.58	1289

26466 rows × 6 columns

Метрики:

В качестве метрик в нашей задаче мы использовали total rewards

$$r\_total_t = \sum_{i=1}^{t} R_i$$

А также `cumulative_regret`

$$reg_t = \sum_{i=1}^{t} R_{opt} - R_t$$

Метрика total rewards была выбрана, поскольку Для торгового бота основная цель — максимизация прибыли. Total rewards предоставляют прямую метрику, которая показывает, насколько хорошо бот зарабатывает деньги. Эффективные стратегии должны приводить к более высоким значениям total rewards.

`A cumulative_regret` измеряет, насколько меньше вознаграждение, полученное торговым ботом, по сравнению с тем, каким бы оно могло быть, если бы бот всегда выбирал наилучшую (оптимальную) рукоятку. Cumulative regret позволяет оценить, насколько хорошо алгоритм обходится с неопределенностью и принимает оптимальные решения на основе доступной информации.

Мы реализовали алгоритм выбора акций с использованием метода многорукого бандита Thompson Sampling.

Мы создаем класс `Strategy` как абстрактную основу для стратегий выбора акций, включающий методы для обновления вознаграждений и выбора акций. В частности, реализован Thompson Sampling, использующий параметры α (успешные выборы) и β (неудачные выборы) для генерации выборок из бета-распределений.

Класс `StockMarketEnv` описывает среду для торговли акциями, отвечает за загрузку и обработку исторических данных для анализа ожидаемых доходностей. Класс `Bandit` управляет выбором акций и обновлением вознаграждений в зависимости от получаемых доходностей.

Функция `calculate_regret` оценивает производительность стратегии, сравнивая накопленные награды с оптимальными доходностями. В завершение вычислений результаты визуализируются через графики кумулятивного сожаления и наград по итерациям.

График кумулятивного рекорда показывает, как с увеличением числа итераций растет накопленный результат. В начале наблюдается небольшое увеличение, затем идет резкий рост, что может указывать на нерегулярный выбор оптимальных акций и упущенную прибыль, а пики — на неудачные выборы. График общих вознаграждений демонстрирует изменения средней доходности портфеля во времени с высокой волатильностью, но в целом с тенденцией к росту. Пики указывают на успешные моменты при выборе доходных акций.