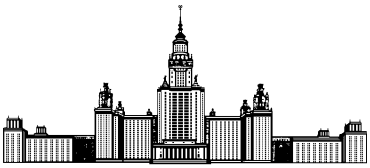

Московский государственный университет имени М. В. Ломоносова
Факультет Вычислительной Математики и Кибернетики
Кафедра Математических Методов Прогнозирования

ДИПЛОМНАЯ РАБОТА

Применение методов обучения с подкреплением к задаче алгоритмической торговли

Выполнил:
студент 517 группы
Сокурский Юрий Валентинович

Научный руководитель:
к.ф-м.н., доцент
Ветров Дмитрий Петрович

Москва, 2015

Содержание

1 Аннотация

В данной работе предложен метод обучения с подкреплением для решения задачи алгоритмической торговли. Метод рассчитан на высокую частоту совершения сделок. Для принятия решений используется агрегированная информация о заявках и сделках других игроков на рынке. Проведено экспериментальное исследование метода и сравнение с альтернативным методом алгоритмической торговли.

2 Введение

Существует много постановок задачи алгоритмической торговли. Они различаются по частоте совершения сделок, количеству используемых биржевых инструментов (например акций), информации, используемой для принятия решений. Методы обучения с подкреплением хорошо подходят для задачи алгоритмической торговли, так как они позволяют работать с отложенными по времени вознаграждениями. Существует много работ по применению методов обучения с подкреплением к задаче алгоритмической торговли [13],[14],[7],[8],[9], постановки задачи в них сильно отличаются.

В данной работе рассматривается постановка задачи маркет-мейкинга [10]. Для неё характерна высокая частота совершения сделок (несколько сотен в день), работа преимущественно с одним инструментом, для принятия решений используется информация о заявках на покупку или продажу других игроков, а также об уже совершённых сделках. В данной работе предложен и реализован метод обучения с подкреплением на базе метода Q-обучение. Отличительной чертой предложенного метода является то, что он сводит задачу обучения с подкреплением к нескольким последовательным восстановлениям регрессии.

В следующем разделе будут формально поставлены задачи алгоритмической торговли и обучения с подкреплением. Далее будет проведён обзор существующих методов обучения с подкреплением. Потом будет представлен разработанный метод. В конце будет проведено экспериментальное исследование представленного метода и сравнение с альтернативной стратегией маркет-мейкинга.

3 Постановка задачи

В этом разделе сначала будет поставлена задача алгоритмической торговли, потом задача обучения с подкреплением. Также будут приведены аргументы в пользу использования методов обучения с подкреплением для задачи алгоритмической торговли.

3.1 Задача алгоритмической торговли

Основными понятиями в задаче алгоритмической торговли являются: инструмент, заявка, книга заявок и сделка. *Инструментом* называется объект торговли, например акция или какой-то товар. *Заявка* отражает желание участника торгов купить или продать инструмент по определённой цене и в определённом объёме. Участники торгов могут посылать заявки в любое время, установленное биржей. Заявки, посланные участниками аккумулируются на бирже в *книге заявок*. Для каждого инструмента существует отдельная книга заявок. Внутри книги заявки отсортированы по цене. Цена может принимать только дискретные значения, поэтому заявки агрегируются и сортируются по возможным значениям цены. Если появляется заявка на продажу по цене, меньшей или равной максимальной цене заявки на покупку, то совершается *сделка*. Заявки, участвовавшие в сделке, удаляются из книги заявок. В случае, если у заявок был разный объём, то заявка с большим объёмом остаётся в книге, и её объём уменьшается. Таким образом все заявки на покупку, хранящиеся в книге заявок имеют цену, меньшую, чем заявки на продажу. Пример книги заявок приведён на рис.1.

Стоит отметить, что участники торгов могут совершать *шорт* заявки, то есть продавать инструменты, которых у них нет в данный момент, и иметь на руках отрицательный объём инструмента. Когда участник торгов в дальнейшем совершает покупку этого инструмента, объём купленных им заявок становится нулевым. Также существует разделение заявок на *пассивные* и *агрессивные*. Агрессивная заявка не имеет цены, и исполняется по оптимальной на данный момент цене. За каждую сделку и за каждую посылку заявки биржа берёт фиксированную комиссию.

Каждый участник торгов может получить информацию о приходящих заявках, их ценах и объёмах. Но при этом заявки разыменованы, это предотвращает точное копирование поведения игроков друг другом. Цель участников торгов – максимизировать прибыль от покупки и продажи инструментов.

Существует много разных постановок задачи алгоритмической торговли, в данной работе рассмотрена задача маркет мейкинга. Торговля ведётся одним инструментом, в день совершается несколько сотен сделок. Отосланные заявки как правило находятся на расстоянии от границы между покупкой и продажей.

Покупка	Цена	Продажа
1	1 609,11	
1	1 609,31	
10	1 609,33	
22	1 609,50	
80	1 609,51	
27	1 610,00	
77	1 610,01	
55	1 610,51	
280	1 610,52	
	1 610,99	15
	1 611,00	532
	1 611,01	20
	1 611,05	42
	1 611,50	25
	1 611,53	70
	1 611,90	1 300
	1 612,00	1 235
	1 612,78	1
	1 612,90	600

Рис. 1: Пример книги заявок: заявки отсортированы по цене. Красным обозначены заявки на продажу, зелёным – заявки на покупку. В центральном столбце отображены возможные значения цены, в правом и левом столбцах – суммарные объёмы заявок, агрегированных по возможным значениям цены

3.2 Задача обучения с подкреплением

Теперь определим задачу обучения с подкреплением. Введём множество состояний среды S, множество действий агента A, стратегию агента $\pi(a|s)$ – распределение на действия, при данном состоянии среды, $a \in A, s \in S$.

Игра агента со средой происходит по схеме, описанной в алгоритме 1.

Инициализация стратегии $\pi_1(a|s)$;

Инициализация состояния среды s_1;

цикл *t = 1,...,T,..* **выполнять**

агент выбирает действие согласно стратегии $a_t \sim \pi_t(a|s_t)$;

среда генерирует вознаграждение $r_{t+1} \sim p(r|a_t, s_t)$ и новое состояние $s_{t+1} \sim p(s|a_t, s_t)$;

агент корректирует стратегию $\pi_{t+1}(a|s)$;

если *состояние s_t – терминальное* **тогда**

игра останавливается;

конец условия

конец цикла

;

Алгоритм 1: Игра агента со средой

Цель агента – подобрать стратегию так, чтобы максимизировать какой-то функционал от будущих вознаграждений. В данной работе рассматривается дисконтированное вознаграждение:

$$R_t = \sum_{\tau=t}^{+\infty} \gamma^{\tau-t} r_\tau \tag{1}$$

где $\gamma \in (0, 1)$ - коэффициент дисконтирования. γ играет роль дальновидности агента.

Выделяют две постановки задачи обучения с подкреплением – on-policy и off-policy [3]. В постановке on-policy агент играет со средой в реальном времени, поэтому скорость обучения стратегии π, определяет суммарное вознаграждение. В постановке off-policy агент имеет накопленные данные по состояниям среды и обучается offline, это позволяет не обращать внимание на скорость обучения. В данной работе используется постановка off-policy. Во многих алгоритмах при обучении используется рандомизированная версия стратегии, что позволяет агенту лучше исследовать среду. Но мы ищем оптимальную стратегию π в множестве детерминированных стратегий, то есть π должна выдавать детерминированное действие на каждое из состояний.

Также стоит отметить, что распределения $p(r|a_t, s_t)$ и $p(s|a_t, s_t)$ заранее неизвестны агенту.

3.3 Применение методов обучения с подкреплением к задаче алгоритмической торговли

В этом разделе совместим постановки задач, определённые выше. В рамках задачи алгоритмической торговли средой является рынок, агентом – торгующий алгоритм. Состоянием среды является описание книги заявок в данный момент времени и исторические данные. Действием агента является выставление заявки, действие определяет выставлять заявку или нет, и на каком расстоянии от границы между покупкой и продажей. В книге заявок между максимальной ценой на покупку и минимальной ценой на продажу как правило существует зазор, поэтому действия определяют расстояния заявки от максимальной заявки на покупку в случае заявок на покупку, и минимальной заявки на продажу в случае заявок на продажу.

Введём понятие *открывающих* и *закрывающих* сделок. Открывающие сделки агент совершает, когда объём, приобретённых им акций равен нулю. Закрывающие сделки – сделки, после которых объём приобретённых акций становится равным нулю. Вознаграждением считается разность между ценами открывающих и закрывающих сделок. В данной работе объём инструмента, который может находиться на руках у алгоритма, принадлежит множеству $\{-1, 0, 1\}$.

В процессе биржевой торговли заявки приходят очень часто. Для того, чтобы сократить объём данных, дискретизуем время по секундам. Стоит отметить, что временной промежуток между открывающей и закрывающей сделкой может быть очень большим.

Основной мотивацией использования методов обучения с подкреплением для задачи алгоритмической торговли является отложенность вознаграждения. То есть оценка качества действия, которое привело к совершению открывающей сделки, будет получена только после совершения закрывающей сделки. В стандартных стратегиях маркет мейкинга этот промежуток в среднем длится несколько сотен секунд.

В данной работе обучение алгоритма происходит по схеме off-policy, для генерации обучающей выборки используется симулятор рынка, который работает на реальных исторических данных. Основным предположением для этого симулятора является то, что агент не оказывает влияния на среду. То есть $s_{t+1} \sim p(s|s_t)$, а не $s_{t+1} \sim p(s|s_t, a_t)$. Также стоит отметить, что в связи с фиксированностью исторических данных состояние детерминированно зависит от момента времени.

4 Существующие методы обучения с подкреплением

В этом разделе будет описан метод Q-обучение, на базе которого был создан метод, используемый в данной работе. Также будет приведён краткий обзор существующих модификаций Q-обучения и других методов.

4.1 Q-обучение

Одним из базовых методов обучения с подкреплением является метод Q-обучение [1].

Этот метод использует понятие функции ценности пары состояние и действие $Q(s, a)$, эта функция определена для стратегии π и является мат. ожиданием дисконтированного вознаграждения $Q^\pi(s, a) = \mathbb{E}_{\pi,s,r}(R_t|s_t = s, a_t = a)$. Также определена функция ценности состояния $V^\pi(s) = \mathbb{E}_\pi(R_t|s_t = s)$, а также оптимальные функции ценности $V^*(s) = \max\limits_\pi V^\pi(s)$, $Q^*(s, a) = \mathbb{E}[r|s, a] + \mathbb{E}_{s^{next}|s,a} V^*(s^{next})$. Детерминированную стратегию можно восстановить по функции ценности $arg\max\limits_a \pi(a|s) = arg\max\limits_a Q(s, a)$.

Алгоритм Q-обучение восстанавливает оптимальную функцию ценности. В алгоритме используется тот факт, что функцию ценности можно выразить через саму себя рекуррентно.

$$Q^\pi(s_t, a_t) = \mathbb{E}(\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}) = \mathbb{E}(r_{t+1} + \gamma Q^\pi(s_{t+1}, a_{t+1})) \tag{2}$$

Аналогично можно выразить оптимальную функцию ценности

$$Q^*(s_t, a_t) = \mathbb{E}(r_{t+1} + \gamma \max_a Q^*(s_{t+1}, a)) \tag{3}$$

Игра агента со средой происходит по следующему алгоритму 2.

Инициализация стратегии $\pi_1(a|s)$;

Инициализация состояния среды s_1;

цикл *t = 1,...,T,..* **выполнять**

агент выбирает действие согласно стратегии $a_t \sim \pi_t(a|s_t)$;

среда генерирует вознаграждение $r_{t+1} \sim p(r|a_t, s_t)$ и новое состояние $s_{t+1} \sim p(s|a_t, s_t)$;

$Q(s_t, a_t) := Q(s_t, a_t) + \alpha_t(r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t))$;

если *состояние* s_t *– терминальное* **тогда**

игра останавливается;

конец условия

конец цикла

;

Алгоритм 2: Q-обучение

В ходе игры используется рандомизированная стратегия $\hat{\pi}$, которая строится для того чтобы агент имел возможность исследовать среду. Существуют разные способы рандомизации стратегии, например $\epsilon - greedy$:

$$\hat{\pi}_{t+1}(a|s) = \frac{1-\epsilon}{|A_t(s)|}[a \in A_t(s)] + \frac{\epsilon}{|A \backslash A_t(s)|} \tag{4}$$

где $A_t(s) = arg \max_{a \in A} Q^{\pi_t}(s, a)$,

или SoftMax:

$$\hat{\pi}_{t+1}(a|s) = \frac{\exp(Q^{\pi_t}(s, a)/\tau)}{\sum_{a'} \exp(Q^{\pi_t}(s, a')/\tau)} \tag{5}$$

В этих стратегиях параметры ϵ и τ определяют компромисс между исследованием среды и использованием уже найденной информации. Значения этих параметров актуальны только для обучения по схеме on-policy.

Видно, что в ходе обучения алгоритм минимизирует разницу между $Q(s_t, a_t)$ и $r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a')$. Параметр α_t является шагом обучения. Доказано, что если $\sum_{t=1}^{\infty} \alpha_t = \infty$, а $\sum_{t=1}^{\infty} \alpha_t^2 < \infty$, то метод сходится к оптимальной стратегии [1].

4.2 Модификации метода Q-обучение и другие методы

Основным недостатком метода Q-обучение является то, что он применим только в случае дискретных состояний среды $s \in S$. В случае если состояния среды описываются действительными векторами, используются приближения функций ценности $\hat{Q}$. В качестве аппроксиматоров используются, например линейные функции [14], или нейросети [2]. Для построения такой аппроксимации нужно последовательно оптимизировать функции потерь, вида ([11], [12]):

$$L_i(\theta_i) = \mathbb{E}_{s,a,r}[\mathbb{E}_{s'}(r + \gamma \max_{a'} Q(s', a'|\theta^*_{i-1}) - Q(s, a|\theta_i))^2]; L_i(\theta_i) \to \min_{\theta_i} \tag{6}$$

Здесь θ_i – параметры аппроксимации, а θ_i^* – оптимальные параметры с точки зрения функционала $L_i(\theta_i)$.

Стоит отметить, что существует много работ, в которых методы обучения с подкреплением применяются для задачи алгоритмической торговли [13], [14], [7], [8], [9]. Но в разных работах задачи ставятся по разному, и редко реализация находится в открытом доступе, это затрудняет сравнение методов. Обучение с подкреплением также используется в робототехнике [4], [5], системах управления светофорами [6] и проч.[11] [12].

5 Разработанный метод

В этом разделе представлены основные трудности применения алгоритма Q-обучение к задаче алгоритмической торговли и модификации метода, позволяющие с ними справиться. Также представлена итоговая схема алгоритма с описанием процесса обучения и тестирования.

5.1 Трудности и модификации

Состояние среды в данной задаче было предсталено вещественным вектором, поэтому была использована аппроксимация функции ценности. В качестве аппроксиматоров были использованы двух и трёхслойные нейросети. Для подбора параметров аппроксиматоров требуется последовательно минимизировать функционалы вида:

$$L_i(\theta_i) = \mathbb{E}_{s,a,r}[\mathbb{E}_{s'}(r + \gamma \max_{a'} Q(s', a'|\theta_{i-1}^*) - Q(s, a|\theta_i))^2] \tag{7}$$

Основной трудностью является оценка мат. ожидания $\mathbb{E}_{s,a,r}$. Оценивание производится с помощью сэмплирования величин из распределений $p(s|s_t, a_t), \pi(a|s_t), p(r|s_t, a_t)$. Если последовательности троек (s,a,r) генерировать, совершая случайные действия, как это рекомендуется в статье [11], то обучающая выборка становится очень большой, а обучение – ресурсоёмким. Но в данной работе мы исходим из предположения, что агент никак не влияет на среду $p(s|s_t, a_t) = p(s|s_t)$, единственное, на что влияют действия агента это текущий объём приобретённого инструмента (действие может привести к сделке). Также в данной работе используются исторические данные, а не реальные торги. Поэтому для оценки $\mathbb{E}_{s,a,r}$ в каждый момент времени можно перебирать все возможные действия, для всех возможных текущих объёмов. Эта модификация сильно уменьшает размер выборки.

Следующей трудностью является переобучение на стратегиях, использующих большие промежутки между открывающими и закрывающими сделками. Такие стратегии имеют высокий уровень риска, и не являются правильным решением задачи маркет мейкинга. Чем больше промежутки между открывающими и закрывающими сделками, тем меньше сделок алгоритм может совершить за день, в то время как стратегия маркет мейкер должна совершать несколько сотен сделок за день. Эту проблему решает ограничение на временной промежуток между открытием и закрытием, обозначим это ограничение T_{max}. Также для удобства разделим стратегию на *внешнюю* и *внутреннюю*

(π^{out}, π^{in}). Внешняя стратегия совершает открывающие сделки, внутренняя – закрывающие. Соответственно обозначим и функции ценности Q^{out}, Q^{in}. Обучающая выборка для внутренней стратегии формируется исходя из того, что внешняя стратегия совершает все возможные действия в каждый момент времени. В обучающей выборке для внешней стратегии в качестве вознаграждений используются результаты работы уже обученной внутренней стратегии. Стоит отметить, что действия внешней стратегии никак не зависят друг от друга, поэтому для внешней стратегии коэффициент дисконтирования можно выбрать равным нулю $\gamma^{out} = 0$. Таким образом обучение внешней стратегии сводится к обычной регрессии.

Ещё одной трудностью является неустойчивая сходимость функций потерь для внутреннего алгоритма. Во первых, процесс обучения функционалов $L_i(\theta_i)$ имеет много локальных оптимумов. Во вторых, значения $Q_i^{in}(s, a|\theta_i^*)$ могут как возрастать, так и убывать при увеличении i (это зависит от начального приближения). Это приводит к тому, что $L_i(\theta_i^*)$ могут как возрастать, так и убывать, это затрудняет определение сходимости (рис. 2). Эту проблему решает введение разных функций ценности $Q_t^{in}(s, a|\theta_t^{in})$ для всех $t \in [1, T_{max}]$. Таким образом обучение внутреннего алгоритма сводится к последовательному обучению T_{max} регрессоров. Сначала обучается функция $Q_{T_{max}}^{in}$, затем каждая t-ая функция обучается, используя результаты t+1 - ой (при обучении t убывает от T_{max} до одного). В качестве целевой переменной для регрессора на каждой итерации используется оценка функции ценности, полученная исходя из действий, полученных с помощью следующих регрессоров.

Довольно существенной трудностью является переобучение на глобальные тренды рынка. Стратегия маркет мейкер должна играть на небольших колебаниях рынка и игнорировать глобальные тренды. В случае, если обучающая выборка состоит преимущественно из дней, в которые цена на инструмент растёт, то алгоритм может сходится к стратегии «купить и ждать» при этом признаки, описывающие состояние среды содержат мало информации о глобальном тренде, таким образом происходит переобучение. Для того, чтобы устранить этот недостаток, можно добавить к имеющейся выборке её «перевёрнутую» копию. Переворачивание выборки осуществляется домножением цены на -1, и заменой покупки на продажу и наоборот. Добавление перевёрнутой выборки делает алгоритм симметричным относительно замены покупки на продажу, это позволяет сократить множество возможных действий: теперь стратегию, выставляющую заявки на покупку, можно использовать для выставления заявок на продажу.

5.2 Итоговая схема алгоритма

Теперь сформулируем итоговую схему алгоритма. Для обучения внутреннего алгоритма сначала нужно составить матрицы состояний среды, индикаторов терминальных состояний и вознаграждений. Формирование этих матриц происходит по алгоритму 3;

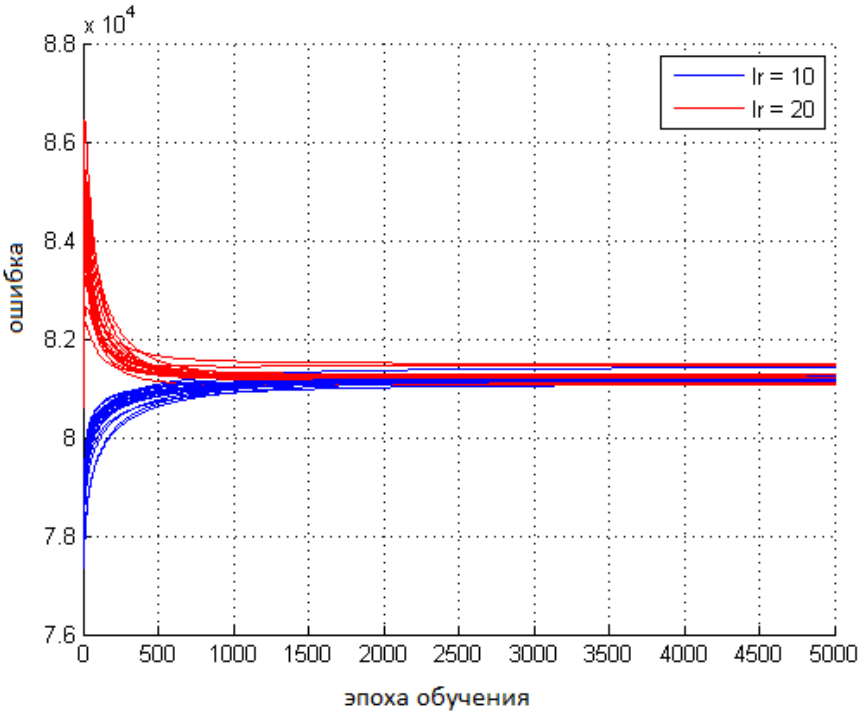

Рис. 2: Сходимость внутреннего алгоритма, в зависимости от шага обучения (lr). На графике представлены несколько запусков алгоритма оптимизации. Видно, что функция ошибок может как возрастать, так и убывать, и что оптимизация сходится к разным локальным оптимумам.

Сформировав матрицы можно произвести обучение внутреннего алгоритма (алг. 4)

Исходные параметры: Входные данные

множество возможных моментов времени τ_{all}; множество состояний среды x_t; описание среды, позволяющее определить совершится сделка или нет; параметр T_{max};

Результат: Выходные данные

матрицы состояний среды Xs, матрицы вознаграждений Rs, матрицы индикаторов терминальных состояний Ts;

Инициализируем T_{max} пустых матриц состояний среды Xs, T_{max} пустых матриц вознаграждений Rs, T_{max} пустых матриц индикаторов терминальных состояний Ts;

Инициализация состояния среды s_1;

цикл *для каждого момента времени* τ **выполнять**

- **цикл** *для каждого возможного действия внешнего алгоритма* **выполнять**
  - **если** *произошла открывающая сделка* **тогда**
    - запоминаем цену открывающей сделки startPrice;
    - **цикл** *для каждого момента времени* $t \in [1, T_{max}]$ **выполнять**
      - добавляем вектор описания среды $x_{\tau+t}$ к матрице Xs[t];
      - инициализируем пустой вектор вознаграждений r;
      - инициализируем пустой вектор индикаторов терминальных состояний $\hat{t}$;
      - **цикл** *для каждого возможного действия внутреннего алгоритма* **выполнять**
        - **если** *произошла закрывающая сделка или* $t == T_{max}$ **тогда**
          - подсчитываем вознаграждение и добавляем его к вектору r;
          - добавляем к вектору $\hat{t}$ единицу;
        - **иначе**
          - добавляем к вектору r ноль;
          - добавляем к вектору $\hat{t}$ ноль;
        - **конец условия**
      - **конец цикла**
      - добавляем вектор вознаграждений r к матрице Rs[t];
      - добавляем вектор индикаторов терминальных состояний $\hat{t}$ к матрице Ts[t];
    - **конец цикла**
  - **конец условия**
- **конец цикла**

конец цикла

;

Алгоритм 3: Создание обучающей выборки

Исходные параметры: Входные данные

Xs,Rs,Ts, параметр T_{max} ;

Результат: Выходные данные

T_{max} регрессоров для значений $Q_t^{in}(s, a)$;

Инициализируем T_{max} новых регрессоров models ;

Инициализируем $t := T_{max}$ и текущую оценку $Q := Rs[t]$;

до тех пор, пока $t \geq 1$ **выполнять**

- **цикл** *для каждого возможного действия внешнего алгоритма* **выполнять**
  - обучим регрессор $models[t] : Xs[t] \to Q$;
  - **если** *произошла открывающая сделка* **тогда**
    - запоминаем цену открывающей сделки startPrice;
    - получим оценки $\hat{Q} = models[t](Xs[t])$;
    - получим оптимальные действия $acts = argmax(\hat{Q})$;
    - обозначим $Q_{max} := Q[acts]$; – столбец максимальных Q;
    - $Q_{max} := Q_{max}$ (копировать столбец |A| раз);
    - инициализируем новое Q: = Rs[t-1];
    - учтём нетерминальные состояния Q[!Ts[t-1]] := Q[!Ts[t-1]] + $Q_{max}[!Ts[t-1]]$;
    - t := t-1
  - **конец условия**
- **конец цикла**

конец цикла

;

Алгоритм 4: Обучение внутреннего алгоритма

Внешний алгоритм обучается на результатах работы внутреннего. Тестирование происходит по схеме, изображённой на рисунках 3 и 4. Пока текущий объём равен нулю, заявки выставляет внешний алгоритм, как только совершается открывающая сделка, заявки начинает выставлять внутренний алгоритм вплоть до закрывающей сделки.

В качестве состояний среды используются следующие признаки:

1. Разница между оптимальными ценами на покупку и продажу
2. Приращение цен на покупку и продажу
3. Суммарный объём заявок с ценами, отличающимися от оптимальной на фиксированную величину
4. Суммарный объём сделок, совершённых за последнюю секунду
5. Расстояние от текущей цены до сглаженной цены

Все признаки берутся в окне за последние три секунды.

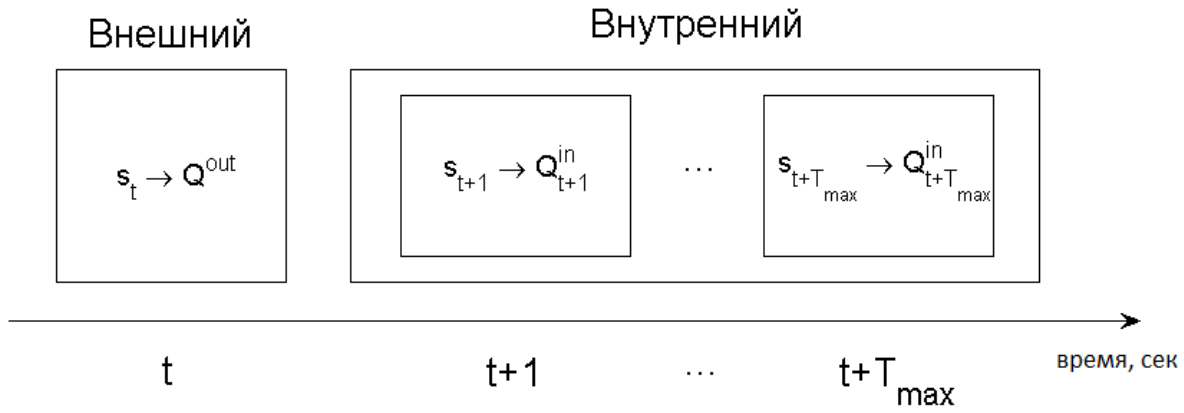

Рис. 3: Схема применения внутреннего и внешнего алгоритмов. Внутренний алгоритм запускается только в том случае, когда внешний совершил открывающую сделку

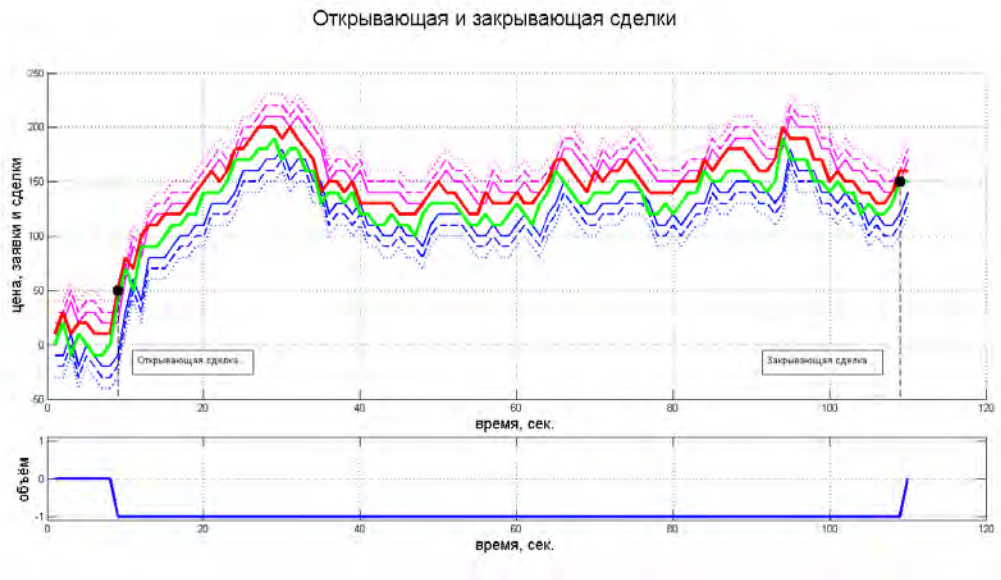

Рис. 4: Процесс алгоритмической торговли. Зелёная и красная линии – оптимальные цены на покупку и продажу в книге заявок. Синии и розовые линии – возможное расположение заявок. Чёрные точки – примеры сделок. На первых девяти секундах заявки выставляет внешний алгоритм, затем происходит открывающая сделка и заявки начинает выставлять внутренний алгоритм. Вслучае, если через T_{max} секунд не происходит закрывающей сделки, то она совершается автоматически

6 Эксперименты

В этом разделе будет продемонстрировано изменение качества алгоритма в зависимости от обучающей выборки, а также сравнение с базовой стратегией. Все эксперименты проводились на инструменте фьючерс индекса РТС, на исторических данных за июнь-июль 2014-го года.

6.1 Качество алгоритма в зависимости от обучающей выборки

Полнота обучающей выборки сильно влияет на обобщающую способность. Если в обучении использовать только дни, соответствующие спокойному рынку, то на днях, содержащих резкие скачки и тренды, алгоритм будет терять, и наоборот. Для того чтобы решить эту проблему нужно иметь либо достаточно большую и разнообразную выборку, либо использовать онлайн обучение. Онлайн обучение заключается в повторном обучении модели на новых приходящих данных. Качество онлайн обучения в среднем превосходит качество алгоритмов, обученных на фиксированном количестве дней (рис. 5, 6). Также на рисунках видно, что алгоритмы, обученные на первых нескольких днях могут показывать плохое качество на последних днях, это связано с тем, что в последних днях глобальные тренды сильнее, чем в первых. Также стоит отметить, что если размер выборки при онлайн обучении будет достаточно большим (рис. 6), то на всех днях алгоритм показывает положительный результат.

В этом эксперименте в качестве регрессоров для внутреннего алгоритма были использованы линейные регрессии, а для внешнего – случайный лес. Использовать случайный лес для внутреннего алгоритма затруднительно, так как он обучается сильно дольше линейной регрессии, а во внутреннем алгоритме требуется обучить T_{max} регрессий. В случае, если в обучении внешнего алгоритма использовать линейную регрессию, то у неё не хватает гибкости, и алгоритм сходится к стратегии "не торговать вообще".

В качестве состояний среды использовались признаки, описанные в предыдущем разделе, параметр $T_{max} = 100$.

6.2 Сравнение с базовой стратегией

Базовая стратегия использует свойство цены – возвращаться к своему среднему. Это фундаментальное свойство биржевых инструментов: если отклонение цены вызвано не глобальным трендом, то цена вернётся в своё изначальное положение. Можно играть на таких отклонениях. Наложим график цены на график её оконного сглаживания, когда цена опускается ниже сглаживания, базовая стратегия покупает инструмент. Когда цена возвращается к сглаживанию, стратегия продаёт купленное. В случае, когда цена поднимается выше сглаживания, стратегия действует наоборот: сначала продаёт, потом покупает. Оконное сглаживание устроено так, что рано или поздно оно пересекается с ценой. Такая стратегия выигрывает, когда инструмент не сильно меняется в цене, и проигрывает, когда инструмент находится под влиянием глобальных трендов.

В этом эксперименте были отобраны участки рынка, на которых базовая стратегия выигрывает, эти участки были поданы на обучение алгоритму. В качестве состояния среды были использованы только те признаки, которые использует базовая стратегия, то есть расстояние до оконного сглаживания. Размер окна для сглаживания в этом эксперименте – 300 секунд. Обучение проводилось на одном дне, тестирование на другом. На рис. 7 видно, что обученная стратегия превосходит базовую. Базовая стратегия не оптимально выставляет свои заявки, из-за того, что её параметры

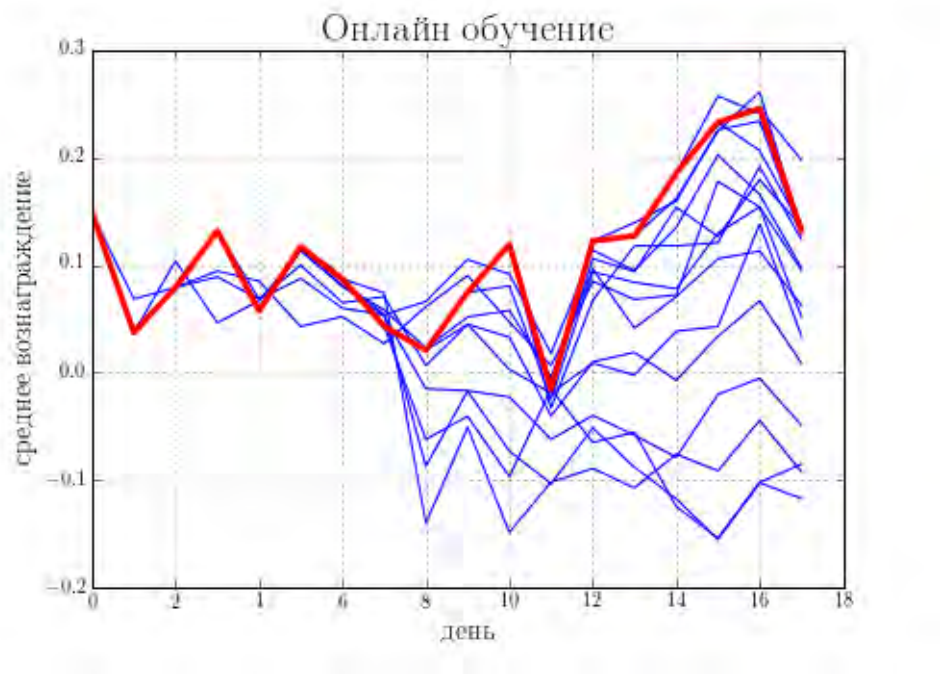

Рис. 5: Среднее вознаграждение, в обучении участвует три дня. Красная линия соответствует онлайн обучению, синии линии соответствуют качеству алгоритмов, обученных на всевозможных подряд идущих тройках дней

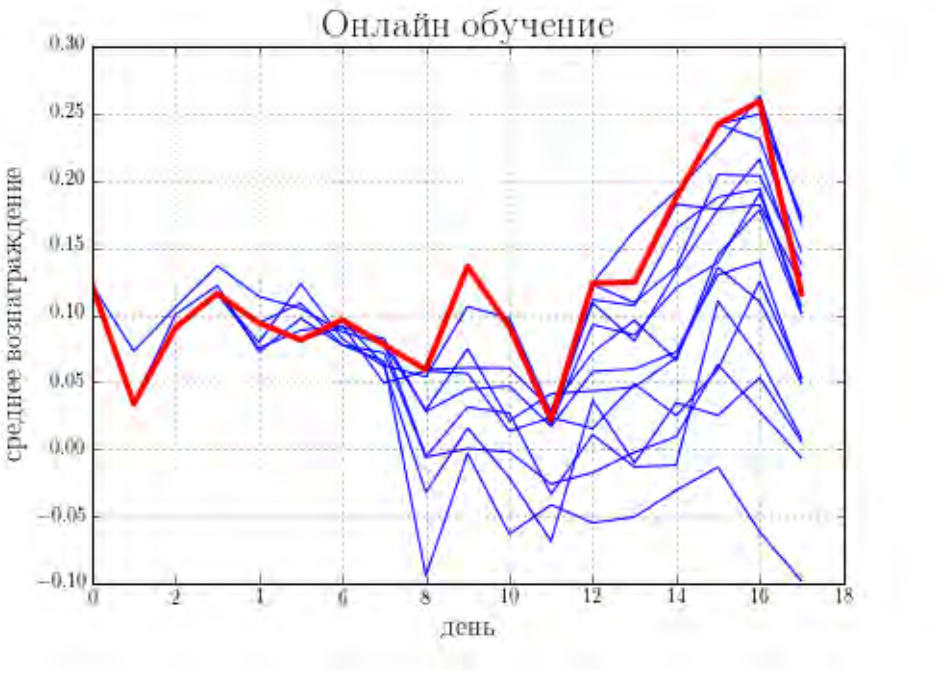

Рис. 6: Среднее вознаграждение, в обучении участвуют пять дней. Красная линия соответствует онлайн обучению, синии линии соответствуют алгоритмам, обученным на всевозможных подряд идущих пятёрках дней

подобраны вручную, а не обучены. Стоит отметить, что если добавить в тест участки, на которых базовая стратегия проигрывает, то у обеих стратегий будет отрицательное вознаграждение, это связано с тем, что в признаке "расстояние до сглаженной цены"слишком мало информации, основная информация, обеспечившая результат предыдущего эксперимента, лежит в объёмах заявок.

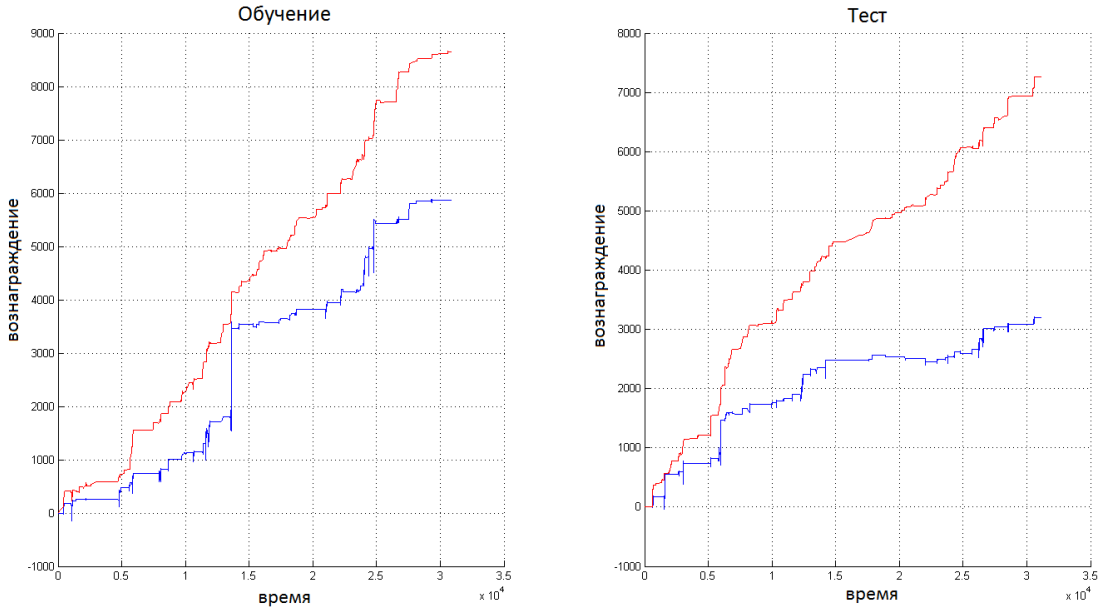

Рис. 7: Суммарное вознаграждение в зависимости от времени внутри дня, красная линия соответствует алгоритму обучения, синяя – базовому

7 Заключение

В рамках данной работы предложен и реализован метод обучения с подкреплением на базе метода Q-обучение. Преодолён ряд трудностей, связанных с применением данного метода к задаче алгоритмической торговли. Предложена идея последовательного обучения регрессоров для аппроксимации функции ценности. Проведено экспериментальное исследование данного алгоритма и сравнение с альтернативным методом на реальных данных.

Список литературы

[1] Christopher JCH Watkins and Peter Dayan. Q-learning. Machine learning, 8(3-4):279–292, 1992.

[2] Martin Riedmiller. Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In Machine Learning: ECML 2005, pages 317–328. Springer,2005.

[3] Sutton, Richard S., and Andrew G. Barto. Introduction to reinforcement learning. MIT Press, 1998.

[4] Hester, Todd, Michael Quinlan, and Peter Stone. "Generalized model learning for reinforcement learning on a humanoid robot."Robotics and Automation (ICRA), 2010 IEEE International Conference on. IEEE, 2010.

[5] Kober, Jens, Erhan Oztop, and Jan Peters. "Reinforcement learning to adjust robot movements to new situations."Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume Three. AAAI Press, 2011.

[6] Schouten, R., and M. Steingraver. "Reinforcement Learning of Traffic Light Controllers under Partial Observability."(2007).

[7] Shelton, Christian R. "Policy improvement for POMDPs using normalized importance sampling."Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence. Morgan Kaufmann Publishers Inc., 2001.

[8] Shelton, Christian R. "Reinforcement learning with partially known world dynamics."Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence. Morgan Kaufmann Publishers Inc., 2002.

[9] Shelton, Christian Robert. "Importance sampling for reinforcement learning with multiple objectives."(2001).

[10] Chan N. T., Shelton C. An electronic market-maker. – 2001.

[11] Mnih V. et al. Playing atari with deep reinforcement learning //arXiv preprint arXiv:1312.5602. – 2013.

[12] Mnih V. et al. Human-level control through deep reinforcement learning //Nature. – 2015. – Т. 518. – №. 7540. – С. 529-533.

[13] Nevmyvaka Y., Feng Y., Kearns M. Reinforcement learning for optimized trade execution //Proceedings of the 23rd international conference on Machine learning. – ACM, 2006. – С. 673-680.

[14] Bertoluzzo F., Corazza M. Reinforcement Learning for automatic financial trading: Introduction and some applications //University Ca'Foscari of Venice, Dept. of Economics Research Paper Series No. – 2012. – Т. 33.