МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ
Кафедра дискретной математики и алгоритмики

ЯКУБОВИЧ Анна Владимировна

МЕТОДЫ МАШИННОГО ОБУЧЕНИЯ В ЗАДАЧАХ ОПТИМАЛЬНОГО УПРАВЛЕНИЯ

Магистерская диссертация

специальность 1-31 80 09 «Алгоритмы и системы обработки больших объемов информации»

Научный руководитель:
Дмитрук Наталья Михайловна
кандидат физ.-мат. наук

Допущена к защите
«____» ________________ 2021 г.
Зав. кафедрой дискретной математики и алгоритмики
________ В.М. Котов
доктор физико-математических наук,
профессор

Минск, 2021

ОГЛАВЛЕНИЕ

РЕФЕРАТ

Магистерская диссертация, 51 стр., 3 рис., 8 таблиц, 13 источников.

Ключевые слова: МАШИННОЕ ОБУЧЕНИЕ, ОБУЧЕНИЕ С ПОДКРЕПЛЕНИЕМ, ЗАДАЧИ ОПТИМАЛЬНОГО УПРАВЛЕНИЯ, НЕЙРОННЫЕ СЕТИ, УПРАВЛЕНИЕ ПО ПРОГНОЗИРУЮЩЕЙ МОДЕЛЯМИ.

Объект исследования – обучение с подкреплением и управление по прогнозирующей модели в рамках задач оптимального управления.

Цель работы – решить задачу оптимального управления методом обучения с подкреплением и управлением по прогнозирующей модели, на основе чего провести сравнительный анализ.

Результаты исследования:

1. Изучены основные задачи оптимального управления, выбрана и поставлена задача.
2. Проведен обзор и выбран метод обучения с подкреплением для решения задачи.
3. Решена поставленная задача с помощью выбранного метода обучения с подкреплением.
4. Проведен обзор решения задач с помощью управления по прогнозирующей модели.
5. Решена задача с помощью управления по прогнозирующей модели.
6. Проведен сравнительный анализ решений, полученных вышеперечисленными методов.

Область применения – выбор решений для более сложных задач оптимального управления.

РЭФЕРАТ

Магістарская дысертацыя, 51 ст., 3 мал., 8 табліц, 13 крыніц.

Ключавыя словы: МАШЫННАЕ НАВУЧАННЕ, НАВУЧАННЕ З ПАДМАЦАВАННЕМ, ЗАДАЧЫ АПТЫМАЛЬНАГА КІРАВАННЯ, НЕЙРОННАЯ СЕТКА, КІРАВАННЕ ПА ПРАГНАЗУЕМАЙ МАДЭЛІ

Аб’ект даследавання – навучанне з падмацаваньнем і кіраванне па прагназуемай мадэлі ў рамках задач аптымальнага кіравання.

Мэта працы – рашыць задачу аптымальнага кіравання метадам навучання з падмацаваньнем і кіраваннем па прагназуемай мадэлі, на аснове чаго правесці параўнальны аналіз.

Вынікі даследавання:

1. Даследаваны асноўныя задачы аптымальнага кіравання, абрана і пастаўлена задача.
2. Праведзены агляд і абран метад навучання з падмацаваньнем для рашэння задачы.
3. Рашана пастаўленая задача з дапамогай абранага метаду навучання з падмацаваньнем.
4. Праведзены агляд рашэння задач з дапамогай кіравання па прагназуемай мадэлі.
5. Рашана задача кіраваннем па прагназуемай мадэлі.
6. Праведзены параўнальны аналіз рашэнняў, атрыманых вышэйпералічанымі метадамі.

Вобласць прымянення – выбар рашэнняў для больш складаных задач аптымальнага кіравання.

ABSTRACT

Master thesis 51 p., 7 pictures, 1 tables, 15 sources

Key words: MACHINE LEARNING, REINFORCEMENT LEARNING, OPTIMAL CONTROL PROBLEMS, NEURAL NETWORKS, MODEL PEDICTIVE CONTROL.

Object of research – reinforcement learning and model predictive control ifor the optimal control problems.

The aim of the work is to solve the problem of optimal control by reinforcement learning and model predictive control. Analyse and compare two solutions.

The results:

1. The main tasks of optimal control have been studied, the task has been selected and set.
2. A review was carried out and a reinforcement learning method was selected for solving the problem.
3. The task was solved using the selected reinforcement learning method.
4. A review of model predictive control for solving problems is carried out.
5. The problem is solved using model predictive control.
6. A comparative analysis of the solutions obtained by the above methods is carried out.

Field of application – the choice of solution method for complex problems of optimal control.

ВВЕДЕНИЕ

Оптимизация и управление на основе данных (data-driven optimization and control) – новое направление исследований в теории оптимизации и теории управления [1, 2]. Актуальность и интерес к этому направлению вызваны двумя факторами.

Во-первых, это рост объемов данных, развитие и доступность различных измерительных устройств и сенсоров, позволяющих записывать ранее недоступные данные, и, соответственно, бурное развитие алгоритмов и систем их обработки.

Во-вторых, объекты управления или оптимизации в прикладных исследованиях становятся все сложнее, и, соответственно, растут размерности и сложность их математических моделей, необходимых для применения существующих методов управления и оптимизации. Также возрастает число ограничений, накладываемых на переменные моделей, и зачастую эти ограничения связаны с безопасностью функционирования системы, т. е. не могут быть нарушены или ослаблены.

В настоящей работе будут исследоваться задачи управления динамическими объектами, в частности одна из центральных проблем теории управления – задача стабилизации. Одним из набирающих популярность в промышленных приложениях подходов к решению этой задачи является так называемый метод управления по прогнозирующей модели (Model Predictive Control – MPC) [4, 5]. Он основан на решении в реальном времени специально подобранной (прогнозирующей) задачи оптимального управления, которая в своей формулировке содержит математическую модель объекта управления в пространстве состояний, различные ограничения на состояния и управления и начальное условие, совпадающее с текущим состоянием объекта стабилизации. Для успеха реализации стратегии MPC, с одной стороны, требуется модель, описывающая процесс управления с высокой точностью, с другой же стороны, для решения задач оптимального управления в реальном времени модель (она задает ограничения-равенства в задаче) должна быть достаточно простой, иначе существующие численные методы решения оптимизационных задач могут не построить решение за требуемое время.

Другим примером может служить классическая задача из теории оптимального управления — задача синтеза оптимальной системы [13] (построение оптимальной обратной связи в задаче оптимального управления). Эта задача до сих пор не решена в классической постановке (построение обратной связи как функции всех позиций системы управления), однако для нее существует подход, называемый оптимальным управлением в реальном времени [12] и близкий по технике методу управления в реальном времени. В связи с приведенными выше факторами, а также описанными примерами, естественной и привлекательной становится следующая идея: 1) непосредственно использовать технологические или экспериментальные данные, полученные в результате наблюдений за поведением динамических систем, в формулировке задачи управления, исключая предварительный шаг идентификации системы по этим данным (этап математического моделирования); 2) привлекать алгоритмы обработки больших данных с целью повышения эффективности схем управления. Один из подходов, реализующих идею 1), можно найти в работе [8]. Еще одним метод, в котором реализована идея 1), является обучение с подкреплением.

В рамках данной работы проводится общий обзор управления по прогнозирующей модели. Также сделан обзор основных свойств методов машинного обучения и выделен подкласс методов, подходящих для решения задач оптимального управления, – обучение с подкреплением. В первой главе поставлена классическая задача оптимального управления, требующая решение. Во второй и третей главе соответственно описаны алгоритмы обучения с подкреплением и управления по прогнозирующей модели для решения поставленной задачи. В четвертой главе приведены практические аспекты их реализации и полученные результаты, произведен сравнительный анализ двух методов.

ГЛАВА 1

ОСНОВНЫЕ ПОНЯТИЯ И ОБЗОР ЛИТЕРАТУРЫ

В настоящей главе изложены общие принципы методов машинного обучения и управления по прогнозируемой модели, используемые для решения задач оптимального управления. Дана общая классификация задач оптимального управления, поставлена решаемая задача.

1.1 Методы машинного обучения

Машинное обучение - это раздел искусственного интеллекта (ИИ), который изучает методы создания алгоритмов, способные обучаться. В машинном обучении алгоритмы «обучаются» находить закономерности и особенности в огромных объемах данных, чтобы принимать решения и делать прогнозы на основе новых данных. Чем лучше алгоритм, тем точнее будут решения и прогнозы.

Сегодня примеры машинного обучения можно встретить повсеместно. Например голосовые помощники воспроизводят музыку по команде и ищут в интернете ответы на наши запросы. Веб-сайты рекомендуют человеку продукты, фильмы и песни на основе того, что он покупал, смотрел или слушал раньше. Кроме того роботы пылесосят полы, детекторы спама предотвращают попадание нежелательных писем в почтовые ящики, системы анализа медицинских изображений помогают врачам определять опухоли, которые они могли пропустить. Первые беспилотные автомобили уже используются в качестве такси.

Для решения задач, описанных выше, применяются разные методы машинного обучения. Формально все они делятся на несколько классов:

1. **Обучение с учителем**

Наиболее распространенный вид задач. Каждый элемент выборки представляет собой пару «объект, ответ». Требуется найти зависимость ответов от описаний объектов и построить алгоритм, принимающий на входе описание объекта и выдающий на выходе ответ. Функционал качества обычно определяется как средняя ошибка ответов, выданных алгоритмом, по всем объектам выборки. В рамках данного вида задач выделяются следующие подзадачи:

• *Задача классификации.* Состоит в получении категориального ответа на основе набора признаков. Имеет конечное количество ответов (часто в виде «да» или «нет»). Например является ли животное на фотографии котом.

• *Задача регрессии.* Состоит в прогнозировании вещественного числа на основе набора признаков. Например цену на квартиру на основе ее характеристик.

• *Задача ранжирования.* Отличается тем, что ответы надо получить сразу на множестве объектов, после чего отсортировать их по значениям ответов. Часто возникает в поисковых системах.

• *Задача прогнозирования.* В рамках данной задачи объектами являются данные за временной интервал. Алгоритм же должен предсказать данные в следующие моменты времени. Часто встречается в задачах предсказания стоимости ценных бумаг.

2. **Обучение без учителя.**

Обучение без учителя включает в себя класс задач обработки данных, в которых известны только описания множества объектов (обучающей выборки). В рамках данной задачи требуется найти зависимости: закономерности, внутренние взаимосвязи, зависимости, которые существуют между объектами. В классе существует несколько основных подклассов:

• *Задача кластеризации.* Основная цель заключается в распределение данных на группы (кластеры). Например разделение людей по уровню платежеспособности.

• *Задача уменьшения размерности.* Состоит в сведении большого числа признаков к меньшему, для удобства их последующего использования и визуализации.

3. **Частичное обучение.**

Частичное обучение предлагает золотую середину между обучением с учителем и обучением без учителя. Каждый объект выборки представляет собой пару «объект, ответ», но ответы известны только для части объектов.

4. **Обучение с подкреплением (reinforcement learning).** В рамках данного класса объектами являются пары «ситуация, принятое решение», ответами же являются значения функционала качества, характеризующего правильность принятых решений. Часто используется в обучении роботов.

Есть четыре основных шага для решения задачи методом машинного обучения:

1. **Выбрать и подготовить набор данных для обучения.**

Обучающие данные (выборка) – это набор данных, которые модель машинного обучения получает для решения поставленной задачи. Обычно выборка делится на тренировочную и тестовую. Тренировочные данные используются для обучения алгоритма, а тестовые для проверки его качества. Алгоритм извлекает из данных признаки, на основе которых выбираются оптимальные параметры модели, при которых функционал качества принимает оптимальное значение.

Для обучения хорошего алгоритма данные для обучения должны быть правильно подготовлены – перемешены случайным образом, из них должны быть удалены дубликаты, устранен дисбаланс и смещение, которые могут влиять на обучение.

2. **Выбрать алгоритм.**

На втором шаге в зависимости от класса, к которому относится исходная задача, из перечня алгоритмов машинного обучения выбирается один или множество, которые будут использоваться для ее решения. Например для обучения с учителем может быть выбран случайный лес, логистическая регрессия или др.

3. **Обучение алгоритма.**

Обучение алгоритма - это итеративный процесс. Он включает в себя прогон переменных через алгоритм, сравнение выходных данных и правильных ответов. На основе полученных результатов корректируются параметры алгоритма таким образом, чтобы он давал более точный результат. Далее этот шаг многократно повторяется, пока не будет достигнута необходимая точность. Полученный в результате точный алгоритм представляет собой *модель* машинного обучения.

4. **Использование и улучшение модели.**

Последним шагом является использование модели на новых данных и, в лучшем случае, повышение ее точности и эффективности с течением времени. Откуда будут поступать новые данные зависит от решаемой проблемы. Например, модель машинного обучения, предназначенная для выявления спама, будет принимать сообщения электронной почты, тогда как модель машинного обучения, которая управляет роботом-пылесосом, будет принимать данные, полученные в результате реального взаимодействия с передвинутой мебелью или новыми объектами в комнате.

1.2 Общая постановка и классификация задач оптимального управления

Исторически оптимизация отождествлялась с программированием, так как раньше слова ”программа” было синонимом детерминированного плана. По этой причине многие классы задач оптимизации получили названия, в которых содержится слово «программа» или «программирование». Например термин ”линейная программа” (LP), который является синонимом задачи линейной оптимизации. Даже крупнейшее сообщество математической оптимизации на протяжении десятилетий носило название «Сообщество математического программирования». Однако в 2011 году оно сменило свое название на «Общество математической оптимизации» (MOS), хоть и их главный журнал «Математическое программирование» сохранил свое имя.

Целью математической оптимизации является поиск лучшего или *оптимального* решения среди всевозможного набора решений, причем оптимальность его определяется через заданную функцию. Решения делятся на допустимые и недопустимые в зависимости от того, соответствуют ли они дополнительным условиям. Данные условия формально описываются в виде функций ограничений, которые накладываются на поставленную задачу. Область математической оптимизации включает в себя множество различных классов задач, которые будут кратко описаны ниже.

В общей постановке задачи оптимального управления существует 5 основных характеристик:

1. **Время.**

Существует два типа задач оптимального управления: те, которые рассматриваются на *непрерывном* промежутке времени $T = |t_0, t_f|$ и те, у которых время задается *дискретным* образом: $T = t_1, t_2, ..., t_N$. Первое часто используется в биологических задачах, второе же в теории игр. Кроме того задача может быть поставлена на бесконечном временном интервале или с фиксированным временем окончания процесса. Во втором случае момент окончания называется *горизонтом планирования*.

2. **Состояние и математическая модель системы.**

Аналогично времени состояние системы X может принадлежать конечномерному пространству R^n или бесконечномерному. В первом случае задача оптимизации называется *конечномерной* , во втором *бесконечномерной* или же задачей *в функциональных пространствах*. Кроме того пространство может быть непрерывным или дискретным, что тоже соответствует классификации задач оптимального управления на *дискретные* и *непрерывные*.

Динамика же изучаемого процесса моделируется чаще всего дифференциальными уравнениями:

$$\dot{x}(t) = f(x(t), u(t), t), \tag{1.1}$$

или разностными уравнениями:

$$x(k+1) = f(x(k), u(k), k), k = 0, 1, ..., \tag{1.2}$$

где n-вектор x – это состояние системы, r-вектор u – управление, функция задана $f : R^n \times R^r \times R \to R^n$. Число n называется порядком системы управления, r – числом входов.

В зависимости от функции f есть два основных класса задач:

• *Линейного программирования.* Это класс задач, где функция f является линейной.

• *Квадратичного программирования.* Это класс задач, где функция f является квадратичной и имеет общий вид $f = c^T + \frac{1}{2}x^T Hx$, где H – симметричная матрица.

3. **Класс управлений и ограничения на них.**

В задачах оптимального управления четко указывается класс функций непрерывного процесса управления , из которого выбираются управления. Это могут быть: кусочно-непрерывные, гладкие, измеримые, импульсные функции и т.д. Также задается множество $U \in R^r$ – множество допустимых значений управления. Как правило U — компакт.

Кусочно-непрерывная (измеримая, дискретная и тд.) функция $u(\cdot) = (u(t), t \in [t_0, t_N])$ называется **доступным управлением**, если

$$u(t) \in U, t \in [t_0, t_N].$$

Аналогичное определение имеет место для дискретных систем управления.

4. **Ограничения на фазовую траекторию.**

Аналогично размерности переменной решения размерность функций ограничений может быть конечной или бесконечной. Если присутствует бесконечное число ограничений неравенства, в то время как переменная решения конечномерна, то задача оптимизации называется полубесконечной. Этот класс часто бывает в робастной оптимизации, где нужно найти лучший выбор переменной решения, которая удовлетворяет ограничениям для всех возможных значений неизвестного, но ограниченного возмущения.

Ограничения на переменные состояния могут накладываться в начальный момент времени t_0 :

$$x(t_0) \in X_0, \tag{1.3}$$

и в конечный момент времени t_N:

$$x(t_N) \in X_N \tag{1.4}$$

Ограничения (1.3) и (1.4) называются *терминальными*.

Так же существуют ограничения в изолированные моменты из промежутка управления $t_i \in [t_0, t_N], i = 1, N$:

$$x(t_i) \in X_i, i = 1, \tag{1.5}$$

Ограничения (1.5) называются *промежуточными фазовыми ограничениями*.

Кроме того могут быть заданы ограничения на всем промежутке управления — фазовые ограничения:

$$x(t) \in X(t), t \in [t_0, t_N], \tag{1.6}$$

где $X_0, X_N, X_i, i = 1, m, X(t), t \in [t_0, t_N],$ — заданные множества пространства состояний.

Доступное управление $u(\cdot) = (u(t), t \in [t_0, t_N])$ называется **допустимым (или программой)**, если оно порождает траекторию $x(\cdot)$, удовлетворяющую всем ограничениям задачи.

5. **Критерий качества задач оптимального управления.**

Множество допустимых управлений содержит более одного элемента, поэтому возникает необходимость сравнивать их между собой. Для этого вводится функционал $J(u)$, называемый критерием качества, и выбирается операция минимизации или максимизации этого функционала, результат которой определяет наилучшее (оптимальное) управление. В теории оптимального управления различают четыре типа критериев качества:

(a) критерий качества Майера (терминальный критерий)

$$J(u) = \phi(x(t_N)),$$

(b) критерий качества Лагранжа (интегральный критерий)

$$J(u) = \int_{t_0}^{t_N} f_0(x(t), u(t), t)dt,$$

(c) критерий качества Больца

$$J(u) = \phi(x(t_N)) + \int_{t_0}^{t_N} f_0(x(t), u(t), t)dt, \tag{1.7}$$

(d) задачи быстродействия

$$J(u) = t_N - t_0 \to \min.$$

Все критерии качества эквивалентны между собой.

Допустимое управление $u_0(\cdot)$ называется **оптимальным управлением (оптимальной программой)**, если на нем критерий качества достигает экстремального значения (min или max):

$$J(u_0) = extr J(u),$$

где минимум (максимум) берется по всем допустимым управлениям.

1.3 Постановка задачи

Основной объект исследования настоящей работы — дискретная стационарная нелинейная динамическая система, вида (1.2) с фазовыми ограничениями (1.6). Рассматривается классическая задача оптимального управления – задача маятника. В ней состояние объекта x состоит из одного параметра:

• угол отклонения маятника от вертикальной оси (ϕ)

На него в каждый момент времени накладываются следующие ограничения:

$$\phi_{min} \leqslant \phi_k \leqslant \phi_{max},$$

$$k = \overline{0, N-1}.$$

Всего существует одно управление $u \in [-2, 2]$. В дифференциальном виде (1.1) общая постановка задачи записывается в виде:

$$\ddot{\phi} + \omega^2 \phi = u, \tag{1.8}$$

где $\omega^2 = \dfrac{g}{l}$ – это постоянный коэффициент. Пример данный задачи можно посмотреть на рисунке 4.2.

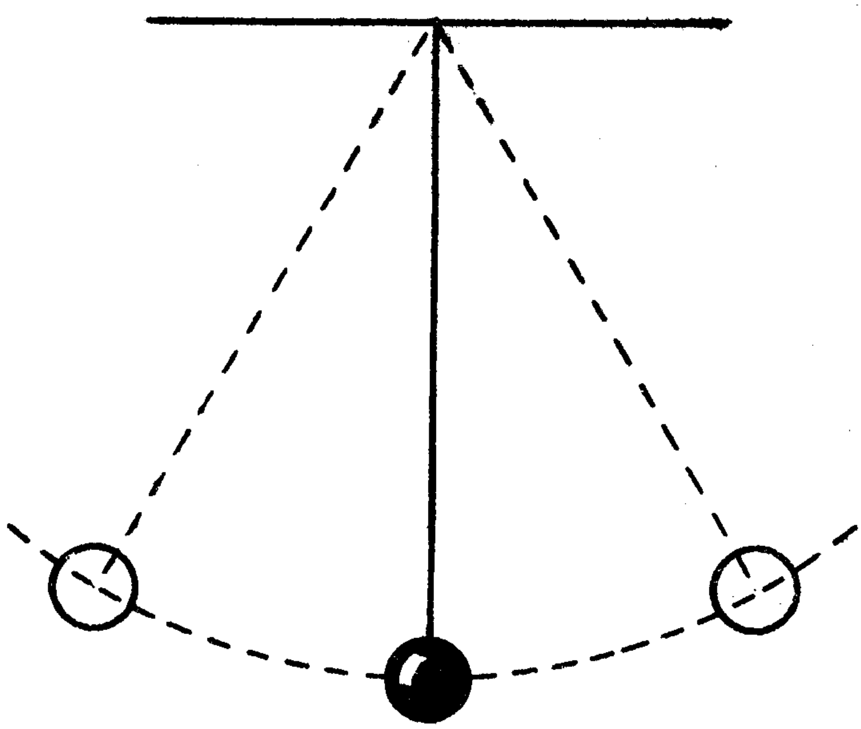

Рисунок 1.1 — Модель маятника

1.4 Управление по прогнозируемой модели (MPC)

Управление по прогнозирующей модели, в англоязычной литературе носящее название Model Predictive Control (MPC), является одним из современных методов теории управления [4, 5]. Подход управления с прогнозирующими моделями начал развиваться в начале 60-х годов XX века для управления процессами и оборудованием в нефтехимическом и энергетическом производстве, для которых применение традиционных методов синтеза было крайне затруднено в связи с исключительной сложностью их математических моделей. В последнее время область применения MPC значительно расширилась, охватывая технологические отрасли и экономику при управлении производством, при решении задач управления запасами и портфелем ценных бумаг.

Основным достоинством MPC-подхода, определяющим его успешное использование в практике построения и эксплуатации систем управления, служит относительная простота базовой схемы формирования обратной связи, сочетающаяся с высокими адаптивными свойствами. Последнее обстоятельство позволяет управлять многомерными и многосвязными объектами со сложной структурой, оптимизировать процессы в режиме реального времени в рамках ограничений на управляющие и управляемые переменные, учитывать неопределенности в задании объектов и возмущений. Кроме того, возможен учет запаздываний, поскольку зачастую решение об управлении принимается в момент времени t – h, а реализация этого решения происходит в момент времени t. Классической областью применения MPC до недавнего времени были задачи стабилизации и слежения в технических приложениях. Теоретические основы метода для задач стабилизации получили строгое обоснование в работах [4, 5].

MPC основывается на последовательном, в каждый момент времени, решении прогнозирующих задач оптимального управления (predictive optimal control problems) с конечным горизонтом управления, сформулированных для математической модели управляемого объекта, начальное условие которой совпадает с измеренным состоянием объекта. Значение оптимального программного управления прогнозирующей задачи на левом конце промежутка управления используется для управления объектом в текущий момент времени и до тех пор, пока не будет получено и обработано следующее измерение состояния.

Управление, которое подается на объект в описанном процессе, представляет собой *обратную связь* (оно зависит от измеряемых состояний), свойства которой зависят от конкретного вида прогнозирующей задачи оптимального управления и целей управления объектом (например, стабилизация, регулирование, слежение и др.).

Методы MPC предполагают построение стабилизирующей обратной связи на основе повторяющегося в каждый текущий момент времени решения задач оптимального управления (ОУ)[10]. Для того, чтобы учесть практическую невозможность мгновенного вычисления решения задач ОУ, предполагается, что состояния среды обрабатываются (измеряются) в дискретные моменты времени $\tau \in \{0, h, 2h, \dots\}$, где $h > 0$ – период квантования, превосходящий время решения задач ОУ. Соответственно, будет строиться дискретная обратная связь, что позволяет определить решение замкнутой системы при заданном начальном состоянии как последовательное решение уравнения (1.1), $x(\tau) = x(\tau - 0)$ на интервалах $t \in [\tau, \tau + h[$.

Общая идея MPC состоит в решении в каждый момент τ так называемой прогнозирующей задачи ОУ с конечным горизонтом $T = Nh$ (N – натуральное число), в которой начальное условие для прогнозирующей модели совпадает с измеренным состоянием $x(\tau)$ объекта управления. В качестве прогнозирующей модели выбирается математическая модель объекта управления, которая может отличаться: это может быть линеаризация, или детерминированная модель, в которой не учитываются возмущения, немоделируемая динамика, другие неопределенности.

Базовый алгоритм MPC состоит в следующем: для каждого t выполнить

1. измерить текущее состояние $x(t)$ объекта управления;
2. решить задачу, получить оптимальное программное управление u
3. подать в среду управляющее воздействие.

Построенная в результате применения данного алгоритма функция $u_{MPC}(t)$, $t \geqslant 0$, является реализацией дискретной обратной связи вида $u = u^0(\tau)$, вдоль траектории измерений, реализовавшейся в конкретном процессе управления. Она обеспечивает асимптотическую устойчивость решения замкнутой системы при ряде дополнительных условий. В частности, известно, что при недостаточно больших горизонтах управления T система может оказаться неустойчивой, поэтому авторами [5] получены нижние оценки параметра T, гарантирующие асимптотическую устойчивость. Нужно отметить, что при таком подходе горизонт управления может оказаться достаточно большим, что отрицательно скажется на трудоемкости решения задачи.

Исторически первый подход состоит в том, чтобы дополнить задачу терминальным ограничением. Недостатками данного подхода являются: 1) при коротких горизонтах планирования прогнозирующая задача ОУ может не иметь решения, 2) двухточечные задачи ОУ являются самыми сложными с вычислительной точки зрения. Устраняют перечисленные недостатки подходы [5], в которых задача дополняется терминальным ограничением , выбирается критерий качества типа Больца (1.7). Асимптотическая устойчивость замкнутой системы в подходах [5] гарантируется в том случае, если существует некоторое множество S, на котором найдется такая локальная обратная связь U, что: 1) множество S является положительно инвариантным для системы $\ddot{x} = f(x, k(x)), x \in S$; 2) при всех $x \in S$ имеет место включение $k(x) \in U$.

В рамках поставленной задачи (1.8) необходимо найти последовательность управлений $u = \{u_0, \ldots, u_{N-1}\}, u_i \in [-2, 2]$, в результате чего будет достигнуто максимальное значение критерия качества. То есть если задано начальное состояние $\overline{x_0} \in \mathbb{R}^{\ltimes}$, то с помощью управления по прогнозирующей модели в каждый момент времени $k = \overline{1, N}$ находится последовательность управляющих воздействий с горизонтом планирования T: $\{u'_0, \ldots, u'_{T-1}\}, u'_i \in [-2, 2]$, в систему подается управление $u_k = u'_0$, возвращается ее будущее состояние и заново перестраивается последовательность управлений с горизонтом планирования T. Таким образом находится траектория решения задачи:

$$x(\overline{x_0}, u),$$

соответствующая начальному состоянию x_0 и последовательности управлений u.

1.5 Выводы

В рамках данной главы была представлена общая характеристика и рассмотрена классификация задач оптимального управления. На основе чего была выбрана и поставлена задача, решаемая в рамках данной работы. Также были рассмотрены основные принципы и классификация методов машинного обучения. В результате из всех разновидностей выбран метод обучения с подкреплением для решения поставленной задачи. Также дан обзор управления по прогнозируемой модели. Более подробно он рассматривается в следующей главе.

ГЛАВА 2
ОБУЧЕНИЕ С ПОДКРЕПЛЕНИЕМ

Методы обучения с подкреплением являются классом методов машинного обучения. В своей основе они предполагают отсутствие учителя и знаний о среде. Таким образом им не требуется наличие источника с базой примеров правильного поведения для обучения, в связи с чем они являются самыми эффективными для задач из теории игр (шахматы, го и другое).

В рамках задачи обучения с подкреплением перед агентом стоит задача о нахождении эффективной стратегии поведения в среде, дающей максимальное суммарное вознаграждение. Основным способом обучения в этом методе является непосредственное взаимодействие со средой, когда агент методом проб и ошибок, от попытки к попытке, улучшает свою стратегию поведения в среде. Такая идея возникла на основе наблюдения за ребенком, который обучается за счет эмпирических данных, полученных от контакта с окружающей средой.

Одним из самых первых выдающихся результатов данного метода является программа TG-Gammon, которая учится играть в нарды. В результате обучения программа смогла играть с действующими на тот момент чемпионами в нарды [3]. Кроме того анализ ее стратегий способствовал изменению общепринятых способов разыгрывания партий.

2.1 Основные понятия обучения с подкреплением

Целью методов обучения с подкреплением является обучение агента, взаимодействующего с окружающей средой. В основе данного обучения лежит гипотеза о вознаграждении: любая цель может быть описана с помощью задачи о максимизации ожидаемого совокупного вознаграждения.

Определение 1. Вознаграждение, или подкрепляющий сигнал (reward, r), – это число, отражающее, насколько удачно поведение агента на текущем шаге.

Таким образом задача агента – максимизировать общее вознаграждение. Для этого агенту нужно научиться выбирать последовательность действий, которые принесут максимальную награду. Важно, что награда может быть отложенной, если действие удачное и вызывает долгосрочные последствия. Поэтому иногда нет смысла обращать внимание на мгновенное вознаграждение и следить за долгосрочной наградой. Например, для задачи инвестирования в ценные бумаги необходимо подождать как минимум неделю, чтобы понять привело ли действие к прибыли.

В основе обучения с подкреплением лежит модель агент-среда (рисунок 2.1). Исходя из нее агент может понимать свое текущее состояние в среде (current state, s), наблюдать окружающее пространство (observation, o), на основе наблюдений предпринимать действия (actions, a), переходить в новое состояние и получать вознаграждение (reward, r). Схематически это можно увидеть на рисунке 2.1.

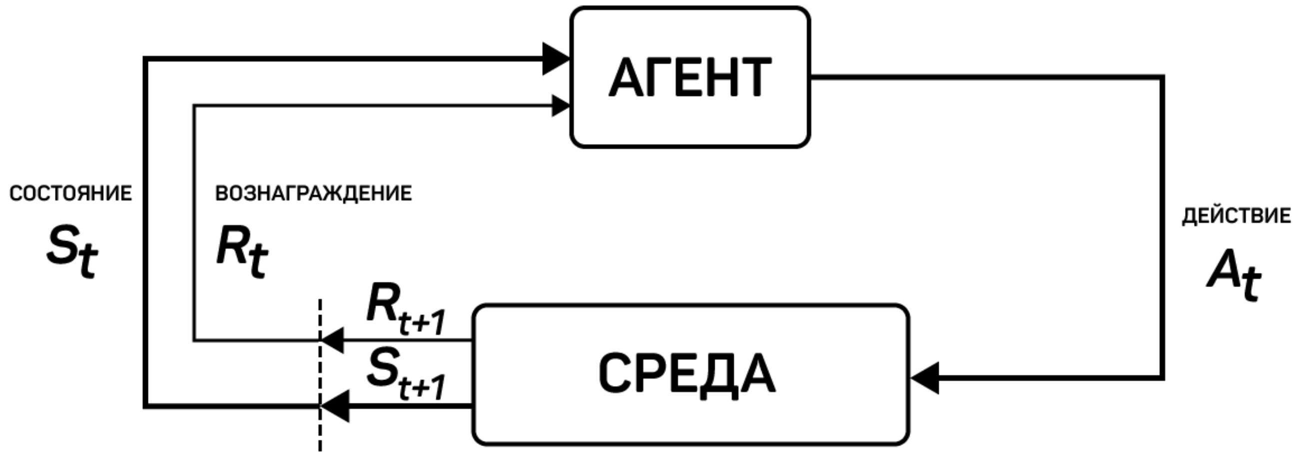

Рисунок 2.1 — Агент и среда

Агент взаимодействует со средой в дискретные моменты времени $t = 0, 1, \ldots$.

В каждый момент времени t агент получает информацию о состоянии среды

$$s_t \in S,$$

где S – множество состояний среды. Состояние может включать в себя низкоуровневые параметры (силу тока) и высокоуровневые (текущее время). Также внутренние параметры агента могут входить в состояние среды, то есть граница между средой и агентом не всегда совпадает с их физической границей.

Далее агент выбирает действие на основе состояния:

$$a_t \in A(s),$$

где $A(s)$ – множество возможных действий, доступных в состояния s.

После исполнения действия агент получает новое состояние s_{t+1}, и награду r_{t+1}.

Такую модель действий агента обычно называют моделью с обратной связью [7].

Выбор агента о том, какое действие предпринять, основывается на истории.

Определение 2. Историей h_t называют последовательность из действий, наблюдений и вознаграждений агента до момента времени t.

$$h_t = \{a_1, s_1, r_1, \ldots, a_t, s_t, r_t\}. \tag{2.1}$$

В ответ на действие среда возвращает наблюдение и награду, которые записываются в историю. Таким образом состояние среды является функцией от истории (2.1):

$$s_{t+1} = f(h_t)$$

В рамках нашей задачи агент получает полное состояние окружающей среды, причем оно является марковским.

Определение 3. Состояние называется марковским тогда и только тогда, когда выполнено свойство марковости:

$$P[s_{t+1}|s_t] = P[s_{t+1}|s_1, \dots, s_t],$$

т.е. будущее состояние s_{t+1} зависит только от текущего состояния s_t и не зависит от прошлых. Таким образом, зная текущее состояние, все дальнейшие состояния могут быть получены без знания истории.

Свойства марковских процессов (Markov decision process – MDP), позволяют формально описывать среду и её модели для различных видов задач обучения с подкреплением.

MDP описывается пятью параметрами (S, A, P_s^a, γ, R):

- S – множество возможных состояний системы,
- A – множество действий агента,
- P_s^a – матрица вероятностей перехода между состояниями. Т. е. состоящую из вероятностей $P_{ss'}^a$ перехода из состояния $s \in S$ в состояние $s' \in S$, предприняв действие $a \in A$.
- γ – дисконтирующий множитель. С помощью него отражается задержка в награде. $\gamma \in [0, 1]$,
- R - функция наград. $R : S \times A \to \mathbb{R}$:

$$R_s^a = \mathbb{E}[r_{t+1}|r_t = s, r_t = a].$$

Пример диаграммы переходов для марковского процесса принятия решений можно увидеть на рисунке 2.2.

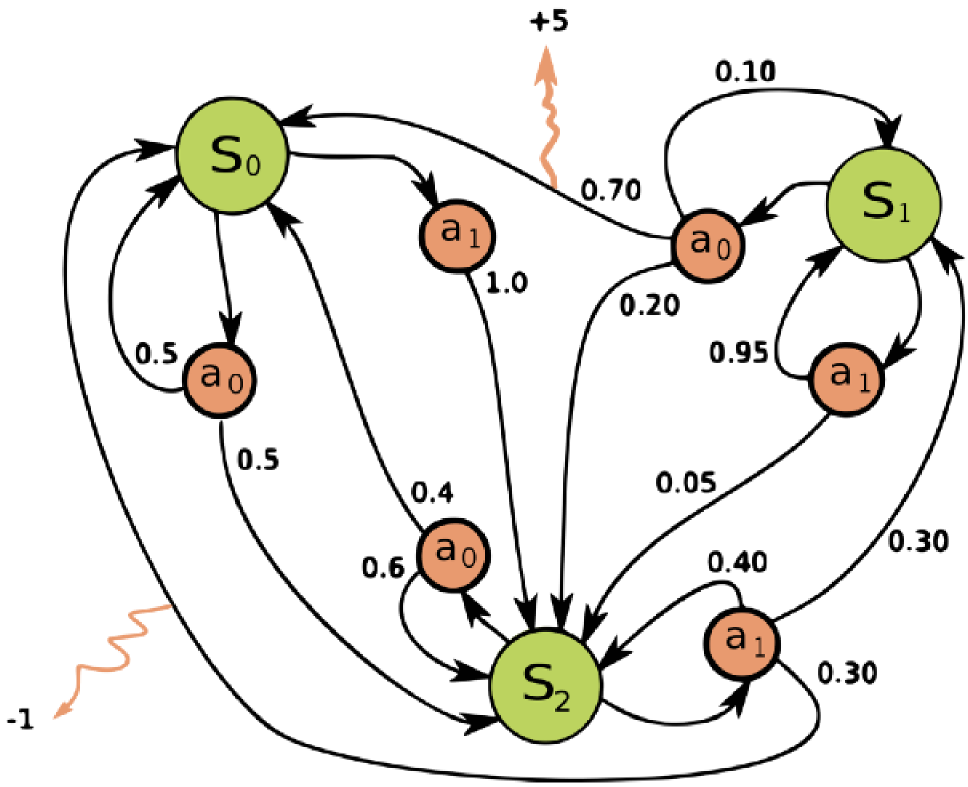

Рисунок 2.2 — Марковский процесс принятия решений

Среда для маятника является непрерывной и сводится к MDP при помощи техники дискретизации [9].

Определение 4. Доходом g_t называется случайная величина, равная дисконтированной сумме вознаграждений агента после момента времени t.

$$g_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots = \sum_{k=1}^{\infty} \gamma^k r_{t+k}. \tag{2.2}$$

Определение 5. Гипотеза о награде Сьюттона: вместо максимизации суммарной награды можно ставить агенту целью максимизацией суммы наград, полученных на последующих шагах при выборе текущего действия.

Таким образом задачу агента можно переопределить как максимизацию ожидания дохода (2.2).

2.2 Структура агента

В задаче оптимального управления ключевыми являются три следующие компоненты:

1. **Policy** – стратегия поведения.

Стратегия π – это стратегия поведения агента по отношению к среде.

Стратегия (Policy function) – функция, которая ставит в соответствие состоянию агента предпринимаемое им действие. Детерминированная стратегия имеет вид:

$$a = \pi(s), s \in S,$$

Стохастическая стратегия:

$$\pi(a|s) = P[a_t = a|s_t = s]. \tag{2.3}$$

2. **Value function** – функция ценности действия.

Функция ценности (Value function) – функция, предсказывающая возможное вознаграждение, которое получит агент, находящийся в момент t в состоянии s_t.

$$\nu(s) = \mathbb{E}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots |s_t = s]. \tag{2.4}$$

(2.4) также можно определить через доход (2.2):

$$\nu(s) = \mathbb{E}[g_t|s_t = s]. \tag{2.5}$$

3. **Model** – модель среды.

Модель среды – модель, которая предсказывает что будет со средой в следующий момент времени.

В основе модели лежат модель переходов и модель вознаграждений, которые показывают вероятность следующего состояния в зависимости от текущего и ожидаемую награду соответственно:

$$\Phi^a_{ss'} = P[s_{t+1} = s'|s_t = s, a_t = a], \tag{2.6}$$

$$\Psi^a_s = E[r|s_t = s, a_t = a]. \tag{2.7}$$

Сама же модель среды описывается как функция:

$$s_{t+1} = m(s_t, a_t).. \tag{2.8}$$

Определение 6. Функция стоимости действия (action-value function) Марковского процесса принятия решений – есть ожидаемый доход, начиная с состояния s, при принятии действия a и в дальнейшем следуя стратегии π.

$$q_\pi(s, a) = \mathbb{E}_\pi[g_t|s_t = s, a_t = a]. \tag{2.9}$$

На основе введенных определений (2.3) и (2.5) помощью уравнения Беллмана выводится следующее представление:

$$\nu_\pi(s, a) = \mathbb{E}_\pi[r_{t+1} + \gamma\nu_\pi(s_{t+1})|s_t = s], \tag{2.10}$$

Из (2.9) и (2.10) получается следующий удобный вид функции q):

$$q_\pi(s, a) = \mathbb{E}_\pi[r_{t+1} + \gamma q_\pi(s_{t+1}, a_{t+1})|s_t = s, a_t = a]. \tag{2.11}$$

2.3 Q-обучение

Определение 7. Q-learning (q-обучение) — это один из алгоритмов реализации метода машинного обучения с подкреплением (reinforcement learning, RL). Часто используется в задачах, в которых отсутствует модель среды.

Задача обучения с подкреплением формулируется как нахождение стратегии $\pi^*(s)$, которая максимизирует математическое ожидание дохода (2.5): $\mathbb{E}[g_t] \rightarrow \max$. Тогда оптимальную стратегию в текущем состоянии s можно записать:

$$\pi^*(s) = \arg\max_a q(s, a). \tag{2.12}$$

На основе (2.9) и (2.12) введем следующую функцию:

$$q^*_\pi(s, a) = \mathbb{E}_{\pi^*}[g_t | s_t = s, a_t = a] \tag{2.13}$$

Таким образом вместо поиска $\pi^*(s)$ наша задача сводится к поиску $q^*(s, a)$

Используя формулу (2.11) и на основе (2.13) метод аппроксимирует значения оптимальной функции выигрыша q^* следующим образом:

$$\begin{aligned} q^*_\pi(s, a) = \mathbb{E}_\pi[r_{t+1} + \gamma q_\pi(s_{t+1}, a_{t+1}) | s_t = s, a_t = a] = \\ \frac{1}{N}\left(\sum_i r_i + \gamma \max_{a'} q(s_i^{next}, a')\right) = \\ \ldots . \end{aligned} \tag{2.14}$$

Алгоритм состоит из следующих шагов:

- Произвольно инициализируются a, s, r, s', функция $q(s, a) \forall s \in S, a \in A$.
- Рассматриваются возможные действия a и находится новое действие $a' = \arg\max_{a \in A} q(s', a)$.
- Берется среднее взвешенное от уже имеющейся функции стоимости и подсчитанной на текущем шаге на основе (2.14):

$$q(s, a) = \gamma[R(s, a) + \gamma q(s', a')] + (1 - \gamma) q(s, a). \tag{2.15}$$

Схему работы алгоритма q-обучения можно увидеть на рисунке 2.3.

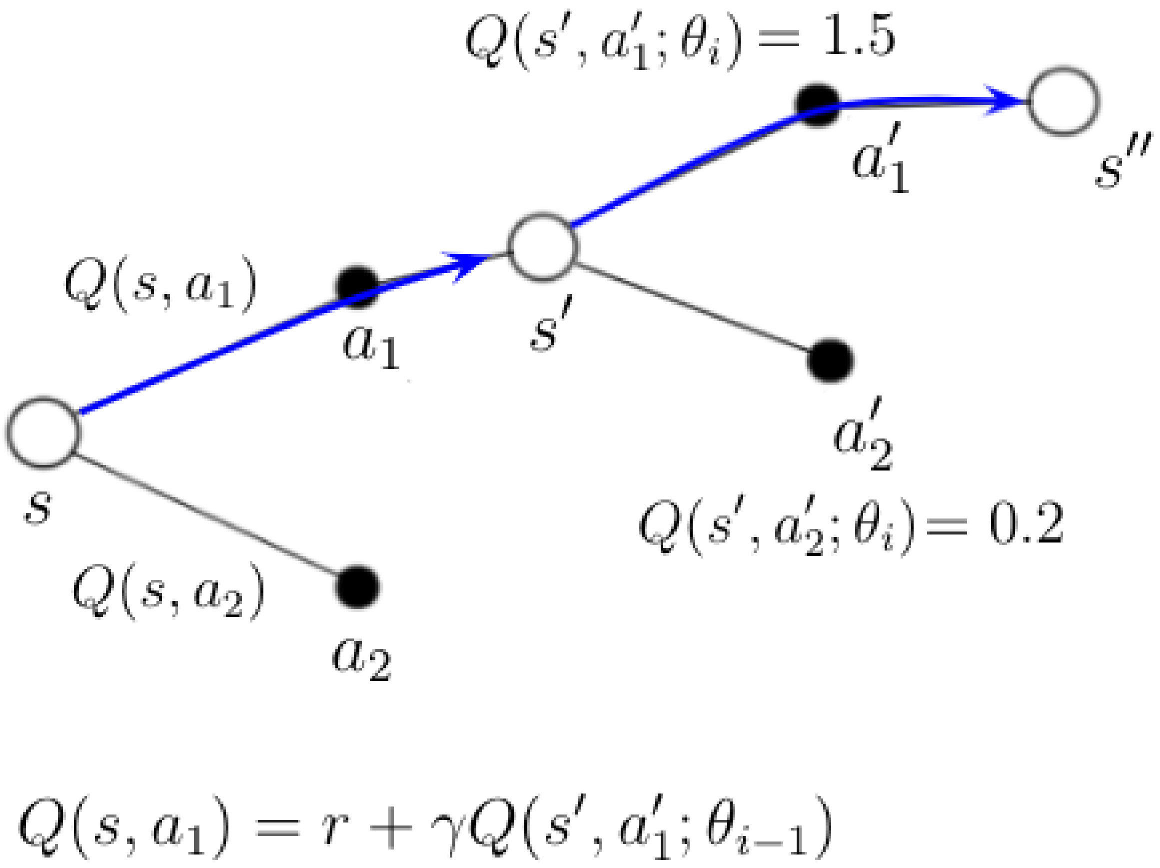

Рисунок 2.3 — Пересчет $q(s, a)$

В качестве метода аппроксимации функции $q(s', a')$ из (2.15) часто используется нейронная сеть. Для оптимизации в качестве функции ошибки выбирается следующая:

$$L = \frac{1}{N} \sum_{i=1}^{N} \left(q(s, a) - [r(s, a) + \gamma \max_{a'} q(s', a')]^2 \right),$$

где s, a, R, s' – текущее состояние, действие, награда и следующее состояние соответственно.

Сама нейронная сеть для $q(s', a')$ представлена на рисунке 2.4.

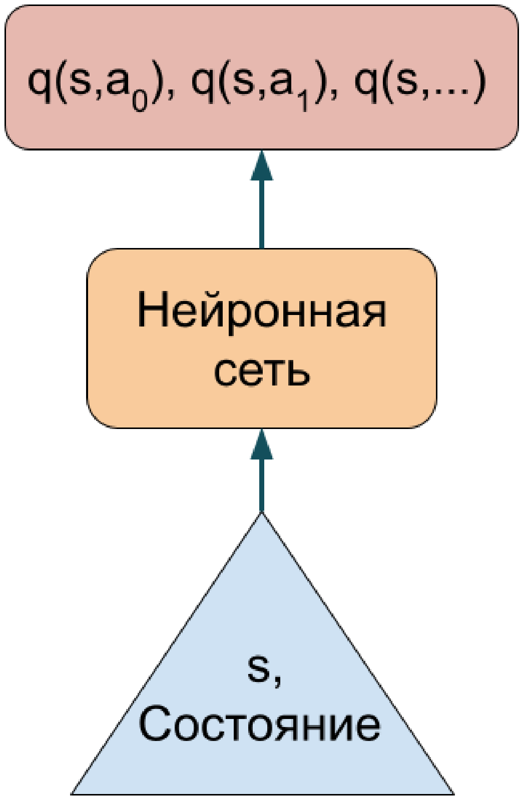

Рисунок 2.4 — Модель обучения нейронной сети

2.4 Епсилон-жадный алгоритм

В рамках подхода q-обучения всегда выбирается действие в зависимости от получаемой награды. Таким образом политика агента заключается в выборе такого действия, которое в тот же момент времени принесет максимальную возможную награду в данном состоянии (2.12). Однако иногда суммарный доход будет больше, если в текущем состоянии взять не максимальную награду, но попасть в состояние, из которого в дальнейшем можно получить награду намного больше.

В рамках эпсилон-жадного алгоритма действий агент не только использует накопленные знания, но и немного исследует среду. Эпсилон-жадный алгоритм большую часть времени выбирает действие с максимальной наградой. Цель его состоит в том, чтобы найти баланс между выбором нового и локально оптимального.

Данный подход заключается в том, чтобы с небольшой вероятностью ϵ производить исследование среды – выбирать случайное действие. А с вероятностью $1-\epsilon$ выбирать действие с максимальной наградой. Схематически это представлено на рисунке 2.5.

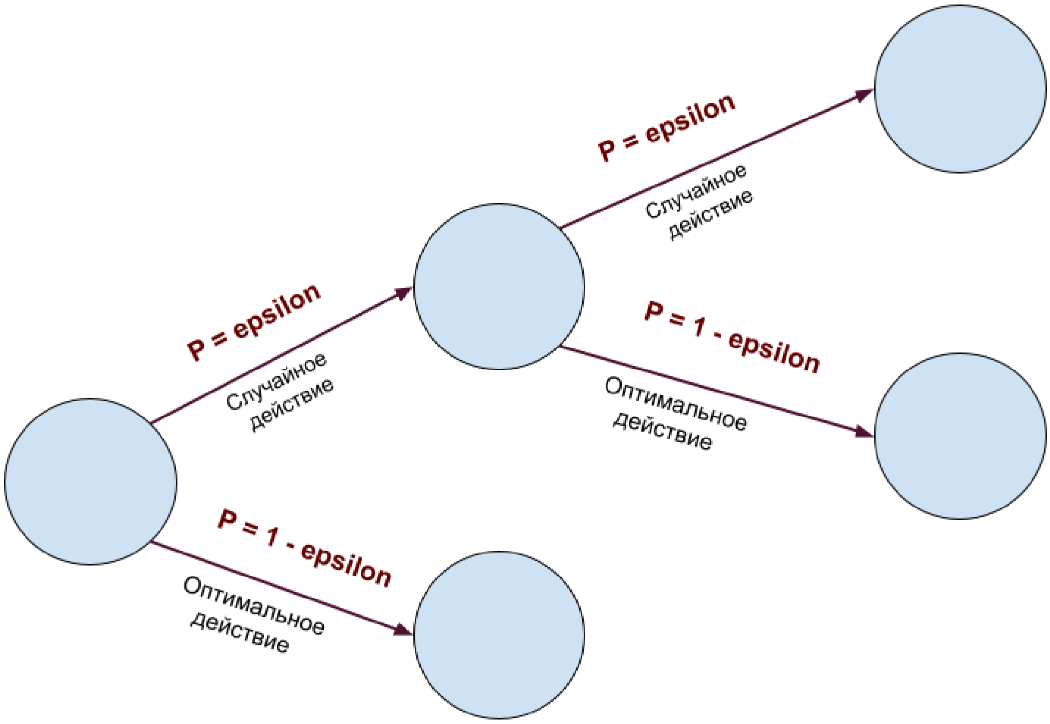

Рисунок 2.5 — Эпсион-жадный алгоритм.

2.5 Выводы

В рамках данной главы была дана общая характеристика обучения с подкреплением. Была описана модель агент-среда, марковский процесс принятия решений и представлен алгоритм q-обучение для решения поставленной задачи. Также было описано его улучшение в виде эпсилон-жадного алгоритма.

ГЛАВА 3

УПРАВЛЕНИЕ ПО ПРОГНОЗИРУЕМОЙ МОДЕЛИ В MODEL-BASED ОБУЧЕНИИ С ПОДКРЕПЛЕНИЕМ

3.1 Марковский процесс принятия решений для в рамках управления с прогнозирующими моделями

В рамках управления по прогнозируемой модели в задачах оптимального управления с ограничениями марковский процесс принятия решений несколько трансформируется. Кроме перечисленных в главе 2 характеристик (состояние среды, множество возможных действий, модель среды, функции наград) к нему добавляется функция стоимости $c : S \to \{0, 1\}$. Она является индикатором, где 0 значит, что ограничения выполнены, а 1 – нарушены.

В рамках данного подхода мы считаем, что модель среды (функции (2.7) и (2.6)) и функция стоимости неизвестны и должны быть найдены на основе собранных данных. Пусть $J_r(\pi)$ обозначает ожидаемую доходность при политике π относительно функции вознаграждения r, а $J_c(\pi)$ – доходность при политике π относительно функции стоимости c:

$$J_r(\pi) = E_\pi[\sum \lim_{t=0}^{T-1} r(s_{t+1})], \tag{3.1}$$

$$J_c(\pi) = E_\pi[\sum \lim_{t=0}^{T-1} c(s_{t+1})], \tag{3.2}$$

где T – это горизонт планирования.

Как было сказано выше, основная цель – максимизировать ожидаемую доходность (2.2). При наложенных ограничениях задача состоит в том, чтобы максимизировать ожидаемую доходность относительно вознаграждения и ограничить ожидаемую доходность относительно функции стоимости. В общем виде это может быть записано следующим образом:

$$\begin{aligned} \pi^* = \arg\max_\pi J_r(\pi), \\ J_c(\pi^*) \leqslant d, \end{aligned} \tag{3.3}$$

где d – заданный лимит на количество нарушений ограничений, а функции $J_r(\pi)$ и $J_c(\pi)$ заданы по формулам (3.1) и (3.2) соответственно.

3.2 Управление с прогнозирующими моделями (MPC)

Напомним, что как следует из главы 1 в основе MPC лежат два основных принципа:

1. Использовать динамическую модель для предсказания состояния системы и улучшения предсказания для принятия оптимального решения – *управления* в текущий момент времени.
2. Использовать предыдущие состояния системы для инициализации состояния динамической модели.

Также напомним, что MPC применяется для решения задач стабилизации и в этом случае выбираются специальные критерии качества, которые на оптимальном значении представляет собой функцию Лапунова для стабилизируемой системы. Прогнозируемая задача оптимального управления в данном случае – задача на минимум.

В терминах исследуемой задачи цель MPC — максимизировать суммарную награду (2.2), выбирая последовательность действий $U = (a_0, \ldots, a_T)$, где T – это горизонт планирования. За один шаг к системе применяется оптимальное управление (действие) и получаются новые наблюдения, после чего заново пересчитываются следующие оптимальные действия. Таким образом находясь в состоянии s_t в момент времени t целью MPC является найти:

$$\begin{aligned} U = \arg \max_{a_0, a_1, \ldots, a_T} [\mathbb{E} \sum_{t=0}^{T} \gamma^t r(s_{t+1})], \\ c(s_{t+1}) = 0, \\ \forall t \in 0, 1, \ldots, T-1. \end{aligned} \tag{3.4}$$

Визуально представить себе работу управления по прогнозирующей модели можно так, как показано на рисунке 3.1.

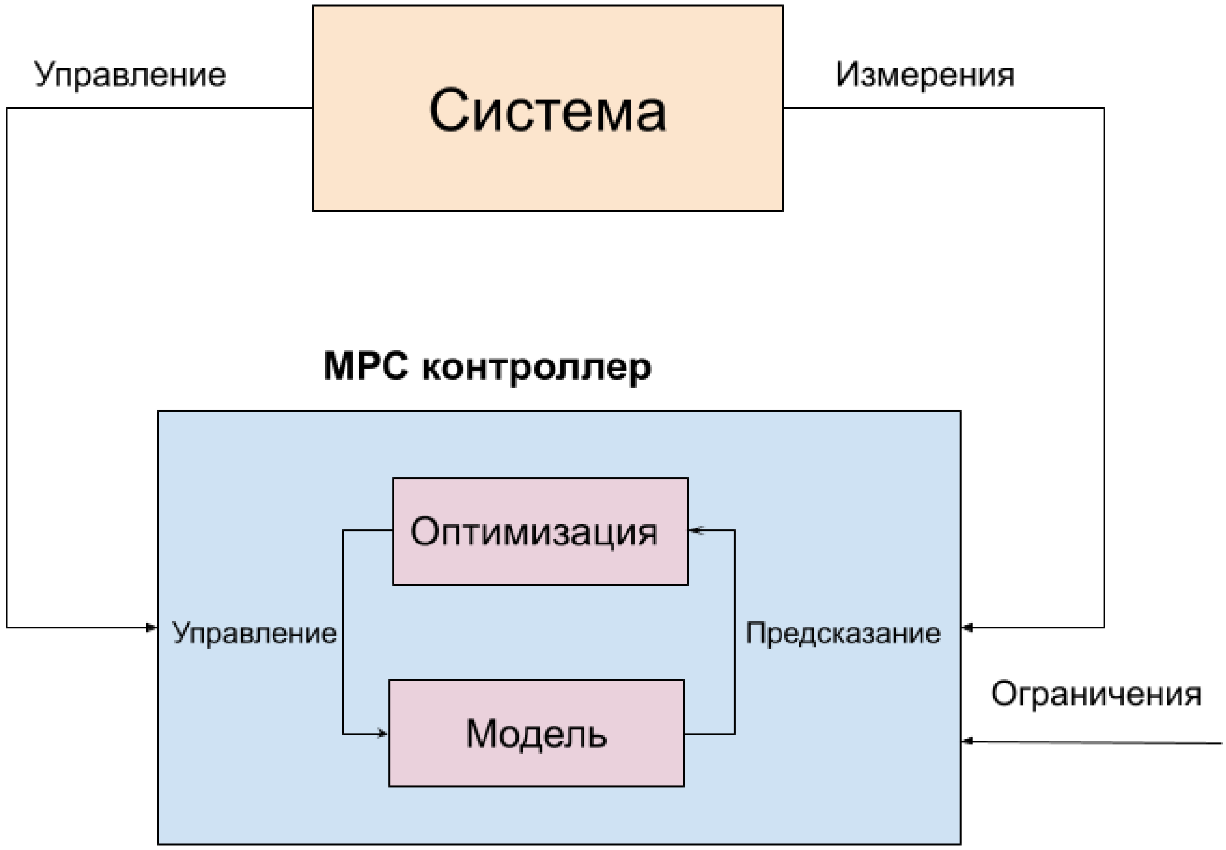

Рисунок 3.1 — MPC

3.3 Model-based обучение с подкреплением

В данный момент существует два основных подхода в обучении с подкреплением: model-based и model-free. Основное их отличием является взаимодействие ос средой и представление о ней. Model-free алгоритмы обучаются на собственном опыте при запуске в реальной среде для обучения (компьютерные игры, реальный мир). Подход model-based же контактирует с реальной средой намного меньше и на этапе обучения старается заменить ее собственным аналогом среды. Таким образом одной из основных задач model-based алгоритмов является построение качественной модели среды (2.8).

Как было сказано в предыдущем разделе, в рамках исследуемой задачи один из способов решения будет основан на отсутствии знаний о среде. Таким образом мы должны получить представление о ней. А именно аппроксимировать функции (2.7) и (2.6) и получить модель (2.8). Это происходит на основе генерации выборок из реальной среды. В качестве аппроксимации модели среды (2.8) часто используется нейронная сеть.

Полный цикл обучения модели и работы управления с прогнозирующими моделями представлен на рисунке 3.2.

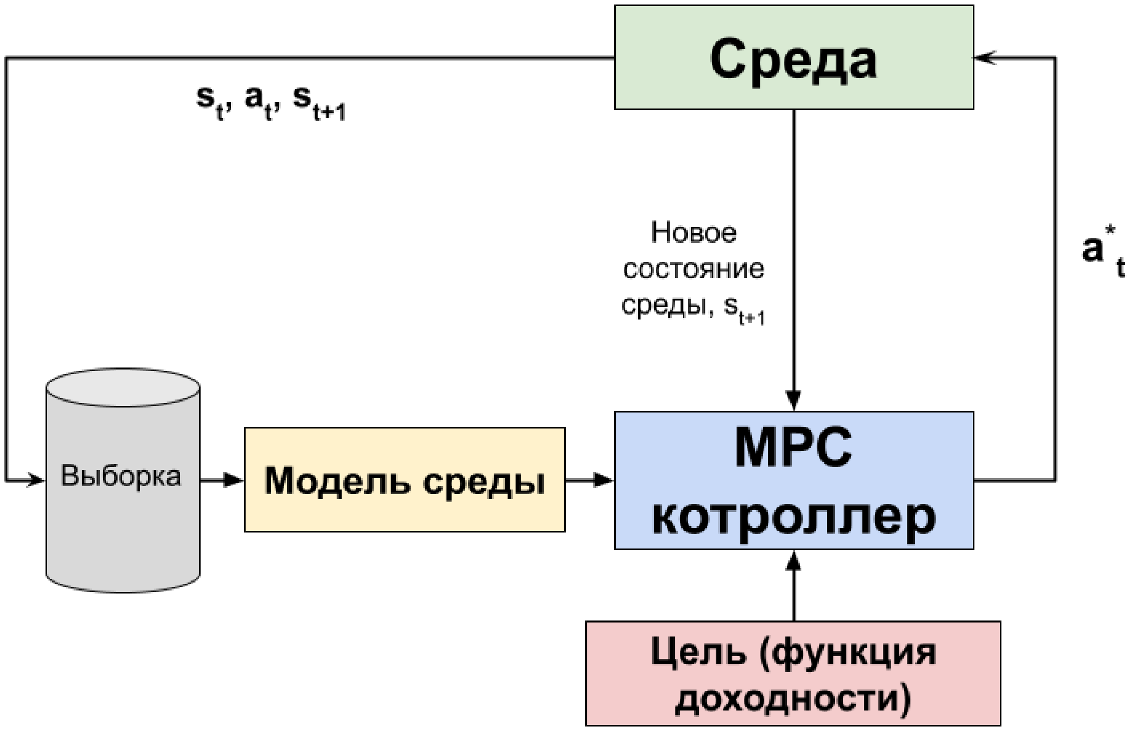

Рисунок 3.2 — Модель обучения MPC

3.4 Метод кросс-энтропии для задач оптимизации

Метод кросс-энтропии (CEM) это метод стохастической оптимизации, основанный на генерации выборок (сэмплировании траекторий, СТ) [11]. С недавнего времени он часто используется в задачах обучения с подкреплением. В рамках данного метода предполагается, что T-размерное решение $U \in R^T$ (3.4) получено из T-размерного факторизированного многомерное распределение Гаусса с параметром θ.

Тогда мы имеем выборку $U \sim N(\theta)$, где $\theta = (\mu, \Sigma)$, μ – T-вектор, а Σ – диагональная TxT матрица ковариаций.

Основная идея метода заключается в том, чтобы генерировать решения итеративно из распределения, которое близко к распределению, из которого получены предыдущие сгенерированные результаты с высокой наградой. Алгоритм останавливается, если достигнуто максимальное количество итераций или $|\Sigma| > \epsilon$, где ϵ – заданный фиксированный порог на матрицу ковариаций.

Одним из способов решения исследуемой задачи оптимизации (3.3) является робастный метод кросс-энтропии (RCE), основанный на методе кросс-энтропии. Он базируется на методе сэмплирования траекторий (СТ) [6] для оценки вознаграждения и стоимости нарушения ограничений. Определим решение (3.4) как последовательность действий размера горизонта планирования

T :

$$U = (a_0, a_1, ..., a_{T-1}).$$

При начальном состоянии s_0 , модели среды $m_\theta(s_t, a_t)$ (2.8), функции стоимости $c(s) \in 0, 1$, мы можем оценить накопленное вознаграждение следующим образом:

$$r(X, s_0) = \sum_{t=0}^{T} \gamma^t r(s_{t+1}), \tag{3.5}$$

$$c(X, s_0) = \sum_{t=0}^{T} \beta^t c(s_{t+1}), \tag{3.6}$$

где $s_{t+1} = m_\theta(s_t, a_t), \forall t \in \{0, \dots, T-1\}$ (2.8), γ и β – дисконтирующие множители. Данные формулы являются аппроксимацией формул (3.1) и (3.2) для решения задачи (3.3) в рамках управления по прогнозируемой модели.

Награда $r(s)$ может быть либо предопределенной, заранее функцией, либо изучена параллельно с моделью.

Интуитивное представление о работе данного метода на примере задачи о роботе показано на рисунке 3.3. Точки на синей линии и точки на оранжевой линии представляют собой две реальные траектории, эллипсы представляют собой неопределенность прогноза модели среды на основе наблюдений и последовательности действий. На рисунке видно, что награда за траекторию B должна быть выше, чем за траекторию A, потому что выбор B приведет к в достижению цели быстрее. Однако траектория A предпочтительнее, потому что робот, следующий по траектории B, может пройти через пламя и нарушить ограничения безопасности.

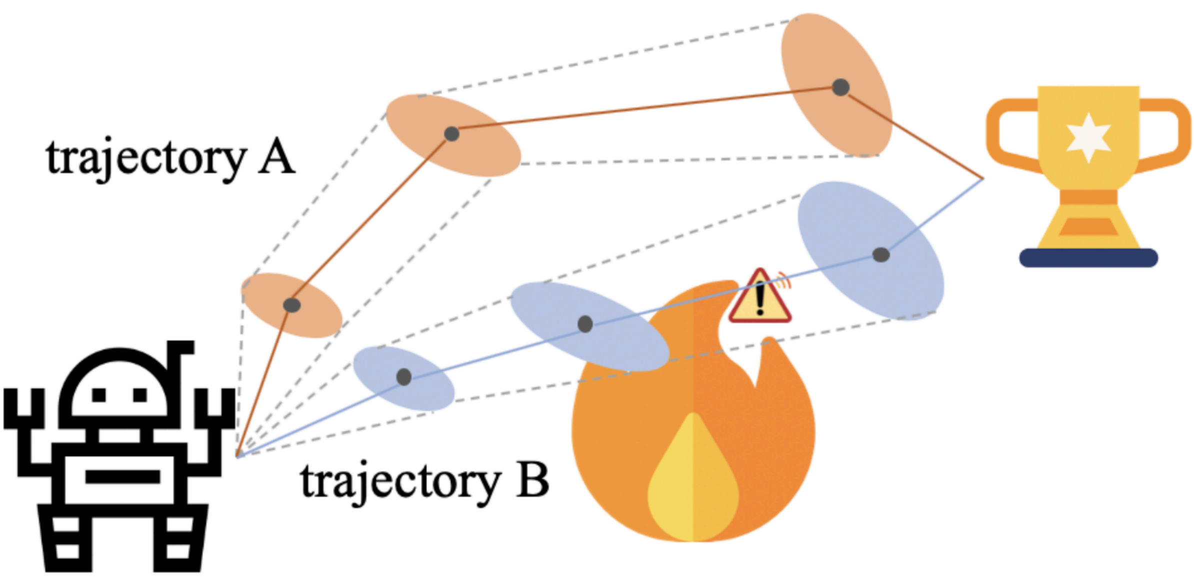

Рисунок 3.3 — Метод кросс-энтропии

Без сэмплирования траекторий, траектория B потенциально может быть предсказана как безопасный маршрут из-за ошибки прогноза модели. С СТ, оценка неопределенности нашей модели имеет небольшой шанс охватить небезопасную зону, поэтому траектория B будет классифицироваться как небезопасная. Поскольку СТ оценивает стоимость траектории при наихудшем сценарии среди всех отобранных маршрутов, он более надежен, когда прогноз модели среды не очень точный.

Обозначим функцию плотности распределения как $p(U, \theta)$. Тогда алгоритм состоит из следующих шагов:

1. Выбираем набор решений.
2. Считаем для него функции r, c по формулам (3.5) и (3.6)
3. Проверяем достижимы ли данные решения.
4. Пересчитываем параметры распределения на основе лучших k решений.

Псевдокод описанного выше алгоритма выглядит следующим образом:

Algorithm 1 Метод робастной кросс-энтропии

Начальная инициализация параметра распределения θ, размерность выборки T, количества отбираемых элементов из выборки k, начального состояния s_0.
Генерация выборки U с высокой наградой
while Критерий остановки не выполнен **do**
    Сгенерировать N примеров из исходного распределения $U_1, U_2, \ldots, U_N \sim N(\theta)$.
    Для каждого элемента выборки U_i вычислить $r(U_i, s_0)$ и $c(U_i, s_0)$.
    Выбрать подходящее подмножество элементов $\Omega \in \{U_i\}_{i=1}^N$ не нарушающих ограничения на основе функции стоимости.
    if $\Omega = \emptyset$ – пустое множество **then**
        Отсортировать $\{U_i\}_{i=1}^N$ в порядке убывания по функции стоимости; задать множество Δ_k – из топ k элементов после сортировки
    else
        Отсортировать множество Ω в порядке убывания по награде. Задать множество Δ_k из первых k элементов, если $|\Omega| > k$, или $\Delta_k = \Omega$.
    Обновить θ с помощью метода максимального правдоподобия: $\theta = arg \max_\theta \prod_{U \in \Delta_k} p(U, \theta)$.
Вернуть U^* с самой высокой наградой из Δ_k.

Для исследуемой задачи мы будем использовать модернизированный робастный метод кросс-энтропии. А именно кроме поиска набора управления U с помощью алгоритма 1, описанного выше, мы параллельно будем обучать модель среды (2.8) и функцию стоимости $c(s_{t+1})$. Псевдокод для нашего алгоритма можно увидеть ниже:

Algorithm 2 Метод робастной кросс-энтропии

Начальная генерация выборки $D = \{s_1, a_1, \ldots, s_N, a_N\}$, инициализация параметра $p(U, \theta)$.
while Количество итераций меньше максимального **do**
    Тренировать модель среды $m(s_t, a_t)$ и функцию стоимости c на основе выборки D.
    for t=0 до конца длины эпизода **do**
        Получить состояние среды s_t
        Оптимизировать действие согласно алгоритму 1:
$\{a_{i^*}\}_{i=t}^{t+T} = RCE(p, s_t)$
        Выбрать действие и подать его в систему
        Получить из среды следующее состояние s_{t+1} и функцию стоимости $c(s_{t+1})$
        Обновить выборку $D = D \bigcup \{s_t, a_t, s_{t+1}, c(s_{t+1})\}$.
Вернуть X^* с самой высокой наградой из Δ_k.

3.5 Выводы

В данной главе представлена теория model-based подхода для решения задач оптимального управления. Описана общая теория управления по прогнозируемой модели. Приведен алгоритм кросс-энтропии для построения горизонта управления и его усовершенствование – метод робастной кросс-энтропии.

ГЛАВА 4

ПОЛУЧЕНННЫЕ РЕЗУЛЬТАТЫ

4.1 Обзор библиотеки gym

Для решения исходной задачи используется библиотека gym, в которой имеется ее реализация. Библиотека gym – это open-source набор инструментов для разработки и сравнения алгоритмов обучения с подкреплением. Данная библиотека не делает никаких предположений о структуре агента и совместима с любой библиотекой численных вычислений, такой как TensorFlow или Pytorch. В ней собраны и реализованы классические задачи оптимального управления и несколько игровых задач.

Сама библиотека симулирует среду. Таким образом агент может подавать в среду действие, а на выходе получать награду и состояние среды. Самый простой способ запустить ее вызов на 1000 шагов с выбором случайного действия из набора всех возможных действий представлен ниже.

```
import gym
env = gym.make('Pendulum-v0')
env.reset()
for i in range(1000):
    env.render()
    env.step(env.action_space.sample()) # take a random action
env.close()
```

Таким образом `env` – это непосредственно среда. Для сброса ее в начальное состояние используется команда `env.reset()`. `env.redner()` рисует картинку полученного состояния текущей среды. Пример можно найти на рисунке 4.1. Множество всех доступных действий можно найти с помощью команды `env.action_space`. Чтобы совершить действие нужно вызвать команду `env.step(action)` и передать в него необходимое действие (`action`). В ответ на это среда вернет четыре параметра `(observation, reward, done, info)`. Которые обозначают следующее:

- observation – состояние среды, наблюдаемое после совершенного действия.
- reward – полученная награда от совершенного действия.
- done – завершена ли текущая задача.
- info – вспомогательная информация, используемая для отладки.

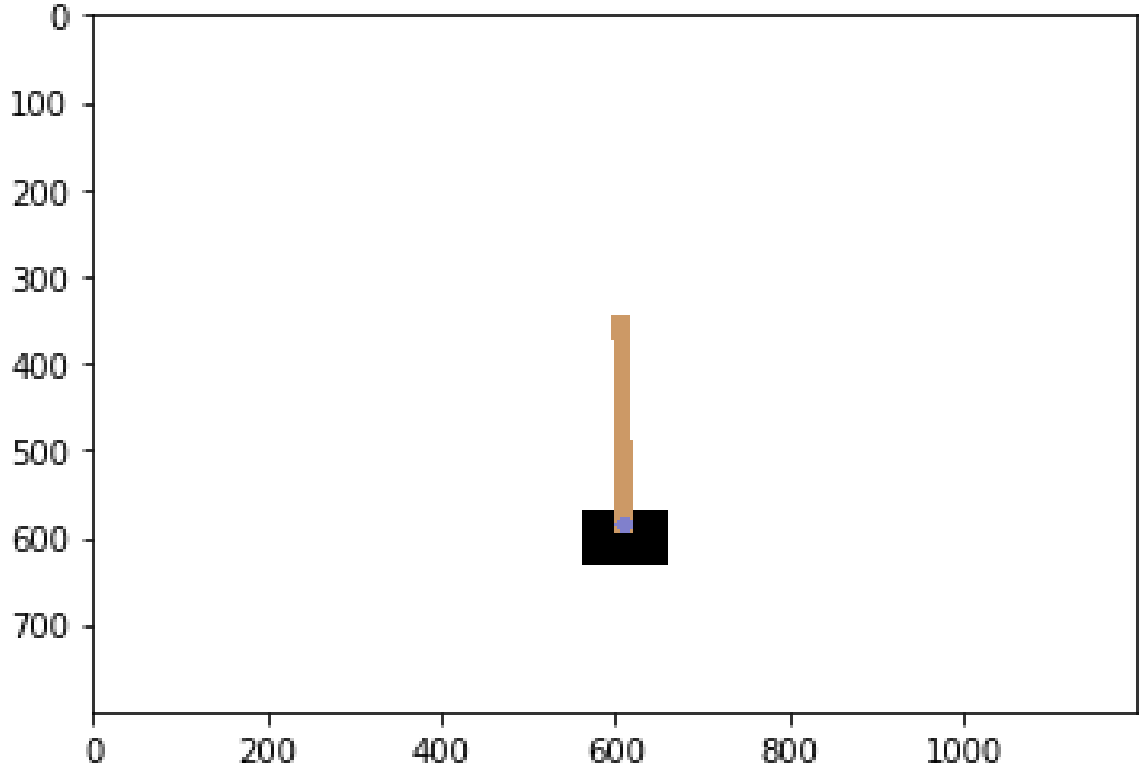

Рисунок 4.1 — Пример обратного маятника

4.2 Формулировка поставленной задачи оптимального управления и ее решением методом q-обучения

В данной работе рассмотрим задачу оптимального управления — маятник. Общая формулировка задана уравнением (**??**.)

Для решения задачи использовалась библитека gym, описанная выше. В ней маятник выглядит как представлен на рисунке 4.2.

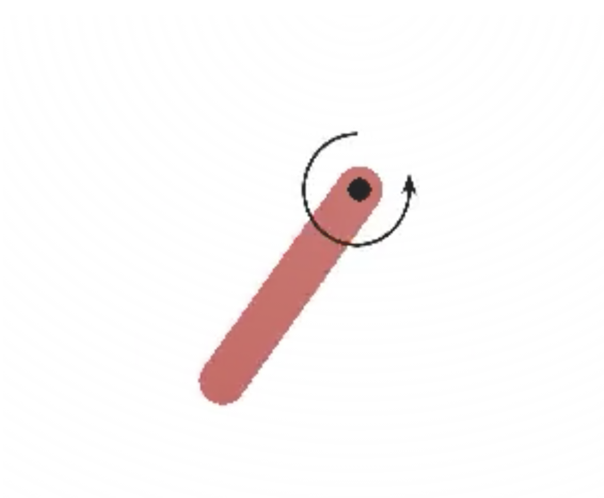

Рисунок 4.2 — Маятник в библиотеке gym

В рамках данной задачи есть одна переменная ϕ – угол отклонения маятника от вертикальной оси. На переменную накладываются следующие ограничения:

$$\phi_{min} \leqslant \phi \leqslant \phi_{max}. \tag{4.1}$$

Визуальное представление того, как угол отсчитывается от оси находится на рисунке 4.3.

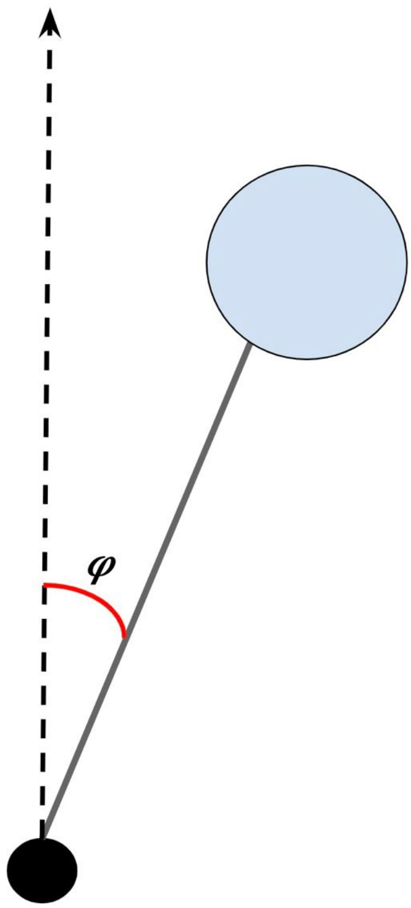

Рисунок 4.3 — Маятник

В нашем случае вводятся следующие ограничения: $\phi_{min} = -180°$, $\phi_{max} = 180°$. Кроме того рассматривается непрерывное управление (действия): $u = [-2, 2]$, то есть либо толкнуть маятник вправо, либо влево с максимальным модулем силы 2.

За каждое действие дается награда $r \in (-\infty, \infty)$. Цель задачи – при 200 запусках сессий привести маятник в вертикальное положение и сохранять его в таком положении.

Для решения задачи использовался алгоритм q-обучения в рамках которого функция $q(s, a)$ считалась с помощью архитектуры нейронной сети представленной на рисунке 4.4.

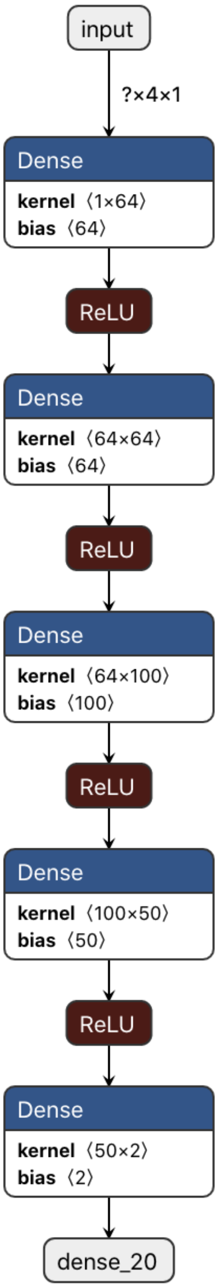

Рисунок 4.4 — Архитектура сети для подсчета $q(s, a)$

В рамках решения задачи использовалась эпсилон-жадная политика. То есть на каждой итерации с вероятностью в ϵ вместо лучшего действия выбиралось случайное. На каждой эпохе обучения мы запускам 10 сессий и считаем среднюю полученную награду. График обучения представлен на рисунке 4.5.

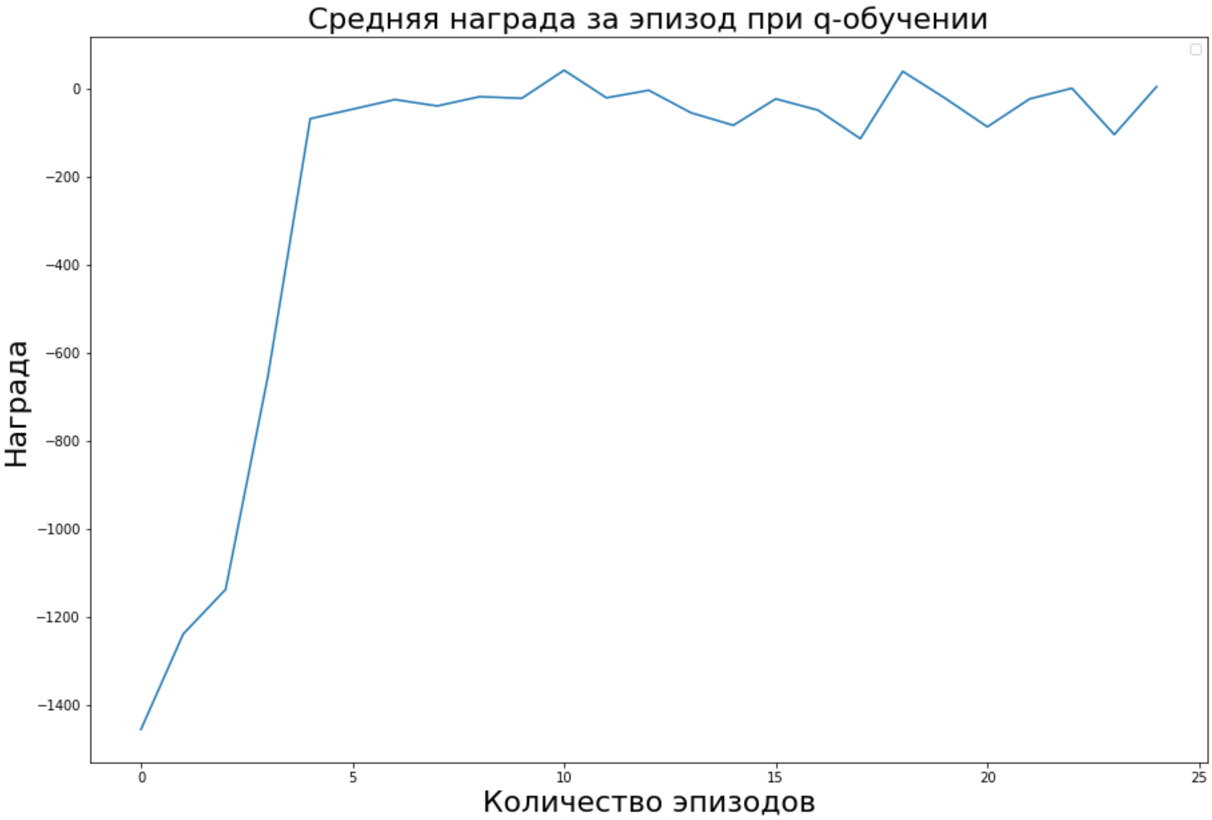

Рисунок 4.5 — График q-обучения

4.3 Решение задачи методом MPC

4.3.1 Решение задачи методом MPC с помощью робастного метода кросс-энтропии

В рамках данной задачи согласно схеме из рисунка 3.2 мы обучали модель среды и параллельно запускали и обучали метод MPC с помощью робастного метода кросс энтропии. Горизонт планирования для MPC составлял $T = 200$, так как эпизод рассчитан на 200 действий. Модель среды (2.8) аппроксимирована нейронной сетью, изображенной на рисунке 4.6.

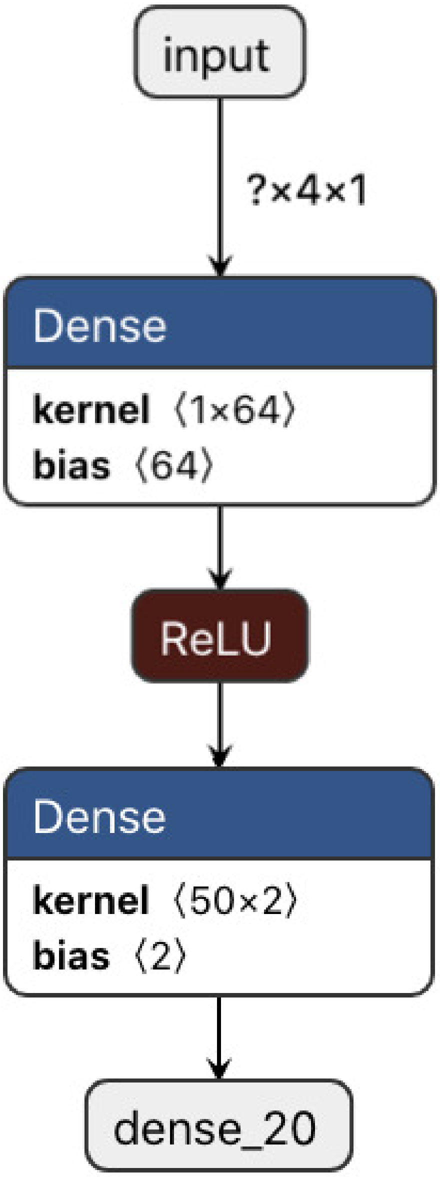

Рисунок 4.6 — Модель среды

В результате обучения модель научилась довольно точно предсказывать следующее состояние среды. График ошибки состояния среды представлен на рисунке 4.7. Кроме того модель среды научилась довольно точно предсказывать награду. График абсолютной ошибки предсказанной награды представлен на рисунке 4.8. Ошибка считалась с помощью средней абсолютной ошибки:

$$err = \frac{\sum_{i=0}^{N} |x_i - y_i|}{N}, \tag{4.2}$$

где x_i – полученное измерение, y_i – истинное значение.

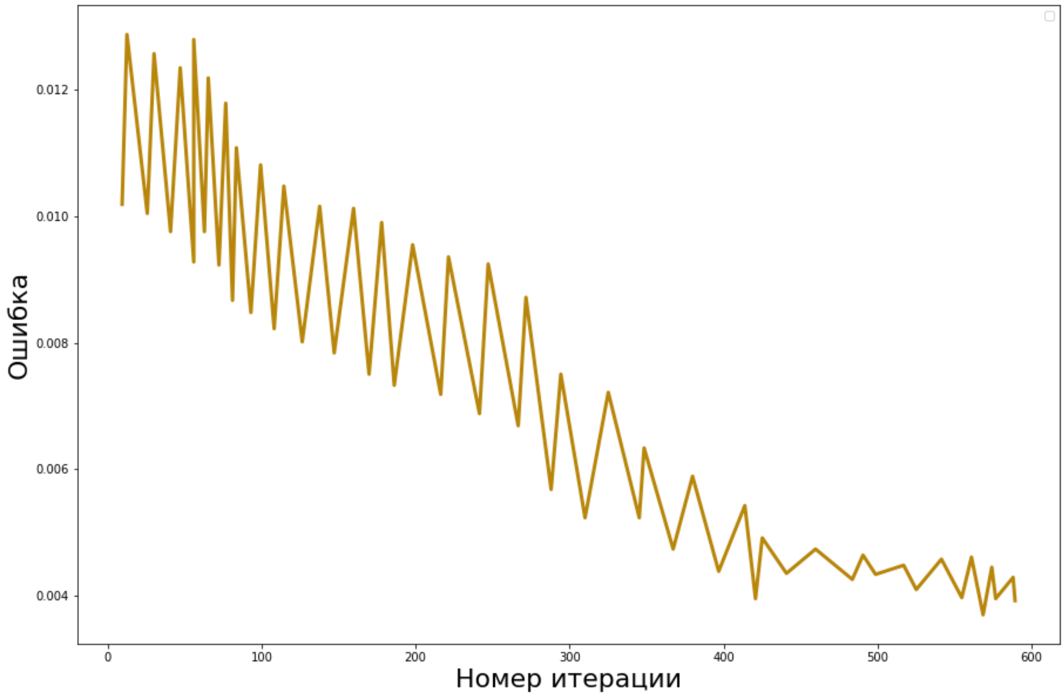

Рисунок 4.7 — График абсолютной ошибки состояния среды

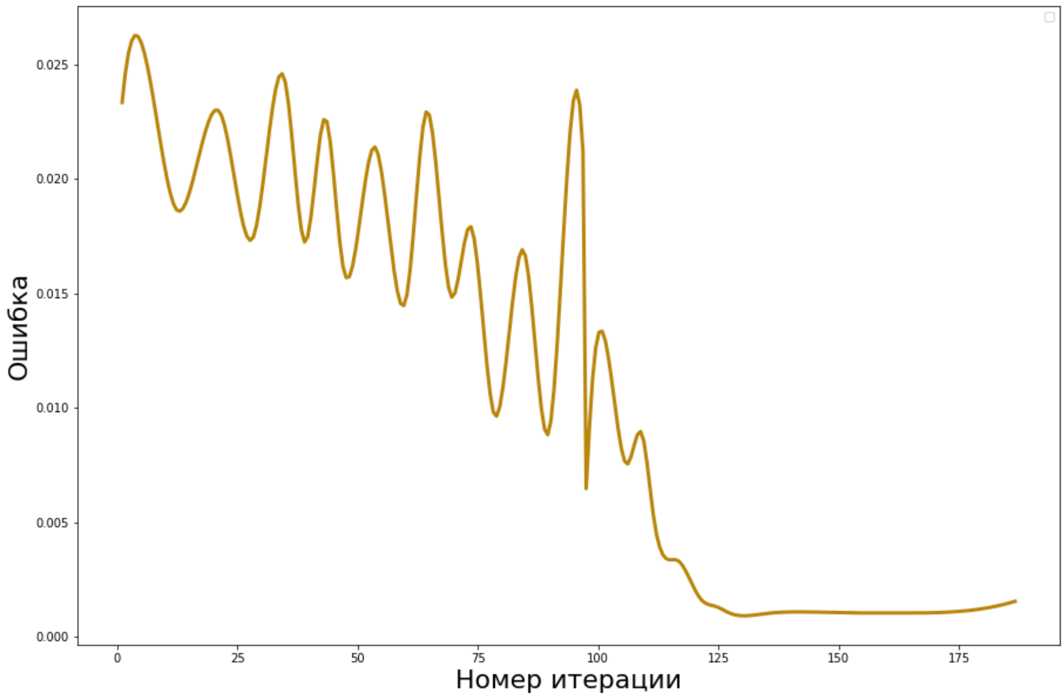

Рисунок 4.8 — График абсолютной ошибки награды

По мере обучения ошибка, используемая нейронной сетью для модели среды падала. График представлен на рисунке 4.9. Таким образом модель среды все больше становится похожа непосредственно на среду.

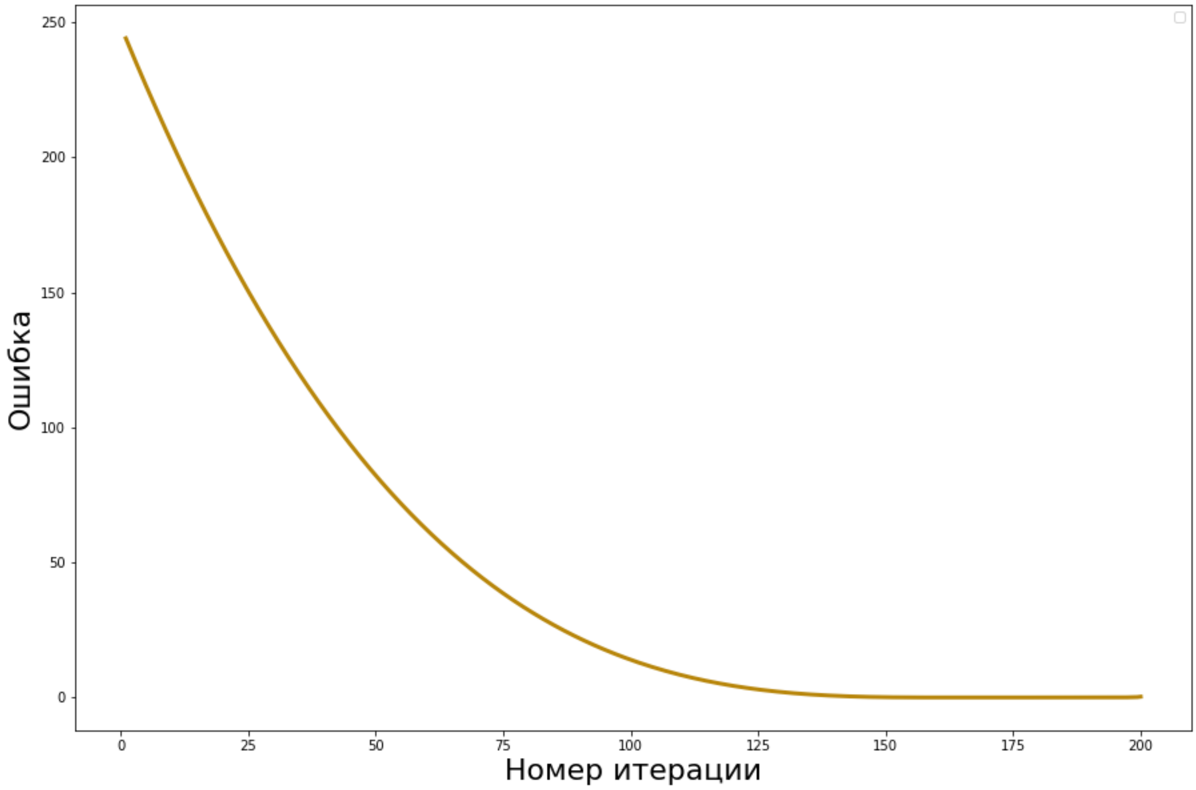

Рисунок 4.9 — График ошибки модели среды

Средняя реальная награда за эпизод (200 действий), которую получает наш агент за каждое действие растет по мере обучения. График чего представлен на рисунке 4.10.

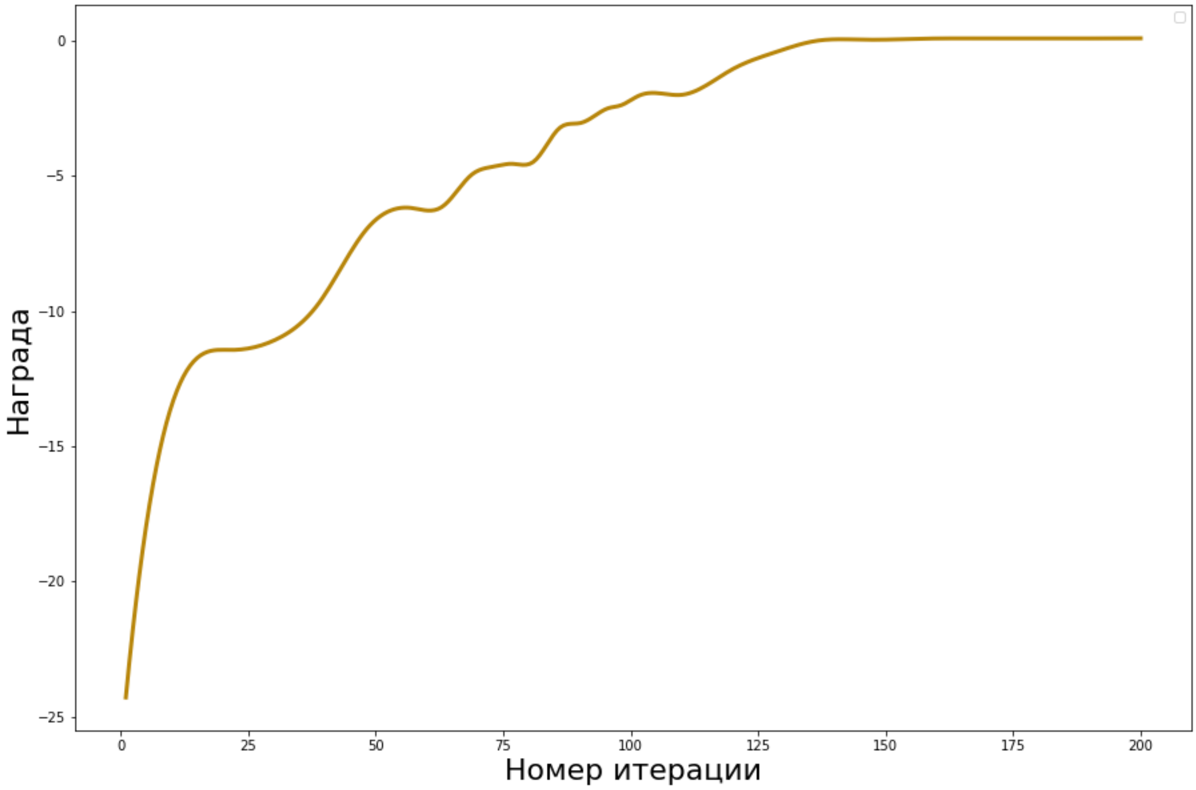

Рисунок 4.10 — Абсолютная награда

4.3.2 Решение задачи методом MPC на предобученной модели среды

Теперь сначала потренируем модель среды, генерируя действия случайно и посмотрим даст ли нам это улучшений. Сначала посмотрим на полученные результаты. Они приведены на рисунках 4.11 и 4.12. Ошибка считается по формуле (4.2).

Видно, что данный метод работает хуже, чем исходный. Из-за предобучения на политике, которая генерирует случайные действия и состояния, наша модель среды хуже предсказывает следующее состояние среды и хуже предсказывает получаемую награду. Это можно легко объяснить тем, что в данном случае наш маятник просто качается из случайного положения в случайное и награда все время становится очень маленькой. Поэтому когда позже мы начинаем применять политику, которую выдает MPC, состояния мало коррелируют с теми, на которых обучалась модель. В результате чего получаются очень плохие предсказания. Поэтому в дальнейшем не будем предобучать модель среды на случайной политике, а будем обучать ее совместно с MPC.

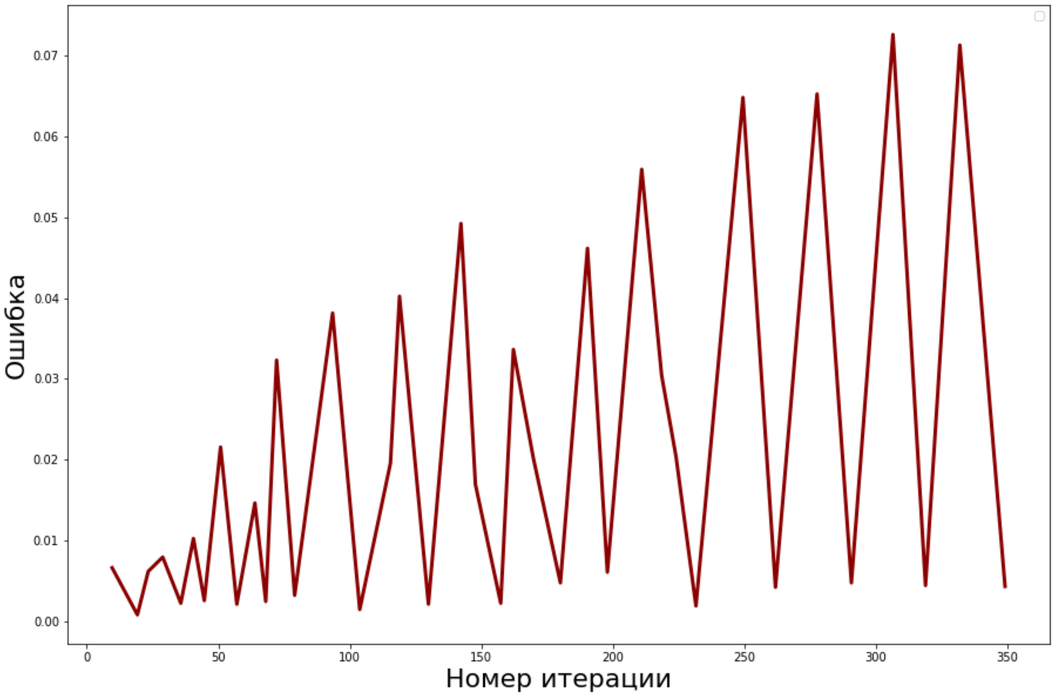

Рисунок 4.11 — График ошибки состояния для предобученной модели.

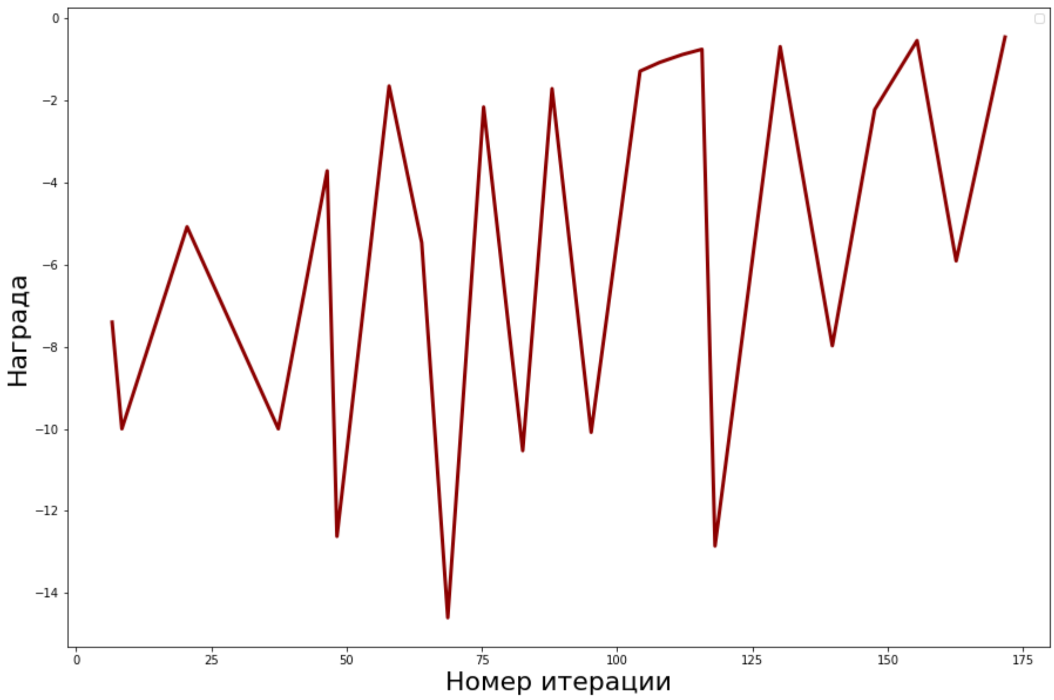

Рисунок 4.12 — График ошибки награды для предобученной модели.

4.3.3 Решение задачи методом MPC с реальной средой

Попробуем посмотреть, насколько хороши наши результаты в сравнении с тем, если бы мы обучались непосредственно на взаимодействии со средой. Результаты получаемой награды для такого обучения представлены на рисунке 4.13. В результате награда также сходится к 0.

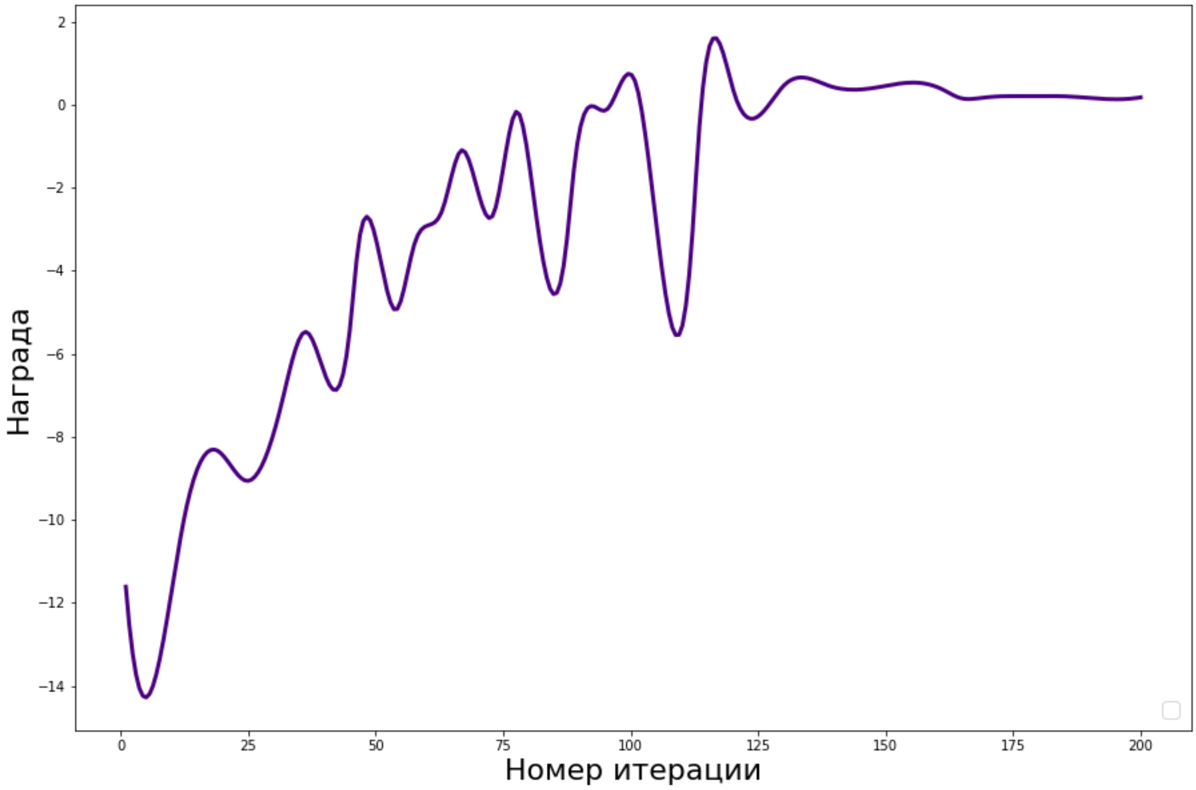

Рисунок 4.13 — Награда в реальной среде

Сравним с наградой, которую получаем на претренированной модели, которая была представлена на рисунке 4.10. График представлен на рисунке 4.14. Видно, что награда, полученная непосредственно на взаимодействии со средой быстрее сходится к 0. Это объяснимо тем, что модель не сразу полностью повторяет среду, а с каждой итерацией все больше приближается к ней. Поэтому и награда растет не так быстро. Однако в итоге и в одном, и во втором случае награда становится сравнимой. Таким образом подход с обучением модели среды не ухудшает политику выбора действия агента, а доход в результате остается сравнимым. То есть model-based алгоритм не ухудшает результаты в сравнении с model-free алгоритмом.

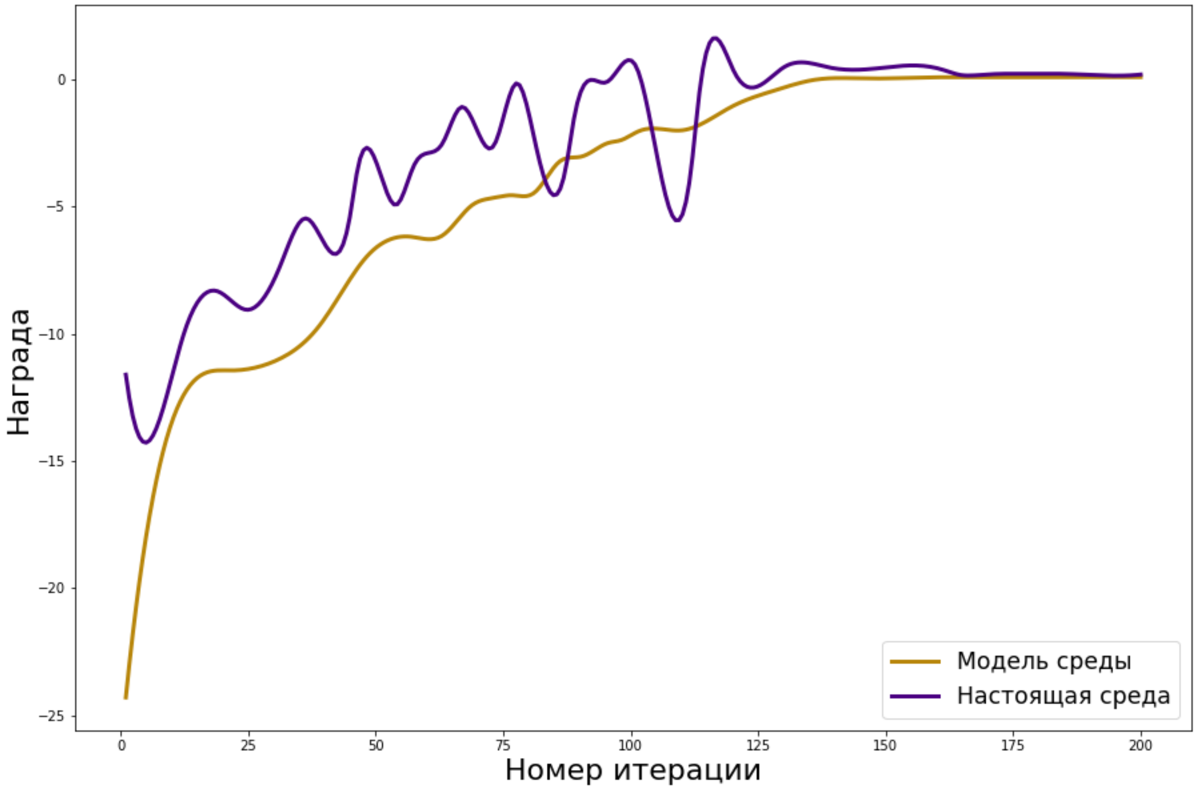

Рисунок 4.14 — График сравнение наград настоящей среды и модели среды

4.3.4 Сравнение двух методов обучения MPC для решения задачи

Теперь будем обучать MPC с помощью второго алгоритма, описанного в главе 3 – эволюционной стратегии адаптации ковариационной матрицы. Полученные результаты в сравнении с результатами обучения методом робастной кросс-энтропии (рисунки 4.7, 4.8, 4.10) представлены на рисунках 4.15, 4.16, 4.17. Ошибка считается по формуле (4.2). На рисунках видно, что новый алгоритм дает более быструю сходимость к получению награды. Кроме того быстрее обучает модель среды, а именно быстрее сходится к меньшей ошибке состояния среды. Таким образом он является предпочтительным для решения данной задач с помощью MPC.

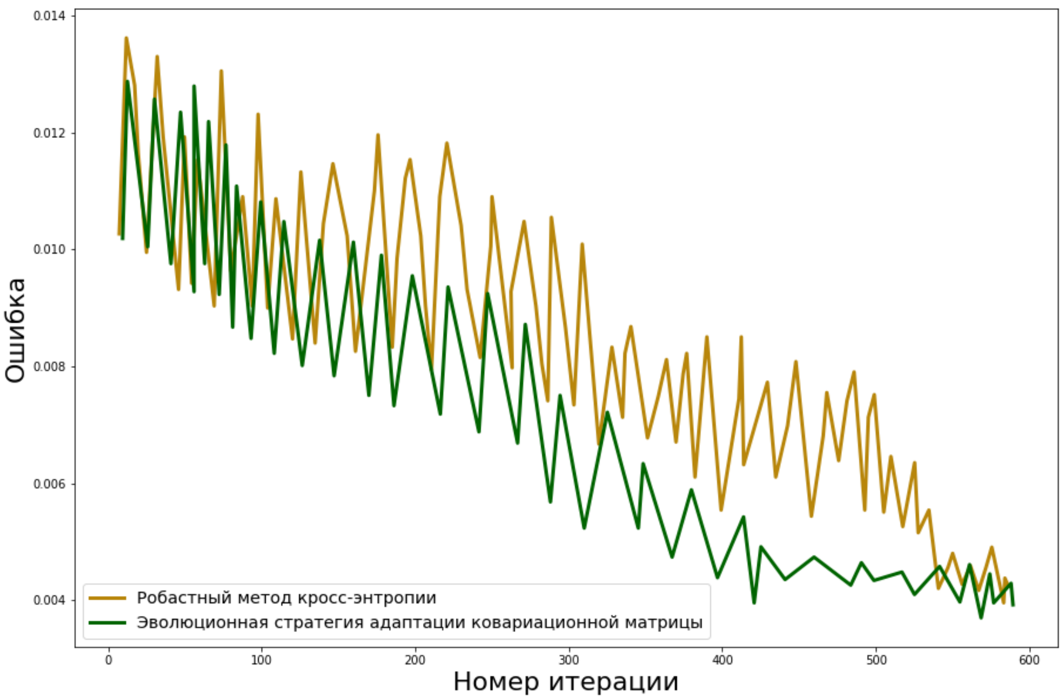

Рисунок 4.15 — График сравнение ошибки состояния

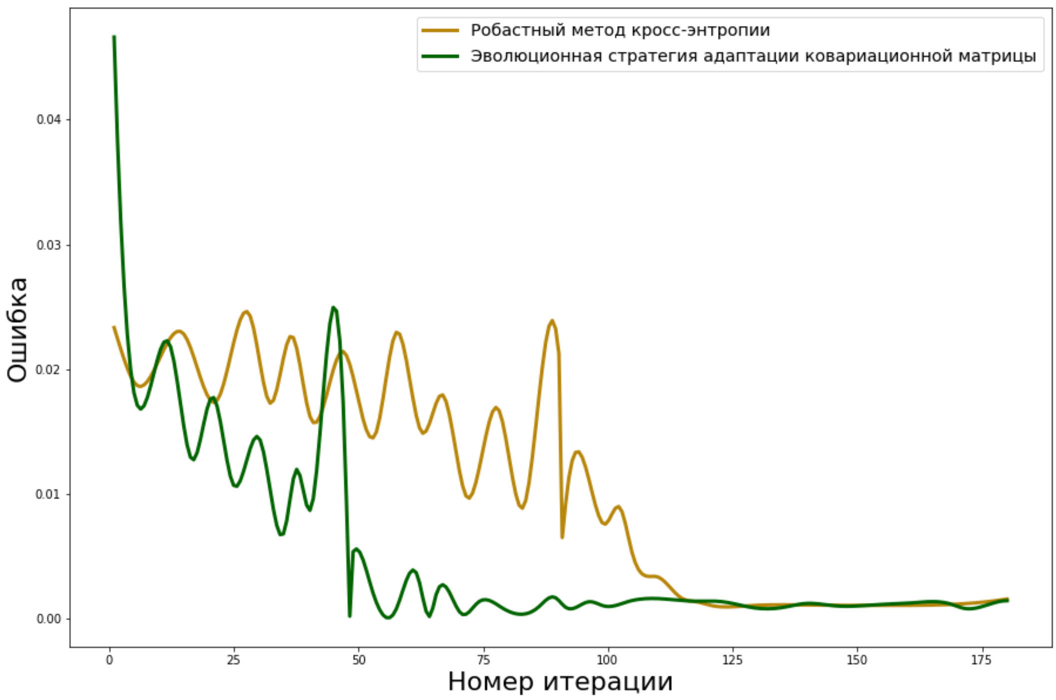

Рисунок 4.16 — График сравнение ошибки награды

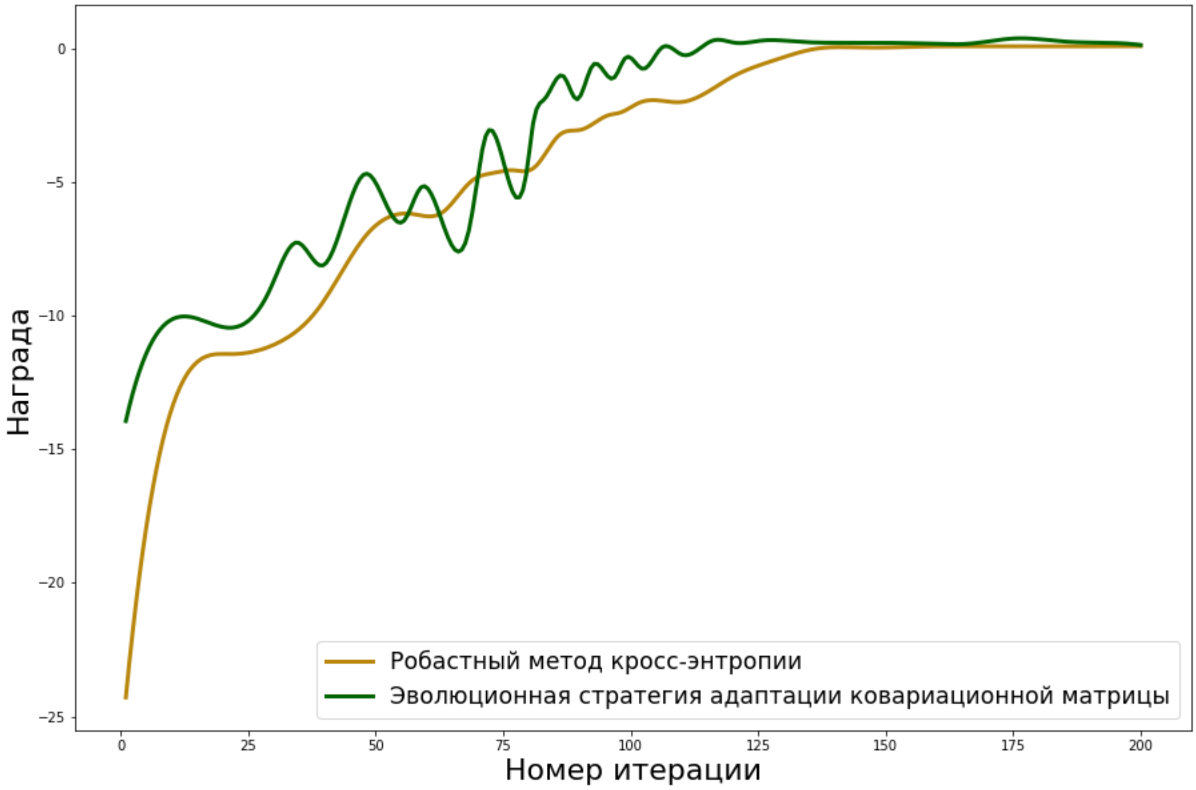

Рисунок 4.17 — График сравнение наград

4.4 Сравнение решения методом MPC и q-обучением

Возьмем MPC, обученный на модели среды с помощью эволюционной стратегии адаптации ковариационной матрицы и используя его стратегию сыграем 1000 сессий маятника и сравним полученные результаты с 1000 сессиями работы маятника, в которых используется политика выбора действия, полученная в результате q-обучения. Результаты среднего полученной награды, ее среднеквадратичного отклонения, минимума и максимума приведены в таблице 4.1. Как видно из нее в среднем MPC работает лучше, к тому же разброс значений (стандартное отклонение) у него меньше. Кроме того экстремальные значения (минимум и максимум) у MPC ближе к среднему. Таким образом решение с помощью MPC выдает лучшие результаты, чем q-обучение. Разница между ними не так велика, однако MPC еще и не нарушает исходные ограничения. Таким образом лучше решает поставленную задачу.

Метод	Mean	Std	Min	Max
MPC	0.5	3	-7	2
Q-обучение	-4	15	-30	5

Таблица 4.1 — Сравнение q-обучения и MPC

4.5 Выводы

В данной главе были представлены результаты экспериментов, результате которых была решена поставленная задача методом q-обучения. Кроме того было получено, что данный метод работает несколько хуже, чем MPC. Также исходная задача была решена методом MPC с помощью алгоритма робастной кросс-энтропии и эволюционной стратегии адаптации ковариационной матрицы. Было получено, что второй алгоритм дает более быструю сходимость и является предпочтительным. Также было найдено, что обучение на модели среды работает несколько хуже, чем на реальной среде. Однако в результате и один, и второй способ сходятся к одинаковой награде. Кроме того было показано, что не имеет смысла предобучать модель на политике, которая выбирает случайное действие.

ЗАКЛЮЧЕНИЕ

В ходе работы получены следующие основные результаты:

1. Изучены основные задачи оптимального управления, выбрана и поставлена задача.
2. Проведен обзор и выбран метод обучения с подкреплением для решения задачи.
3. Решена поставленная задача с помощью выбранного метода обучения с подкреплением.
4. Проведен обзор метода решения задач с помощью управления с прогнозирующими моделями.
5. Решена задача с помощью метода с прогнозирующими моделями.
6. Проведен сравнительный анализ решений, полученных вышеперечисленными методов.

В результате сравнения двух подходов выявлено, что эволюционная стратегия адаптации ковариационной матрицы в рамках управления по прогнозируемой модели работает лучше, чем алгоритм робастной кросс-энтропии. Кроме того средняя награда, получаемая данным алгоритмам, в среднем лучше, чем при помощи q-обучения. Также обучение по прогнозируемой модели учитывает нарушения ограничений в виде функции стоимости в отличие от q-обучения. Таким образом значительно реже нарушает ограничения. В результате чего является более предпочтительным и более безопасным.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Dimitri P. Bertsekas. Dynamic programming and optimal control volume one. 2017.
2. Dimitri P. Bertsekas. Dynamic programming and optimal control volume two. 2017.
3. Tesauro G. a self-teaching backgammon program, achieves master-level play. In *Neural Computation*, page 215–219, 1994.
4. Moritz M. Diehl James B. Rawlings, David Q. Mayne. Model predictive control: Theory, computation, and design. 2012.
5. Lars Grüne Jürgen Pannek. Nonlinear model predictive control. 2017.
6. R. McAllister K. Chua, R. Calandra and S. Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*, page 4754–4765, 2018.
7. M. Leslie Pack Kaelbling. Reinforcement learning: A survey. In *Journal of Artificial Intelligence Research*, pages 237–285, 1996.
8. N. Kastsiukevich, D. Dmitruk. Data-driven optimal control of linear time-invariant systems. In *21th IFAC World Congress*, 2020.
9. R. S. Sutton and A. G. Barto. Reinforcement learning: An introduction. second edition. 2016.
10. A.L. Warren Thomas Marlin. Constrained mpc under closed-loop. In *Conference: American Control Conference*, 2004.
11. R. Y. Rubinstein Z. I. Botev, D. P. Kroese and P. L'Ecuyer. The cross-entropy method for optimization. In *Handbook of statistics*, page 35–59, 2013.
12. О. И. Костюкова Р.Ф. Габасов. Синтез оптимальных обратных связей для систем с внутренними ограничениями на управление. 1997.
13. Понтрягин Л.С. и др. Математическая теория оптимальных процессов. 1983.

ПРИЛОЖЕНИЕ

https://github.com/yakubanna/master_code