Содержание

1 Введение в дисциплину “Искусственный интеллект”

Искусственный интеллект (ИИ) - чрезвычайно широкая область знаний, которая включает математическую логику, теорию вероятностей, теорию непрерывных функций и практические системы, способные имитировать когнитивные функции человека, такие как восприятие, рассуждение, обучение и действие. Тематика области **искусственного интеллекта** в настоящее время охватывает огромный перечень научных направлений, от задач самого общего характера (**обучение**, **рассуждение**, **восприятие** и т.д.) и до таких конкретных задач, как игра в шахматы, доказательство математических теорем, сочинение стихов, вождение автомобиля или диагностика заболеваний. Достижения в области **ИИ** могут найти себе применение при решении любой интеллектуальной задачи, это универсальная научная область.

2 Основные задачи и понятия

2.1 Задачи ИИ

1. **Решение плохо формализуемых задач**.
    - Такие задачи характеризуются высоким уровнем неопределенности, отсутствием полного формального описания и наличием множества параллельных критериев оценки.
    - Пример: принятие стратегических управленческих решений, где невозможно задать чёткий алгоритм из-за динамичности внешних факторов.
2. **Моделирование поведения интеллектуальных агентов**.

- Разработка моделей, способных имитировать когнитивные процессы, такие как **восприятие**, **память**, **обучение** и **планирование**.
- Пример: автономный робот, который анализирует данные с датчиков, планирует маршрут и принимает решения в режиме реального времени.

3. **Представление и обработка знаний**.

   - Формирование структурированных моделей знаний, которые используются для **логического вывода** и **вероятностного рассуждения**.
   - Пример: экспертная система для диагностики заболеваний, где знания представлены в виде правил и фактов.

4. **Интерактивное взаимодействие с внешней средой**.

   - Создание систем, способных воспринимать, анализировать и реагировать на окружающие сигналы.
   - Пример: системы распознавания образов, анализирующие визуальную информацию и принимающие решения на её основе.

Ключевая задача **ИИ** заключается в выяснении того, как создавать программы, которые в рамках возможного обеспечивают рациональное поведение **агента** с использованием небольшого объема программного кода, а не обширных таблиц с множеством записей.

2.2 Понятия и термины

Плохо формализуемая задача – это задача, для которой сложно или невозможно заранее задать строгие математические модели, четкие правила или алгоритмы решения [1].

Агент — это просто что-то, что действует (слово агент произошло от латинского слова agere - "действовать"). Конечно, все компьютерные программы что-то делают, но ожидается, что компьютерные агенты будут делать больше: работать автономно, воспринимать окружающую среду, сохранять свое существование в течение длительного периода времени, приспосабливаться к изменениям, устанавливать и преследовать определенные цели [1].

Внешняя среда – окружение, в котором действует агент. Среда может быть детерминированной или стохастической, статической или динамической, с полной или частичной информацией.

Поведение – совокупность действий агента в ответ на изменения внешней среды. Может быть заранее заданным, адаптивным или обучаемым.

Модель – формализованное представление реального объекта, процесса или системы, используемое для анализа, прогнозирования или управления. В **ИИ** модель может описывать среду, агента, данные или связи между ними.

Обучение – процесс улучшения поведения или предсказательных способностей системы (агента, алгоритма) на основе опыта, данных или взаимодействия со средой [1].

Знание – информация, которая представлена в виде, пригодном для обработки системой **искусственного интеллекта** (например, правила, факты, связи, вероятности).

Искусственный нейрон. Элемент нейронной сети, имитирующий биологический нейрон, суммирующий входные сигналы с весовыми коэффициентами и применяющий нелинейную функцию активации [4].

Искусственная нейронная сеть (ИНС). Система взаимосвязанных нейронов, способная моделировать сложные зависимости и решать задачи классификации, регрессии, аппроксимации и оптимизации.

3 Задачи обучения

3.1 Классификация подходов обучения

1. **Обучение с учителем (Supervised Learning)** – система обучается на размеченных данных, где каждому входу соответствует правильный выход. Примеры: классификация, регрессия.

2. **Обучение без учителя (Unsupervised Learning)** – система обучается на неразмеченных данных, выявляя структуры и закономерности. Примеры: кластеризация, понижение размерности.

3. **Обучение с подкреплением (Reinforcement Learning, RL)** – агент обучается через взаимодействие с внешней средой, получая награды или штрафы за свои действия.

4. **Обучение с частичным привлечением учителя (Semi-supervised Learning)** – комбинированный подход, где используются как размеченные, так и неразмеченные данные.

5. **Обучение с переносом знаний (Transfer Learning)** – использование знаний, полученных в одной задаче, для решения другой.

3.2 Формальная постановка задачи обучения с подкреплением

Обучение с подкреплением – это обучение тому, что делать, т. е. как отобразить ситуации на действия, чтобы максимизировать численный сигнал – вознаграждение. Обучаемому не говорят, какие действия предпринимать, он должен сам понять, какие действия

приносят максимальное вознаграждение, пробуя их. В наиболее интересных и трудных случаях действия могут влиять не только на непосредственное вознаграждение, но и на следующую ситуацию, а значит, на все последующие вознаграждения. Эти две характеристики – поиск методом проб и ошибок и отложенное вознаграждение – являются наиболее важными отличительными чертами **обучения с подкреплением** [3].

Мы формализуем задачу **обучения с подкреплением**, применяя идеи из теории динамических систем, а точнее как задачу оптимального управления не полностью известным **марковским процессом принятия решений**. Основную мысль можно сформулировать просто – требуется уловить наиболее важные аспекты реальной проблемы, стоящей перед обучающимся агентом, который взаимодействует во времени с окружающей средой для достижения некоторой цели. Обучающийся агент должен уметь в какой-то степени воспринимать состояние среды и предпринимать действия, изменяющие это состояние. У агента также должна быть цель или несколько целей, как-то связанных с состоянием окружающей среды. **Марковские процессы принятия решений** включают все три аспекта – восприятие, действие и цель – в простейшей возможной форме, не сводя, однако, ни один аспект к тривиальному. Любой метод, подходящий для решения таких задач, будет рассматриваться нами как метод **обучения с подкреплением**.

3.3 Классификация задачи обучения с подкреплением

В целом возможные подходы можно классифицировать следующим образом.

- **Обучение с подкреплением на основе модели**. В этих подходах агент использует модель перехода среды как инстру-

мент, помогающий интерпретировать сигналы вознаграждения и принимать решения о том, как действовать. Изначально эта модель может быть неизвестна, и тогда агент изучает ее посредством наблюдения результатов своих действий, либо она может быть уже определена, - например, программе игры в шахматы могут быть известны все правила этой игры, даже если она еще не обучена выбирать хорошие ходы. В частично наблюдаемых средах модель перехода также будет полезна для оценки состояния.

- **Обучение с подкреплением без модели**. В этих подходах агент изначально не знает и не изучает модель перехода для окружающей среды. Вместо этого он учится непосредственно представлению о том, как себя вести. Подобный подход возможен в двух вариантах.
  - **Изучение полезности действий**. Наиболее распространенной формой изучения полезности действий является **Q-обучение**, когда агент изучает **Q-функцию** или функцию ожидаемой полезности действия $Q(s,)$, определяющую сумму вознаграждений от состояния s и далее, если будет выполнено действие . При известной **Q-функции** агент может выбрать, что ему делать в состоянии s, посредством поиска действия с самым высоким значением ожидаемой полезности Q.
  - **Поиск стратегии**. Агент изучает стратегию, непосредственно отображающую состояния на действия.

Рассмотрим методы решения.

4 Динамическое программирование

4.1 Введение в ДП

- **ДП** — семейство алгоритмов для вычисления оптимальных стратегий в **марковских процессах принятия решений (МППР)**.
- Предполагает идеальную модель среды, что ограничивает практическое применение в **обучении с подкреплением** из-за вычислительной сложности.
- Теоретически важен: формирует основу для методов **обучения с подкреплением**, стремящихся достичь эффекта **ДП** с меньшими затратами и без идеальной модели.

4.2 Ключевые элементы МППР

- **Конечный МППР**: задается множествами состояний S, действий A, вознаграждений R.
- **Динамика среды** описывается вероятностями перехода $p(s', r|s, a)$.
- **Функции ценности** — инструмент для структурирования поиска стратегий.

4.3 Оценивание стратегии (Предсказание)

- Цель: вычислить функцию ценности $v_\pi(s)$ для произвольной стратегии π.
- **Алгоритм итеративного оценивания**:

- На каждой итерации обновляются ценности всех состояний.
- **Полное обновление**: учитывает все возможные переходы (математическое ожидание по всем следующим состояниям).
- **Обновление на месте**: перезапись значений в одном массиве, что может ускорить сходимость.

- Сходимость гарантируется только в пределе, но на практике алгоритм останавливают при малых изменениях.

4.4 Улучшение стратегии

- **Жадная стратегия** π': выбирает действие, максимизирующее $q_\pi(s, a)$ (ожидаемую ценность).
- **Теорема об улучшении стратегии**: если π' жадная относительно v_π, то $v_\pi \geq v_{\pi'}$.
- **Стохастические стратегии**: допускают распределение вероятностей между действиями, максимизирующими ценность.

4.5 Итерация по стратегиям

- Процесс:
  1. Оценивание текущей стратегии π для получения v_π.
  2. Улучшение стратегии до π', жадной относительно v_π.
- Для конечных **МППР** гарантирует сходимость к оптимальной стратегии за конечное число шагов.

- **Алгоритм итерации по стратегиям**
  Инициализация
  Для всех состояний $s \in S$:

$$V(s) \in \mathbb{R} \quad \text{и} \quad \pi(s) \in \mathcal{A}(s) \quad \text{выбираются произвольно}$$

  Оценивание стратегии
  Повторять:

$$\Delta \leftarrow 0$$

  Для каждого $s \in S$:

$$v \leftarrow V(s)$$
$$V(s) \leftarrow \sum_{s',r} p(s', r|s, \pi(s)) \left[r + \gamma V(s')\right]$$
$$\Delta \leftarrow \max(\Delta, |v - V(s)|)$$

  Пока $\Delta < \theta$ (где θ — параметр точности).

  Улучшение стратегии
  policy-stable $\leftarrow$ true
  Для каждого $s \in S$:

$$\text{old-action} \leftarrow \pi(s)$$
$$\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r|s, a) \left[r + \gamma V(s')\right]$$
$$\text{Если old-action} \neq \pi(s): \quad \text{policy-stable} \leftarrow \text{false}$$

4.6 Итерация по ценности

- Объединяет шаги оценивания и улучшения в одной операции.
- **Усеченное оценивание**: выполняется один проход обновления (вместо полной сходимости).

- Основана на **уравнении оптимальности Беллмана**.
- Практически останавливается при малых изменениях функции ценности.
- **Алгоритм итерации по ценности**
  Параметр алгоритма: небольшая пороговая величина $\theta > 0$, определяющая точность оценки.
  Инициализация:

  $V(s)$ — произвольные значения для всех $s \in S^+$, где $V(\text{terminal}) = 0$.

  Повторять:

  $$\Delta \leftarrow 0$$

  Для каждого $s \in S$:

  $$v \leftarrow V(s)$$
  $$V(s) \leftarrow \max_a \sum_{s',r} p(s', r|s, a) \left[r + \gamma V(s')\right]$$
  $$\Delta \leftarrow \max(\Delta, |v - V(s)|)$$

  Пока $\Delta < \theta$.

  Вывод: детерминированная стратегия $\pi \approx \pi_*$, где

  $$\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r|s, a) \left[r + \gamma V(s')\right].$$

4.7 Асинхронное ДП

- Обновления производятся в произвольном порядке, без систематических проходов по состояниям.
- Преимущества:

- **Гибкость**: можно интегрировать с реальным взаимодействием агента со средой.
- **Эффективность** для больших пространств состояний.

- Условие сходимости: все состояния должны обновляться бесконечно часто.

4.8 Обобщенная итерация по стратегиям (ОИС)

- Два процесса:
  - **Оценивание**: делает функцию ценности, согласованной со стратегией.
  - **Улучшение**: делает стратегию жадной относительно функции ценности.
- Взаимодействие процессов приводит к оптимальности:
  - Стабилизация происходит только при достижении оптимальных v^* и π^*.

4.9 Эффективность ДП

- Сложность: полиномиальная от числа состояний n и действий k (в худшем случае).
- Проблемы:
  - **Проклятие размерности**: экспоненциальный рост состояний с увеличением переменных.
  - Однако **ДП** эффективнее прямого поиска и линейного программирования для больших задач.

- Практика: методы **ДП** решают задачи с миллионами состояний, особенно при хороших начальных приближениях.

5 Методы Монте-Карло

5.1 Введение в методы Монте-Карло

- **Определение**: Методы **МК** решают задачи обучения с подкреплением через усреднение выборочного дохода.
- **Область применения**: Эпизодические задачи (опыт делится на завершаемые эпизоды).
- **Особенности**:
  - Не требуют знания модели среды.
  - Обновления производятся поэпизодно, а не пошагово.
  - Основаны на полных доходах, полученных после посещения состояний или пар **состояние-действие**.

5.2 Предсказание методами Монте-Карло

- **Цель**: Оценка функции ценности состояний $v_\pi(s)$ для заданной стратегии π.
- **Методы**:
  - **МК первого посещения**: усредняет доходы, полученные после первого посещения состояния s в эпизоде.
  - **МК всех посещений**: усредняет доходы после всех посещений s.

- **Сходимость**: Оба метода сходятся к $v_\pi(s)$ при бесконечном числе посещений.
- **Преимущества**:
  - Метод первого посещения проще и имеет меньшую дисперсию.
  - Метод всех посещений квадратично сходится, но сложнее в реализации.
- **Предсказание методом MC первого посещения для оценивания** $V \approx v_\pi$

  Вход: стратегия π, подлежащая оцениванию.

  Инициализация:

  $V(s) \in \mathbb{R}$ (произвольные начальные значения для всех $s \in S$)
  $Returns(s) \leftarrow$ пустой список для всех $s \in S$

  Повторять бесконечно (для каждого эпизода):

  1. Сгенерировать эпизод, следуя π:

     $$S_0, A_0, R_1, S_1, A_1, R_2, \ldots, S_{T-1}, A_{T-1}, R_T$$

  2. Инициализировать возврат:

     $$G \leftarrow 0$$

  3. Повторять для каждого шага эпизода $t = T-1, T-2, \ldots, 0$:

     $G \leftarrow \gamma G + R_{t+1}$
     Если $S_t \notin \{S_0, S_1, \ldots, S_{t-1}\}$:
     Добавить G в $Returns(S_t)$
     $V(S_t) \leftarrow$ среднее$(Returns(S_t))$

5.3 Оценивание ценности действий

- **Цель**: Оценка $q_\pi(s, a)$ (ценности пар **состояние-действие**).
- **Проблема**: При детерминированной стратегии π некоторые действия не выбираются, что делает их оценку невозможной.
- **Решение**:
  - **Исследовательские старты**: гарантируют, что все пары **состояние-действие** посещаются с ненулевой вероятностью.
  - **Требование**: Эпизоды начинаются со случайных пар **состояние-действие**.

5.4 Управление методом Монте-Карло

- **Общая идея**: Использование **обобщенной итерации по стратегиям (ОИС)**:
  1. **Оценивание**: Точная оценка q_{π_k}.
  2. **Улучшение**: Построение жадной стратегии π_{k+1} относительно q_{π_k}.
- **Гарантии**: Сходимость к оптимальной стратегии π^* и функции q^*.
- **Ограничения**:
  - Требует бесконечного числа эпизодов и исследовательских стартов (непрактично).
- **Метод Монте-Карло ИС (с исследовательскими стартами) для оценивания** $\pi \approx \pi_*$

Инициализация:

$\pi(s) \in \mathcal{A}(s)$ (произвольная стратегия для всех $s \in S$)
$Q(s,a) \in \mathbb{R}$ (произвольные значения для всех $s \in S, a \in \mathcal{A}(s)$)
$Returns(s,a) \leftarrow$ пустой список для всех $s \in S, a \in \mathcal{A}(s)$

Повторять бесконечно (для каждого эпизода):

1. Выбрать $S_0 \in S$, $A_0 \in \mathcal{A}(S_0)$ случайным образом, так что вероятность любой пары > 0.
2. Сгенерировать эпизод, следующий π:

$$S_0, A_0, R_1, S_1, A_1, R_2, \ldots, S_{T-1}, A_{T-1}, R_T$$

3. Инициализировать возврат:

$$G \leftarrow 0$$

4. Повторять для каждого шага эпизода $t = T-1, T-2, \ldots, 0$:

   $G \leftarrow \gamma G + R_{t+1}$
   Если $(S_t, A_t) \notin \{(S_0, A_0), (S_1, A_1), \ldots, (S_{t-1}, A_{t-1})\}$:
      Добавить G в $Returns(S_t, A_t)$
      $Q(S_t, A_t) \leftarrow$ среднее$(Returns(S_t, A_t))$
      $\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$

5.5 Методы без исследовательских стартов

- **Проблема**: как обеспечить исследование без искусственных стартов?

- **Решение**:
  - **ε-жадные стратегии**:
    - С вероятностью $1 - \varepsilon$ выбирается жадное действие.
    - С вероятностью ε — случайное действие.
  - **Методы с единой стратегией (on-policy)**:
    - Обучают и улучшают текущую стратегию π, которая остается ε-мягкой.
    - Гарантируют исследование за счет случайных действий.
- **Метод управления MC первого посещения (для ε-мягких стратегий) для оценивания $\pi \approx \pi_*$**

  Параметр алгоритма: небольшое $\varepsilon > 0$.

  Инициализация:

  $\pi \leftarrow$ произвольная ε-мягкая стратегия
  $Q(s, a) \in \mathbb{R}$ (произвольные значения для всех $s \in S, a \in \mathcal{A}(s)$)
  $Returns(s, a) \leftarrow$ пустой список для всех $s \in S, a \in \mathcal{A}(s)$

  Повторять бесконечно (для каждого эпизода):

  1. Сгенерировать эпизод, следующий π:

     $$S_0, A_0, R_1, S_1, A_1, \ldots, S_{T-1}, A_{T-1}, R_T$$

  2. Инициализировать возврат:

     $$G \leftarrow 0$$

3. Повторять для каждого шага эпизода $t = T-1, T-2, \ldots, 0$:

   $G \leftarrow \gamma G + R_{t+1}$

   Если $(S_t, A_t) \notin \{(S_0, A_0), (S_1, A_1), \ldots, (S_{t-1}, A_{t-1})\}$:

   Добавить G в $Returns(S_t, A_t)$

   $Q(S_t, A_t) \leftarrow$ среднее$(Returns(S_t, A_t))$

   $A^* \leftarrow \arg\max_a Q(S_t, a)$ (неоднозначности разрешаются произвольно

   Для всех $a \in \mathcal{A}(S_t)$:

$$\pi(a|S_t) \leftarrow \begin{cases} 1 - \varepsilon + \dfrac{\varepsilon}{|\mathcal{A}(S_t)|} & \text{если } a = A^* \\ \dfrac{\varepsilon}{|\mathcal{A}(S_t)|} & \text{если } a \neq A^* \end{cases}$$

5.6 Предсказание с разделенной стратегией

- **Цель**: Оценка v_π или q_π с использованием данных от другой стратегии b.
- **Условие покрытия**: b должна выбирать все действия, возможные при π.
- **Выборка по значимости**:
  - Коэффициент значимости
  - **Типы оценок**:
    - Обыкновенная выборка
    - Взвешенная выборка

5.7 Управление с разделенной стратегией

- **Цель**: Обучение оптимальной стратегии π^* на данных от стратегии b.
- **Требования**:
  - b должна быть мягкой (выбирать все действия с ненулевой вероятностью).
  - π — жадная стратегия относительно текущей оценки Q.
- **Алгоритм**:
  1. Генерация эпизодов по стратегии b.
  2. Обновление Q с использованием взвешенной выборки по значимости.
  3. Улучшение π до жадной стратегии.
- **Управление методом MC с разделенной стратегией для вычисления оценки** $\pi \approx \pi_*$

  Инициализация: для всех $s \in S$, $a \in \mathcal{A}(s)$:

  $$\begin{aligned} &Q(s,a) \in \mathbb{R} \quad \text{(произвольные значения)} \\ &C(s,a) \leftarrow 0 \\ &\pi(s) \leftarrow \operatorname*{argmax}_a Q(s,a) \end{aligned}$$

  Повторять бесконечно (для каждого эпизода):

  1. $b \leftarrow$ произвольная мягкая стратегия.
  2. Сгенерировать эпизод, следующий b:

  $$S_0, A_0, R_1, S_1, A_1, \ldots, S_{T-1}, A_{T-1}, R_T$$

3. Инициализировать переменные:

$$G \leftarrow 0, \quad W \leftarrow 1$$

4. Повторять для каждого шага эпизода $t = T-1, T-2, \ldots, 0$:

$G \leftarrow \gamma G + R_{t+1}$
$C(S_t, A_t) \leftarrow C(S_t, A_t) + W$
$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \dfrac{W}{C(S_t, A_t)} [G - Q(S_t, A_t)]$
$\pi(S_t) \leftarrow \underset{a}{\operatorname{argmax}} Q(S_t, a)$ (неоднозначности разрешаются произвольно
Если $A_t \neq \pi(S_t)$: выйти из внутреннего цикла
$W \leftarrow W \cdot \dfrac{1}{b(A_t|S_t)}$

5.8 Инкрементная реализация

- **Особенности**:
  - Обновление оценок происходит поэпизодно.
  - Подходит для методов с разделенной стратегией.
- **Алгоритм**:
  - Для каждой пары **состояние-действие** обновляется средневзвешенное значение $Q(s, a)$.
  - Используется коэффициент значимости для коррекции весов.

6 Обучение на основе временных различий

Ключевые термины

- **TD-обучение**: Инкрементальное обновление на основе временных различий.
- **Бутстрэппинг**: Использование текущих оценок для обновления.
- **Sarsa/Q-обучение**: Алгоритмы управления с единой/разделенной стратегией.
- **Смещение максимизации**: Артефакт использования максимума оценок.
- **Послесостояния**: Состояния после действия с известной динамикой.

6.1 Введение в TD-обучение

- **TD-обучение** (обучение на основе временных различий) объединяет идеи методов **Монте-Карло (МК)** и **динамического программирования (ДП)**:
  - Как **МК**: обучается на опыте без модели окружения.
  - Как **ДП**: использует **бутстрэппинг** (обновляет оценки на основе других оценок, не дожидаясь конечного результата).
- **Ключевое отличие от МК**:

  - **МК** ждет конца эпизода для обновления оценки (цель — полный доход).
  - **TD** обновляет оценки после каждого шага, используя наблюдаемое вознаграждение R_{t+1} и оценку следующего состояния $V(S_{t+1})$.

- **Пример**: **TD(0)** (одношаговый TD) — базовая версия, где обновление происходит сразу после перехода.

6.2 Преимущества TD-методов

- **Против ДП**: не требует модели окружения (распределений переходов и вознаграждений).
- **Против МК**:
  - **Инкрементальность**: Обновления происходят после каждого шага, а не в конце эпизода.
  - Эффективен в задачах с длинными эпизодами или непрерывных средах.
- **Сходимость**: На практике TD-методы сходятся быстрее МК на стохастических задачах, хотя строгих доказательств нет.

6.3 Оптимальность TD(0)

- При пакетном обновлении **TD(0)** сходится к детерминированному решению (при достаточно малом шаге).
- Стохастически эквивалентная оценка (оптимальное решение) требует больших вычислительных ресурсов, но **TD-методы** аппроксимируют ее с линейной сложностью.

- **Табличный TD(0) для оценивания** v_π

  Вход: оцениваемая стратегия π.
  Параметр алгоритма: размер шага $\alpha \in (0, 1]$.

  Инициализация:

  $V(s)$ — произвольные значения для всех $s \in S^+$, где $V(\text{terminal}) = 0$.

  Повторять для каждого эпизода:

  1. Инициализировать начальное состояние S.
  2. Повторять для каждого шага эпизода:

     $A \leftarrow$ действие, выбранное π для S
     Предпринять A, наблюдать R, S'
     $V(S) \leftarrow V(S) + \alpha \left[R + \gamma V(S') - V(S)\right]$
     $S \leftarrow S'$

  3. Продолжать, пока S не станет терминальным состоянием.

6.4 Sarsa: TD-управление с единой стратегией

- **Цель**: Оценка функции ценности действий $q_\pi(s, a)$ для текущей стратегии π.

- **Сходимость**: Гарантируется при условии бесконечного посещения всех пар (**состояние–действие**) и сходимости стратегии к жадной (например, через ε-жадность).

- **Sarsa (TD-управление с единой стратегией) для оценивания** $Q \approx q_*$

  Параметры алгоритма: - Размер шага $\alpha \in (0, 1]$ - Небольшое $\varepsilon > 0$.

Инициализация:

$Q(s, a)$ — произвольные значения для всех $s \in S^+, a \in \mathcal{A}(s)$, где $Q(\text{termina}$

Повторять для каждого эпизода:

1. Инициализировать начальное состояние S.
2. Выбрать действие A в состоянии S, следуя стратегии на основе Q (например, ε-жадной).
3. Повторять для каждого шага эпизода:

   Предпринять действие A, наблюдать R, S'
   Выбрать действие A' в S', следуя стратегии на основе Q
   $Q(S, A) \leftarrow Q(S, A) + \alpha \left[R + \gamma Q(S', A') - Q(S, A)\right]$
   $S \leftarrow S', \quad A \leftarrow A'$

4. Продолжать, пока S не станет терминальным состоянием.

6.5 Q-обучение: TD-управление с разделенной стратегией

- **Цель**: Непосредственная аппроксимация оптимальной функции ценности действий q^*.
- **Особенности**:
  - Стратегия влияет на выбор действий, но оценка Q не зависит от текущей стратегии.
  - Сходится к q^* при условии обновления всех пар и выполнения условий стохастической аппроксимации.

- **Q-обучение (TD-управление с разделенной стратегией) для оценивания $\pi \approx \pi_*$**

  Параметры алгоритма:

  – Размер шага $\alpha \in (0, 1]$

  – Небольшое $\varepsilon > 0$

  Инициализация:

  $Q(s, a)$ — произвольные значения для всех $s \in S^+, a \in \mathcal{A}(s)$, где $Q(\text{termina}$

  Повторять для каждого эпизода:

  1. Инициализировать начальное состояние S.
  2. Повторять для каждого шага эпизода:

     Выбрать действие A в состоянии S, следуя стратегии на основе Q (на
     Предпринять действие A, наблюдать R, S'
     $Q(S, A) \leftarrow Q(S, A) + \alpha \left[R + \gamma \max_a Q(S', a) - Q(S, A) \right]$
     $S \leftarrow S'$

  3. Продолжать, пока S не станет терминальным состоянием.

6.6 Expected Sarsa

- **Улучшение Sarsa**: заменяет случайное следующее действие A_{t+1} на математическое ожидание по стратегии:

- **Преимущества**:

  – Снижает дисперсию за счет устранения стохастичности A_{t+1}.

- **Гибридность**: при использовании жадной стратегии совпадает с **Q-обучением**.

6.7 Смещение максимизации и двойное обучение

- **Проблема**: Максимизация по оценкам Q приводит к положительному смещению (например, из-за шумных оценок).
- **Решение**:
  - **Двойное Q-обучение**: использует две независимые оценки Q_1 и Q_2.

6.8 Специальные случаи: Послесостояния

- **Послесостояния**: Состояния после совершения действия, где известны непосредственные последствия (например, ходы в играх).
- **Функция ценности послесостояний**:
  - Оценивает позиции после действия, а не пары (**состояние**–**действие**).
  - Эффективна в задачах с частично известной динамикой (например, шахматы: известен результат хода, но не ответ противника).
- **Пример**: В крестиках-ноликах разные действия могут приводить к одинаковым послесостояниям, что упрощает обучение.

7 Исполнитель-критик

7.1 REINFORCE с базой

- **Основные характеристики**:
  - Обучает как стратегию (политику), так и функцию ценности состояний.
  - Не относится к методам **исполнитель-критик**, так как функция ценности используется только как база (для снижения дисперсии), но не для бутстрэппинга.
  - **Бутстрэппинг** — обновление оценки ценности состояния на основе оценок последующих состояний. В **REINFORCE** отсутствует, что делает алгоритм несмещенным, но увеличивает дисперсию.
- **Преимущества и недостатки**:
  - Асимптотическая сходимость к локальному минимуму.
  - Медленное обучение из-за высокой дисперсии (характерно для методов **Монте-Карло**).
  - Неудобен для онлайн-режима и непрерывных задач.

7.2 Методы исполнитель-критик

- **Ключевые особенности**:
  - **Критик** использует бутстрэппинг, что вводит смещение, но снижает дисперсию и ускоряет обучение.
  - Сочетает преимущества **TD-методов** (временных различий) и градиента стратегии.

- **Одношаговые методы**:
  - Аналоги **TD(0)**, **Sarsa(0)** и **Q-обучения**.
  - Полностью онлайн и инкрементны: обрабатывают состояния, действия и вознаграждения по мере поступления.
  - Заменяют полный доход в **REINFORCE** на одношаговый доход (с использованием обученной функции ценности в качестве базы).
  - Для обучения функции ценности применяется полуградиентный **TD(0)**.
- **Одношаговый метод исполнитель–критик (эпизодический) для оценивания** $\pi_\theta \approx \pi_*$

  Вход:
  - Дифференцируемая параметризация стратегии $\pi(a|s,\theta)$
  - Дифференцируемая функция ценности состояний $\hat{v}(s,w)$

  Параметры алгоритма: размеры шагов $\alpha^\theta > 0$, $\alpha^w > 0$.

  Инициализация:

  $\theta \in \mathbb{R}^d, \quad w \in \mathbb{R}^d$ (например, инициализировать нулями).

  Повторять бесконечно (для каждого эпизода):

  1. Инициализировать начальное состояние S.
  2. $I \leftarrow 1$.

3. Повторять, пока S не станет терминальным:

   $A \sim \pi(\cdot|S, \theta)$
   Предпринять действие A, наблюдать S', R
   $\delta \leftarrow R + \gamma\hat{v}(S', w) - \hat{v}(S, w)$ (если S' терминальное, то $\hat{v}(S', w) = 0$)
   $w \leftarrow w + \alpha^w \delta \nabla \hat{v}(S, w)$
   $\theta \leftarrow \theta + \alpha^\theta I \delta \nabla \ln \pi(A|S, \theta)$
   $I \leftarrow \gamma I$
   $S \leftarrow S'$

- **Обобщения**:
  - **n-шаговые методы**: заменяют одношаговый доход на $G_{t:t+n}$ (многошаговый доход).
  - **λ-доходный алгоритм**: использует G_t^λ для гибкого выбора степени бутстрэппинга.
  - **Обратное представление**: реализуется через раздельные следы приемлемости для исполнителя и критика (аналогично главе 12).

7.3 Сравнение подходов

- **REINFORCE с базой** сохраняет теоретическую корректность (несмещенность), но на практике уступает методам **исполнитель-критик** из-за высокой вычислительной нагрузки и ограниченной применимости.

- Методы **исполнитель-критик**, несмотря на смещение, обеспечивают быстрое обучение, низкую дисперсию и гибкость, что делает их предпочтительными в большинстве реальных сценариев, особенно там, где требуется онлайн-адаптация.

7.4 Дополнительные замечания

- **Полуградиентный TD(0)**: Используется для обучения функции ценности в одношаговых методах.
- **Следы приемлемости**: Позволяют гибко управлять влиянием предыдущих состояний на обновления.
- **Алгоритмы**: Псевдокод (не приведен в тексте) описывает полностью инкрементные и онлайн-процессы, что устраняет необходимость хранения полной истории эпизодов.

8 Прогнозирование и аппроксимация функций ценности

8.1 Концепция обновления функции ценности

- Все методы предсказания сводятся к корректировке оценки ценности состояний в направлении цели обновления $s \to u$, где:
  - s — состояние, для которого обновляется ценность;
  - u — целевое значение (новая оценка), к которому сдвигается текущая ценность s.
- **Примеры методов**:
  - **Монте-Карло**.
  - **TD(0)** (цель — немедленная награда + дисконтированная оценка следующего состояния).
  - **n-шаговое TD** (цель — промежуточный возврат за n шагов).

– **Динамическое программирование (ДП)** (цель — математическое ожидание для произвольного состояния s, а не только встреченного в опыте).

8.2 Обобщение обновлений через аппроксимацию функций

- **Интерпретация обновлений**:
  - Каждое обновление $s \to u$ рассматривается как обучающий пример, где s — вход, u — желаемый выход.
  - Задача: сделать оценку ценности s ближе к u, используя методы обучения с учителем (аппроксимация функции).
- **Отличия от табличных методов**:
  - В табличном случае обновляется только элемент, соответствующий s.
  - При аппроксимации обновление распространяется на множество состояний, что позволяет обобщать опыт.

8.3 Применение методов машинного обучения

- **Используемые методы**:
  - Искусственные нейронные сети, решающие деревья, многомерная регрессия и др.
  - Обучающие данные: пары $s \to u$ (состояние — цель) для каждого обновления.
- **Требования к методам в обучении с подкреплением**:

- **Онлайн-обучение**: данные поступают постепенно в процессе взаимодействия со средой (а не статический набор).
- **Работа с нестационарными целями**: целевые значения u могут меняться со временем из-за:
  * Изменения стратегии π (например, в обобщенной итерации по стратегиям).
  * Использования бутстрэппинга (методы ДП и TD), где оценка ценности зависит от текущих весов модели.

8.4 Проблемы традиционных методов аппроксимации

- **Недостатки классических подходов**:
  - Многие методы (например, сложные нейросетевые архитектуры) рассчитаны на многопроходное обучение по фиксированным данным.
  - Невозможность адаптации к нестационарности целевой функции (например, при изменении стратегии или обновлении весов модели).
- **Ключевые ограничения**:
  - Методы, не способные обучаться инкрементально (постепенно) или учитывать изменяющиеся цели, плохо подходят для задач обучения с подкреплением.

8.5 Итоговые требования к методам аппроксимации

1. **Эффективность в онлайн-режиме**: быстрое обновление модели на лету без полного пересчета.
2. **Устойчивость к нестационарности**: адаптация к изменяющимся целевым значениям u.
3. **Способность к обобщению**: перенос информации между состояниями для уменьшения объема вычислений.
4. **Совместимость с бутстрэппингом**: работа в условиях, когда цели зависят от текущих оценок модели.

9 Стохастические градиентные и полуградиентные методы

9.1 Основные принципы СГС-методов

- **Цель**: Корректировка вектора весов $\mathbf{w}$ для минимизации ошибки аппроксимации функции ценности $v(s, \mathbf{w})$.
- **Структура**:
  - Вектор весов $\mathbf{w} = (w_1, w_2, \ldots, w_d)^T$ имеет фиксированную размерность.
  - Функция $v(s, \mathbf{w})$ дифференцируема по $\mathbf{w}$ для всех состояний $s \in S$.
- **Обновление весов**:

- На каждом шаге t обрабатывается пример $S_t \to v_\pi(S_t)$, где S_t — состояние, $v_\pi(S_t)$ — его истинная ценность (цель).
- Обновление весов происходит по формуле: **формула** (место для формулы).

9.2 Условия сходимости СГС

- **Убывающий шаг** α:
  - Для гарантии сходимости к локальному оптимуму параметр α должен удовлетворять условиям стохастической аппроксимации: **формула** (место для формулы).
  - Это обеспечивает баланс между исследованием и эксплуатацией.
- **Ограничения аппроксиматора**:
  - Из-за ограниченности ресурсов точное описание всех состояний невозможно.
  - Решение должно обобщаться на состояния, не встречавшиеся в примерах.

9.3 Работа с неточными целями U_t

- **Общий случай**:
  - Если цель U_t (например, зашумленная оценка $v_\pi(S_t)$ или бутстрэппинговая цель) несмещенная, то СГС сходится к локальному оптимуму.
  - Пример: Метод Монте-Карло, где $U_t = G_t$ (полный возврат), является несмещенной оценкой $v_\pi(S_t)$.

- **Бутстрэппинговые методы**:
  - Цели зависят от текущих весов $\mathbf{w}_t$ (например, **формула** в TD(0)).
  - Это приводит к смещению, так как цель U_t не является независимой от $\mathbf{w}_t$.
  - Такие методы называются полуградиентными, так учитывается только часть градиента (игнорируется влияние $\mathbf{w}_t$ на цель).

9.4 Полуградиентные методы: особенности и примеры

- **Особенности**:
  - Не гарантируют сходимость, но на практике часто работают быстрее СГС.
  - Позволяют онлайн-обучение без ожидания завершения эпизода (например, TD(0)).
- **Примеры**:
  - **TD(0)**:
    - Псевдокод включает обновление весов после каждого шага взаимодействия со средой.
  - **n-шаговые методы и ДП**:
    - Используют цели, зависящие от нескольких шагов или модели среды, что усиливает смещение.
- **Полуградиентный алгоритм TD(0) для оценивания** $\hat{v} \approx v_\pi$

Вход:

- Оцениваемая стратегия π
- Дифференцируемая функция $\hat{v} : S^+ \times \mathbb{R}^d \to \mathbb{R}$, где $\hat{v}(\text{terminal}, \cdot) = 0$

Параметр алгоритма: размер шага $\alpha > 0$.

Инициализация:

$$w \in \mathbb{R}^d \text{ — произвольные веса (например, } w = 0).$$

Повторять для каждого эпизода:

1. Инициализировать начальное состояние S.
2. Повторять для каждого шага эпизода:

   Выбрать действие $A \sim \pi(\cdot|S)$
   Предпринять действие A, наблюдать R, S'
   $w \leftarrow w + \alpha \left[R + \gamma \hat{v}(S', w) - \hat{v}(S, w)\right] \nabla \hat{v}(S, w)$
   $S \leftarrow S'$

3. Продолжать, пока S не станет терминальным состоянием.

9.5 Агрегирование состояний как частный случай СГС

- **Принцип работы**:
  - Состояния группируются, каждой группе соответствует один элемент вектора $\mathbf{w}$.
  - Оценка ценности состояния равна весу его группы.

  - При обновлении корректируется только вес группы текущего состояния S_t.

- **Математическая интерпретация**:
  - Градиент **формула** (место для формулы) равен 1 для элемента группы S_t и 0 для остальных.

9.6 Преимущества и недостатки методов

- **СГС**:
  - **Плюсы**: Гарантированная сходимость к локальному оптимуму при несмещенных целях.
  - **Минусы**: требует точных целей $v_\pi(S_t)$, что редко доступно на практике.

- **Полуградиентные методы**:
  - **Плюсы**:
    * Возможность онлайн-обучения.
    * Высокая скорость обучения (например, TD(0) быстрее Монте-Карло).
  - **Минусы**:
    * Нет гарантий сходимости из-за смещения.
    * Зависимость от текущих весов усложняет анализ.

10 Нелинейная аппроксимация функций: искусственные нейронные сети

10.1 Основные понятия ИНС

- **Структура сетей**:
  - **Сети прямого распространения (feedforward)**:
    * Не содержат циклов: выходы не влияют на входы.
    * Состоят из входного, скрытых и выходного слоев.
  - **Рекуррентные сети**:
    * Содержат циклы, позволяющие обрабатывать последовательности данных.
    * В книге рассматриваются только сети прямого распространения для упрощения.
- **Функции активации**:
  - **Сигмоидная**: $f(x) = \frac{1}{1+e^{-x}}$.
  - **Ректификатор (ReLU)**: $f(x) = \max(0, x)$.
  - **Ступенчатая функция**: $f(x) = 1$ при $x \geq 0$, иначе 0.
  - **Линейные функции** (не используются в скрытых слоях, так как делают сеть эквивалентной однослойной).

10.2 Универсальная аппроксимация

- **Теорема Cybenko (1989)**:
  - ИНС с одним скрытым слоем, содержащим достаточное количество сигмоидных блоков, может аппроксимировать

любую непрерывную функцию на компактной области входного пространства.

  - Условие: функция активации должна быть нелинейной.

- **Глубокие сети**:

  - На практике глубокие ИНС (со многими скрытыми слоями) эффективнее для сложных задач (распознавание образов, иерархические признаки).
  - Каждый слой формирует более абстрактные представления данных (например, края → формы → объекты в изображениях).

10.3 Обучение ИНС

- **Метод обратного распространения (backpropagation)**:

  - Основан на стохастическом градиентном спуске (СГС).
  - **Прямой проход**: вычисление активаций всех блоков.
  - **Обратный проход**: вычисление градиентов ошибки по весам для их обновления.
  - Проблема: градиенты затухают или взрываются в глубоких сетях, что замедляет обучение.

- **Целевые функции**:

  - В обучении с учителем: минимизация ошибки на обучающих данных.
  - В обучении с подкреплением: TD-ошибка, максимизация ожидаемого дохода.

10.4 Проблемы обучения глубоких сетей

- **Переобучение**:
  - Сети с большим числом параметров (весов) запоминают шум в данных.
  - Методы борьбы:
    * **Прореживание (dropout)**: случайное отключение блоков во время обучения для повышения обобщения.
    * **Регуляризация**: добавление штрафа за большие веса (например, L1/L2).
    * **Ранняя остановка**: прекращение обучения при ухудшении качества на валидационных данных.
- **Затухание/взрыв градиентов**:
  - Градиенты становятся слишком малыми (или большими) при передаче через слои.
  - Решение: методы инициализации весов, пакетная нормировка, остаточные связи.

10.5 Методы улучшения обучения глубоких сетей

- **Пакетная нормировка (batch normalization)**:
  - Нормировка выходов слоев перед передачей на следующий слой (нулевое среднее, единичная дисперсия).
  - Ускоряет обучение и стабилизирует градиенты.
- **Глубокое остаточное обучение (residual learning)**:
  - Обучение разности между входом и выходом группы слоев: $F(x) + x$.

- Решает проблему затухания градиентов в глубоких сетях.
- Пример: добавление прямых связей в обход слоев (He et al., 2016).

- **Инициализация весов**:
  - Предобучение слоев с помощью обучения без учителя (например, сети глубокого доверия Hinton et al., 2006).
  - Помогает найти хорошие начальные точки для градиентного спуска.

10.6 Архитектуры ИНС

- **Глубокие сверточные сети (CNN)**:
  - Предназначены для обработки пространственных данных (изображения, аудио).
  - **Сверточные слои**:
    * Выделяют локальные признаки с помощью фильтров (карты признаков).
    * Веса фильтров разделяются между всеми позициями в карте.
  - **Слои подвыборки (pooling)**:
    * Уменьшают размерность данных (например, усреднение или максимум по областям 2×2).
    * Повышают инвариантность к сдвигам и поворотам.
  - **Пример**: сеть LeNet-5 (LeCun et al., 1998) для распознавания рукописных цифр:
    * Чередование сверточных слоев (5×5 фильтры) и слоев подвыборки (2×2).

* Завершается полносвязными слоями.

10.7 Применение в обучении с подкреплением

- **Роль ИНС**:
  - Аппроксимация функций ценности и стратегий в задачах с большими пространствами состояний.
  - Примеры: AlphaGo (глава 16), где глубокие CNN использовались для оценки позиций.
- **Преимущества**:
  - Автоматическое извлечение иерархических признаков.
  - Возможность обработки неструктурированных данных (изображения, текст).
- **Ограничения**:
  - Высокие вычислительные затраты.
  - Сложность интерпретации внутренних представлений.

11 Практические примеры применения обучения с подкреплением

11.1 TD-Gammon (Нарды)

- **Основная идея**: Программа для игры в нарды, использующая алгоритм TD(λ) с нейронной сетью (ИНС) и методом обратного распространения ошибок.
- **Особенности**:

  - Оценка ценности позиции (вероятность выигрыша) через многослойную ИНС с 198 входными параметрами, кодирующими состояние доски.
  - Обучение через самоигры: программа генерирует партии, играя против себя, и обновляет веса сети после каждого хода.
  - Успехи: TD-Gammon 3.0 достигла уровня сильнейших игроков мира, повлияла на стратегии профессионалов.

- **Инновации**:
  - Использование специализированных признаков для нард (начиная с версии 1.0) улучшило качество игры.
  - Комбинация функций ценности, поиска и методов Монте-Карло.

11.2 Программа Сэмюэла (Шашки)

- **Исторический контекст**: Одна из первых самообучающихся программ (1950-е гг.).
- **Методы**:
  - **Зубрёжка**: Сохранение оценок позиций и их повторное использование для увеличения глубины поиска.
  - **Обучение обобщением**: Корректировка параметров функции ценности через игры против самой себя.
  - Минимаксный поиск с эвристиками для оценки позиций.
- **Результаты**: Программа достигла уровня выше среднего начинающего, но не мастерского.

- **Проблемы**: Отсутствие явного вознаграждения и риск деградации функции ценности.

11.3 Управление памятью (DRAM)

- **Задача**:Планирование запросов к памяти для минимизации задержек.
- **Методы**:
  - Алгоритм Sarsa с линейной аппроксимацией функции ценности действий.
  - Плиточное кодирование для представления состояний (6 признаков: количество запросов чтения/записи, ожидающих строк и т.д.).
- **Результаты**:
  - Самообучающийся контроллер (RL) показал на 7–33% выше производительность, чем традиционные методы (FR-FCFS).
  - Онлайновое обучение улучшило результат на 8% по сравнению с фиксированной стратегией.

11.4 Персонализация веб-служб

- **Задача**: Максимизация кликов (CTR) и пожизненной ценности (LTV).
  - **Жадная оптимизация**: Случайный лес для предсказания кликов.
  - **LTV-оптимизация**: Алгоритм FQI (подобранная Q-итерация) с учётом долгосрочных последствий.

- **Результаты**: LTV-стратегия увеличила вовлечённость пользователей, что подтверждено тестами с высокой достоверностью.

11.5 Парение в восходящих потоках

- **Моделирование**:Турбулентная атмосфера + аэродинамика планера.
- **Метод**: Алгоритм Sarsa с агрегированием состояний (вертикальная скорость/ускорение ветра, крутящий момент).
- **Результаты**:
  - Обученная стратегия позволяет планеру набирать высоту по спирали в восходящем потоке.
  - Ключевые признаки: вертикальное ускорение ветра и крутящий момент.

12 Ограничения подхода обучения с подкреплением

12.1 Эволюционные методы

- **Примеры**: генетические алгоритмы, генетическое программирование, имитация отжига.
- **Принцип работы**:
  - Используются статические стратегии, которые взаимодействуют со средой в течение длительного времени.
  - Стратегии с максимальным вознаграждением и их модификации передаются следующему поколению.

- Процесс напоминает биологическую эволюцию.

- **Преимущества**:
  - Эффективны, если пространство стратегий небольшое, хорошо организовано, или поиск может занимать много времени.
  - Полезны в задачах, где агент не воспринимает полное состояние среды.

- **Недостатки**:
  - Игнорируют структуру задачи ОП: не используют информацию о состояниях и действиях в процессе взаимодействия.
  - Не учитывают, что стратегия должна быть функцией, отображающей состояния в действия.

12.2 Сравнение методов обучения с подкреплением и эволюционных подходов

- **Методы ОП**:
  - Обучаются через взаимодействие со средой, анализируя детали отдельных актов (например, переходы между состояниями, выбор действий).
  - Используют информацию о состояниях и действиях, что повышает эффективность поиска оптимальных стратегий.

- **Эволюционные методы**:
  - Не способны извлекать пользу из последовательности состояний и действий.

– В большинстве случаев менее эффективны, чем методы ОП, из-за игнорирования структуры задачи.

12.3 Заключение

- Фокус на методах обучения с подкреплением, так как они лучше адаптированы для задач, требующих анализа взаимодействий со средой.
- Эволюционные методы рассматриваются как вспомогательные, но не как основное направление из-за их ограничений в использовании информации о состояниях и действиях.

Список литературы

[1] Рассел С., Норвиг П. *Искусственный интеллект: современный подход. 4-е изд., том 1. Решение проблем: знания и рассуждения.* СПб.: Диалектика, 2021.

[2] Рассел С., Норвиг П. *Искусственный интеллект: современный подход. 4-е изд., том 3. Обучение, восприятие и действие.* СПб.: Диалектика, 2022.

[3] Саттон Р. С., Барто Э. Д. *Обучение с подкреплением: Введение. 2-е изд.* М.: ДМК Пресс, 2020. 552 с.

[4] «Искусственные нейронные сети» [Электронный ресурс]. `http://bigor.bmstu.ru/?cnt/?doc=NN/base.cou`.