Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИУ_____

КАФЕДРА ИУ5_____

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

НА ТЕМУ:

Обучение с подкреплением

_____

_____

_____

_____

Студент ИУ5-63Б _____ **Ларионова А.П.**
(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель _____ **Гапанюк Ю.Е.**
_____
(Подпись, дата) (И.О.Фамилия)

Консультант _____ _____
(Подпись, дата) (И.О.Фамилия)

2022 г.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ

Заведующий кафедрой _____
(Индекс)

_____ _____
(И.О.Фамилия)

« _____ » _____ 20 ____ г.

З А Д А Н И Е
на выполнение научно-исследовательской работы

по теме “ Обучение с подкреплением”

_____

_____

Студент группы ИУ5-63Б_____

Ларионова Амина Павловна

_____
(Фамилия, имя, отчество)

Направленность НИР (учебная, исследовательская, практическая, производственная, др.)

_____учебная _____

Источник тематики (кафедра, предприятие, НИР) _____

График выполнения НИР: 25% к ___ нед., 50% к ___ нед., 75% к __ нед., 100% к ___ нед.

Техническое задание _____

_____

_____

Оформление научно-исследовательской работы:

Расчетно-пояснительная записка на _____ листах формата А4.
Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)

_____

_____

_____

Дата выдачи задания « ___ » _____ 2022 г.

Руководитель НИР _____ _____**Гапанюк Ю.Е.**
_____
(Подпись, дата) (И.О.Фамилия)

Студент _____ _____**Ларионова А.П.**
_____
(Подпись, дата) (И.О.Фамилия)

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

Оглавление

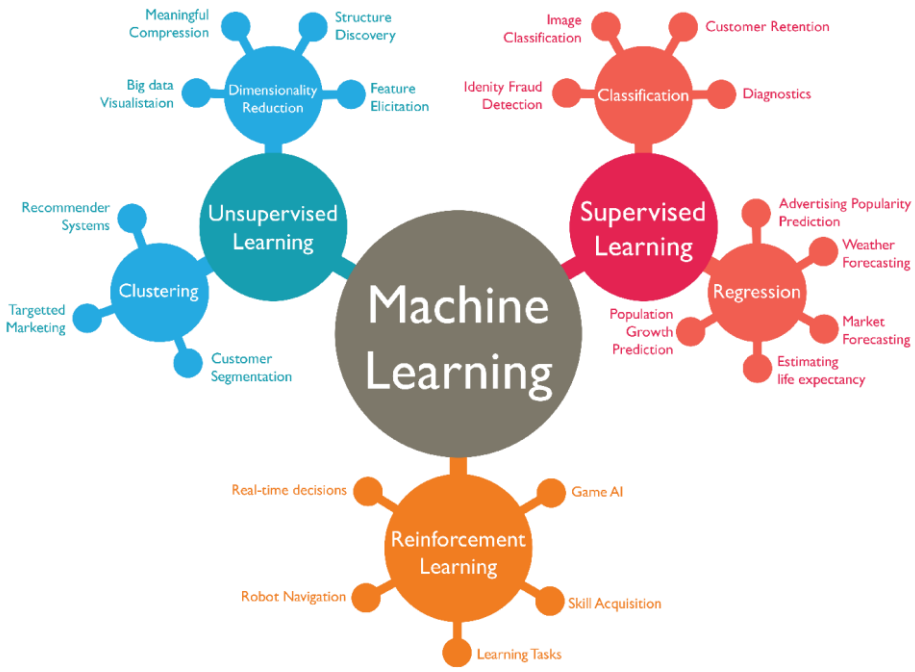
Идея обучения с подкреплением возникла десятки лет назад, но этой дисциплине предстояло пройти долгий путь, прежде чем она стала одним из самых активных направлений исследований в области машинного обучения и нейронных сетей. Сегодня это предмет интереса ученых, занимающихся психологией, теорией управления, искусственным интеллектом и многими другими отраслями знаний.

Обучение с подкреплением – это один из разделов машинного обучения, исследующий вычислительный подход к обучению агента, который пытается максимизировать свою совокупную накопленную награду путем взаимодействия со сложной, зачастую стохастической средой. Последние несколько лет исследования этого подхода переживают настоящий ренессанс – ни одна научная конференция по искусственному интеллекту не обходится без секции на эту тему. Каждый год публикуются сотни научных статей, и все больше компаний в России и за рубежом начинают применять последние достижения этой области в своем бизнесе для улучшения различных внутренних процессов – от рекомендательных систем до оптимизации цепей поставок.

Введение в обучение с подкреплением

Для начала рассмотрим машинное обучение и его виды.
Машинное обучение — это специализированный способ, позволяющий обучать компьютеры, не прибегая к программированию. Для простоты восприятия типы машинного обучения принято разделять на три категории:

1) Обучение с учителем (supervised learning)
2) Обучение без учителя (unsupervised learning)
3) Обучение с подкреплением (reinforcement learning)



Обучение с учителем (Supervised learning) — один из способов машинного обучения, в ходе которого испытуемая система принудительно обучается с помощью примеров «стимул-реакция». С точки зрения кибернетики, является одним из видов кибернетического эксперимента. Между входами и эталонными выходами (стимул-реакция) может существовать некоторая зависимость, но она неизвестна. Известна только конечная совокупность прецедентов — пар «стимул-реакция», называемая обучающей выборкой. На основе этих данных требуется восстановить зависимость (построить модель отношений стимул-реакция, пригодных для прогнозирования), то есть построить алгоритм, способный для любого объекта выдать достаточно точный ответ. Для измерения точности ответов, так же как и в обучении на примерах, может вводиться функционал качества.

Обучение без учителя (Unsupervised learning) — один из способов машинного обучения, при котором испытуемая система спонтанно обучается выполнять поставленную задачу без вмешательства со стороны экспериментатора. С точки зрения кибернетики, это является одним из видов кибернетического эксперимента. Как правило, это пригодно только для задач, в которых известны описания множества объектов (обучающей выборки), и требуется обнаружить внутренние взаимосвязи, зависимости, закономерности, существующие между объектами.

Обучение без учителя часто противопоставляется обучению с учителем, когда для каждого обучающего объекта принудительно задаётся «правильный ответ», и требуется найти зависимость между стимулами и реакциями системы.

Обучение с подкреплением (reinforcement learning) — один из способов машинного обучения, в ходе которого испытуемая система (агент) обучается, взаимодействуя с некоторой средой. С точки зрения кибернетики, является одним из видов кибернетического эксперимента. Откликом среды (а не специальной системы управления подкреплением, как это происходит в обучении с учителем) на принятые решения являются сигналы подкрепления, поэтому такое обучение является частным случаем обучения с учителем, но учителем является среда или её модель. Также нужно иметь в виду, что некоторые правила подкрепления базируются на неявных учителях, например, в случае искусственной нейронной среды, на одновременной активности формальных нейронов, из-за чего их можно отнести к обучению без учителя.

Обучение с подкреплением

Обучение с подкреплением- обучение интеллектуального агента хорошей последовательности принятия решения в условиях неполной информации.

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.

Основная мысль- 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.

Отличия от обучения с учителем и от обучения без учителя

Обучение с подкреплением отличается от **обучения с учителем**, еще одного вида обучения, который изучается в большинстве современных работ по машинному обучению. В случае обучения с учителем имеется обучающий набор помеченных примеров, подготовленный квалифицированным внешним учителем. Каждый пример представляет собой описание ситуации и спецификацию – метку – правильного действия, которое система должна предпринять в этой ситуации. Часто метка определяет категорию, которой принадлежит ситуация. Цель такого обучения – добиться, чтобы система смогла экстраполировать, или обобщить, свою реакцию на ситуации, которые не были предъявлены в обучающем наборе. Это важный вид обучения, но, взятый сам по себе, он не подходит для обучения с помощью взаимодействия. В интерактивных задачах часто практически невозможно получить примеры желаемого поведения, которые правильно представляли бы все ситуации, в которых агенту предстоит действовать. На неизведанной территории – там, где от обучения как раз и ожидают плодов, – агент должен уметь действовать, исходя из своего опыта.

Обучение с подкреплением отличается и от так называемого **обучения без учителя**, которое обычно имеет целью обнаружение структуры, скрытой в наборе непомеченных данных. Кажется, что термины «обучение с учителем» и «обучение без учителя» исчерпывают все возможные парадигмы машинного обучения, однако это не так. Может возникнуть соблазнительная мысль о том, что обучение с подкреплением – разновидность обучения без учителя, поскольку отсутствуют примеры правильного поведения. Но в действительности **цель обучения с подкреплением** – максимизировать вознаграждение, а не выявить скрытую структуру. Выявление структуры

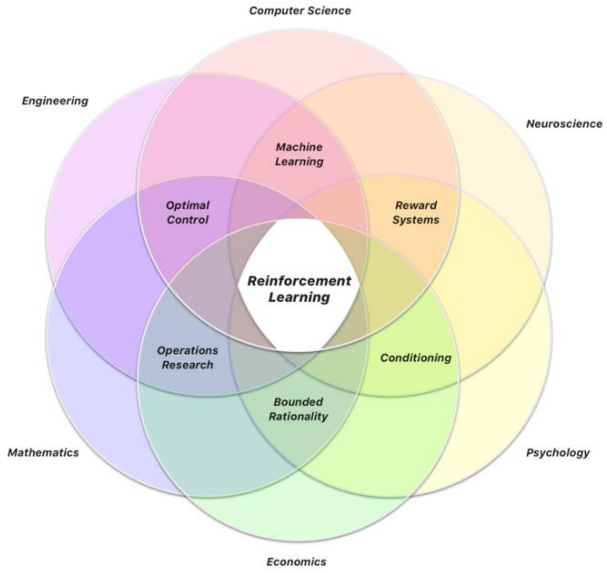
в опыте агента, конечно, может быть полезно, но само по себе не решает задачу максимизации вознаграждения, стоящую перед обучением с подкреплением. Поэтому мы считаем обучение с подкреплением третьей парадигмой машинного обучения наряду с обучением с учителем, без учителя и, возможно, еще какими-то парадигмами.

Взаимосвязь с другими дисциплинами

Один из самых захватывающих аспектов современного обучения с подкреплением – его содержательные и плодотворные связи с другими инженерными и научными дисциплинами. Обучение с подкреплением – часть наблюдающейся уже несколько десятилетий тенденции к более тесной интеграции между искусственным интеллектом и машинным обучением, с одной стороны, и статистикой, оптимизацией и другими разделами математики – с другой. Например, способность некоторых методов обучения с подкреплением обучаться с помощью параметризованных аппроксиматоров решает классическую проблему «проклятия размерности» в исследовании операций и теории управления. Еще более заметна сильная связь обучения с подкреплением с психологией и нейронауками, приносящая ощутимые выгоды обеим сторонам. Из всех видов машинного обучения именно обучение с подкреплением ближе всего к способам обучения людей и животных, и истоки многих ключевых алгоритмов обучения с подкреплением следует искать в биологических самообучающихся системах. Обучение с подкреплением также вернуло долг – как в виде психологической модели обучения животных, лучше соответствующей некоторым эмпирическим данным, так и в виде влиятельной модели частей системы вознаграждения мозга.

Особенность RL

С классической точки зрения обучение с подкреплением является частью машинного обучения, однако это нечто большее. Обучение с подкреплением является наукой на стыке нескольких различных ветвей, а именно: психология, программирование, инженерия, компьютерные науки и экономика.
При пересечение все науки, перечисленные выше, дают нам обучение с подкреплением.



Главные особенности обучения с подкреплением

1) Нет учителя, т.е. ошибка не задается явно, а косвенно передается через вознаграждение. Однако это нельзя назвать обучением без учителя, так как мы все же получаем ответ от среды, но не сразу, а через какой-то нефиксированный промежуток времени.
2) Обратная связь от среды может поступать с задержкой.
3) Параметр времени имеет особое значение, то есть наши данные поступают последовательно. Наша задача-предсказание последовательности действий, каждое последующее действие зависит от предыдущего и оптимальность всего решения зависит от каждого из шагов.
4) Действия агента влияют на поступающие в дальнейшем данные.

Применение:

1) AlphaGo- игра в Go лучше человека.
2) Игра в игры Atari, StarCraft, Dota, MineRL – иногда лучше человека.
3) Контроль движения человекоподобного робота.
4) Управление технологическим процессами, заводами и станциями.
5) Управление инверсионными пакетами, игра на бирже.
6) Автоматизация “холодных звонков”.
7) Реализация сложных движений техники (например, вертолета).

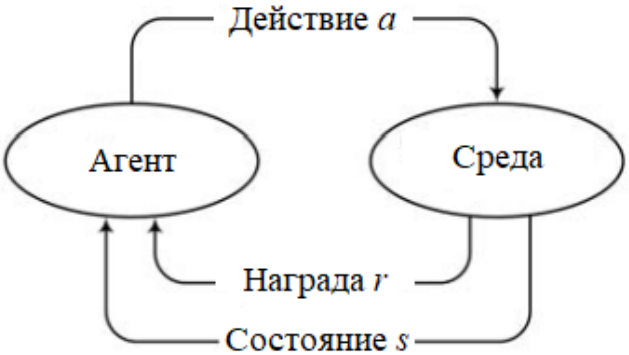
Основные понятия:

1) *Агент*- обучается посредством взаимодействия с окружающей средой для достижения некоторой цели. Агент — человек или объект, который играет активную роль. На практике это фрагмент кода, в котором реализована стратегия. В основном данная стратегия принимает решение о том, какое действие необходимо, исходя из наблюдений.

2) *Окружающая среда*. Некая модель мира, являющаяся внешней по отношению к агенту и отвечающая за предоставление нам наблюдений и вознаграждений. Ее состояние изменяется в зависимости от наших действий.
3) s ∈ S – *состояние*. Наша среда в определенный момент времени может принимать только одно состояние.
4) a ∈ A – *действие*, которое совершает агент по отношению с среде.
5) p (s' | a, s) – модель переходов в среде, которую мы не знаем. В зависимости от текущего состояния и действия, которое было предпринято в этом состоянии мы переходим в некоторое состояние s'.
6) π (a | s)- *политика / стратегия*. То есть какое действие a мы принимаем, находясь в состоянии s.
7) r (s, a) – *награждения*.

Конечные марковские процессы принятия решений (МППР)

МППР предназначены для прямолинейной формулировки задачи обучения в результате взаимодействия для достижения цели. Сторона, которая обучается и принимает решения, называется агентом. Сторона, с которой агент взаимодействует, включающая в себя все, что находится вне агента, называется окружающей средой, или просто средой. Обе стороны взаимодействуют непрерывно – агент выбирает действия, а среда реагирует на эти действия и предлагает агенту новые ситуации. Среда также генерирует вознаграждения – числовые значения, которые агент стремится со временем максимизировать посредством выбора действий.



Траектория: {s0a0, s1a1, …}

Постановка задачи:

Суммарное дисконтированное награждение, $\gamma \in [0,1]$
R_t: =r (s_t, a_t) + γ r(s_{t+1}, a_{t+1})+ γ^2 r(s_{t+2},a_{t+2})+…-> max

В этой задаче присутствует оценочная обратная связь, но у нее имеется и ассоциативная сторона – выбор различных действий в разных ситуациях.

МППР – классическая формализация последовательного принятия решений, когда от действий зависит не только немедленное вознаграждение, но и последующие ситуации, или состояния, а через них и будущие вознаграждения. Таким образом, в МППР присутствует отложенное вознаграждение и необходимость искать компромисс между немедленным и отложенным вознаграждениями. В МППР мы оцениваем ценность q* (s, a) действия a в состоянии s или ценность v* (s) каждого состояния при условии оптимального выбора действия. Эти зависящие от состояния величины необходимы для правильного распределения поощрения между отдаленными последствиями выбора тех или иных действий.

МППР – математически идеализированная форма задачи обучения с подкреплением, для которой можно высказывать точные теоретические утверждения. Основными элементами математической постановки задачи являются: доход, функции ценности и уравнения Беллмана.

Под обучением с подкреплением понимается обучение в процессе взаимодействия с целью научиться правильно вести себя для достижения цели. Обучающийся с подкреплением агент и окружающая среда взаимодействуют на протяжении последовательности дискретных временных шагов. Описание их интерфейса определяет конкретную задачу: действия – это варианты, выбираемые агентом, состояния – основа для принятия решений о выборе, а вознаграждения – основа для оценивания выбора. Все, что находится внутри агента, полностью известно и контролируется агентом, а все, что вне, абсолютно не контролируется, но может быть или не быть полностью известно.

Стратегия – это стохастическое правило, руководствуясь которым, агент выбирает действия в зависимости от состояния.

Цель агента – максимизировать сумму вознаграждения, полученного на протяжении периода времени.

Если задача обучения с подкреплением сформулирована с помощью корректно определенных вероятностей переходов, то она образует **марковский процесс принятия решений (МППР).** Конечный МППР – это МППР с конечными множествами состояний, действий и вознаграждений (в том виде, в каком мы определили их здесь). Современная теория обучения с подкреплением по большей части ограничена конечными МППР, но методы и идеи имеют более широкое применение.

Доход – это функция будущих вознаграждений, которую агент стремится максимизировать (точнее, ее математическое ожидание). У него имеется несколько определений, зависящих от природы задачи и того, хотим ли мы обесценивать отложенное вознаграждение. Формулировка без обесценивания годится для эпизодических задач, когда взаимодействие между агентом и средой естественно распадается на эпизоды, а формулировка с обесцениванием – для непрерывных задач, когда взаимодействие не распадается на эпизоды, а продолжается безгранично. Мы хотим определить доход в задачах обоих видов, так чтобы к эпизодическому и непрерывному случаям был применим один и тот же набор уравнений.

Функции ценности стратегии сопоставляют каждому состоянию или паре состояние–действие ожидаемый доход, который агент может получить, применяя данную стратегию. Оптимальные функции ценности сопоставляют

каждому состоянию или паре состояние–действие наибольший ожидаемый доход, достижимый при использовании любой стратегии. Стратегия, для которой функции ценности оптимальны, называется оптимальной стратегией.

Оптимальные функции ценности для состояний и пар состояние–действие однозначно определены для данного МППР, но оптимальных стратегий может быть много. Любая стратегия, жадная относительно оптимальных функций ценности, должна быть оптимальной стратегией.

Уравнения оптимальности Беллмана – это специальные условия согласованности, которым должны удовлетворять оптимальные функции ценности и которые в принципе можно решить и тем самым найти оптимальные функции ценности, после чего можно сравнительно просто вычислить оптимальную стратегию.

Задачу обучения с подкреплением можно поставить разными способами в зависимости от предположений об уровне знаний, доступных агенту в начальный момент. В задачах с полным знанием агент располагает полной и точной моделью динамики окружающей среды. Если окружающая среда представляет собой МППР, то такая модель состоит из функции динамики p с четырьмя аргументами. В задачах с неполным знанием полная и точная модель среды отсутствует. Даже если у агента имеется полная и точная модель окружающей среды, он, как правило, все равно не может произвести на одном шаге достаточный объем вычислений, чтобы воспользоваться ей в полной мере.

Другим важным ограничением является доступная память. Память необходима для построения хороших аппроксимаций функций ценности, стратегий и моделей. В большинстве практически интересных случаев состояний гораздо больше, чем можно представить с помощью записей таблицы, поэтому необходима аппроксимация. В обучении с подкреплением нас очень интересуют случаи, в которых оптимальное решение найти невозможно, но его можно каким-то способом аппроксимировать.

Value-based reinforcement learning.

V- функция полезности состояний.

Также введем понятие функции полезности действий. Это предсказание будущего вознаграждения и используется для оценки состояний. Это математическое ожидание от вознаграждения, находясь в данный момент времени в состоянии s.

$$V^{\pi}(s) = \mathbb{E}_{\pi}\left[R_t | s_t = s\right]$$

Чтобы понять какое максимальное математическое ожидание мы можем получить, введем следующее понятие:

$$V^{*}(s) = \max_{\pi} V^{\pi}(s)$$

Лучшая стратегия в таком случае будет иметь вид:
Value iteration method.

$$\pi^*(a|s) = \arg\max_a \left[r(s_0, a_0) + \gamma \mathbb{E}_{p(s'|s_0,a_0)} V^*(s')\right]$$

Такой метод подходит далеко не для всех задач, так как в данной оценке все состояния у нас детерминированные.

Q- функция.

Функция полезности действий Q (s, a) – математическое ожидание отдачи, полученной, начиная с состояния s и выбранного действия a, при выполнении стратегии π.

$$Q^\pi(s, a) = \mathbb{E}_\pi \left[R_t | s_t = s, a_t = a\right]$$

$$Q^*(s, a) = \max_\pi Q^\pi(s, a)$$

Стратегия при этом будет иметь следующий вид:

$$\pi^*(a|s) = \arg\max Q^*(s, a)$$

Q-learning.
Данное выражение выполняется для всех состояний и всех действий.

$$Q^*(s, a) = r(s, a) + \gamma \mathbb{E}_{p(s'|s,a)} \max_{a'} Q^*(s', a'), \forall s, a$$

Перенесем правую часть налево и получим следующее равенство:

$$Q^*(s, a) - r(s, a) - \gamma \mathbb{E}_{p(s'|s,a)} \max_{a'} Q^*(s', a') = 0$$

Перейдем к задаче машинного обучение, то есть мы возведем данное выражение в квадрат и будем его минимизировать для уменьшения ошибки:

$$\frac{1}{|S||A|} \sum_{s,a} \left[Q^*(s, a) - r(s, a) - \gamma \mathbb{E}_{p(s'|s,a)} \max_{a'} Q^*(s', a')\right]^2 \to \min_{Q^*}$$

Данную функцию мы минимизируем, значит движемся в сторону антиградиента:

$$Q^*_{new}(s,a) = Q^*_{old}(s,a) - \alpha \nabla_{Q^*} \mathcal{L}(Q^*)$$

Распишем градиент и получим формулу Q-learning:

$$Q^*_{new}(s,a) = Q^*_{old}(s,a) - 2\alpha[Q^*_{old}(s,a) - r(s,a) - \gamma \max_{a'} Q^*_{old}(s',a') \times 1$$

Динамическое программирование

Под оцениванием стратегии понимается итеративное вычисление функций ценности для заданной стратегии. Под улучшением стратегии понимается вычисление улучшенной стратегии при заданной функции ценности для этой стратегии. Объединяя оба вычисления, мы получаем **алгоритм итерации по стратегиям и алгоритм итерации по ценности** – два наиболее популярных метода ДП.

Любой из них можно использовать для надежного вычисления оптимальных стратегий и функций ценности для конечных МППР, если вся информация о МППР известна. Классические методы ДП выполняют несколько проходов по множеству состояний, производя операцию полного обновления каждого состояния. Каждая такая операция обновляет ценность одного состояния на основе ценности всех возможных последующих состояний и их вероятностей. Полные обновления тесно связаны с уравнениями Беллмана: они лишь немногим больше, чем эти уравнения, преобразованные в операторы присваивания. Если в результате обновления ценность перестает изменяться, значит, алгоритм сошелся к ценностям, удовлетворяющим соответствующему уравнению Беллмана. Четырем основным функциям ценности (V_π, V^*, Q_π и Q^*) соответствуют четыре уравнения Беллмана и четыре полных обновления.

Интуитивное представление о работе обновлений в алгоритмах ДП дают их диаграммы предшествующих состояний. Лучше понять методы ДП, да и почти все методы обучения с подкреплением, можно, если рассматривать их как обобщенную итерацию по стратегиям (ОИС). Это общая идея двух взаимодействующих процессов, аппроксимирующих стратегию и функцию ценности. Один процесс принимает на входе стратегию и занимается ее оцениванием, изменяя функцию ценности, так чтобы она больше походила на истинную функцию ценности для этой стратегии. Другой процесс принимает на входе функцию ценности и производит некое улучшение стратегии, стремясь сделать ее лучше в предположении, что заданная функция ценности совпадает с функцией ценности стратегии. Хотя каждый процесс изменяет исходные данные для другого, вместе они находят общее решение: стратегию и функцию ценности, которые не изменяются обоими процессами и, следовательно, оптимальны.

В некоторых случаях можно доказать, что ОИС сходится, и прежде всего это справедливо для классических методов ДП. В других случаях сходимость не доказана, но все равно идея ОИС позволяет лучше понять методы. Методы ДП необязательно должны выполнять полные проходы по множеству состояний. Асинхронные методы ДП – это итеративные методы, обновляющие состояние на месте в произвольном порядке, быть может, стохастическом и с использованием неактуальной информации. Многие такого рода методы можно рассматривать как мелкоструктурные формы ОИС. Наконец, отметим одно **специальное свойство методов ДП**. Все они обновляют оценки ценности состояний на основе оценок ценности последующих состояний. То есть оценки обновляются на основе других оценок. Эта общая идея называется **бутстрэппингом**. Многие методы обучения с подкреплением выполняют бутстрэппинг, даже те, которым, в отличие от ДП, не требуется полная и точная модель окружающей среды.

Методы Монте-Карло

Методы Монте-Карло, обучаются функциям ценности и оптимальным стратегиям на опыте в форме выборочных эпизодов. Это дает им по меньшей мере три преимущества над методами ДП.

Во-первых, их можно использовать для обучения оптимальному поведению непосредственно на взаимодействии с окружающей средой, не имея модели динамики среды.

Во-вторых, они применимы совместно с имитационными или выборочными моделями. Есть на удивление много приложений, для которых легко смоделировать выборочные модели, даже если трудно построить явную модель вероятностей переходов, необходимую методам ДП.

В-третьих, методы Монте-Карло легко и эффективно применяются к небольшому подмножеству состояний. Область, представляющую особый интерес, можно точно обсчитать, не прибегая к точному оцениванию полного множества состояний.

Четвертое преимущество методов Монте-Карло, то, что они менее уязвимы к нарушениям марковского свойства. Это связано с тем, что они не обновляют оценки ценности на основе оценок ценности последующих состояний. Иными словами, в них отсутствует бутстрэппинг.

ОИС подразумевает взаимодействие процессов оценивания и улучшения стратегии. Методы Монте-Карло предлагают альтернативный процесс оценивания стратегии. Вместо того чтобы использовать модель для вычисления ценности каждого состояния, они просто усредняют много доходов, полученных при старте из этого состояния. Поскольку ценность состояния – это ожидаемый доход, такое среднее может оказаться хорошей аппроксимацией ценности. В методах управления нас особенно интересует аппроксимация функций ценности действий, потому что их можно использовать для улучшения стратегии, не имея модели динамики переходов в окружающей среде.

В методах Монте-Карло шаги оценивания и улучшения стратегии чередуются в соседних эпизодах и могут быть реализованы инкрементно на

поэпизодной основе. Обеспечение достаточного исследования – проблема для методов управления Монте-Карло. Недостаточно просто выбирать действия с максимальной текущей оценкой, поскольку тогда от альтернативных действий не будет получено никакого дохода, и мы можем никогда не узнать, что они на самом деле лучше.

Один из возможных подходов – игнорировать эту проблему, предполагая, что эпизоды начинаются в парах состояние–действие, случайно выбранных так, что они покрывают все возможности. Такие исследовательские старты иногда можно организовать в приложениях, в которых эпизоды смоделированы, но маловероятно, что так случится при обучении на реальном опыте. В методах с единой стратегией агент обязуется не пренебрегать исследованием и пытается найти оптимальную стратегию, которая продолжает исследовать. В методах с разделенной стратегией агент также занимается исследованием, но обучается детерминированной оптимальной стратегии, которая может быть никак не связана со стратегией, которой он следует.

Под предсказанием с разделенной стратегией понимают обучение функции ценности целевой стратегии на данных, сгенерированных другой поведенческой стратегией. Такие методы обучения основаны на той или иной форме выборки по значимости, т. е. на сопоставлении доходам весов в виде отношения вероятностей выбрать наблюдаемые действия при следовании обеим стратегиям; тем самым их математические ожидания преобразуются от поведенческой стратегии к целевой. В обыкновенной выборке по значимости используется простое среднее взвешенных доходов, а во взвешенной выборке по значимости – взвешенное среднее.

Обыкновенная выборка по значимости порождает несмещенные оценки, имеющие, однако, большую, быть может, даже бесконечную дисперсию, тогда как взвешенная выборка по значимости всегда дает оценки с конечной дисперсией и на практике более предпочтительна.

Несмотря на концептуальную простоту, применение методов Монте-Карло с разделенной стратегией для предсказания и управления еще не устоялось и остается темой активных исследований. Методы Монте-Карло, отличаются от методов ДП в двух главных отношениях.

Во-первых, они работают с выборочным опытом и потому могут использоваться для прямого обучения без модели.

Во-вторых, в них отсутствует бутстрэппинг, т. е. обновление оценок ценности на основе других оценок ценности.

Эти два различия не являются тесно связанными и могут быть разделены.

Обучение на основе временных различий

TD-обучение – это сочетание идей, заложенных в методах Монте-Карло и динамическом программировании (ДП).

Как и методы Монте-Карло, методы TD позволяют обучаться непосредственно на опыте, не требуя модели динамики окружающей среды. Как и ДП, методы TD обновляют оценки, основываясь в том числе на других обученных оценках, не дожидаясь конечного результата (бутстрэппинг).

Связь между TD, ДП и методами Монте-Карло – тема, пронизывающая все обучение с подкреплением. Эти идеи и методы проникают друг в друга и допускают комбинирование разными способами.

Для решения задачи управления (нахождения оптимальной стратегии) все методы – ДП, TD и Монте-Карло – пользуются тем или иным вариантом обобщенной итерации по стратегиям (ОИС). Различие между методами связано в основном с подходом к задаче предсказания.

Как обычно, мы разбили задачу на две части: предсказание и управление. TD-методы представляют альтернативу методам Монте-Карло для решения задачи предсказания. В обоих случаях в основе обобщения на задачу управления лежит идея обобщенной итерации по стратегиям (ОИС), которую мы вынесли из динамического программирования. Идея эта заключается в том, что приближенные стратегия и функции ценности должны взаимодействовать так, чтобы одновременно приближаться к своим оптимальным значениям. Один из двух процессов, составляющих ОИС, побуждает функцию ценности точно предсказывать доход для текущей стратегии; это задача **предсказания**. Второй процесс побуждает стратегию к **локальному улучшению** (например, ценой ε-жадности) относительно текущей функции ценности. Если первый процесс основан на опыте, то возникает осложнение, связанное с необходимостью поддерживать достаточный уровень исследования. Мы можем классифицировать TD-методы управления по тому, как они справляются с этой трудностью – применяя подход с единой или с разделенной стратегией.

Sarsa – метод с единой стратегией, а Q-обучение – с разделенной стратегией. Метод Expected Sarsa в том виде, в котором мы его описали, также является методом с разделенной стратегией.

Существует третий способ обобщения TD-методов на управление. Это методы типа **исполнитель–критик**.

Это общие методы обучения, позволяющие делать долгосрочные предсказания о динамических системах. Например, TD-методы могут применяться к предсказанию финансовых показателей, продолжительности жизни, исхода выборов, погоды, поведения животных, спроса на электроэнергию или действий покупателей. Лишь после того, как TD-методы начали анализировать как методы чистого предсказания, независимые от обучения с подкреплением, стали наконец понятны их теоретические свойства. Но, несмотря на это, существуют потенциальные применения методов TD-обучения, которые еще только ждут досконального исследования.

Reinforce (policy gradient)

Метод REINFORCE непосредственно вытекает из теоремы о градиенте стратегии. Добавление функции ценности состояний в качестве базы уменьшает дисперсию REINFORCE, не внося при этом смещения. Использование функции ценности состояний для бутстрэппинга вносит смещение, но зачастую это даже желательно по той же причине, по какой бутстрэппинговые TD-методы превосходят методы Монте-Карло (значительное уменьшение дисперсии).

Нейросеть-наша политика.

$\pi(a|s,\theta)$ - нейросеть
$\pi(a|s,\theta) = \mathbb{N}(a_j|\mu_j, \sigma_j^2(s,\theta))$ - при непрерывных a
$\pi(a|s,\theta) = \text{SoftMax}(\text{Output}_a(s\theta))$ - при дискретных a

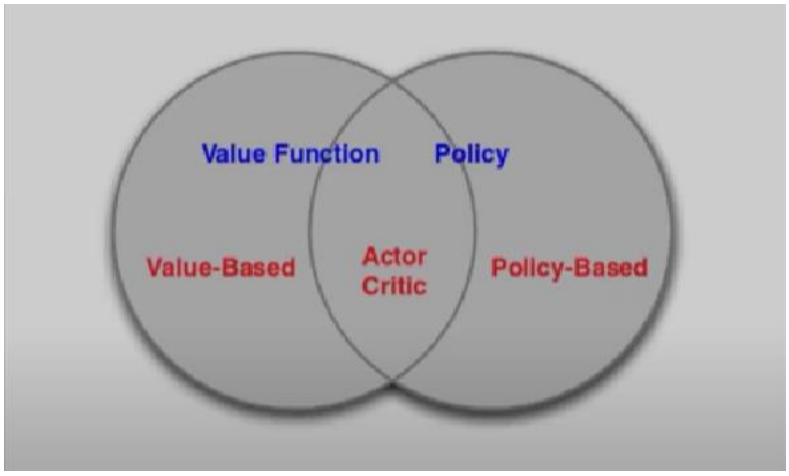
Политику мы будем оценивать с помощью следующей функции:

$$F(\theta) = \mathbb{E}_{p(s)}\mathbb{E}_{\pi(a|s,\theta)}Q^{\pi}(s,a) \to \max_{\theta}$$

$$\nabla_{\theta}F(\theta) \approx \mathbb{E}_{p(s)}\mathbb{E}_{\pi(a,s|\theta)}\nabla_{\theta}\log\pi(a|s,\theta)Q^{\pi}(s,a)$$

Данный алгоритм довольно долго сходится.

Подходы обучения с подкреплением



Actor Critic

Методы, которые обучаются аппроксимировать одновременно стратегию и функции ценности, часто называют методами исполнитель–критик, причем под «исполнителем» понимается обученная стратегия, а под «критиком» – обученная функция ценности, обычно функция ценности состояний.

В данном методе у нас будет две нейронных сети, а именно:

$\pi(a|s,\theta)$: Actor (нейронная сеть, например)
$Q(s,a|w)$: Critic (нейронная сеть, например) - оценивает Q для текущей π

Сеть Actor будет генерировать нам состояния, она параметризована параметрами θ и есть другая нейронная сеть, параметризованная параметрами w.

$$\nabla_\theta F(\theta) \approx \mathbb{E}_{p(s)} \mathbb{E}_{\pi(a,s|\theta)} \nabla_\theta \log \pi(a|s,\theta) Q^\pi(s,a|w)$$

Схема QAC:

- Инициализируем θ, w
- Итерируемся до сходимости:
  - Сэмплируем (s, a, r, s')
  - $a' \sim \pi(a|s', \theta)$
  - $y = r(s,a) + \gamma Q(s'a'|w_{old})$
  - $\theta_{new} = \theta_{old} + \alpha \nabla_\theta \log \pi(a|s,\theta_{old}) Q(s,a|w_{old})$
  - $w_{new} = w_{old} - 2\beta(Q(s,a|w_{old}) - y)\nabla_w Q(s,a|w_{old})$

Схема A2C:

В данной схеме у нас сохраняется только одна нейросеть:

$V(s|v)$ - нейронная сеть (для Q теперь нет сети)

- Инициализируем θ, v
- Итерируемся до сходимости:
  - Сэмплируем (s, a, r, s')
  - $a' \sim \pi(a|s', \theta)$
  - $y = r(s,a) + \gamma V(s'|v_{old})$
  - $\theta_{new} = \theta_{old} + \alpha \nabla_\theta \log \pi(a|s,\theta_{old}) \underbrace{(y - V(s|v_{old}))}_{\text{Advantage}}$
  - $v_{new} = v_{old} - 2\beta(V(s|v_{old}) - y)\nabla_v V(s|v_{old})$

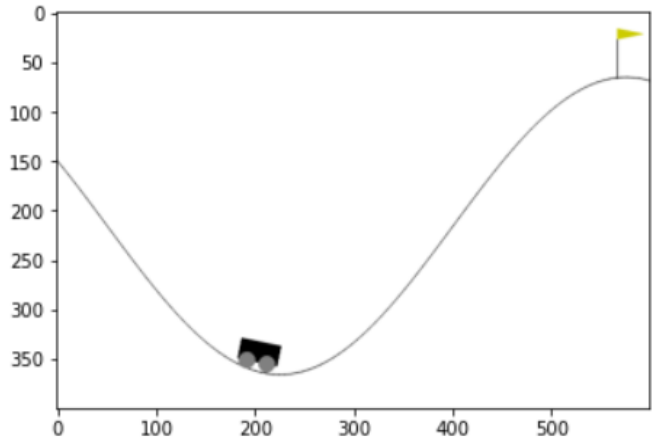
Практика

Будем пользоваться стандартными средами, реализованными в библиотеке OpenAI Gym.

Задача: необходимо доехать до флага на каретке.

```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import sys
sys.path.append('c:\\users\\amina\\anaconda3\\lib\\site-packages')
import gym
```

```
# создаем окружение
env = gym.make("MountainCar-v0")
env_screen = env.render(mode = 'rgb_array')
plt.imshow(env_screen)
env.close()
```




```
obs0 = env.reset()
print("изначальное состояние среды:", obs0)
# выполняем действие 2
new_obs, reward, is_done, _ = env.step(2)
print("новое состояние:", new_obs,
      "вознаграждение", reward)


def act(s):
    actions = {'left': 0, 'stop': 1, 'right': 2}
    # в зависимости от полученного состояния среды
    # выбираем действия так, чтобы тележка достигла флага
    action = actions['left'] if s[1] < 0 else actions["right"]
    return action

# создаем окружение, с ограничением на число шагов в 249
env = gym.wrappers.TimeLimit(
    gym.make("MountainCar-v0").unwrapped,
    max_episode_steps=250)
# проводим инициализацию и запоминаем начальное состояние
s = env.reset()
done = False
while not done:
    # выполняем действие, получаем s, r, done
    s, r, done, _ = env.step(act(s))
    # визуализируем окружение
    env.render()

env.close()
if s[0] > 0.47:
    print("Успех!")
else:
    raise NotImplementedError("""!""")
```

Q-обучение

Одним из наиболее популярных алгоритм обучения на основе временных различий является Q-обучение. Агент, который принимает решения на основе Q-функции, не требует модель для обучения и выбора действий, т.е. такой агент также свободен от модели (model-free), как и TD-агент. Уравнение Беллмана для значения Q-функции в равновесии записывается как:

$$Q(s,a) = r(s) + \gamma \sum_{s'} T(s,a,s') \max_{a'} Q(a',s')$$

Уравнение для итерационного обновления значений Q-функции выглядит следующим образом:

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r(s) + \gamma \max_{a'} Q(a',s') - Q(s,a)).$$

Оформим этот алгоритм в виде класса QLearningAgent:

```
import random, math

import numpy as np
from collections import defaultdict


class QLearningAgent():
    """
    Q-Learning агент

    Замечание: избегайте прямое использование
    self._q_values, для этого определены
    функции: get_q_value, set_q_value
    """

    def __init__(self, alpha, epsilon, discount,
                 get_legal_actions):
        self.get_legal_actions = get_legal_actions
        self._q_values = \
            defaultdict(lambda: defaultdict(lambda: 0))
        self.alpha = alpha
        self.epsilon = epsilon
        self.discount = discount

    def get_q_value(self, state, action):
        return self._q_values[state][action]

    def set_q_value(self, state, action, value):
        self._q_values[state][action] = value
```

Добавим нашему агенту возможность вычислять оценки V:

```
def get_value(self, state):
    """
      Возвращает значение функции полезности,
      рассчитанной по Q[state, action],
    """
    possible_actions = self.get_legal_actions(state)
    value = max([self.get_q_value(state, action) for action in
possible_actions])
    return value

QLearningAgent.get_value = get_value
```

Стратегия нашего агента будет заключаться в выборе лучшего действия, в соответствии с оценками Q:

```
def get_policy(self, state):
    """
      Выбирает лучшее действие, согласно стратегии.
    """
    possible_actions = self.get_legal_actions(state)

    # выбираем лучшее действие, согласно стратегии
    best_action = None
    for action in possible_actions:
        if best_action is None:
            best_action = action
        elif self.get_q_value(state, action) > self.get_q_value(state,
best_action):
            best_action = action

    return best_action

QLearningAgent.get_policy = get_policy
```

Для конкретной ситуации мы будем выбирать действие, используя ϵ-жадный подход:

```
def get_action(self, state):
    """
      Выбирает действие, предпринимаемое в данном
      состоянии, включая исследование (eps greedy)
      С вероятностью self.epsilon берем случайное
      действие, иначе действие согласно стратегии
      (self.get_policy)
    """
    possible_actions = self.get_legal_actions(state)

    # выбираем действие, используя eps-greedy подход
    if np.random.random() < self.epsilon:
        action = np.random.choice(possible_actions, 1)[0]
```

```
    else:
        action = self.get_policy(state)

    return action

QLearningAgent.get_action = get_action

def update(self, state, action, next_state, reward):
    """
      функция Q-обновления
    """
    # выполняем Q-обновление,
    # используем методы getQValue и setQValue
    t = self.alpha*(reward + self.discount*self.get_value(next_state) -
self.get_q_value(state, action))
    reference_qvalue = self.get_q_value(state, action) + t
    self.set_q_value(state, action, reference_qvalue)

QLearningAgent.update = update
```

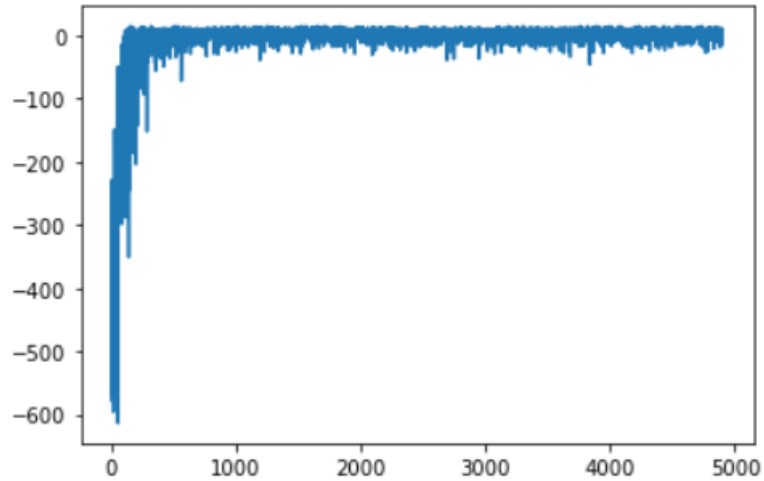
Тестируем нашего агента на задаче Taxi

Задача: машина, пассажир и пункт назначения появляются на карте в рандомных местах при каждом запуске программы. Необходимо доехать до пассажира (синий квадрат), забрать его и довести до пункта назначения (красный квадрат).

```
import gym
env = gym.make("Taxi-v3")

n_actions = env.action_space.n

def play_and_train(env, agent, t_max=10**4):
    """функция запускает полную игру,
    используя стратегию агента (agent.get_action(s))
    выполняет обновление агента (agent.update(...))
    и возвращает общее вознаграждение
    """
    total_reward = 0.0
    s = env.reset()

    for t in range(t_max):
        # выбираем действие
        a = agent.get_action(s)
        next_s, r, done, _ = env.step(a)

        # выполняем обновление стратегии
        agent.update(s, a, next_s, r)

        s = next_s
        total_reward +=r
        if done:
```

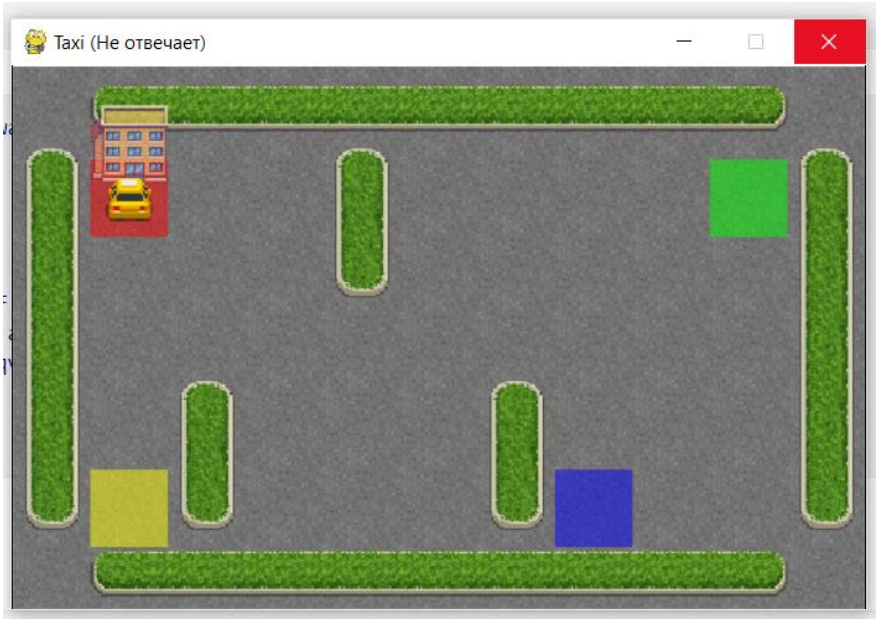
```
            break

    return total_reward

import matplotlib.pyplot as plt
%matplotlib inline
from IPython.display import clear_output

agent = QLearningAgent(alpha=0.5, epsilon=0.1,
                       discount=0.9,
                       get_legal_actions=lambda s: range(
                           n_actions))

assert 'get_policy' in dir(agent)
rewards = []
for i in range(5000):
    rewards.append(play_and_train(env, agent))

    if i % 100 == 0:
        clear_output(True)
        print('eps =', agent.epsilon,
              'mean reward =', np.mean(rewards[-10:]))
        print("alpha=", agent.alpha)
        plt.plot(rewards)
        plt.show()
```

```
eps = 0.1 mean reward = 0.0
alpha= 0.5
```




```
total_reward = 0.0
s = env.reset()

for t in range(1000):
    # выбираем действие
    a = agent.get_action(s)
    next_s, r, done, _ = env.step(a)

    s = next_s
    total_reward +=r
```

```
    if done:
        break

    env.render()
total_reward
```



Источники:

1) Ричард С. Саттон, Эндрю Дж. Барто “Обучение с подкреплением. Введение”.
   Ричард_С_Саттон,_Эндрю_Дж_Барто_Обучение_с_подкреплением_Введение.pdf
2) Обучение с подкреплением Q-learning, Policy Gradient (Reinforce), Actor-Critic Практика на gym.
   https://www.youtube.com/watch?v=kz6I-H5rsOU
3) https://plyus.pw/ru/ml2020/
4) https://blog.paperspace.com/getting-started-with-openai-gym/
5) https://sbercloud.ru/ru/warp/blog/machine-learning-about
6) Лапань М. Глубокое_обучение_с_подкреплением_AlphaGo_и_други.pdf
7) Supervised vs Unsupervised vs Reinforcement Learning.
   https://www.youtube.com/watch?v=1FZ0A1QCMWc