**DQN**

**Обучение с подкреплением (RL далее ОП) относится к разновидности метода машинного обучения, при котором агент получает отложенное вознаграждение на следующем временном шаге, чтобы оценить свое предыдущее действие. Он в основном использовался в играх (например, Atari, Mario), с производительностью на уровне или даже превосходящей людей. В последнее время, когда алгоритм развивается в комбинации с нейронными сетями, он способен решать более сложные задачи.**

**Окружение – это среда или объект, на который воздействует Агент (например игра), в то время как Агент представляет собой алгоритм ОП. Процесс начинается с того, что Окружение отправляет свое начальное состояние (state = s) Агенту, который затем, на основании своих значений, предпринимает действие (action = a ) в ответ на это состояние. После чего Окружение отправляет Агенту новое состояние (state’ = s’) и награду (reward = r) Агент обновит свои знания наградой, возвращенной окружением, за последнее действие и цикл повторится. Цикл повторяется до тех пор, пока Окружение не отправит признак конца эпизода.**

**Большинство алгоритмов ОП следуют этому шаблону. В следящем параграфе я кратко расскажу о некоторых терминах, используемых в ОП, чтобы облегчить наше обсуждение в следующем разделе.**

**Определения:**

**1.      Action (A, a):** все возможные команды, которые агент может передать в Окружение (среду)

**2.      State (S,s):** текущее состояние возвращаемое Окружением

**3.      Rewrd (R,r):** мгновенная награда возвращаемое Окружением, как оценка последнего действия

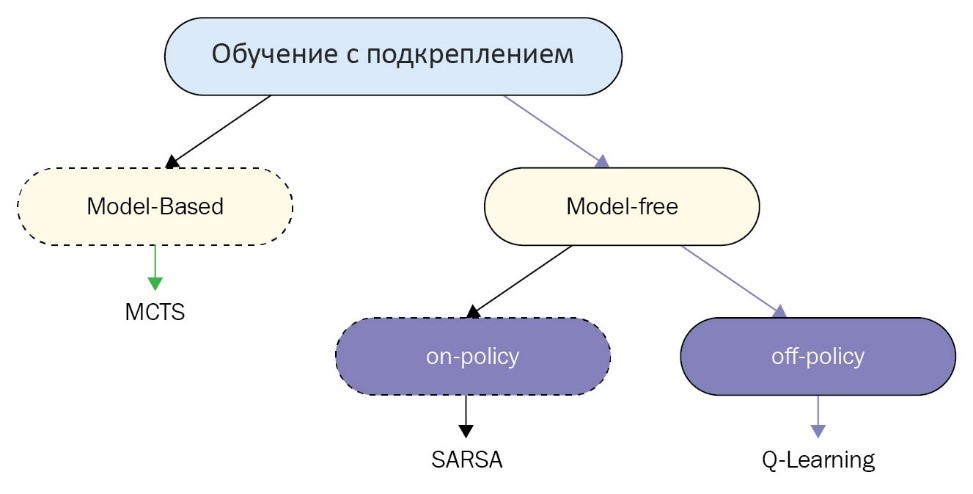
**4.    Policy (**π **):** Политика -  стратегия, которую использует Агент, для определения следующего действия (a’) на основе текущего состояния среды.

5.      **Value (V) или Estimate (E) :** ожидаемая итоговая (награда) со скидкой, в отличии от мгновенной награды **R,** является функцией политики Eπ(s) и определяется, как ожидаемая итоговая награда Политики в текущем состоянии s. (Встречается в литературе два варианта Value – значение, Estimate – оценка, что в контексте предпочтительней использовать E – оценка. Прим. переводчика)

6.      **Q-**value (Q): оценка Q аналогична оценки V, за исключением того, что она принимает дополнительный параметр a (текущее действие). Qπ(s, a)  является итоговой оценкой политики π от состояния s и действия a

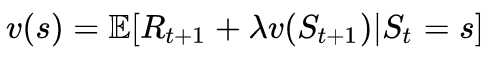
**2.1.    Q-learning**

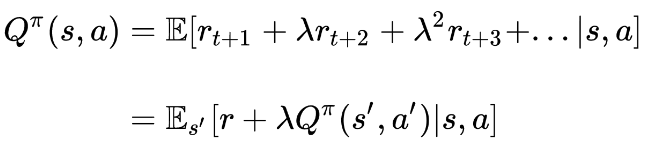
Q-learning это не связанный с политикой без модельный алгоритм ОП, основанный на хорошо известном уравнении Беллмана:

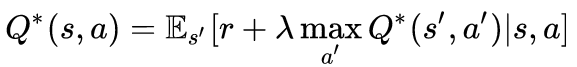


**2.1.    Q-learning**

Q-learning это не связанный с политикой без модельный алгоритм ОП, основанный на хорошо известном уравнении Беллмана:

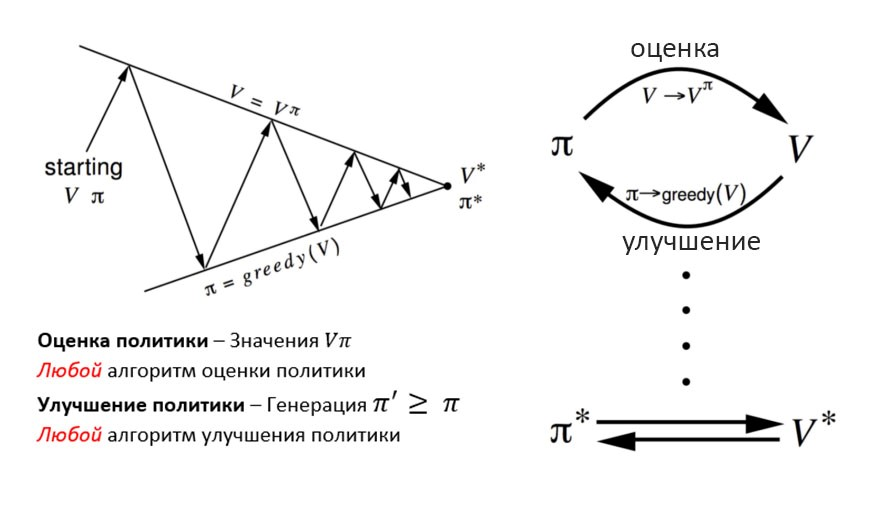
**Мы можем переписать это уравнение в форме Q-value:**

**Оптимальное значение Q, обозначенное как Q\*, может быть выражено как:**

**Цель состоит в том, чтобы максимизировать Q-значение. Прежде чем углубиться в метод оптимизации Q-value, я хотел бы обсудить два метода обновления значений, которые тесно связаны с Q-learning.**

**Итерация политики**

Итерация политики представляет собой цикл между оценкой политики и ее улучшением.

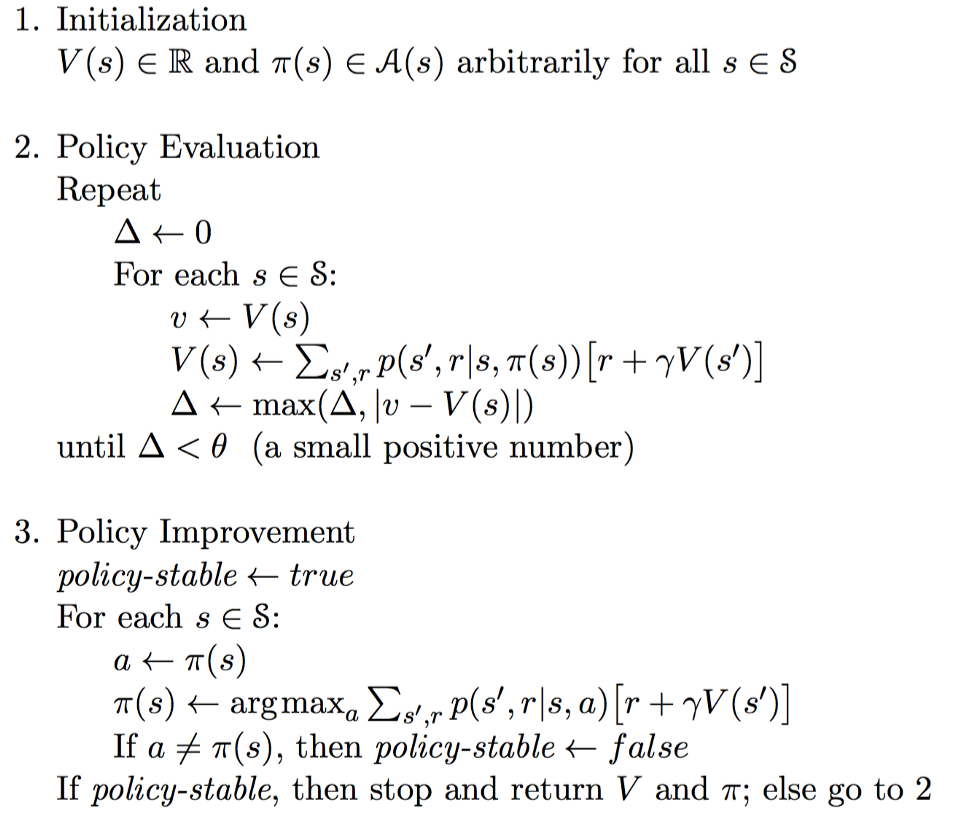
**Оценка политики оценивает значения функции V с помощью «жадной политики» полученной в результате последнего улучшения политики. С другой стороны, улучшение политики обновляет политику, генерирующую действия (action – a), что максимизирует значения V для каждого состояния (окружения). Уравнения обновления основаны на уравнении Беллмана. Итерации продолжаются до схождения.**

**Итерация Оценок (V)**

Итерация оценок содержит только один компонент, который обновляет функцию оценки значений V, на основе Оптимального уравнения Беллмана.

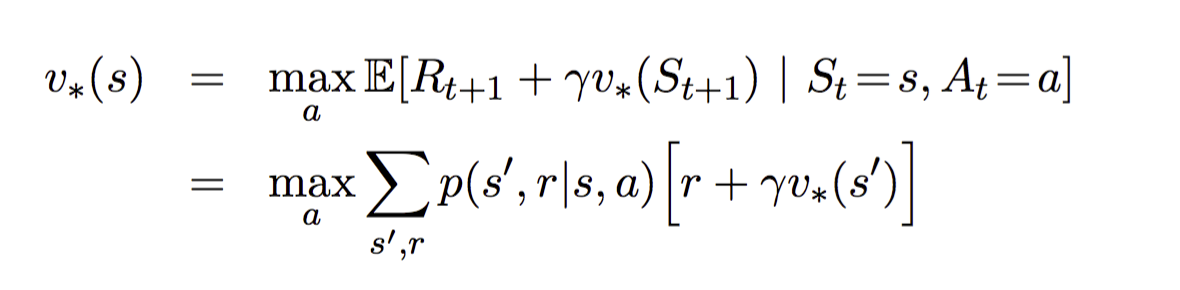
**Итерация Оценок (V)**

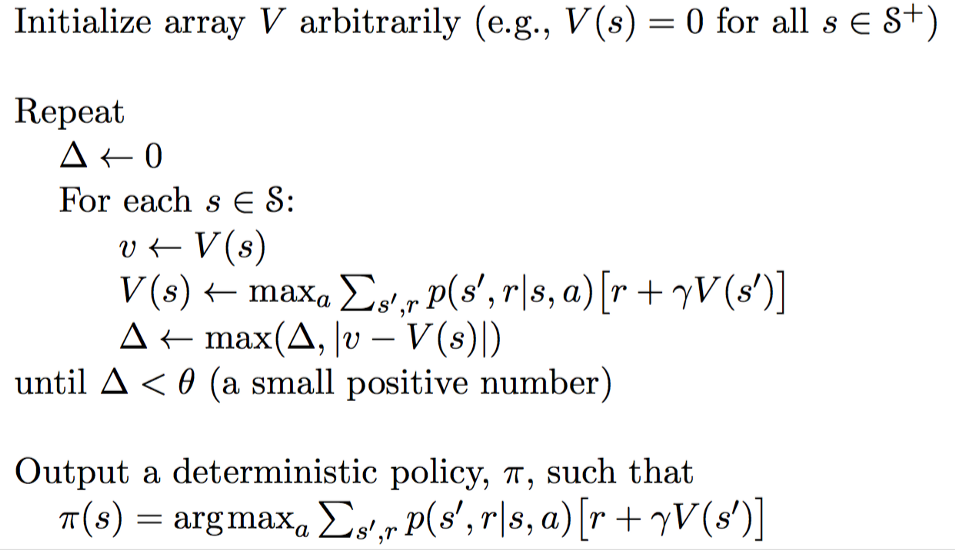
Итерация оценок содержит только один компонент, который обновляет функцию оценки значений V, на основе Оптимального уравнения Беллмана.



**Итерация Оценок (V)**

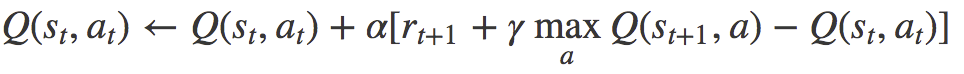
Итерация оценок содержит только один компонент, который обновляет функцию оценки значений V, на основе Оптимального уравнения Беллмана.



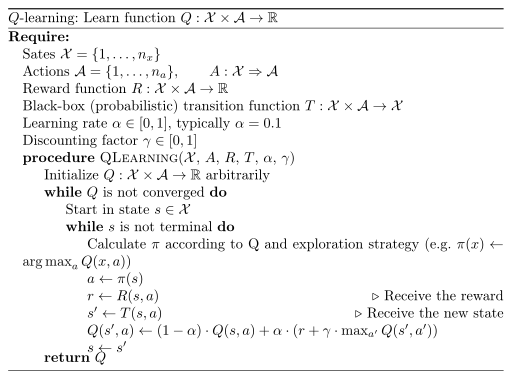


**После того, как итерация сходится, оптимальная политика напрямую выводится путем применения функции максимального аргумента для всех состояний.**

Обратите внимание, что эти два метода требуют знания вероятности перехода p, что указывает на то, что это алгоритм на основе модели. Однако, как я упоминал ранее, алгоритм, основанный на модели, страдает проблемой масштабируемости. Так как же Q-Learning решает эту проблему?



**Здесь** a (альфа) **скорость обучения (т.е. как быстро мы приближаемся к цели) Идея Q-learning во многом основана на итерациях оценок (v). Однако уравнение обновления заменяется приведенной выше формулой. В результате нам больше не нужно думать о вероятности перехода (p).**



**Обратите внимание, что следующее действие a’ выбирается для максимизации Q-значения следующих состояний вместо того, чтобы следовать текущей политике. В результате Q-learning относится к категории вне политики (off-Policy).**

**2.3.    Deep Q Network (DQN)**

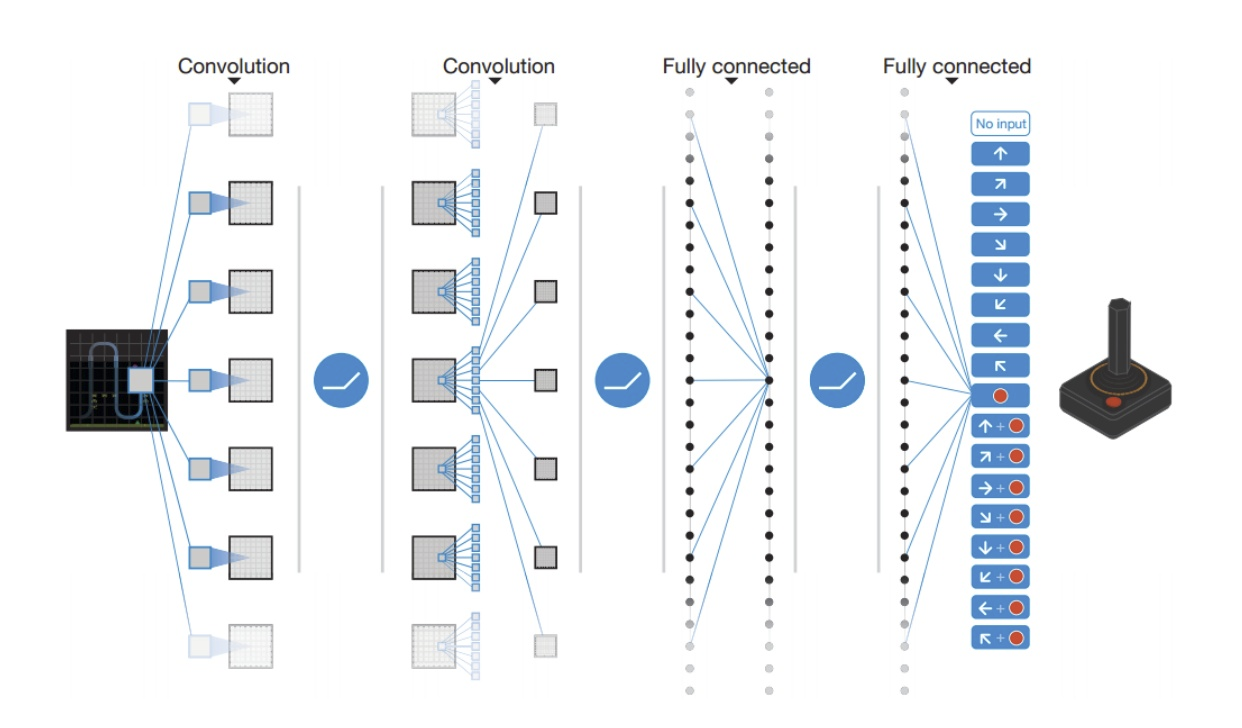
**Хотя Q-learning - очень мощный алгоритм, его главная слабость - отсутствие общности. Если вы рассматриваете Q- learning, как обновление чисел в двумерном массиве (пространство действий \* пространство состояний (action space \* state space)), оно фактически напоминает динамическое программирование. Это указывает на то, что для состояний, которые агент Q-Learning не видел раньше, он не знает, какое действие предпринять. Другими словами, агент Q-Learning не имеет возможности оценивать значение для невидимых состояний. Чтобы справиться с этой проблемой, DQN избавляется от двумерного массива, введя нейронную сеть.**

DQN использует нейронную сеть для оценки значений Q-функции. На вход сети подаются текущие кадры игрового поля, а выходом - соответствующее значение Q для каждого возможного действия.

**аваВ 2013 году DeepMind применил DQN к игре Atari, как показано на рисунке выше. Входными данными является необработанное изображение текущей игровой ситуации. Оно проходит через несколько сверхточных слоев, а затем через полно связный слой. Результатом является Q-значение для каждого действия, которое может предпринять агент.**

Вопрос сводится к следующему: **как мы обучаем сеть?**

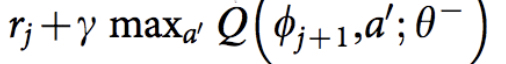
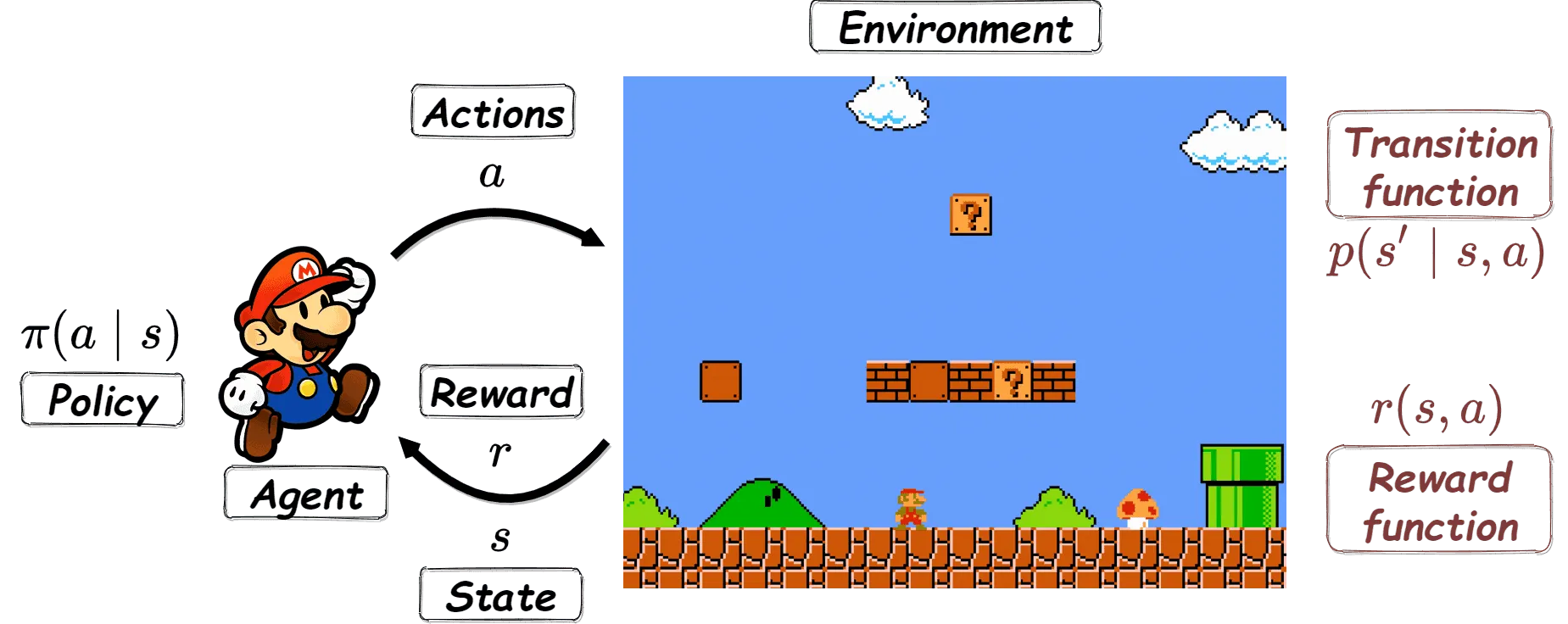
Ответ заключается в том, что мы обучаем сеть на основе уравнения обновления Q-learning. Напомним, что целевое значение Q для Q-learning:



**аваВ 2013 году DeepMind применил DQN к игре Atari, как показано на рисунке выше. Входными данными является необработанное изображение текущей игровой ситуации. Оно проходит через несколько сверхточных слоев, а затем через полно связный слой. Результатом является Q-значение для каждого действия, которое может предпринять агент.**

Вопрос сводится к следующему: **как мы обучаем сеть?**

Ответ заключается в том, что мы обучаем сеть на основе уравнения обновления Q-learning. Напомним, что целевое значение Q для Q-learning:



**φ эквивалентно состоянию s, в то время как θ обозначает параметры в нейронной сети, что не входит в область нашего обсуждения. Таким образом, функция потерь для сети определяется как квадрат ошибки между целевым значением Q и выходным значением Q из сети.**

**Еще два метода также важны для обучения DQN:**

1.      Воспроизведение опыта: поскольку обучающие батчи в типичной настройке ОП(RL) сильно коррелированы и менее эффективны для обработки данных, это приведет к более сложной конвергенции для сети. Одним из способов решения проблемы выборки батчей является воспроизведение опыта. По сути, батчи переходов сохраняются, а затем случайным образом выбираются из «пула переходов» для обновления знаний.

2.     Отдельная целевая сеть: целевая сеть Q имеет ту же структуру, что и сеть, которая оценивает значение. Каждый шаг C, в соответствии с приведенным выше псевдокодом, целевая сеть принимает значения основной сети. Таким образом, колебания становятся менее сильными, что приводит к более стабильным тренировкам.

**Описание окружения Lunar-Lander-V3**

## **Описание**

**Эта среда представляет собой классическую задачу оптимизации траектории ракеты. Согласно принципу максимума Понтрягина, оптимальным является включение двигателя на полную мощность или его выключение. Именно по этой причине в этой среде есть дискретные действия: включение или выключение двигателя.**

**Существует два варианта среды: дискретная или непрерывная. Посадочная площадка всегда находится в координатах (0,0). Координаты - это первые два числа в векторе состояния. Возможна посадка за пределами посадочной площадки. Запас топлива бесконечен, поэтому агент может научиться летать, а затем приземлиться с первой попытки.**

**Чтобы увидеть эвристическую посадку, запустите:**

**Поле действий**

**Доступно четыре отдельных действия:**

**• 0: ничего не делать**

**• 1: запустить левый механизм ориентации**

**• 2: запустить основной двигатель**

**• 3: запустить правый механизм ориентации**

**Пространство для наблюдения**

**Состояние представляет собой 8-мерный вектор: координаты посадочного модуля в координатах x и y, его линейные скорости в координатах x и y, его угол, его угловая скорость и два логических значения, которые представляют, соприкасается ли каждая опора с землей или нет.**

**Награды**

**После каждого шага выдается награда. Общая награда за эпизод - это сумма наград за все шаги в этом эпизоде.**

**За каждый шаг награда:**

**• увеличивается/уменьшается по мере приближения/удаления посадочного модуля к посадочной площадке.**

**• увеличивается/уменьшается, чем медленнее/быстрее движется посадочный модуль.**

**• уменьшается, чем больше наклонен посадочный модуль (угол не горизонтальный).**

**• увеличивается на 10 пунктов за каждую опору, соприкасающуюся с землей.**

**• уменьшается на 0,03 балла за каждый кадр, который запускает боковой движок.**

**• уменьшается на 0,3 балла за каждый кадр, на котором запускается основной двигатель.**

**За эпизод, который потерпел аварию или благополучно приземлился, вы получаете дополнительную награду в размере -100 или +100 баллов соответственно.**

**Эпизод считается решенным, если он набрал не менее 200 баллов.**

**Начальное состояние**

**Посадочный модуль стартует в верхней части экрана, к центру масс которого приложена произвольная начальная сила.**

**Окончание эпизода**

**Эпизод заканчивается, если:**

**1. посадочный модуль падает (корпус модуля соприкасается с Луной).;**

**2. посадочный модуль выходит за пределы видового экрана (координата x больше 1).;**

**3. посадочный модуль не активен. Согласно документации Box2D, тело, которое не активировано, - это тело, которое не движется и не сталкивается ни с каким другим телом:**

**История версий**

**• v3:**

**• Сбрасывайте смещение ветра и турбулентности (C) всякий раз, когда среда сбрасывается, чтобы обеспечить статистическую независимость между последовательными эпизодами (связанная проблема с GitHub).**

**• Исправьте недетерминированное поведение из-за неполного уничтожения мира (связанная проблема с GitHub).**

**• Изменено пространство наблюдения для координат x, y с**

**на , скоростей с на и углов с на на**

**• (связанная с этим проблема на GitHub).**

**• версия 2: подсчет затраченной энергии, а в версии 0.24 добавлена турбулентность с параметрами "энергия ветра" и "turbulence\_power".**

**• версия 1: в вектор состояния добавлен контакт ног с землей; контакт с землей дает +10 призовых очков и -10, если затем контакт теряется; награда увеличена до 200; более сильный первоначальный случайный толчок.**

**• версия 0: Начальная версия**

**Записи**

**Есть несколько неожиданных ошибок, связанных с реализацией среды.**

**1. Положение боковых подруливающих устройств на корпусе посадочного модуля изменяется в зависимости от ориентации посадочного модуля. Это, в свою очередь, приводит к тому, что к посадочному модулю прикладывается крутящий момент, зависящий от ориентации.**

**2. Единицы измерения состояния не согласованы. Т.е.**

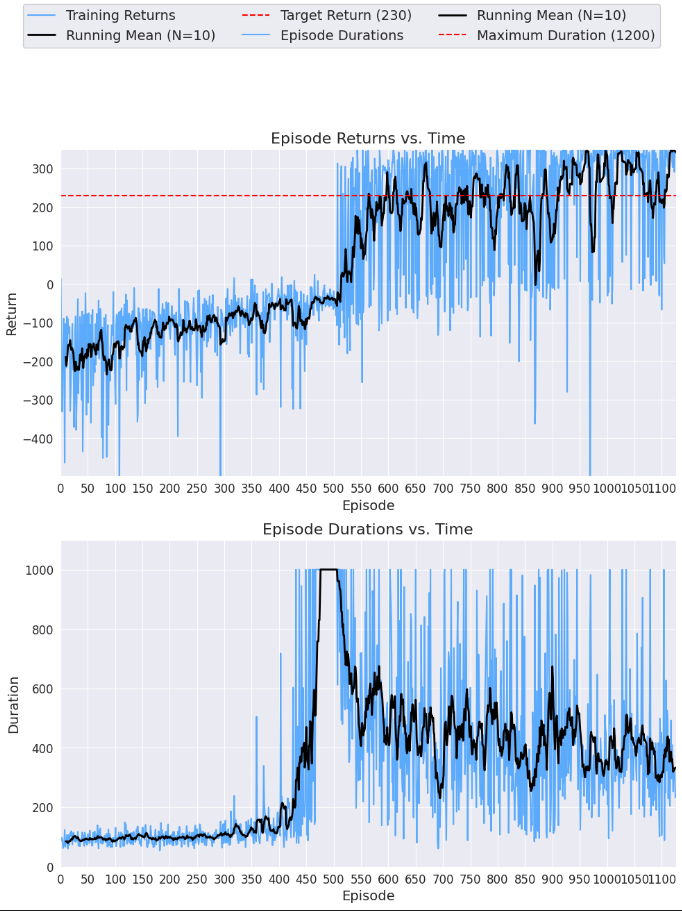
**• Угловая скорость указана в единицах измерения, равных 0,4 радиан в секунду. Для преобразования в радианы в секунду это значение необходимо умножить на коэффициент 2,5.**

**Для значений по умолчанию VIEWPORT\_W, VIEWPORT\_H, SCALE и FPS масштабные коэффициенты равны: "x": 10, "y": 6,666, "vx": 5, "vy": 7,5, "угол": 1, "угловая скорость": 2,5**

**После внесения корректировки единицы измерения состояния будут следующими: "x": (единицы измерения), "y": (единицы измерения), "vx": (единицы измерения в секунду), "vy": (единицы измерения в секунду), "угол": (радианы), ‘угловая скорость": (в радианах в секунду)**

**custom\_reward()**

**Эта функция** custom\_reward**формирует поведение агента в задаче посадки, используя комбинацию штрафов и бонусов. Она непрерывно оценивает состояние агента, измеряя его удаленность от цели (горизонтального центра), угол наклона и скорость. За отклонения от идеальных параметров (нулевое расстояние, нулевой угол, низкая скорость) начисляются штрафы, которые подталкивают агента к более аккуратной посадке. В случае успешного завершения эпизода (то есть, посадки без крушения), агент получает бонус, но только если к моменту посадки он уже сумел заработать положительную награду, что исключает возможность "обучения" катастрофически быстрым, но неконтролируемым приземлениям. Результирующая награда является суммой всех штрафов и, при выполнении условий, бонуса за успешную посадку. Величины штрафов и бонуса настраиваются через параметры** reward\_params**, позволяя гибко корректировать приоритеты в процессе обучения.**



**Умеренное поощрение за выживание и точность (**reward\_survive\_and\_be\_accurate**)**

**Компоненты Награды:**

**--------------------**

**- Штраф за Расстояние: Штрафует за горизонтальное расстояние от центра (x=0).**

**Стимулирует агента располагаться рядом с местом посадки.**

**- Штраф за Угол: Штрафует за большие углы наклона, стимулируя агента оставаться**

**в вертикальном положении.**

**- Штраф за Скорость: Штрафует за высокие скорости, стимулируя мягкую посадку.**

**- Бонус/Штраф за Посадку:**

**- Бонус за успешную посадку (завершенный эпизод) ЕСЛИ агент накопил**

**положительную награду.**

**- Штраф за неудачную посадку (завершенный эпизод) ЕСЛИ агент НЕ накопил**

**положительную награду.**

**- Штраф за Прерывание: Штрафует за преждевременное завершение эпизода из-за**

**прерывания (например, истечение времени).**

**- Награда за Выживание: Предоставляет небольшую положительную награду за каждый**

**шаг, когда агент остается в живых и в полете.**

**Гиперпараметры (Определены в self.reward\_params):**

**------------------------------------------------**

**- distance\_penalty (float): Весовой коэффициент для штрафа за расстояние.**

**- angle\_penalty (float): Весовой коэффициент для штрафа за угол.**

**- velocity\_penalty (float): Весовой коэффициент для штрафа за скорость.**

**- landing\_bonus (float): Величина бонуса/штрафа за посадку.**

**Стратегия:**

**----------**

**Эта функция балансирует между стимулированием точной посадки и предотвращением**

**преждевременного завершения. Она использует стратегию формирующего награждения,**

**основанную на штрафах, чтобы направить агента к желаемому поведению при посадке.**

**Бонус/штраф за посадку зависит от накопленной агентом награды, а штраф за**

**прерывание стимулирует агента избегать преждевременного завершения из-за**

**истечения времени. Небольшая награда за выживание предоставляет постоянный**

**стимул оставаться в живых и продолжать обучение.**

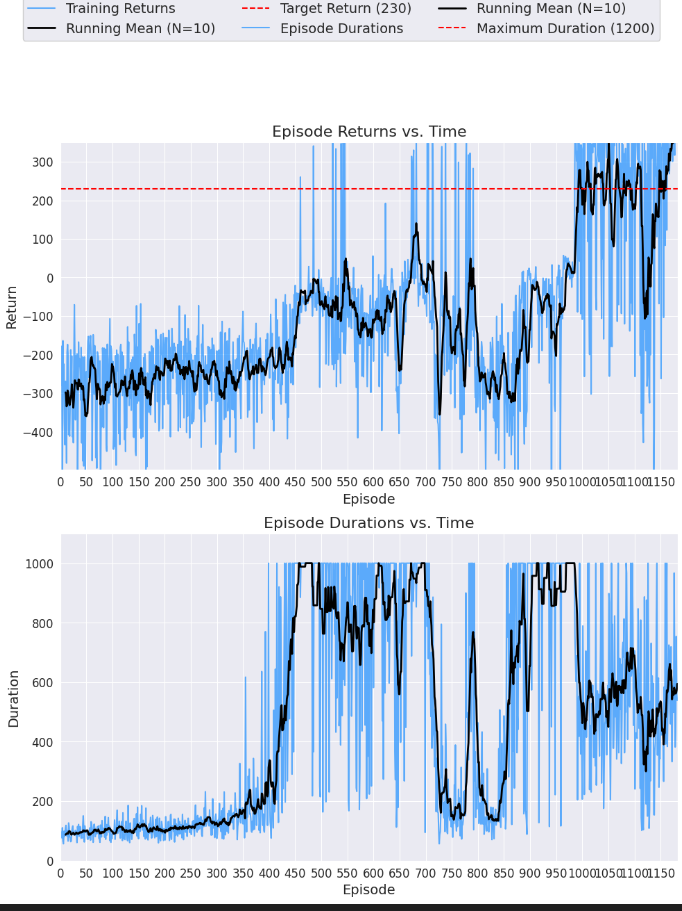
**Ключевые Отличия от Предыдущей Функции Награды (на русском):**

**1. Штраф за Неудачную Посадку: Предыдущая функция давала только \*бонус\* за посадку и \*не штрафовала\* за крушения явно (кроме кумулятивных штрафов за плохое положение/угол/скорость). Эта версия \*штрафует\* завершенный эпизод (например, крушение) \*отрицательным\* landing\_bonus, если накопленная награда не положительная. Это делает агента более нетерпимым к крушениям.**

**2. Штраф за Прерывание: Эта версия добавляет штраф за \*прерванный\* эпизод (например, истечение времени). Это препятствует агенту просто ждать до истечения времени и стимулирует его активно пытаться приземлиться.**

**3. Награда за Выживание: Эта версия добавляет небольшую постоянную награду (+0.1) за каждый шаг, который агент выживает (т.е., эпизод не terminated и не truncated). Это обеспечивает небольшой постоянный стимул оставаться в полете и исследовать, особенно на ранних этапах обучения. Это помогает предотвратить быстрое нахождение плохого решения (например, немедленное крушение) и прекращение исследования.**

**В заключение, эта функция награды более сложная. Она пытается непосредственно устранить потенциальные проблемы в предыдущей версии путем:**



**Агрессивное поощрение за скорость и точность (риск) (**reward\_fast\_and\_accurate

**)**

**## Описание функции награды: Комплексная стратегия для точной и стабильной посадки**

**Эта функция награды разработана для обучения агента выполнять быструю, точную и стабильную посадку. Она использует многокомпонентный подход, включающий штрафы, бонусы и стимулы, чтобы сформировать желаемое поведение. В отличие от предыдущих версий, эта функция явно наказывает за неудачи и поощряет активное стремление к посадке, а также выживание в процессе обучения.**

**Компоненты Награды:**

**Штраф за Расстояние: Штрафует агента за горизонтальное удаление от целевой точки посадки (x=0). Этот штраф стимулирует агента занимать позицию ближе к месту посадки, готовясь к безопасному приземлению.**

**Штраф за Угол: Штрафует агента за большие углы наклона. Это мотивирует агента сохранять вертикальное положение, что важно для контроля и стабильности, особенно при подходе к посадке.**

**Штраф за Скорость: Штрафует агента за высокую скорость. Стимулирует более мягкий и контролируемый спуск, уменьшая риск крушения при контакте с землей.**

**Бонус/Штраф за Посадку:**

**\* Бонус за успешную посадку: Предоставляется агенту, успешно завершившему эпизод (например, благополучно приземлился) и при этом накопившему положительную награду за предыдущие шаги. Этот бонус усиливает желаемое поведение и поощряет к повторению успешных стратегий.**

**\* Штраф за неудачную посадку: Назначается агенту при завершении эпизода с отрицательным общим вознаграждением (например, в результате крушения). Этот штраф делает агента более чувствительным к ошибкам и стимулирует избегать ситуаций, приводящих к крушениям. Ключевое отличие от предыдущих версий: явное наказание за неудачи.**

**Штраф за Прерывание: Штрафует агента за эпизоды, которые заканчиваются преждевременно из-за прерывания (например, истечение времени). Это препятствует пассивному поведению и стимулирует агента активно пытаться приземлиться до истечения времени. Ключевое отличие от предыдущих версий: стимулирует активное стремление к посадке.**

**Награда за Выживание: Предоставляет небольшую положительную награду за каждый шаг, когда агент остается в живых и находится в полете (эпизод не завершен и не прерван). Этот небольшой, но постоянный стимул поощряет агента оставаться в воздухе и исследовать окружающую среду, особенно на ранних этапах обучения, предотвращая преждевременное принятие плохих стратегий (например, быстрое крушение). Ключевое отличие от предыдущих версий: обеспечивает постоянный стимул для исследования.**

**Гиперпараметры (Определены в self.reward\_params):**

**distance\_penalty (float): Весовой коэффициент, определяющий силу штрафа за расстояние.**

**angle\_penalty (float): Весовой коэффициент, определяющий силу штрафа за угол.**

**velocity\_penalty (float): Весовой коэффициент, определяющий силу штрафа за скорость.**

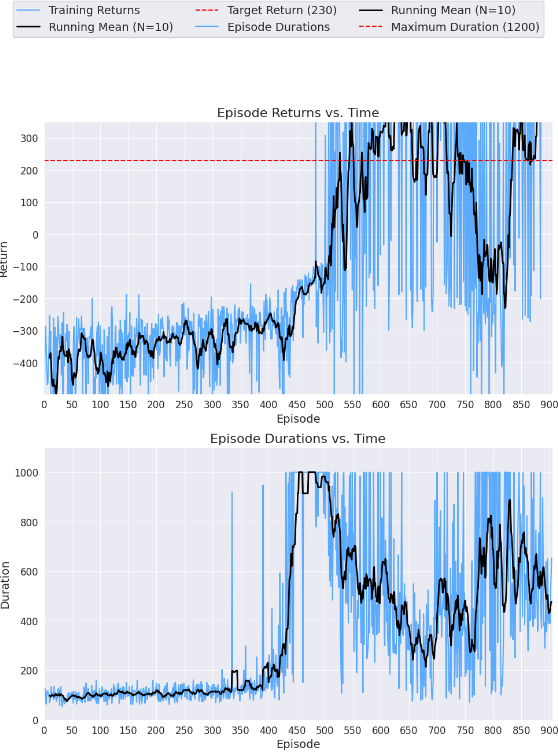
**landing\_bonus (float): Величина бонуса, начисляемого за успешную посадку (и абсолютная величина штрафа за неудачную посадку).**

**Стратегия:**

**Эта функция награды использует стратегию формирующего награждения, чтобы направить агента к желаемому поведению при посадке. Штрафы за расстояние, угол и скорость формируют траекторию полета и помогают агенту подготовиться к мягкой посадке. Бонус за успешную посадку закрепляет удачные попытки, а штраф за неудачную посадку мотивирует агента избегать крушений. Штраф за прерывание стимулирует активные попытки приземлиться, а награда за выживание обеспечивает постоянный стимул для обучения и исследования. Зависимость бонуса/штрафа за посадку от накопленной награды позволяет дифференцировать успешные и неудачные завершения эпизодов.**

**Ключевые Отличия от Предыдущей Функции Награды:**

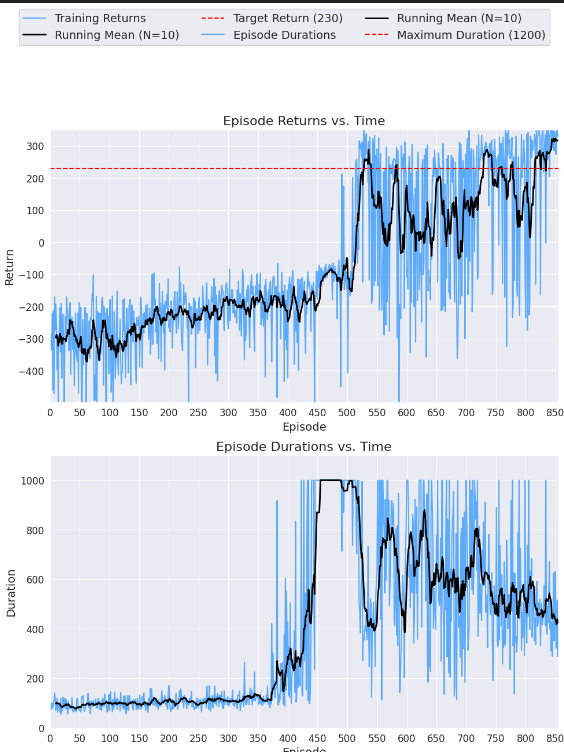
**1. Штраф за Неудачную Посадку: Предыдущая функция давала только \*бонус\* за успешную посадку и \*не штрафовала\* явно за крушения (кроме кумулятивного эффекта штрафов за плохое положение, угол и скорость). Эта версия \*штрафует\* завершенный эпизод (например, крушение) \*отрицательным\* landing\_bonus, если накопленная награда не положительная. Это делает агента более нетерпимым к крушениям и более склонным к поиску безопасных посадок.  
  
2. Штраф за Прерывание: Эта версия добавляет штраф за \*прерванный\* эпизод (например, из-за истечения времени). Это препятствует агенту просто ждать до истечения времени и стимулирует его активно пытаться приземлиться, даже если времени осталось мало.  
  
3. Награда за Выживание: Эта версия добавляет небольшую постоянную награду за каждый шаг, который агент выживает (т.е., эпизод не** terminated**и не** truncated**). Это обеспечивает небольшой постоянный стимул оставаться в полете и исследовать, особенно на ранних этапах обучения. Это помогает предотвратить быстрое нахождение плохого решения (например, немедленное крушение) и прекращение исследования.  
  
В заключение, эта функция награды представляет собой более сложную и сбалансированную систему, направленную на обучение агента выполнению точной, стабильной и быстрой посадки. Она явно наказывает за ошибки, поощряет активное обучение и исследование, а также формирует желаемое поведение с помощью многокомпонентного подхода. Она непосредственно устраняет потенциальные проблемы, замеченные в предыдущих версиях, делая обучение более эффективным и стабильным.**



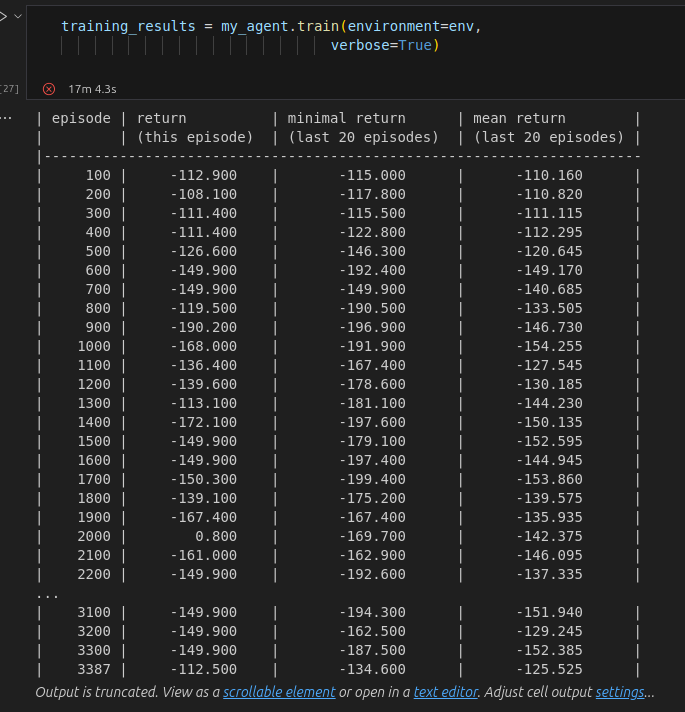
**Награда, зависящая от высоты (y) (**reward\_based\_on\_height

**)**

**## Описание функции награды** reward\_based\_on\_height**(награда, основанная на высоте)  
  
Эта функция награды направлена на обучение агента мягкой и точной посадке с учетом высоты над землей. Она использует сочетание штрафов за отклонение от целевой позиции и угла, а также штрафов за скорость, которые усиливаются по мере приближения к земле, стимулируя замедление перед посадкой.  
  
Входные параметры:**self**: Ссылка на объект класса, содержащий параметры награды (**self.reward\_params**).**observation**: Наблюдение за окружающей средой, содержащее:  
\*** x**: Координата x позиции агента.  
\*** y**: Координата y позиции агента (высота).  
\*** vx**: Скорость агента по оси x.  
\*** vy**: Скорость агента по оси y.  
\*** angle**: Угол ориентации агента.  
\*** \_**,** \_**,** \_**: Заполнители для неиспользуемых параметров.**reward**: Текущая награда, к которой добавляются новые бонусы и штрафы.**terminated**: Булево значение, указывающее, завершился ли эпизод (успешно или неудачно).**truncated**: Булево значение, указывающее, был ли эпизод прерван по какой-либо причине (например, истечение времени).**info**: Дополнительная информация об эпизоде (не используется напрямую).  
  
Логика работы:  
  
1. Базовые штрафы:  
\*** distance\_reward**: Линейный штраф за отклонение по горизонтали от целевой точки посадки (x=0). Рассчитывается как** -abs(x) \* self.reward\_params['distance\_penalty']**.  
\*** angle\_penalty**: Линейный штраф за отклонение от вертикального положения (угол = 0). Рассчитывается как** -abs(angle) \* self.reward\_params['angle\_penalty']**.  
\*** velocity\_penalty**: Линейный штраф за скорость, учитывающий абсолютные значения скоростей по обеим осям. Рассчитывается как** -(abs(vx) + abs(vy)) \* self.reward\_params['velocity\_penalty']**.  
\* Эти штрафы суммируются и добавляются к текущей награде:** reward += distance\_reward + angle\_penalty + velocity\_penalty**.  
  
2. Штраф за скорость в зависимости от высоты:  
\*** if y < 50:**: Если высота агента (**y**) меньше 50 (метров, единиц измерения), применяется дополнительный штраф за скорость.  
\*** reward += -(abs(vx) + abs(vy)) \* self.reward\_params['velocity\_penalty'] \* 2**: Штраф за скорость удваивается, если агент находится достаточно близко к земле. Это мотивирует агента замедлиться перед посадкой. Ключевая особенность: контекстная награда, зависящая от высоты.  
  
3. Бонусы и штрафы за завершение эпизода:  
\*** terminated and reward > 0**: Если эпизод завершился успешно (и накоплена положительная награда), добавляется бонус за посадку:** reward += self.reward\_params['landing\_bonus']**.  
\*** elif terminated**: Если эпизод завершился неудачно (и/или накоплена отрицательная награда), применяется штраф за неудачу:** reward -= self.reward\_params['landing\_bonus']**.  
\*** elif truncated**: Если эпизод был прерван, назначается небольшой штраф:** reward -= self.reward\_params['landing\_bonus'] / 2**.  
  
Возвращаемое значение:**reward**: Обновленное значение награды после применения штрафов и бонусов.  
  
Гиперпараметры (Определены в self.reward\_params):**distance\_penalty (float)**: Весовой коэффициент штрафа за расстояние.**angle\_penalty (float)**: Весовой коэффициент штрафа за угол.**velocity\_penalty (float)**: Весовой коэффициент штрафа за скорость.**landing\_bonus (float)**: Величина бонуса за успешную посадку (и штрафа за неудачную посадку).  
  
Стратегия:  
  
Функция** reward\_based\_on\_height**использует стратегию формирующего награждения, которая учитывает высоту агента. Линейные штрафы за расстояние и угол направляют агента к целевой точке и ориентации. Увеличение штрафа за скорость при приближении к земле способствует мягкой посадке. Бонусы и штрафы за завершение эпизода усиливают желаемое поведение и наказывают за неудачи. Использование линейных штрафов вместо квадратичных может привести к более стабильному и предсказуемому обучению, особенно на начальных этапах.  
  
Ключевые особенности и отличия от предыдущих версий:  
  
Зависимость от высоты:  Основное отличие заключается в том, что функция \*усиливает\* штраф за скорость, когда агент находится на определенной высоте. Это побуждает агента замедлиться перед посадкой, что критически важно для безопасного и успешного приземления.  
Линейные штрафы:  Функция использует \*линейные\* штрафы за расстояние, угол и скорость, в отличие от квадратичных штрафов в некоторых предыдущих версиях. Линейные штрафы могут быть более стабильными и менее чувствительными к выбросам в данных, что может способствовать более плавному обучению.  
Простая структура: Функция имеет более простую и прямолинейную структуру по сравнению с более сложными стратегиями, такими как явное наказание за крушения или награда за выживание. Это может сделать ее легче для понимания и отладки.  
  
В заключение, эта функция награды направлена на обучение агента безопасной и точной посадке, акцентируя внимание на замедлении скорости при приближении к земле. Она представляет собой более простой и, возможно, более стабильный подход к формированию желаемого поведения по сравнению с более сложными стратегиями награждения.**



**Разреженная награда (**reward\_sparse**)**



**## Описание функции награды** reward\_sparse**(редкая награда)  
  
Эта функция награды реализует стратегию редкого награждения (sparse reward), при которой агент получает значительную награду только при выполнении целевой задачи (успешная посадка) или наказание при неудаче. В отличие от функций с формирующим награждением, здесь отсутствует постепенное направление агента к цели через промежуточные награды. Основная цель - заставить агента исследовать среду и самостоятельно обнаружить способ успешной посадки, полагаясь в основном на случайные действия.  
  
Входные параметры:**self**: Ссылка на объект класса, содержащий параметр награды** landing\_bonus**(**self.reward\_params**).**observation**: Наблюдение за состоянием окружающей среды (не используется непосредственно в вычислениях награды).**reward**: Текущая награда (перезаписывается в этой функции).**terminated**: Булево значение, указывающее, завершился ли эпизод (успешно или неудачно).**truncated**: Булево значение, указывающее, был ли эпизод прерван (например, из-за истечения времени).**info**: Дополнительная информация об эпизоде (не используется непосредственно).  
  
Логика работы:  
  
1. Проверка условий завершения эпизода:  
\*** if terminated and reward > 0:**: Если эпизод завершился успешно (предположительно, с положительной накопленной наградой), назначается большой положительный бонус:** reward = self.reward\_params['landing\_bonus']**.  
\*** elif terminated:**: Если эпизод завершился неудачно (например, крушение), назначается большой отрицательный штраф:** reward = -self.reward\_params['landing\_bonus']**.  
\*** elif truncated:**: Если эпизод был прерван (например, из-за истечения времени), назначается небольшой отрицательный штраф:** reward = -self.reward\_params['landing\_bonus'] / 2**.  
\*** else:**: Если эпизод продолжается, назначается небольшой отрицательный штраф за каждый шаг:** reward = -0.1**.  
  
2. Перезапись награды:  Важно отметить, что функция \*перезаписывает\* входное значение** reward**, а не добавляет к нему какие-либо бонусы или штрафы, как это делалось в предыдущих функциях награды.  
  
Возвращаемое значение:**reward**: Обновленное (перезаписанное) значение награды.  
  
Гиперпараметры (Определены в self.reward\_params):**landing\_bonus (float)**: Величина бонуса за успешную посадку (и штрафа за неудачную).  
  
Стратегия:  
  
Функция** reward\_sparse**реализует стратегию редкого награждения, которая характеризуется:  
  
Минимальным формирующим награждением: Агенту не предоставляется никакой информации о том, как продвигаться к цели, кроме как через итоговый результат (успешная посадка или неудача).  
Большими, но редкими наградами: Бонус за успешную посадку и штраф за падение имеют значительную величину, чтобы привлечь внимание агента к этим исходам.  
Штрафом за бездействие: Небольшой штраф за каждый шаг, когда эпизод не завершается, мотивирует агента к активным действиям.  
  
Такая стратегия может быть полезна в задачах, где трудно определить промежуточные цели или где экспертные знания ограничены. Однако, она может быть очень сложной для обучения, так как агенту необходимо случайно наткнуться на решение (успешную посадку), прежде чем начать улучшать свою стратегию.  
  
Ключевые особенности и отличия от предыдущих версий:  
  
Отсутствие формирующего награждения:  Это самое важное отличие. Нет штрафов за расстояние, угол или скорость. Агент получает награду только в конце эпизода.  
Перезапись, а не добавление: Функция \*перезаписывает\* значение награды, а не добавляет к нему штрафы или бонусы.  
Сильный акцент на исследовании: Такая стратегия подразумевает, что агент должен в основном полагаться на случайное исследование, чтобы найти успешные стратегии.  
Потенциальная сложность обучения: Обучение с редким награждением может быть гораздо сложнее и требовать больше вычислительных ресурсов и времени, чем с формирующим награждением.  
  
В заключение, эта функция награды использует стратегию редкого награждения, что может быть эффективным подходом в задачах, где трудно или нежелательно предоставлять агенту подробную информацию о том, как достичь цели. Однако, этот подход может быть сложным для обучения и требует от агента значительного количества исследований. Она подходит для ситуаций, когда желаемый результат четко определен, но путь к нему неизвестен или не может быть легко сформулирован в виде промежуточных целей.**

**Награда, зависящая от действия (Action-Dependent Reward) (**reward\_action\_dependent**)**

*****#***# Описание функции награды** reward\_action\_dependent**(награда, зависящая от действия)  
  
Эта функция награды расширяет базовый подход к награждению, учитывая \*действие\*, предпринятое агентом, при расчете награды. В отличие от функций, которые основываются только на состоянии окружающей среды (**observation**), эта функция пытается формировать поведение агента путем наказания за определенные действия, которые считаются нежелательными (в данном примере, слишком большая тяга).  
  
Входные параметры:**self**: Ссылка на объект класса, содержащий параметры награды (**self.reward\_params**).**observation**: Наблюдение за окружающей средой, содержащее:  
\*** x**: Координата x позиции агента.  
\*** y**: Координата y позиции агента (не используется напрямую).  
\*** vx**: Скорость агента по оси x.  
\*** vy**: Скорость агента по оси y.  
\*** angle**: Угол ориентации агента.  
\*** \_**,** \_**,** \_**: Заполнители для неиспользуемых параметров.**reward**: Текущая награда, к которой добавляются новые бонусы и штрафы.**terminated**: Булево значение, указывающее, завершился ли эпизод (успешно или неудачно).**truncated**: Булево значение, указывающее, был ли эпизод прерван по какой-либо причине (например, истечение времени).**info**: Дополнительная информация об эпизоде, которая должна содержать информацию о действии, предпринятом агентом (info['action']).  
  
Логика работы:  
  
1. Базовые штрафы (не зависящие от действия):  
\*** distance\_reward**: Линейный штраф за отклонение по горизонтали от целевой точки посадки (x=0). Рассчитывается как** -abs(x) \* self.reward\_params['distance\_penalty']**.  
\*** angle\_penalty**: Линейный штраф за отклонение от вертикального положения (угол = 0). Рассчитывается как** -abs(angle) \* self.reward\_params['angle\_penalty']**.  
\*** velocity\_penalty**: Линейный штраф за скорость, учитывающий абсолютные значения скоростей по обеим осям. Рассчитывается как** -(abs(vx) + abs(vy)) \* self.reward\_params['velocity\_penalty']**.  
\* Эти штрафы суммируются и добавляются к текущей награде:** reward += distance\_reward + angle\_penalty + velocity\_penalty**.  
  
2. Штраф за действие (зависит от действия):  
\*** try...except KeyError:**: Функция пытается получить информацию о действии из словаря** info**под ключом** 'action'**. Использование блока** try...except**обеспечивает обработку ситуации, когда информация о действии отсутствует, предотвращая сбой программы.  
\*** action = info['action']**: Получение значения действия. Предполагается, что action представляет собой значение тяги (throttle).  
\*** if abs(action) > 0.8:**: Если абсолютное значение действия (тяги) превышает 0.8 (предположительно, в диапазоне от -1 до 1), применяется штраф.  
\*** reward -= 0.05**: Применяется небольшой штраф за использование слишком большой тяги. Это мотивирует агента использовать более умеренную тягу.  
  
3. Обработка отсутствия информации о действии:  
\*** except KeyError:**: Если ключ** 'action'**отсутствует в словаре** info**, выводится предупреждение:** print("Предупреждение: Информация о действии отсутствует в 'info'.")**. Это позволяет обнаружить проблему, когда окружение не предоставляет необходимую информацию.  
  
4. Бонусы и штрафы за завершение эпизода:  
\*** if terminated and reward > 0:**: Если эпизод завершился успешно (и накоплена положительная награда), добавляется бонус за посадку:** reward += self.reward\_params['landing\_bonus']**.  
\*** elif terminated:**: Если эпизод завершился неудачно (и/или накоплена отрицательная награда), применяется штраф за неудачу:** reward -= self.reward\_params['landing\_bonus']**.  
\*** elif truncated:**: Если эпизод был прерван, назначается небольшой штраф:** reward -= self.reward\_params['landing\_bonus'] / 2**.  
  
Возвращаемое значение:**reward**: Обновленное значение награды после применения штрафов и бонусов, включая штраф, зависящий от действия (если он был применен).  
  
Гиперпараметры (Определены в self.reward\_params):**distance\_penalty (float)**: Весовой коэффициент штрафа за расстояние.**angle\_penalty (float)**: Весовой коэффициент штрафа за угол.**velocity\_penalty (float)**: Весовой коэффициент штрафа за скорость.**landing\_bonus (float)**: Величина бонуса за успешную посадку (и штрафа за неудачную посадку).  
  
Стратегия:  
  
Функция** reward\_action\_dependent**расширяет стратегию формирующего награждения, вводя компонент, зависящий от действия. Это позволяет формировать поведение агента не только на основе состояния окружающей среды, но и на основе конкретных действий, которые он предпринимает. В данном случае, функция штрафует за чрезмерную тягу, что может быть полезно для предотвращения агрессивных маневров и стимулирования более плавного и контролируемого полета. Обратите внимание на важность наличия информации о действии в словаре** info**.  
  
Ключевые особенности и отличия от предыдущих версий:  
  
Учет действия агента: Самое главное отличие - учет действия, предпринятого агентом, при вычислении награды. Это позволяет формировать поведение агента более целенаправленно.  
Предположение о значении action:  Функция предполагает, что** action**представляет собой значение тяги (throttle). В других задачах** action**может представлять другие управляющие параметры.  
Обработка ошибок:  Использование** try...except KeyError**позволяет обрабатывать ситуации, когда информация о действии отсутствует, что повышает устойчивость кода.  
Возможность расширения: Функция может быть легко расширена для учета других действий и применения различных штрафов и бонусов в зависимости от этих действий.  
  
В заключение, эта функция награды представляет собой более продвинутый подход, который позволяет формировать поведение агента не только на основе состояния окружающей среды, но и на основе конкретных действий, которые он предпринимает. Это может быть полезно для обучения агента более сложным стратегиям и предотвращения нежелательного поведения. Важно убедиться, что окружение предоставляет информацию о действиях агента в словаре** info**.**