Test task: RL for gridworld

В данном задании предлагается пройти следующий путь: а) создать простую клеточную среду для задачи поиска и навигации, б) реализовать для нее два базовых решения, в) добавить усложнение в среду в виде более сложных неоднозначных стохастических наблюдений и адаптировать ваши решения под нее, г) векторизовать среду и код взаимодействия со агента со средой, д) провести тестирование решений и проанализировать полученные результаты.

Задача 1. Базовая среда

Реализуйте виртуальную клеточную среду-лабиринт, аналогичную классическому GridWorld:

- Среда представляет собой прямоугольную область размера (h, w) (гиперпараметры инициализации), окруженной стеной. Внутри области могут быть расположены препятствия (задается бинарной 2д маской `obstacle_mask` при инициализации).
- Цель агента — добраться из текущей позиции до целевой позиции. Позиция цели `pos_goal` фиксированная и задается на этапе инициализации. Другой параметр инициализации `pos_agent` задает позиции, в которых агент может случайным образом оказаться в начале эпизода. Пусть `pos_agent` может быть задан а) либо одной парой координат — фиксированной позицией старта агента в каждом эпизоде, б) либо 2д массивом размера (w, h), где записаны любые числа — при нормировке данный массив будет задавать вероятность стартовать эпизод из каждой клетки (но нужно не забыть наложить на этот массив маску препятствий!)
- В качестве наблюдения среда выдает представление текущей клетки агента (не координаты! но можно добавить такую опцию для дебага среды). В простейшем варианте, агент видит тип клетки, на которой находится, закодированный в one-hot вектор. Типы следующие: пол, цель, стена (=граница области), препятствие (внутри области). Далее типы я буду называть цветами, как будто тип задается покраской. И есть еще нюанс: пол покрашен в `n_colors` разных цветов. Таким образом, всего `n_colors + 3` цветов (=фичей) в наблюдении.
- Среда принимает в кач-ве параметра инициализации 2д массив размера (h, w) с покраской внутренней области лабиринта
  - для удобства стоит реализовать функцию-хелпер, осуществляющий случайную покраску пола для заданных параметров среды (размер, цвет, рнг) и возвращающий нужный для инициализации среды 2д массив.
- Возможно, у вас к этому моменту есть вопрос — как может агент “видеть” цвет клетки препятствия/стены, если он не может на ней стоять? Давайте реализуем флаг инициализации среды `see_obstacle`, который определяет логику, что видит агент в момент попытки пойти в стену/препятствие: `true` — наблюдает цвет препятствия, `false` — наблюдает клетку, в которой окажется (=текущую).
- Действия агента: одно из четырех направлений (пронумеруйте по часовой, начиная с вверх).
- Среда реализует интерфейс gymnasium.Env. Можно не прям досконально — например, работу с рнг сделать по-своему. Важно, чтобы длина эпизода задавалась так же, в среде корректно задавались `observation_space` и `action_space`, а также интерфейс `reset()` и `step()` был правильным.

Предлагаю следующие сетапы среды для тестирования (по возрастанию сложности):

1. Среда 5x5 с уникальными цветами для каждой позиции (25 цветов) без препятствий. Это полностью наблюдаемая задача (MDP). Позволит провести поверхностное тестирование корректности решения и дебаг.
2. Среда 5x5 с 5 цветами пола без препятствий. Относительно простая задача, тестирующая способность агента учиться ориентироваться в частично наблюдаемой среде.
3. Среда 10x10 7 цветов пола со случайно расставленными 10% препятствий. Проверка масштабирования метода на более крупный размер среды.
4. Среда 10x10 4 цвета пола со случайно расставленными 10% препятствий. Увеличенная неопределенность в наблюдении за счет большей повторяемости цветов.

Будем использовать их и в последующих заданиях.

Задача 2. Базовые решения

Реализуйте методы DQN и PPO, а также сам pipeline обучения. Проведите серию экспериментов, показывающих качество решения и небольшой сравнительный анализ.

Рекомендации и пояснения:

- можно использовать любые источники для реализации методов, но реализация должна быть целиком внутри сданного вами репозитория (т.е. подключить либу и вызвать полностью весь цикл обучения или целиковую реализацию архитектуры в ней не подойдет)
- (к предыдущему пункту) в рамках данного задания ожидается, что вы разберетесь с предложенными методами хотя бы в общих чертах и будете способны объяснить свой код
- подумайте, какие архитектурные решения нейронной сети точно стоит использовать для данной задачи (нужен ли сложный энкодер наблюдений, нужна ли память и тп)
- в качестве визуализации результатов можно привести следующие графики обучения: средняя отдача за эпизод, компоненты лосса. Также можно привести пример записи эпизода работы обученного агента (здесь можно по-разному, например визуализировать траекторию агента на карте среды).

Задача 3. Кодирование цвета изображениями из MNIST

Добавьте версию среды, в которой наблюдения теперь кодируются следующим образом. До этого цвет кодировался one-hot вектором из дискретного множества (N+3 цвета). А теперь будет кодироваться случайным изображением из MNIST (например, цвет 2 -> случайное изображение двойки из мниста). *NB: это ограничит нас в числе возможных цветов, т.е. максимум 7 цветов покраски пола (+3 служебных), так что уже нельзя будет использовать этот вариант среды для 1го сетапа.*

Рекомендации:

- опционально для ускорения можно предусмотреть небольшое уменьшение масштаба изображения (но чтобы изображение было не меньше 12x12)
- выберите как вам самим удобно, стоит ли отдавать наблюдение в виде 2д массива или во flatten виде;
- лучше не изменять имеющуюся у вас среду из первого задания, а добавить рядом еще одну (для этого и след пункта);
- как думаете, на что влияет данное изменение среды? что оно тестирует?
- подумайте, нужно ли / стоит ли использовать альтернативную архитектуру сети у агента для данной модификации.

Цель данного задания: проверить работу на среде с более сложными/неопределенными наблюдениями.

Задача 4. Векторизация среды

Добавьте векторизованную версию среды (см. документацию gymnasium) — версию, которая поддерживает симуляцию нескольких `num_envs` параллельно идущих эпизодов.

Обратите внимание, gymnasium из коробки позволяет "симулировать" векторизацию среды из коробки с помощью классов `SyncVectorEnv` и `AsyncVectorEnv`. Это неполноценная векторизация — только на уровне API, — где сами вычисления для каждого параллельного эпизода производятся отдельно, а следовательно преимущества в скорости работы среды вы не получите. В данном задании предлагается сделать два шага:

1. Сделайте наивную векторизацию (если к этому моменту еще не) базовыми средствами gymnasium (лучше использовать `SyncVectorEnv`) и адаптируйте ваш пайплайн обучения модели под работу в batch-режиме. *NB: выше было упомянуто, что это почти не ускорит среду, но будет ли это верно для всего цикла обучения в совокупности?*
2. Реализуйте полноценную векторизацию среды, например, средствами `numpy`. *При работе над данным пунктом вам скорее всего пригодится данная статья.*

Сравните скорость и качество обучения с обычной версией. При сравнении графиков обучения обязательно используйте общее число шагов (т.е. сумму по всем параллельным средам).

Цель данного задания: проверить работу на среде с более сложными/неопределенными наблюдениями.

Формат сдачи

Ссылка на репозиторий с кодом (мой гитхаб `pkuderov`), по возможности ссылка на проект в wandb с графиками запусков, а также небольшой отчет о проделанной работе (краткое описание как воспроизвести запуски, комментарии к экспериментам и результатам, свои мысли на этот счет). Последние два пункта можно собрать в wandb reports и дать дополнительно ссылку на него. Если нет возможности использовать wandb, воспользуйтесь доступными инструментами для формирования и отображения графиков.

Дополнительные рекомендации и пояснения:

1. Цель всего задания проверить комплекс ваших навыков: умение быстро разобраться с новой темой, умение писать/оформлять код и справляться с техническими сложностями, умение провести и оформить исследование, умение выбрать некоторое законченное подмножество большой задачи, умение попросить помощь.
2. Частичное/неудачное решение тоже решение. Задание большое и в целом нет ожидания, что будет сделано все, еще и на хорошем уровне. Поэтому учитывайте заранее ограниченность времени и свою загруженность. Вам самим нужно решить что, как, в каком объеме и насколько качественно выполнять. Рассматривайте это тоже как часть общего задания и будьте готовы пояснить свой выбор.
3. Со мной можно консультироваться и делиться промежуточным решением до дедлайна.