Рекомендации по выполнению лабораторных работ

Рассмотрим рекомендации на примере проекта змейка.

В файле **snakeHuman.py** расположен программный код игры змейка, написанный на языке python. При запуске этого файла открывается игра «змейка», в которой управляя стрелками, змейкой (синей цепочкой квадратиков) требуется собирать яблоки (красные квадратики). За каждое собранное яблоко добавляется 1 очко. При этом змейка погибает, если врезается в край карты или любую часть своего хвоста.

Требуется написать проект, обучающий искусственный интеллект, управляющий змейкой. Проект должен содержать среду включающую змейку и яблоки (environment, snake и food в файле **environment.py**); агента управляющего змейкой и получающего информацию (state) из среды (agent в файле **snakeAI.py**); модель обучения агента (model в файле **model.py**). Также следует добавить построение графиков (statistics в файле **statistics.py**), отображающих эффективность разработанного метода.

Среда, в разрабатываемом проекте, почти полностью берется из файла **snakeHuman.py**, и помещается в файл **environment.py**. Следует выделить три класса: Environment, Snake, Food. Рассмотрим содержимое указанных классов.

Класс Environment включает поля:

self.fieldW – ширина поля (в клетках).

self.fieldH – высота поля (в клетках).

self.cellSize – размер клетки поля.

self.display – окно в котором происходит отрисовка игры.

self.clock – часы (необходимые для смены кадров игры).

self.snake – змейка.

self.food – яблоко.

self.score – количество очков набранных змейкой.

и методы:

__init__(self, window_width=640, window_height=480, cell_size=20) – конструктор, заполняющий поля класса начальными значениями.

reset(self) – сброс параметров игры к начальным значениям. Например, в начале очередного эпизода наступившего после смерти змейки.

play_step(self, action) – выполнение очередного шага игры (шага змейки).

update_ui(self) – отрисовка нового кадра.

Класс Snake включает поля:

self.direction – направление в котором двигается змейка.

self.head – координаты головы змейки.

self.body – тело змейки.

и методы:

__init__(self, w, h) – конструктор, заполняющий поля класса начальными значениями.

move(self, action) – функция, выполняющая шаг змейки в соответствии с выбранным значением action.

is_collision(self, w, h) – проверка того столкнулась ли змейка со стенкой или своим хвостом.

is_collide_with_something(self, list_of_something) – проверка того столкнулась ли змейка с какими-то объектами.

Класс Food включает поля

self.pt – координаты клетки в которой расположено яблоко

self.fieldW – ширина поля (в клетках)

self.fieldH – высота поля (в клетках)

и методы:

__init__(self, field_width, field_height, x, y) – конструктор, заполняющий поля класса начальными значениями

place_food(self) – размещение яблока в начале игры или после того как предыдущее было съедено змейкой.

Environment должны присутствовать функции: помимо конструктора, нужно перенести функции play_step и update_ui

Агент (в файле **snakeAI.py**) должен уметь:

1) Получать состояние (с помощью функции get_state) из среды (environment).
2) Выбирать действие (с помощью функции get_action), которое должна совершить змейка.
3) Работать с памятью агента (с помощью функций remember, train_long_memory, train_short_memory), если она используется алгоритмом.
4) Запускать цепочку обучения агента (с помощью функции train).

Модель обучения (в файле **model.py**), в разрабатываемом проекте должна:

1) Для алгоритмов value-based и actor-critic (с помощью функции train_step), улучшать значения qvalue для выбранных state и action.
2) Для алгоритмов policy-based и actor-critic обучать стратегию (с помощью функции forward).
3) Для алгоритмов policy-based и actor-critic сохранять лучшую из найденных стратегий (с помощью функции save).

Код для построения графиков и формирования статистики можно предоставить студентам в виде файла **statistics.py** и куска функции train (файла **agentAI.py**). Или же посвятить отдельное время обучению работы с графиками (библиотекой matplotlib) и статистикой в python.