Применение машинного обучения в задачах теории игр

Пуговкина Диана Алексеевна, гр.20.Б04-мм

Санкт-Петербургский государственный университет
Прикладная математика и информатика
Вычислительная стохастика и статистические модели

Научный руководитель: к.ф.-м.н., доцент Шпилев П.В.
Рецензент: к.ф.-м.н., доцент Пепелышев А.Н.

Санкт-Петербург, 2024

Введение

Цели работы:

1. Создать агента искусственного интеллекта, который с помощью Q-обучения в сочетании с нейронными сетями сможет научиться играть в Pac-Man, взаимодействуя со средой без знания правил игры.
2. Найти оптимальные параметры и архитектуру нейронной сети алгоритма Deep Q-learning.

Обучение с подкреплением

Простейшая модель обучения с подкреплением состоит из:

- множества состояний окружения (states) S;
- множества действий (actions) A;
- множества скалярных наград (rewards).

Определение

Политика $\pi : S \to A$ — это стратегия, которую использует агент, для определения следующего действия a' на основе текущего состояния среды.

Задача максимизировать величину

$$R = \sum_t \gamma^t r_t,$$

где $\gamma \in [0, 1]$ – дисконтирующий множитель для предстоящей награды.

Pac-Man

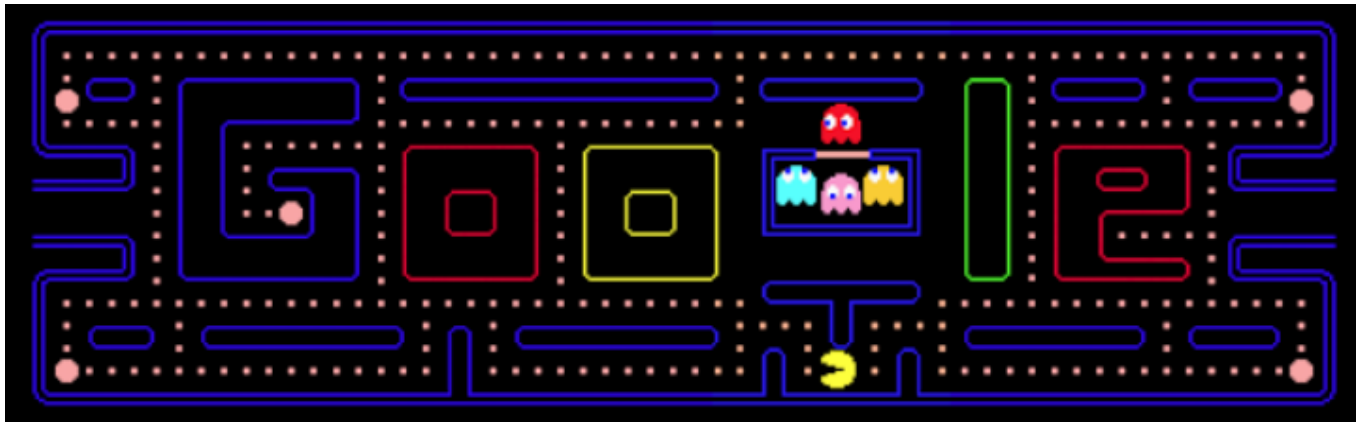

Рис.: Игровое поле

Ситуация	**Награда**
Съесть призрака	50
Съесть еду	20
Проиграть	-500
Выиграть	200

Q-learning

Q-learning называется табличный алгоритм обучения с подкреплением:

- Проинициализировать $Q^*(s, a)$ произвольным образом.
- Пронаблюдать s_0 из среды.
- Для $k = 0, 1, 2, \ldots$:
  1) с вероятностью ε выбрать действие a_k случайно, иначе жадно:
  $$a_k = \text{argmax}_{a_k} Q^* (s_k, a_k)$$
  2) отправить действие a_k в среду, получить награду за шаг r_k и следующее состояние s_{k+1}.
  3) обновить одну ячейку таблицы:

$$Q^* (s_k, a_k) \leftarrow (1-\alpha) Q^* (s_k, a_k) + \alpha \left(r_k + \gamma \max_{a'} Q^* \left(s_{k+1}, a' \right) \right)$$

Deep Q-learning

Входными данными является необработанное изображение текущей игровой ситуации. Оно проходит через несколько сверточных слоев, а затем через полносвязный слой. Результатом является Q-значение для каждого действия, которое может предпринять агент.

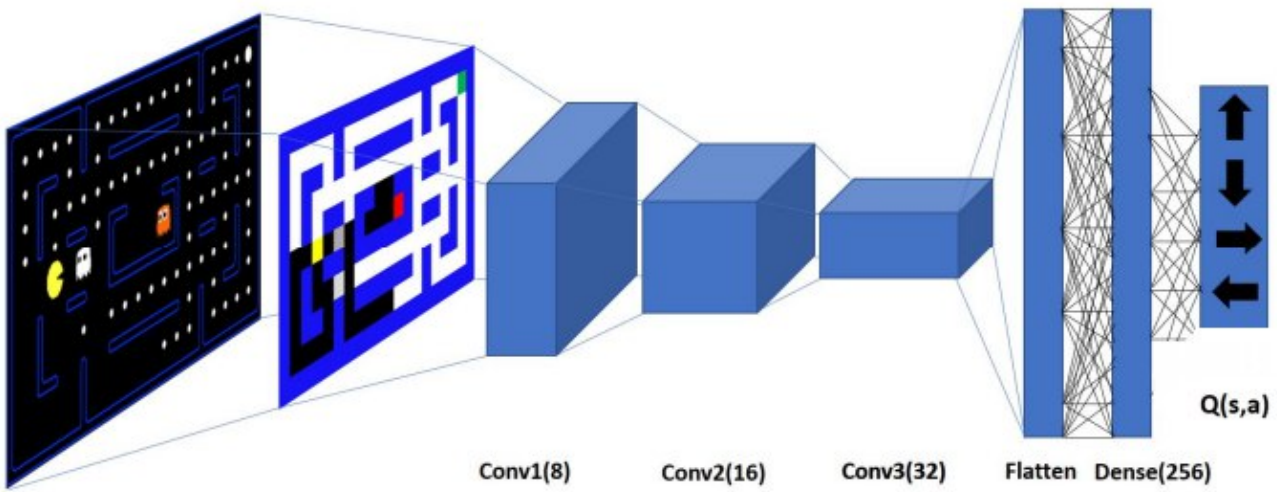

Рис.: DQN

Нейронные сети

Полносвязный слой можно описать следующей формулой:

$$f(x) = K\left(\Sigma w_i x_i + b\right)$$

- K – **функция активации**, которая применяется к выходным данным, чтобы добавить нелинейность,
- x – входные данные (вектор), полученные из предыдущего слоя,
- f – выходные данные слоя,
- w – вектор весов (по одному для каждого входа),
- b – член смещения.

Функция активации

Функция активации определяет выходной сигнал нейрона на основе его входа.

Сигмоидная функция (логистическая)

Логистическая функция нелинейна и сжимает входные значения от 0 до 1 по формуле:

$$F(x) = (1 + exp(-x))^{-1}$$

Функция активации ReLU

Формула ReLU:

$$f(x) = max(0, x),$$

где x — входной сигнал, а $f(x)$ — выходной.

Sigmoid Vs Rectifier

Определение

Если в среде есть терминальные состояния, одна итерация взаимодействия от начального состояния до попадания в терминальное состояние называется **эпизодом** (episode).

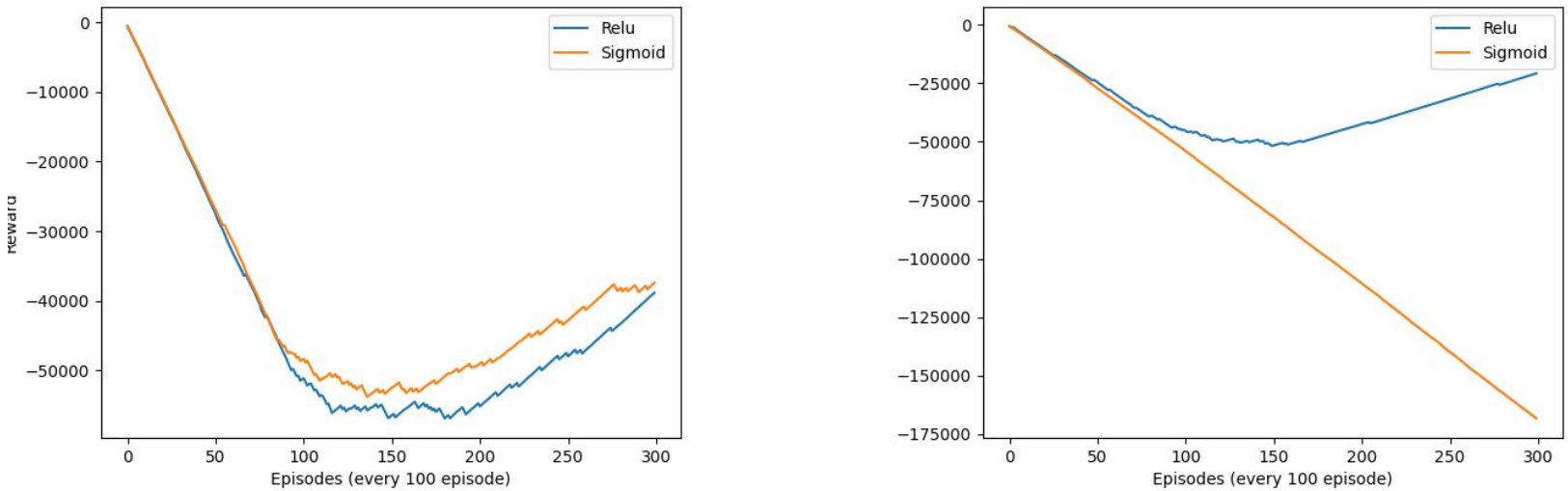

Рис.: Сравнение функций активации на сетях с разной архитектурой

Модификации Rectifier

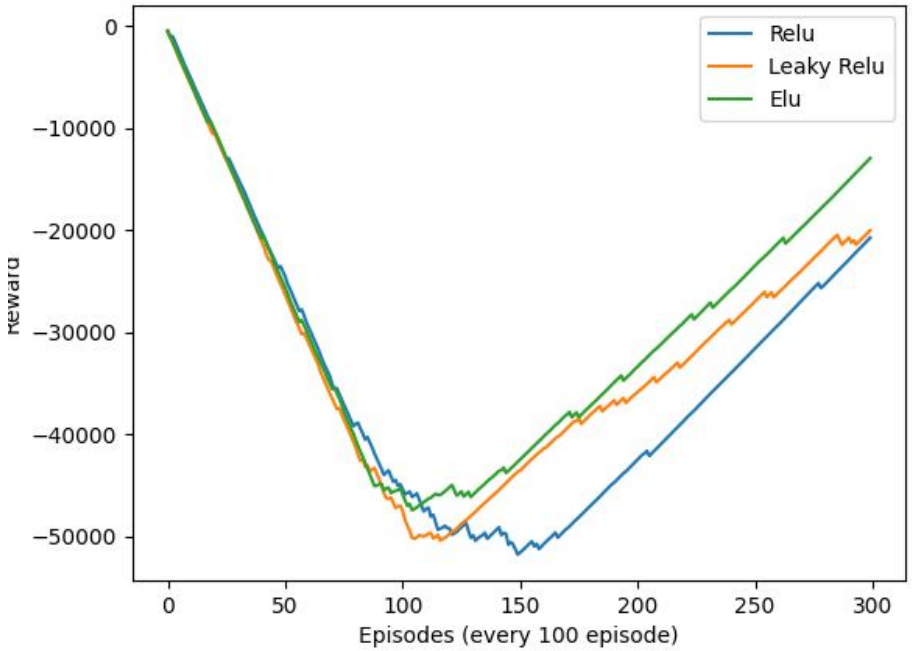

Рис.: Сравнение агентов с разными модификациями ReLU

Функция активации	Формула	Win rate
ReLU	$max(0, x)$	0.97
Leaky ReLU	$max(0, 0.01x, x)$	0.96
ELU	$f(x) = x$ при $x > 0$, иначе $a(e^x - 1)$	0.98

Adam Vs RMSProp

Оптимизаторы моделей в глубоком обучении определяют оптимальный набор параметров модели, таких как вес и смещение, чтобы при решении конкретной задачи модель выдавала наилучшие результаты.

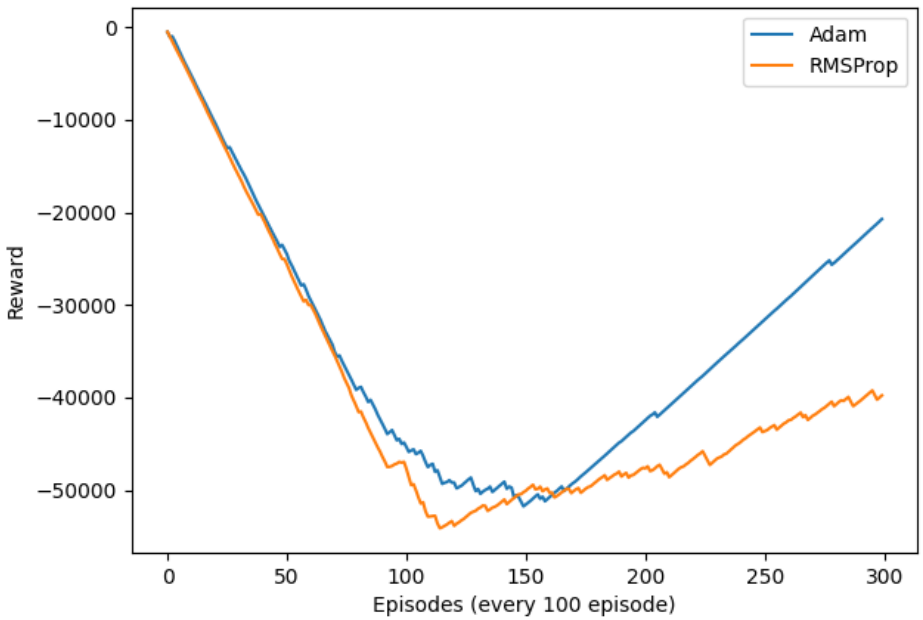

Рис.: Adam vs RMSProp

Полученные результаты

Сыграем 200 игр на разных по количеству слоев сверточных нейронных сетях с функцией активации ELU, алгоритмом оптимизации Adam и с коэффициентом скорости обучения равным 0.001.

Сетка	**Тренировочные**	Average score	**Тестовые**	Average score
DQN2	0.94	459.06	0.71	225.96
DQN3	0.98	505.08	0.76	294.835
DQN4	0.91	432.37	0.31	-177.18
DQN5	0.96	489.315	0.36	-152.28

Заключение

- Найдены оптимальные параметры и архитектура нейронной сети алгоритма Deep Q-learning для Pac-Man.
- Наилучший результат для тренировочных и тестовых карт показала архитектура сети DQN3, то есть архитектура с двумя сверточными слоями, одним полносвязным и одним слоем вывода.

Конец

Спасибо за внимание!