Маршрутизация в компьютерной сети с применением глубокого обучения

Панчишин Иван Романович, группа М41381с

2021-06-28

Задача 5

Реализован алгоритм DQN. Код агента находится в файле `agents/dqn.py`. Агент использует нейросеть для предсказания времени доставки через каждого соседа, а также дообучает модель на более точных предсказаниях, полученных от соседей в качестве вознаграждения.

Нейросеть реализована при помощи библиотеки PyTorch. Она состоит из 2 полносвязных скрытых слоев с выпрямителем (ReLU). Выходной слой состоит из 1 нейрона, который выдает оценку времени доставки. Входной слой может быть двух типов, как показано на рисунке 1.

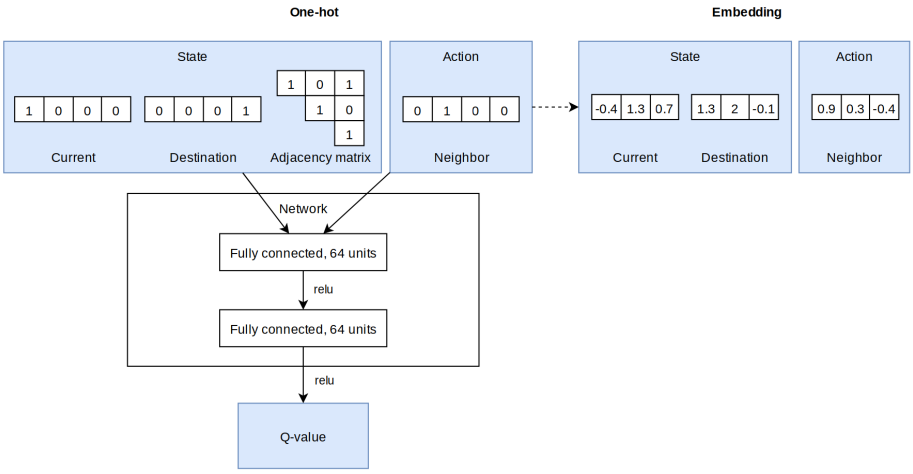

Рис. 1: Архитектура сети

Исходный код модели находится в файле `networks/qnetwork.py`.

Для предварительного обучения модели написан генератор обучающей выборки. Каждый элемент выборки имеет время доставки по кратчайшему пути без очередей, которое является целевым значением, и следующие признаки:

- `dst` - номер узла назначения пакета,
- `src` — номер узла отправки пакета,

- `pkg_id` — идентификатор пакета,
- `nbr` — номер соседнего узла, через который будет отправлен пакет,
- `amatrix_0, amatrix_1, ...` — матрица смежности.

Обучение выполнено на 165293 состояниях, по 64 штуки. Во время генерации обучающей выборки, последовательно обрывалось и восстанавливалось каждое соединение. Кривая обучения представлена на рисунке 2.

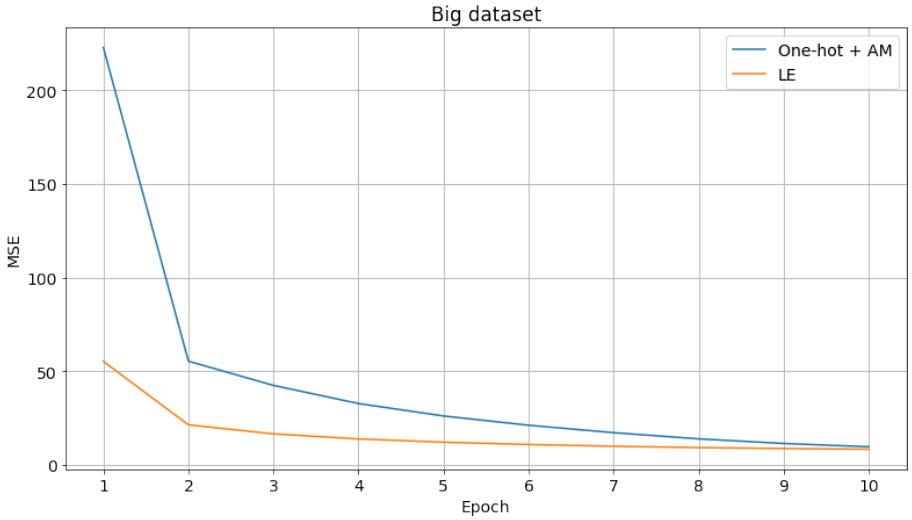

Рис. 2: Обучение моделей

Задача 6

Реализован метод получения эмбеддингов Laplacian Eigenmaps. В графе компьютерной сети каждое ребро имеет вес w, равный времени прохождения пакета через узел:

$$w = \mathrm{sz} \,/\, \mathrm{bw} + \mathrm{proc},$$

где sz — размер пакета, bw — пропускная способность канала, proc — время обработки пакета узлом.

Чем больше вес ребра, тем ближе получаются эмбеддинги соответствующих узлов, поэтому вес каждого ребра инвертируется.

Чтобы эмбеддинги в графах, отличающихся только весами ребер, были различными, рассчитывается средний вес ребра. На него делится вес каждого ребра и домножается матрица эмбеддингов.

Исходный код энкодеров узлов находится в файле `networks/nodeenc.py`.

Задача 7

На рисунке 3 показана работа DQN в новой сети того же размера под низкой нагрузкой.

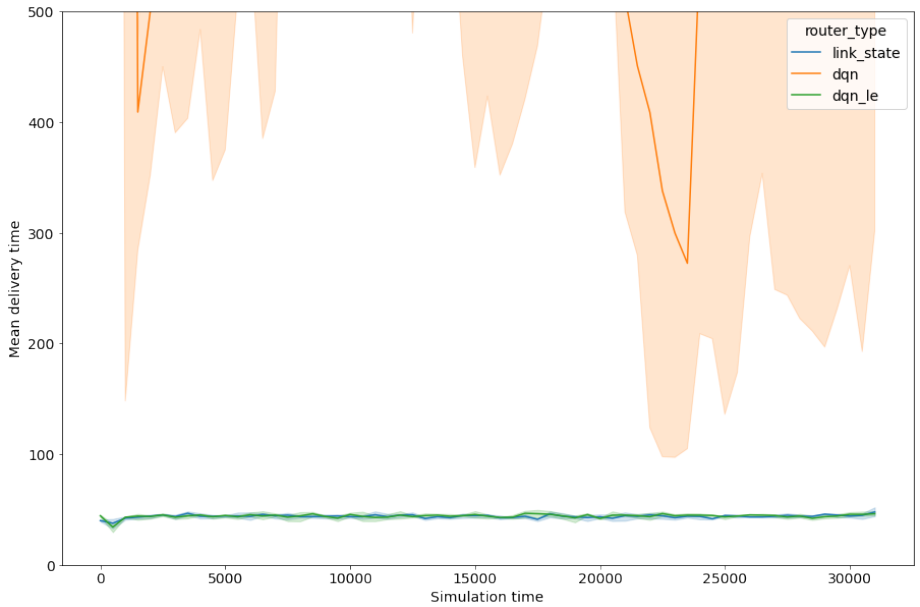

Рис. 3: Работа агентов в неизвестном окружении

На рисунке 4 показана структура новой сети:

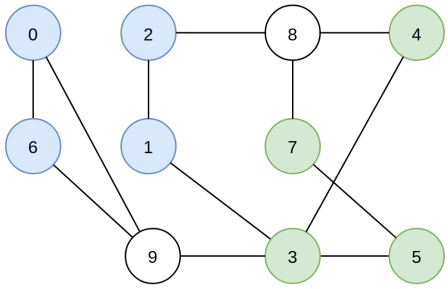

Рис. 4: Новая сеть

Версия с использованием LE-эмбеддингов работает оптимально и аналогично link-state. Версия с one-hot представлением узлов работает хаотично и не сходится к стабильной стратегии. Опыт нейросети, обученной на LE-эмбеддингах, хорошо обобщается на графы того же размера.

Задача 8

- Размер пакета: 1000 байт
- Пропускная способность канала: 100 байт/шаг
- Время обработки пакета: 5 шагов

Размер пакета: 1000

Адаптация к изменению нагрузки на сеть

Сценарий отправки пакетов:

- 100 пакетов с задержкой 12 шагов между любыми узлами,
- затем 500 пакетов с задержкой 12 шагов от узлов 0, 1, 2, 6 к узлам 3, 4, 5, 7,
- затем 1500 пакетов с задержкой 7 шагов между теми же узлами,
- затем 500 пакетов с задержкой 12 шагов между теми же узлами.

Результат симуляции приведен на рисунке 5.

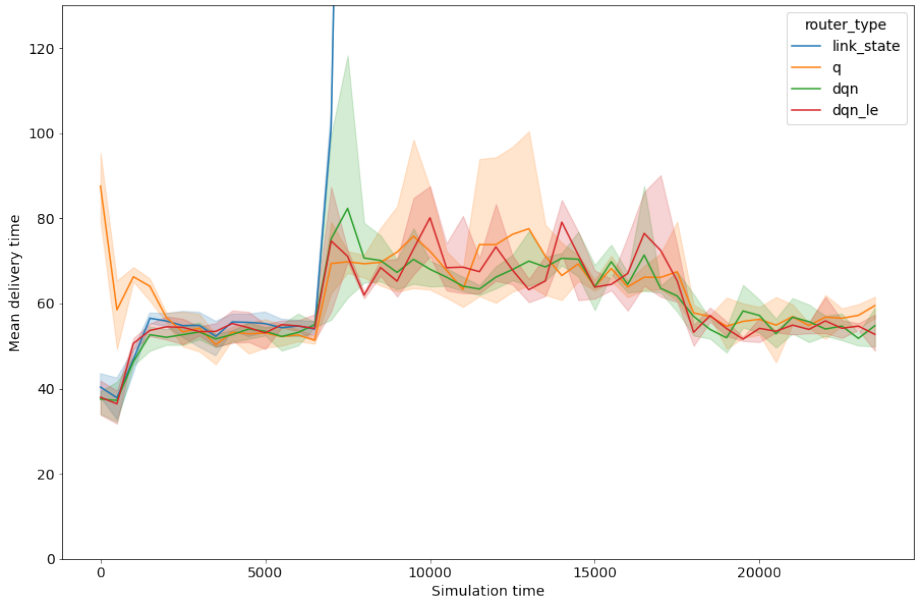

Рис. 5: Изменение нагрузки

DQN работает аналогично Q.

Обрыв соединений

Сценарий отправки пакетов:

- 100 пакетов с задержкой 12 шагов между любыми узлами,
- затем 3500 пакетов с задержкой 12 шагов от узлов 0, 1, 2, 6 к узлам 3, 4, 5, 7.

Соединения (6, 7), (0, 1), (4, 5) последовательно обрываются спустя каждые 500 пакетов, начиная со 101 пакета, затем восстанавливаются в том же порядке, с тем же интервалом.

Результат симуляции приведен на рисунке 6.

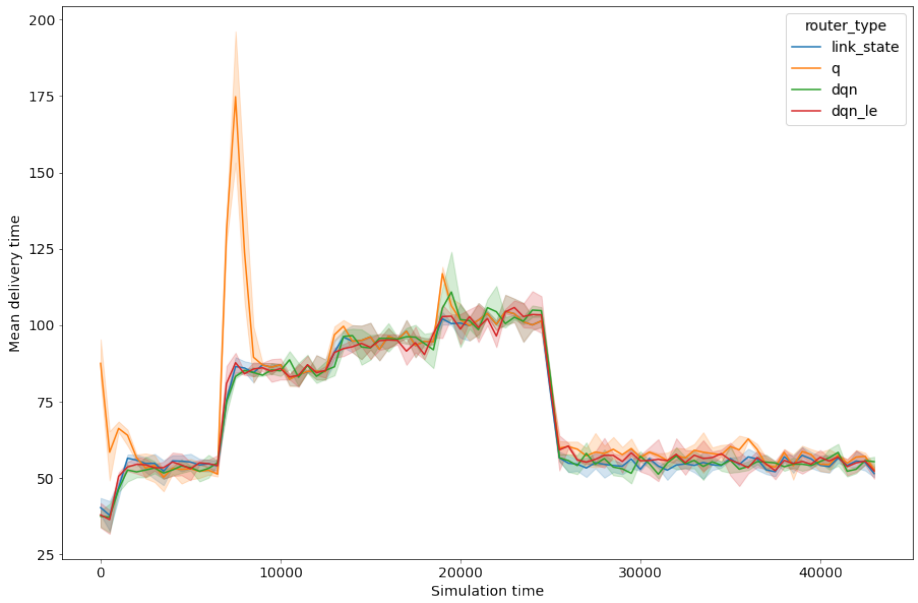

Рис. 6: Обрыв соединений

DQN не имеет выраженных скачков времени доставки, которые наблюдаются с Q. Благодаря предобучению, нейронная сеть узнает обрывы и восстановления соединений и позволяет агенту быстрее прийти к оптимальной стратегии.