СОДЕРЖАНИЕ

1 Рекуррентные нейронные сети

Что же делает рекуррентные сети такими особенными? Вопиющим ограничением классических нейронных сетей, рассмотренных ранее (а также сверточных нейронных сетей) является то, что их API слишком ограничен: они принимают на вход вектор фиксированного размера (например, изображение) и возвращают вектор фиксированного размера (например, вероятности различных классов). Более того, эти модели выполняют заданное отображение за фиксированное количество вычислительных шагов (например, количество слоев в модели). Основная причина, по которой рекуррентные нейронные сети настолько интересны заключается в том, что они позволяют нам работать с *последовательностями* векторов: последовательность входных данных, выходных данных, или вообще и того, и другого. Рассмотрим несколько примеров:

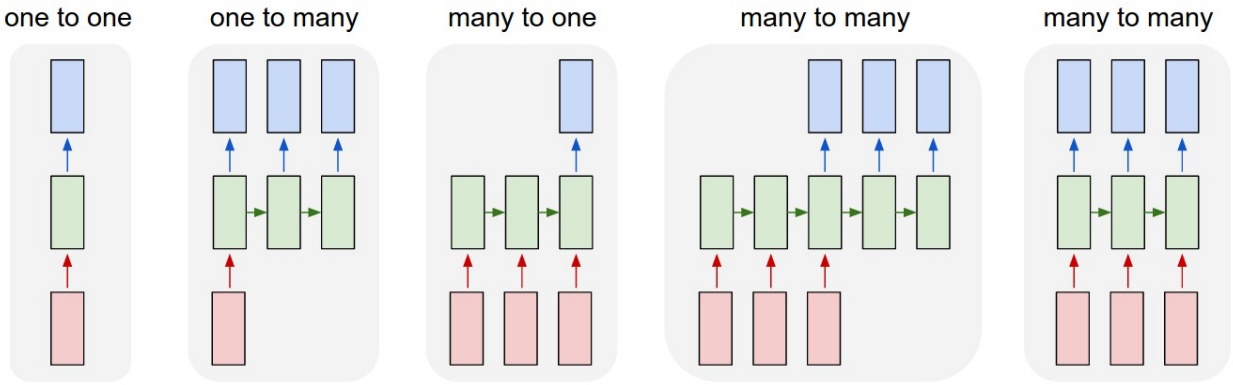

Рис. 1.1: Каждый прямоугольник представляет собой вектор, а стрелки - функции (например, матричное умножение). Векторы входных данных представлены красным цветом, выходных - синим, а зеленые содержат в себе состояние RNN (подробнее об этом далее). Слева направо: (1) Классический метод обработки без RNN, на основании входа конечного размера производит вывод конечного размера (например, классификация изображений). (2) Последовательность в качестве выхода (например, описание изображений). (3) Последовательность в качестве входа (например, анализ эмоциональной окраски, при котором заданное предложение классифицируется как выражающее положительное или негативное настроение). (4) Последовательность в качестве как входа, так и выхода (например, машинный перевод: RNN считывает предложение на английском языке и возвращает его перевод на французском языке). (5) Синхронизированная последовательность в качестве как входа, так и выхода (например, классификация видео, где мы хотим назвать каждый кадр видео). Заметим, что ни в одном из случаев на последовательности длин не наклыдывается никаких предварительных ограничений, т.к. рекуррентное преобразование (зеленое) фиксированно и может применяться сколько угодно раз.

Очевидно, что режим работы с последовательностями гораздо более мощный, по сравнению с фиксированными сетями, которые изначально обречены фиксированным количеством вычислительных шагов. По своей

сути, RNN описывают программы. Вообще говоря, если проводить аналогию с универсальными теоремами аппроксимации, то известно, что RNN Тьюринг-полны в том смысле, что они могут симулировать поведение произвольных программ [1].

Если обучение классических нейронных сетей можно назвать оптимизацией над функциями, то обучение рекуррентных сетей можно назвать оптимизацией над программами.

Примечание Несмотря на то, что классические RNN сами по себе являются очень интересными, они обычно служат, своего рода, переходной ступенью к пониманию более продвинутых моделей, таких как LSTM и трансформеры (см. главы todo):

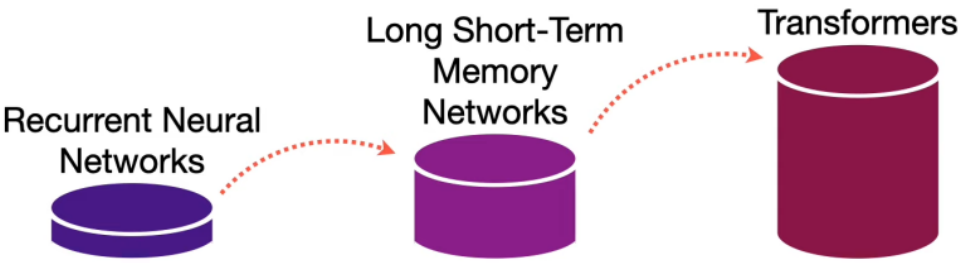

Рис. 1.2

1.1 Архитектура рекуррентной нейронной сети

Как же рекуррентным нейронным сетям удается работать с последовательностями произвольных размеров? У них это получается благодаря циклам, встроенным в них, позволяющим сохранять информацию.

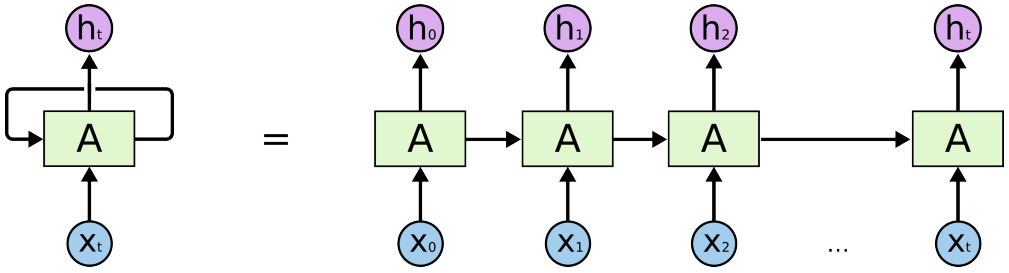

Рис. 1.3: Схема рекуррентного слоя, где (слева) фрагмент нейронной сети A принимает на вход некоторое значение x_t и возвращает некоторое значение h_t. Цикл позволяет информации передаваться от одного шага сети к другому. Порой подобная цикловая структура может сбить с толку, однак если его развернуть (справа), то оказывается, что это очень похоже на классическую нейронную сеть. О RNN можно думать как о множестве копий одной и той же сети, каждая из которых передает сообщение своему последователю [2].

Подобная цикловая структура позволяет обрабатывать последовательности данных произвольных размеров (стоит упомянуть, что в RNN мы используем одни и те же значения параметров (весов и смещений) внутри одного рекурентного слоя. Подобное разделение параметров позволяет применять и обобщать модель на примеры различной формы (в данном случае длины). Если бы у нас были различные параметры для каждого временного индекса, мы не смогли бы ни обобщить модель на длины последовательностей, не встречавшиеся на этапе обучения, ни распространить статистическую силу на последовательности разной длины и на разные моменты времени).

1.2 Прямой проход

Разумеется на практике чаще используют RNN, состоящие не из одного рекуррентного слоя, а нескольких, чтобы получить, так называемые, **глубокие рекуррентые нейронные сети (depp RNNs)**. Такой подход позволяет нам, как и в случае многих слоев в FFNN и CNN, обучить иерархические зависимости между признаками, что на практие работает лучше однослойных сетей.

1.3 Обратный проход

В силу своей природы, в RNN используется не классический метод обратного распространения ошибки (backpropagation) (см. главу todo), а его модификация: **метод обратного распространения ошибки во времени (backpropagation through time (BPTT))** (разумеется существуют и другие алгоритмы, например RTRL, но в данной работе мы сфокусируемся на BPTT, тк он концепутально проще, популярнее и вычислительно эффективнее по времени (но не по памяти)). Принцип работы у BPTT такой же, как и у традиционного backprop. Единственное отличие заключается в том, что BPTT суммирует ошибки в каждый момент времени, когда, сети прямого распространения, в свою очердь, в этом не нуждаются, тк они не сохраняют значения параметров между слоями.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Andrej Karpathy. The unreasonable effectiveness of recurrent neural networks. https://karpathy.github.io/2015/05/21/rnn-effectiveness/#:~:text=models%20perform%20this%20mapping%20using,A%20few%20examples%20may, 2015. [Online; accessed 2025-15-05].

[2] Christopher Olah. Understanding lstm networks. https://colah.github.io/posts/2015-08-Understanding-LSTMs/#:~:text=Recurrent%20neural%20networks%20address%20this,them%2C%20allowing%20information%20to%20persist, 2015. [Online; accessed 2025-15-05].