Вопрос 1

Выполнен

Баллов: 1,00 из 1,00

Почему рекуррентные нейросети часто медленнее свёрточных?

Выберите один ответ:

a. Потому что нет хороших программных реализаций

b. Потому что очередной элемент может быть обработан только после завершения обработки предыдущего элемента, так как зависит от значения скрытого состояния

Ваш ответ верный.

Вопрос Инфо

И, самое главное, рекуррентки обучаются не так легко. Здесь появляются проблемы затухания или взрыва градиента. Поэтому современные работы в области рекурренток, в основном, связаны с поиском баланса между мощностью и стабильностью процесса обучения. Давайте попробуем понять, откуда берутся эти самые проблемы — затухание и взрыв градиента. Мы это будем делать на примере классической рекуррентки (такие сети ещё называют "vanilla"). При этом мы будем считать, что размерность всех переменных (входных данных и скрытого состояния) равна единице, то есть мы будем работать только со скалярными операциями. Для простоты будем считать, что мы предсказываем одну единственную величину по всему тексту — например, мы решаем задачу классификации и класс предсказываем на основе значения состояния на последнем шаге, то есть после прочтения всего входного текста. Соответственно, предсказание подаётся в функцию потерь или функционал качества, исходя из которого мы будем градиентным спуском настраивать параметры нейросети. Здесь мы не будем специфицировать, какую конкретно функцию активацию мы используем — для того, что мы хотим сделать, это неважно. Итак, до прочтения первого слова у нас уже есть начальное состояние — это тоже параметр сети. Затем мы читаем первое слово и находим новое значение скрытого состояния. Затем — ещё слово, и при этом важно помнить, что h1, на самом деле — это функция от начального состояния и первого слова. Ну, и так далее... Таким образом мы получим глубокую композицию функций.

Вопрос 2

Выполнен

Баллов: 1,00 из 1,00

Отметьте переменные, от которых зависит значение скрытого состояния h_5 на шаге 5 в классической однонаправленной рекуррентной сети.

Выберите один или несколько ответов:

- [ ] a. Вход на шестом шаге x_6
- [x] b. Вход на первом шаге x_1
- [x] c. Скрытое состояние на четвёртом шаге h_4
- [ ] d. Скрытое состояние на шестом шаге h_6
- [x] e. Вход на пятом шаге x_5

Ваш ответ верный.

Вопрос Инфо

Итак, прямой проход по сети сделали. Ну, пора и домой — то есть обратно. Раскручиваем цепочку вызовов, начиная со значения функции потерь. Предсказание (yy) — это функция от последнего состояния, которое, в свою очередь, является функцией предпоследнего состояния и последнего слова, и так далее. Мы применяем градиентный спуск, поэтому попробуем посчитать производную функции потерь по весам нейросети. В первую очередь нас интересует производная по весам рекуррентного блока — в нашем случае это одно число (ww). yy — это сложная функция, поэтому применяем правило цепочки.

Вопрос 3

Выполнен

Баллов: 1,00 из 1,00

Значение ошибки для некоторых данных вычисляется по формуле $Loss(g(f(x;w)))$, где x - входные данные, w - параметры модели, $Loss, g, f$ - некоторые функции.

Выберите правильное применение правила цепочки для нахождения производной функции ошибки по весам модели $\frac{\partial Loss}{\partial w}$.

Выберите один ответ:

a. $\frac{\partial Loss}{\partial w} = \frac{\partial Loss}{\partial g} + \frac{\partial g}{\partial f} + \frac{\partial f}{\partial w}$

b. $\frac{\partial Loss}{\partial w} = \frac{\partial Loss}{\partial g}\frac{\partial g}{\partial f}\frac{\partial f}{\partial w}$

c. $\frac{\partial Loss}{\partial w} = \frac{\partial Loss}{\partial g}\frac{\partial g}{\partial f}$

d. $\frac{\partial Loss}{\partial w} = \frac{\partial Loss}{\partial g}\frac{\partial f}{\partial w}$

Ваш ответ верный.

Вопрос Инфо

Мы дошли до производной последнего рекуррентного состояния по весам. Вот это значение равно $h_t$thtt. Так так! Давайте здесь остановимся поподробнее. Мы опять берём производную сложной функции, поэтому результат будет равен произведению производной самой функции и производной её аргумента. Производная самой функции нас пока что не сильно интересует, поэтому введём обозначение для краткости, в качестве "f штрих" с нижним индексом t (f'_tft'). z от w тоже не зависит, поэтому оно сокращается. И тут мы вспоминаем, что h — это функция, которая зависит от w, поэтому мы применяем правило дифференцирования произведения функций. Производная w по w равна единице. Чтобы найти производную предыдущего состояния, раскроем его (получим такое вот выражение). Это выражение очень похоже на то, с чего мы начали. Отлично, давайте тогда просто возьмём и подставим, только индексы, заменим на новые. Мы подставили сюда производную предпоследнего скрытого состояния. Мы можем продолжить эту процедуру дальше до самого первого элемента и начального состояния — тогда получим следующую формулу. Вполне логично — производная w зависит от всех шагов. А ещё, внутри есть произведение всех производных функции активации на нескольких шагах — вот тут-то собака и зарыта.

Вопрос 4

Выполнен

Баллов: 1,00 из 1,00

Допустим, $w = 1.1$, тогда за 100 шагов в последней формуле на предыдущем видео накопится множитель w^{99}. Найдите его значение. Ответ округлите до целого значения. Напомним:

$$\frac{\partial f(w \cdot h_{t-1} + z_t)}{\partial w} = \sum_{i=1}^{t} \left(h_{i-1} \cdot w^{t-i} \prod_{j=i}^{t} f'_j \right).$$

Ответ: 12528

Ваш ответ верный.

Вопрос Инфо

Часто в качестве функции активации используется гиперболический тангенс. Его производная лежит в диапазоне от нуля до единицы. Если мы возьмём много таких чисел, лежащих от нуля до единицы, и перемножим их, мы получим значение, очень близкое к нулю. Это приводит к затуханию градиента — информация с первых шагов почти никак не учитывается при вычислении обновления весов. И в этом случае весь смысл использования рекуррентности исчезает. И, наоборот, если модуль вот этой части больше единицы, то, возводя степень, мы получим очень большое по модулю число, что приводит к переполнению и катастрофическому падению точности вычислений. Со взрывом градиента борются очень просто — сначала честно считают градиентные шаги для всех параметров, а потом, если какой-то градиент по модулю превышает некоторый порог, то он заменяется на значение порога со знаком. То есть слишком большие градиенты просто обрезаются.[1,2] Бороться с затуханием градиента гораздо сложнее — этому посвящено множество работ. Давайте рассмотрим парочку.

[1] https://machinelearningmastery.com/how-to-avoid-exploding-gradients-in-neural-networks-with-gradient-clipping/
[2] http://www.wildml.com/deep-learning-glossary/

Вопрос 5

Выполнен

Баллов: 1,00 из 1,00

Отметьте условия, которые могут привести к затуханию градиента весов рекуррентного блока с ростом количества элементов в последовательности.

Выберите один или несколько ответов:

- [ ] a. Использование функции активации с производной $max \left| \frac{\partial f(x)}{\partial x} \right| > 1$
- [x] b. Малое по модулю значение w
- [x] c. Использование функции активации с производной $\left| \frac{\partial f(x)}{\partial x} \right| < 1$
- [ ] d. Усталость разработчика
- [ ] e. Большое по модулю значение w

Ваш ответ верный.

Вопрос 6

Выполнен

Баллов: 1,00 из 1,00

Допустим, скрытое состояние вычисляется по формуле $h_t = Wh_{t-1}$, где $h_t, h_{t-1} \in \mathbb{R}^{\mathrm{d}}$ - новое состояние и предыдущее, а $W \in \mathbb{R}^{\mathrm{d} \times \mathrm{d}}$ - матрица перехода, Wh - операция матричного произведения матрицы на вектор-столбец.

Можно ли вычислить хотя бы некоторые компоненты вектора h_t до того как все компоненты h_{t-1} будут вычислены?

Выберите один ответ:

- a. Нет
- b. Да

Ваш ответ верный.

Вопрос Инфо

Нашлись ребята, которые решили упростить LSTM, и придумали "gated recurrent unit" или "грушку". Ключевая идея здесь ровно такая же — поток ошибки постоянного объёма. Однако теперь потоком ошибки управляют не два шлюза, а один — по сути, этот шлюз на каждом шаге осуществляет выбор между двумя альтернативами: оставить предыдущее значение, или обновить. Обновление вычисляется похожим образом (тоже с тангенсом). Однако тут внутри есть ещё один шлюз, который управляет чувствительностью вектора "g" к предыдущему значению состояния. В результате, количество параметров сократилось на треть — ну что ж неплохо. На практике GRU и LSTM, в большинстве задач, работают практически одинаково и дают очень близкое качество. То есть сеть упростилась, стала учиться лучше, но при этом осталось достаточно мощной. Кажется, одна из проблем рекурренток частично решена. Однако есть вторая проблема — скорость. По-прежнему, рекуррентки относительно плохо распараллеливаются — в первую очередь из-за вот этих вот зависимостей. В результате каждый элемент вектора скрытого состояния зависит от всего вектора предыдущего состояния. Это заметили авторы ещё одного вида рекурренток, который называется "simple recurrent unit".[1] Они заменили матрицу и матричное произведение на вектор и поэлементное произведение. Таким образом, теперь можно параллельно вычислять значение разных элементов рекуррентных векторов в рамках одного шага. С учётом того, что, на практике, размерность этих векторов составляет от нескольких сотен до пары тысяч, это даёт очень хороший прирост производительности. Кроме того, в этой сети ещё в два раза меньше параметров, поэтому она ещё меньше переобучается. Да, она слабее, чем LSTM, но, на практике, за счёт более простой структуры, процесс обучения идёт более эффективно и качество решения задачи остаётся прежним, или может даже немного улучшится. В этом видео мы разобрались, что такое рекуррентные нейросети, какие сложности в процессе их обучения возникают — в первую очередь это взрыв градиента (тогда градиенты просто отсекают[2]) и затухание градиента. Для борьбы со второй проблемой придумывают специальные архитектуры. Самая старая, проверенная и популярная — LSTM. Относительно недавно был предложен облегченный вариант — "грушка". И ещё более недавно придумали, как можно упростить и, при этом, ускорить рекуррентки — примером такой работы является simple recurrent unit.

[1] Lei T. et al. Simple recurrent units for highly parallelizable recurrence //arXiv preprint arXiv:1709.02755. – 2017. (https://arxiv.org/abs/1709.02755)
[2] How to Avoid Exploding Gradients With Gradient Clipping https://machinelearningmastery.com/how-to-avoid-exploding-gradients-in-neural-networks-with-gradient-clipping/

Вопрос 7

Выполнен

Баллов: 1,00 из 1,00

Теперь скрытое состояние вычисляется по формуле $h_t = v * h_{t-1}$, где $h_t, h_{t-1} \in \mathbb{R}^d$ - новое состояние и предыдущее, а $v \in \mathbb{R}^d$ - вектор перехода, $v * h$ - операция поэлементного произведения двух векторов.

Отметьте компоненты вектора h_{t-1}, от которых зависит третья компонента вектора $h_t[3]$.

Выберите один или несколько ответов:

- a. $h_{t-1}[2]$
- b. $h_{t-1}[4]$
- c. $h_{t-1}[3]$

Ваш ответ верный.

◄ 3.8 Вопросы по теме: Свёрточные нейросети в обработке текста

Перейти на...

4.2 Моделирование языка ►