Санкт-Петербургский государственный университет
Прикладная математика и информатика
Статистическое моделирование

Глубокое обучение

Рукавишникова Анна
Страшко Владислав
Сандул Михаил

Санкт-Петербург
2019

Оглавление

1. Глубокое обучение в парадигме методов машинного обучения

Глубокое обучение — это совокупность широкого семейства методов машинного обучения, основанных на имитации работы человеческого мозга, а именно, на взаимодействии нейронов. Термин «глубокое обучения» появился ещё в 1980-х, но до середины 2000-х для реализации методов глубокого обучения не хватало вычислительных мощностей существующих компьютеров. Стоит отметить, что часто под данным термином подразумеваются многослойные нейронные сети.

Существенное отличие глубокого обучения от машинного обучения как такового заключается в том, что часть, касаемая feature engineering (feature extraction) в классических методах машинного обучения решается вручную, а в глубоком обучении она решается (может решаться) автоматически.

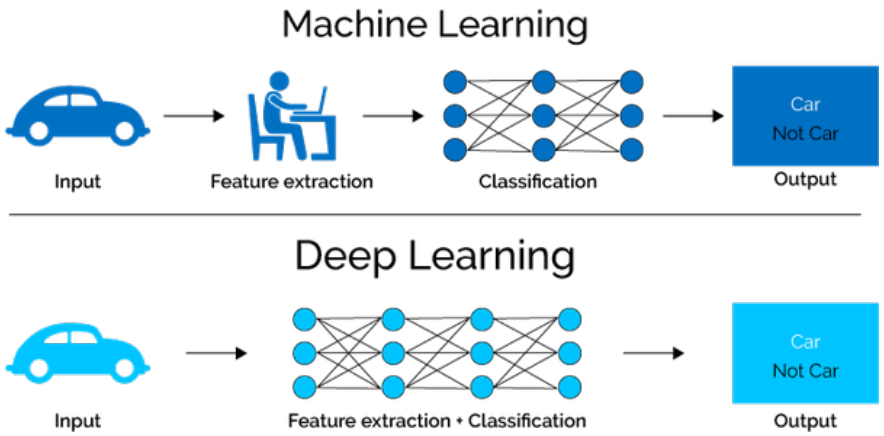

Рис. 1. Отличие между машинным обучением и глубоким обучением

Основными представителями алгоритмов глубокого обучения являются:

- свёрточные нейронные сети (CNN),
- рекуррентные нейронные сети (RNN).

Свёрточные нейронные сети используются, как правило, для анализа изображений, текстов и речи, а рекуррентные — для обработки объектов, характеризующихся наличием последовательной структуры. Например, рекуррентные нейронные сети часто используют для обработки текстов на естественном языке, обработки аудио и видео.

2. Рекуррентные нейронные сети (RNN)

В связ с тем, что свёрточным нейронным сетям был посвящён один из прошлых семинаров, сейчас поподробнее остановимся на рекуррентных нейронных сетях.

Идея, заложенная в создание рекуррентных нейронных сетей состоит в желании имитации человеческой памяти. Иными словами, при обработке тех или иных объектов с последовательной структурой нам бы хотелось, чтобы нейронная сеть сохраняла информацию о своём предыдущем состоянии. Представим, например, что мы хотим классифицировать события, происходящие в фильме. Непонятно, как традиционная нейронная сеть могла бы использовать рассуждения о предыдущих событиях фильма, чтобы получить информацию о последующих. Решить эту проблемы помогают рекуррентные нейронные сети, благодаря тому, что они содержат обратные связи и позволяют сохранять информацию.

2.1. Архитектура рекуррентной нейронной сети

Рассмотрим устройство рекуррентных нейронных сетей. Пусть x_t — входной вектор в момент времени t, h_t — вектор скрытого состояния в момент времени t, y_t — выходной вектор в момент времени t. Стоит заметить, что в некоторых приложениях $y_t \equiv h_t$. Таким образом, схема рекуррентной нейронной сети может быть представлена следующим образом:

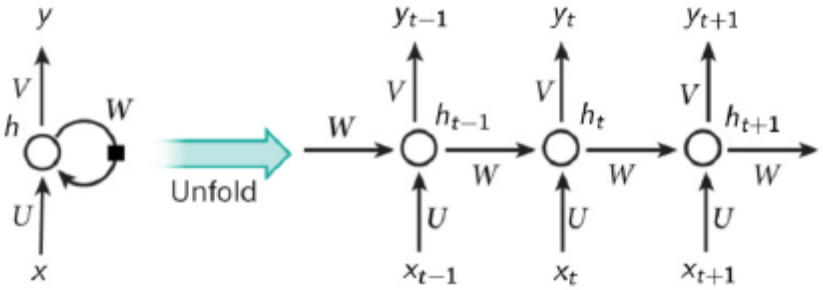

Рис. 2. Схема рекуррентной нейронной сети с развёрткой обратной связи

На схеме выше слева обратная связь схематично обозначена стрелкой, выходящей из фрагмента (модуля) сети со скрытым состоянием h и входящей в него же. Наличие такой обратной связи позволяет передавать информацию от одного шага сети к другому. $\mathbf{W}, \mathbf{U}$ и $\mathbf{V}$ — это матрицы параметров, для которых справедливы следующие

соотношения:

$$\begin{aligned} h_t &= \sigma_h(\mathbf{U}x_t + \mathbf{W}h_{t-1}), \\ y_t &= \sigma_y(\mathbf{V}h_t), \end{aligned}$$

где σ_h и σ_y — функции активации.

Поясним разворачивание (unfolding) обратной связи на примере работы рекуррентной нейронной сети с предложением, положив для определённости $t = 1$. В таком случае x будет являться самим предложением, а x_0 будет первым словом в данном предложении. Это слово мы подаём на вход нейрону. Обработав его, нейрон выдаст значение y_0. Переходя к обработке следующего слова x_1, нейрон получит на вход не только само это слово, но и информацию, полученную от обработки первого слова этого предложения x_0. Тем самым, сеть сможет уловить некоторую взаимосвязь между первым и вторым словом в предложении. В качестве примера можно также рассмотреть временной ряд. В этом случае x — рассматриваемый временной ряд целиком, а h — его неизвестный сигнал. В то время как x_t — вектора вложения данного ряда (размера 1).

2.2. Обучение рекуррентной нейронной сети

В процессе обучения рекуррентной сети стоит задача минимизации следующего функционала:

$$\sum_{t=0}^{T} \mathcal{L}_t(\mathbf{U}, \mathbf{V}, \mathbf{W}) \to \min_{\mathbf{U},\mathbf{V},\mathbf{W}},$$

где $\mathcal{L}_t(\mathbf{U}, \mathbf{V}, \mathbf{W}) = \mathcal{L}(y_t(\mathbf{U}, \mathbf{V}, \mathbf{W}))$ — потеря от предсказания y_t.

Также в отличие от обычных нейронных сетей (не рекуррентных) используется специальный вариант обратного распространения ошибок — Backpropagation Through Time (BPTT):

$$\frac{\partial \mathcal{L}_t}{\partial \mathbf{W}} = \frac{\partial \mathcal{L}_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \sum_{k=0}^{t} \left(\prod_{i=k+1}^{t} \frac{\partial h_i}{\partial h_{i-1}} \right) \frac{\partial h_k}{\partial \mathbf{W}}.$$

Поскольку рекуррентная сеть получается очень глубокой, то становится острой проблема затухающего градиента. Это происходит засчёт того, что подсчитывая градиент в методе обратного распространения ошибки (backpropagation) при обновлении весов, переходя от одного скрытого слоя к другому, у нас появляется множитель в виде некоторой матрицы в очень большой степени. Элементы этой матрицы могут быть

очень близки к нулю, тогда возводя эту матрицу в очень большую степень мы получим вырождение градиента с экспоненциальной скоростью в ноль (затухание). Существует и противоположная крайность такой проблемы, называемая, взрывом градиента. Одним из решений данной проблемы является ограничение частной производной: $\frac{\partial h_i}{\partial h_{i-1}} \to 1$. Именно эта идея реализована в сетях долгой краткосрочной памяти (Long short-term memory, LSTM).

3. Нейронные сети долгой краткосрочной памяти (LSTM)

Мотивацией к созданию LSTM сетей послужило следующее соображение. Рекуррентные нейронные сети хорошо справились бы, например, со следующей задачей: мы хотели бы предсказать последнее слово в предложении «облака плывут по небу» на основании предыдущих. В таком случае для предсказания нам не нужен более широкий контекст, довольно очевидно из предыдущего контекста, что последним будет слово «небо». То есть в таком примере, где дистанция между актуальной информацией и местом, где она понадобилась, невелика, рекуррентные нейронные сети могут обучиться, используя информацию из прошлого. Однако, можно столкнуться со случаем, когда необходимо больше контекста для предсказания последнего слова, как, например, в предложении «Я много лет жил во Франции со своими родителями и старшими братьями, поэтому я бегло говорю по-французски». Ближайшие к последнему слову контекст предполагает, что последним словом будет название языка, но, чтобы установить, какого именно языка, нам необходимо помнить о слове «Франции» из более отдалённого контекста.

Таким образом, известно, что по мере роста расстояния между актуальной информацией и точкой её применения, рекуррентные нейронные сети теряют способность находить смысловые связи. Эта проблема именуется проблемой долговременных зависимостей. Сети LSTM разработаны специально, чтобы избежать проблемы долговременной зависимости.

3.1. Устройство LSTM сети

Детально рассмотрим устройство и принцип работы сетей LSTM. Структура LSTM, как и для любой RNN сети, напоминает цепочку, состоящую из повторяющихся модулей. Однако, в отличие от рекуррентных сетей, где модуль состоит из одного слоя ней-

ронов, LSTM сеть имеет уже четыре взаимодействующих между собой слоя. Ключевым состоянием LSTM сети является вектор состояния ячейки C_t в момент времени t.

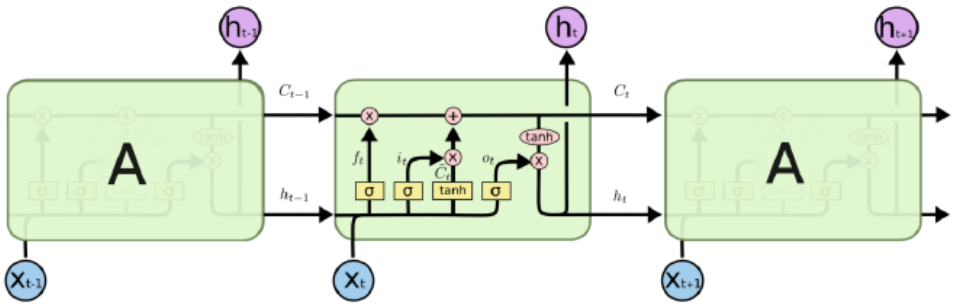

Рис. 3. Структура LSTM сети

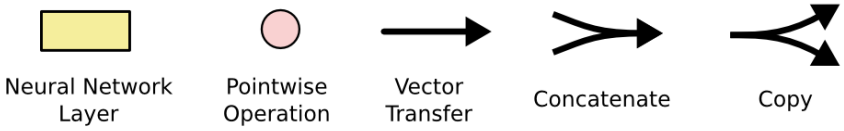

Рис. 4. Условные обозначения

Перейдём к пошаговому разбору алгоритма работы LSTM сети.

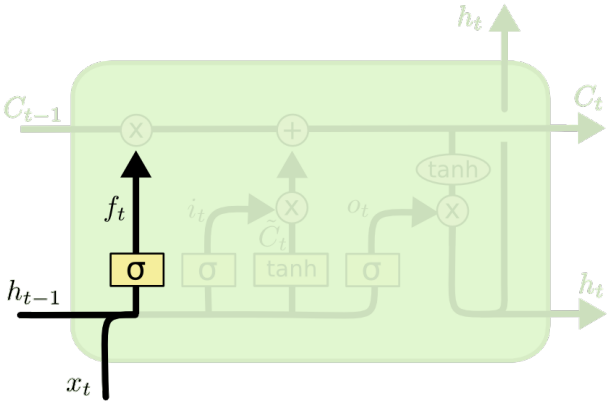

Рис. 5. Слой фильтра забывания

Итак, на первом шаге первому слою LSTM ячейки в момент времени t поступают на вход два значения: h_{t-1} — вектор скрытого состояния с предыдущего момента времени и x_t — входной вектор в момент времени t. Данный слой называется слоем фильтра забывания (forget gate layer), так как на этом шаге определяется, какие координаты век-

тора состояния C_{t-1} нужно запомнить, а какие — нет. Преобразование входных данных осуществляется следующим образом:

$$f_t = \sigma(\mathbf{W}_f[h_{t-1}, x_t] + b_f), \tag{1}$$

где σ — сигмоида, $\mathbf{W}_f$ и b_f — матрица и вектор параметров, соответственно, а операция $[h_{t-1}, x_t]$ означает конкатенацию двух векторов.

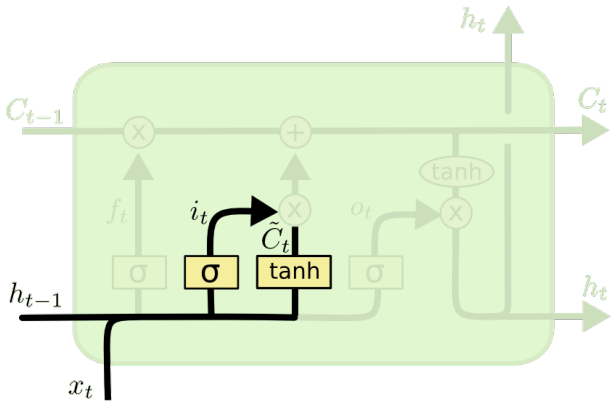

Рис. 6. Слой входного фильтра и tanh-слой

На следующем шаге необходимо решить, какая новая информация будет храниться в состоянии ячейки. Данный шаг будет осуществляться в два этапа. Сначала слой фильтра входных данных (input gate layer) с параметрами $\mathbf{W}_i$, b_i определяет, какие координаты вектора состояния нужно обновить:

$$i_t = \sigma(\mathbf{W}_i \cdot [h_{t-1}, x_t] + b_i). \tag{2}$$

Затем tanh-слой с параметрами $\mathbf{W}_C$, b_C строит вектор новых значений-кандидатов $\widetilde{C}_t$, которые можно добавить в состояние ячейки:

$$\widetilde{C}_t = tanh(\mathbf{W}_C \cdot [h_{t-1}, x_t] + b_C).$$

На третьем шаге новое состояние C_t формируется как смесь старого состояния C_{t-1} с фильтром f_t и вектора значений-кандидатов $\widetilde{C}_t$ с фильтром i_t (используем покоординатное умножение):

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \widetilde{C}_t.$$

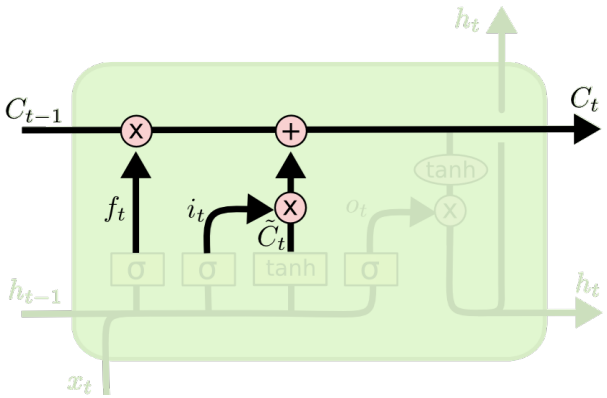

Рис. 7. Обновление состояния ячейки

На этом шаге. очевидно, настраиваемых параметров нет, так как мы используем уже полученные ранее результаты.

Теперь осталось только решить, какую информацию мы хотим получить на выходе. Выходные данные будут основаны на нашем состоянии ячейки, к которому будут применены некоторые фильтры.

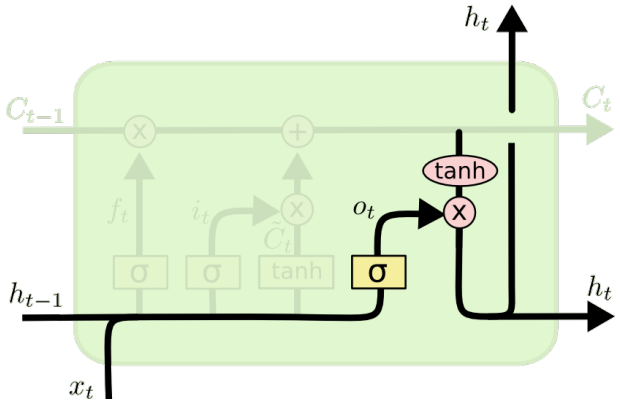

Рис. 8. Обновление состояния ячейки

Сначала мы применяем фильтр выходных данных (output gate) с параметрами $\mathbf{W}_o$, b_o, который решает, какую информацию из состояния ячейки мы будем выводить, то есть какие координаты вектора состояния C_t нужно выдать:

$$o_t = \sigma(\mathbf{W}_o[h_{t-1}, x_t] + b_o). \tag{3}$$

Затем значения состояния ячейки проходят через tanh-слой, чтобы получить на выходе значения из диапазона от -1 до 1 в качестве выходного сигнала h_t, и пере-

множаются с выходными значениями o_t, что позволяет выводить только требуемую информацию:

$$h_t = o_t \cdot tanh(C_t).$$

3.2. Вариации LSTM

Существует множество различных вариаций LSTM сетей. Отличие между ними незначительны, однако, о некоторых из них стоит упомянуть.

Одна из популярных вариаций LSTM, предложенная Герсом и Шмидхубером в работе [1], характеризуется добавлением так называемых «смотровых глазков» или «замочных скважин» (peephole connections). С их помощью слои фильтров могут видеть состояние ячейки.

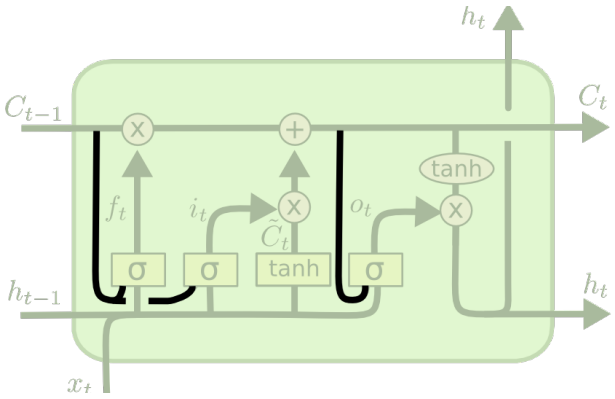

Рис. 9. LSTM с “замочными скважинами”

На схеме выше «замочные скважины’» есть у каждого слоя, но в некоторых случаях они добавляются лишь к некоторым слоям. В такой вариации LSTM число параметров модели увеличивается, а, кроме того, формулы (1), (2) и (3) видоизменяются следующим образом:

$$\begin{aligned} f_t &= \sigma(\mathbf{W}_f \cdot [C_{t-1}, h_{t-1}, x_t] + b_f), \\ i_t &= \sigma(\mathbf{W}_i \cdot [C_{t-1}, h_{t-1}, x_t] + b_i), \\ o_t &= \sigma(\mathbf{W}_o \cdot [C_t, h_{t-1}, x_t] + b_o). \end{aligned}$$

Немного больше от стандартных LSTM сетей отличается следующая модификация — управляемые рекуррентные нейроны (блоки) (Gated Recurrent Unit, GRU). Впервые

эта вариация LSTM была описана в работе [2]. Важной особенностью данной нейронной сети является объединение фильтра забывания (forget gate) и входа (input gate) в один фильтр обновления (update gate). Кроме того, состояние ячейки C_t объединяется со скрытым состоянием h_t. Построенная в результате модель проще, чем стандартная LSTM (она имеет меньше параметров), и популярность ее неуклонно возрастает.

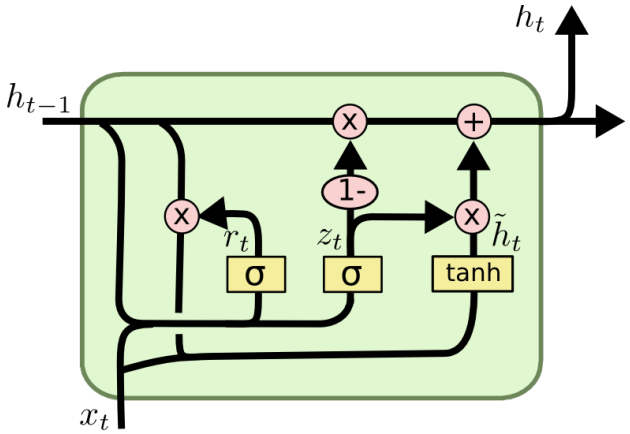

Рис. 10. LSTM: Gated Recurrent Unit (GRU)

Действия фильтров в GRU сети описываются следующими уравнениями:

$$\begin{aligned} z_t &= \sigma(\mathbf{W}_z \cdot [h_{t-1}, x_t]), \\ r_t &= \sigma(\mathbf{W}_r \cdot [h_{t-1}, x_t]), \\ \widetilde{h}_t &= tanh(\mathbf{W} \cdot [r_t \cdot h_{t-1}, x_t]), \\ h_t &= (1 - z_t) \cdot h_{t-1} + z_t \cdot \widetilde{h}_t. \end{aligned}$$

Существует множество других модификаций LSTM сетей, как, например, глубокие управляемые рекуррентные нейронные сети (Depth Gated RNNs), представленные в работе [3]. Сравнение самых популярных вариаций LSTM представлено в работе [4], авторы которой приходят к выводу о том, что все модификации приблизительно одинаковы.

4. Применение RNN к временным рядам

Одной из задач, для которых используются рекуррентные нейронные сети, является задача предсказания временных радов. В таком случае на вход рекуррентной ней-

ронной сети подаются отдельно точки временного ряда, и для обработки используется архитектура RNN сетей, называемая one to many.

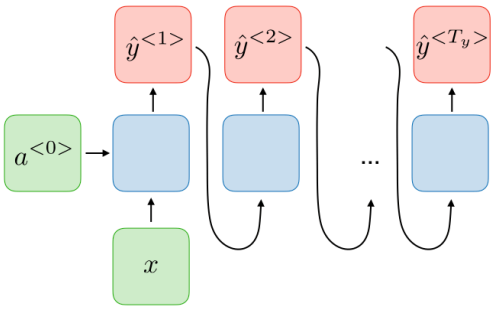

Рис. 11. RNN: архитектура one to many

Здесь x — последняя точка временного ряда, $a^{<0>}$ — скрытое состояние сети, полученное на предыдущем шаге (в прошлый момент времени), и мы строим прогноз на T_y точек вперёд. Для получения каждого следующего прогноза мы используем полученное предсказание на предыдущем шаге. Если бы мы хотели сделать прогноз только на одну точку вперёд, то, как частный случай, мы бы использовали структуру one to one.

Список литературы

1. Gers F. A., Schmidhuber J. Recurrent nets that time and count // Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium. — Vol. 3. — 2000. — July. — P. 189–194.
2. Learning phrase representations using RNN encoder–decoder for statistical machine translation / Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre et al. // Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). — Doha, Qatar : Association for Computational Linguistics, 2014. — October. — P. 1724–1734. — Access mode: https://www.aclweb.org/anthology/D14-1179'.
3. Depth-gated LSTM / Kaisheng Yao, Trevor Cohn, Katerina Vylomova et al. // CoRR. — 2015. — Vol. abs/1508.03790. — 1508.03790.
4. Jozefowicz R., Zaremba W., Sutskever I. An empirical exploration of recurrent network architectures // Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37. — ICML'15. — Lille, France : JMLR.org, 2015. — P. 2342–2350. — Access mode: http://dl.acm.org/citation.cfm?id=3045118.3045367.