Машинное обучение в экономике

Рекуррентные нейронные сети

Потанин Богдан Станиславович

доцент, научный сотрудник, кандидат экономических наук

2023–2024

Рекуррентная нейронная сеть

Постановка проблемы

- Многие данные удобно рассматривать как последовательность связанных между собой значений.
- Например, в экономике исследователи часто анализируют временные ряды, чтобы спрогнозировать динамику определенных показателей: цену акций, инфляцию и т.д.
- **Проблема** – архитектура классических нейронных сетей не учитывает связь между последовательными значениями целевой переменной.
- **Решение** – вернуться к классическим моделям временных рядов, таким как, например, ARIMA.
- **Проблема** – классические эконометрические модели временных рядов часто накладывают достаточно сильные предпосылки и иногда оказываются недостаточно гибкими для того, чтобы описывать очень сложные зависимости.
- **Решение** – применить рекуррентные нейронные сети.

- Вспомним о классических способах анализа временных рядов на примере авторегрессионного процесса AR(1) с экзогенными регрессорами:

$$y_t = \mu + \alpha y_{t-1} + x_t \beta + \varepsilon_t$$

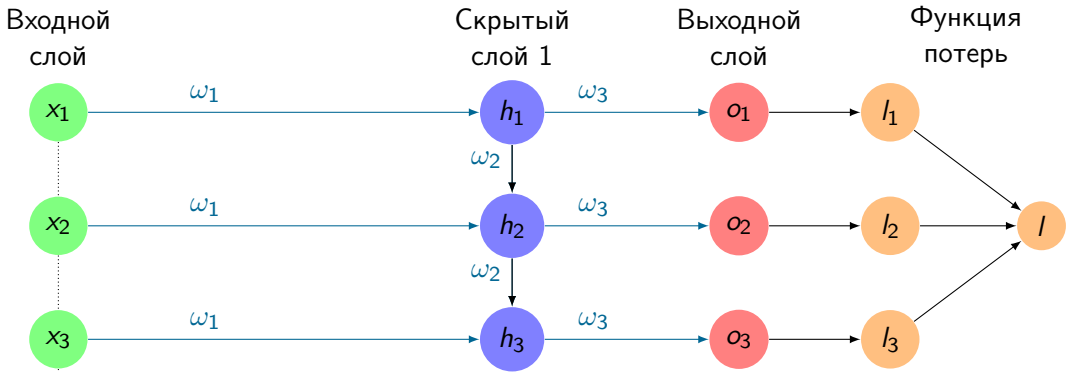
где:

- y_t - значение целевой (зависимой) переменной в момент времени t.
- x_t – вектор строка значений признаков (независимых переменных) в момент времени t.
- μ - константа.
- α - авторегрессионный коэффициент.
- β - вектор столбец коэффициентов признаков (независимых переменных).
- ε_t – случайная ошибка (шок) в момент времени t. Как правило предполагается, что случайные ошибки независимы и одинаково распределены.

- **Проблема** – в некоторых случаях форма, в которой признаки x_t и лаговые значения y_{t-1} влияют на текущее значение целевой переменной y_t может существенно отличаться от линейной, что осложняет использование классических методов (но все еще возможно непараметрическое оценивание с помощью, например, сплайнов).

Рекуррентная нейронная сеть

Графическая репрезентация идеи на простом примере



$x_t = (x_{1t}, ..., x_{tm})$ - признаки в момент времени t

$h_t = h(s_t) = h(x_t\omega_1^T + h_{t-1}\omega_2^T + b)$

h - функция активации, b - смещения, ω_j - веса

- Обучение рекуррентных нейронных сетей происходит по аналогии с классическими:
  - Выбирается структура нейросети: признаки, функции активации, число слоев, количество нейронов и т.д.
  - С помощью метода численной оптимизации, например, мини-пакетного градиентного спуска, функция потерь минимизируется по параметрам (веса и смещение) нейросети. При этом градиент рассчитывается с помощью метода обратного распространения ошибки (backpropagation).
  - При необходимости изменяется структура нейросети, например, исходя из результатов кросс-валидации.
- **Проблема** – градиенты весов, с которыми значения скрытых слоев одних периодов времени входят в скрытые слои других периодов времени часто могут взрываться или затухать. Например, для рассмотренной ранее реккурентной нейросети с одним скрытым слоем производная по ω_2 (для простоты рассмотрим скалярный вес) считается рекурсивно и имеет вид:

$$\frac{\partial l_t}{\partial \omega_2} = \frac{\partial l_t}{\partial o_t}\frac{\partial o_t}{\partial h_t}\frac{\partial h_t}{\partial s_t}\underbrace{\left(h_{t-1} + \omega_2 \frac{\partial h_{t-1}}{\partial \omega_2}\right)}_{\partial s_t / \partial \omega_2} \qquad \frac{\partial h_{t-1}}{\partial \omega_2} = \frac{\partial h_{t-1}}{\partial s_{t-1}}\underbrace{\left(h_{t-2} + \omega_2 \frac{\partial h_{t-2}}{\partial \omega_2}\right)}_{\partial s_{t-1} / \partial \omega_2}$$

Долгая краткосрочная память (Long short-term memory - LSTM)

Описание слоя

- Введем специальные обозначения для следующих функций активации:

$$\textbf{Сигмоида: } \sigma(s) = \frac{1}{1+e^{-s}} \in (0,1) \qquad \textbf{Гиперболический тангенс: } \tanh(s) = \frac{e^{s}-e^{-s}}{e^{s}+e^{-s}} \in (-1,1)$$

- Определим следующие векторы, именуемые **фильтрами** (вентилями или вратами):

$$f_t = \sigma(x_t\omega_{1f}^T + o_{t-1}\omega_{2f}^T + b_f) \qquad \textbf{забывания} \text{ (forget gate)}$$
$$i_t = \sigma(x_t\omega_{1i}^T + o_{t-1}\omega_{2i}^T + b_i) \qquad \textbf{входа} \text{ (input gate)}$$
$$\tilde{o}_t = \sigma(x_t\omega_{1o}^T + o_{t-1}\omega_{2o}^T + b_o) \qquad \textbf{выхода} \text{ (output gate)}$$

Где ω_* и β_* обозначают матрицы весов и векторы строки смещений соответственно.

- Память хранится в **векторе состояний** C_t, используемом для получения значений выходного слоя:

$$C_t = \underbrace{\tanh\left(x_t\omega_{1a}^T + o_{t-1}\omega_{2a}^T + b_f\right)}_{a_t} \odot i_t + f_t \odot C_{t-1} \qquad o_t = \tanh(C_t) \odot \tilde{o}_t \qquad C_0 = \tilde{o}_0 = (0,...,0)$$

Где $\odot$ означает поэлементное перемножение матриц (в частности векторов).

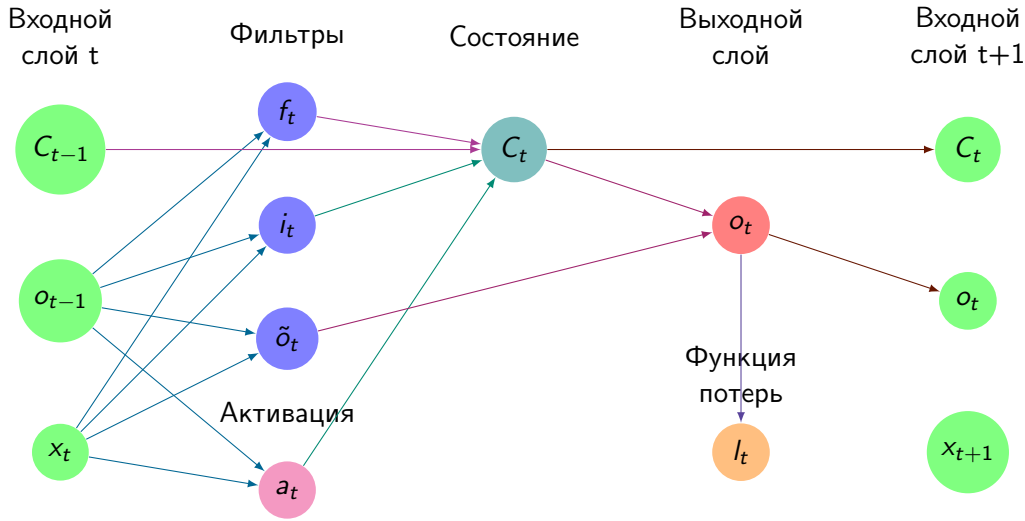
Долгая краткосрочная память (Long short-term memory - LSTM)

Интуиция

- Иногда вектор состояний C_t интерпретируется как долгосрочная память, а вектор значений выходного слоя o_{t-1} как краткосрочная память.
- Вследствие применения сигмоидной функции значения фильтров забывания f_t, входа i_t и выхода $\tilde{o}_t$ находятся в диапазоне от 0 до 1.
- В произведении $f_t \odot C_{t-1}$, вопреки названию, фильтр забывания f_t интерпретируется как доля сохраняемой долгосрочной памяти C_t.
- В произведении $a_t \odot i_t$ фильтр входа i_t отвечает за долю новой информации a_t, используемой для обновления долгосрочной памяти C_t.
- В произведении $C_t \odot \tilde{o}_t$ фильтр выхода $\tilde{o}_t$ определяет вклад долгосрочной памяти C_t в краткосрочную o_t.
- Долгая краткосрочная-память позволяет использовать информацию о событиях, которые произошли достаточно давно, но при этом избегать проблемы затухающих и взрывающихся градиентов.

Долгая краткосрочная память (Long short-term memory - LSTM)

Графическая репрезентация идеи



- При прогнозировании временных рядов в качестве признаков часто используются m лаговых значений целевой переменной, например:

$$x_t = (y_{t-1}, y_{t-2}, ..., y_{t-m})$$

- Также, можно включать и дополнительные признаки, например, прогнозировать инфляцию в зависимости от ее лаговых значений и лаговых значений выпуска.
- **Важно** – если модель используется для прогнозирования на q периодов вперед, то признаки x_t нужно брать с лагом не менее q по отношению к целевой переменной. В противном случае мы не сможем прогнозировать в момент времени $t + q$, поскольку не будем знать значений признаков в этот же момент времени.
- Например, если в качестве признака для прогнозирования инфляции y_t на один период вперед $q = 1$ используется выпуск z_t, то необходимо положить $x_t = z_{t-1}$. Если бы мы положили $x_t = z_t$, то смогли бы прогнозировать инфляцию в период времени $t + 1$ лишь зная выпуск в этот же период времени, что бессмысленно, поскольку к моменту времени $t + 1$ мы будем знать не только выпуск z_{t+1}, но и саму настоящую инфляцию y_{t+1}.

- Часто исследователь заинтересован в моделировании совместной динамики нескольких показателей, например, цен двух акций:

$$y_t = \begin{bmatrix} y_{1t} & y_{2t} \end{bmatrix}^T$$

- В классическом эконометрическом анализе временных рядов совместная динамика моделируется с помощью многомерных моделей временных рядов, таки как, например, векторные авторегрессионные модели VAR.
- В качестве альтернативы можно использовать, например, рекуррентные нейронные сети, у которых выход o_t является не числом, а вектором:

$$o_t = \begin{bmatrix} o_{1t} & o_{2t} \end{bmatrix}^T$$

- Таким образом, выходной слой состоит из нескольких нейронов и функция потерь зависит от них всех. Иногда ее считают с помощью усреднения двух функций потерь, посчитанных отдельно по каждому выходу:

$$\mathrm{L}(y_t, o_t) = \frac{\mathrm{L}_1(y_{1t}, o_{1t}) + \mathrm{L}_2(y_{2t}, o_{2t})}{2}$$

- **Важно** – веса не обязательно должны быть равными, часто их имеет смысл корректировать на единицы измерения и относительную важность прогнозов каждого из показателей.

- Рекуррентные нейронные сети можно использовать не только непосредственно для анализа временных рядов, но и для создания новых признаков x_t, которые затем могут быть полезны при прогнозировании.
- **Анализ сантиментов** позволяет охарактеризовать тональность текста и может быть использован для создания новых признаков:
  - Является ли отзыв на фильм положительным или отрицательным.
  - Автор статьи склоняется скорее к позитивному или негативному сценарию экономического развития.
  - Новости в день t скорее можно охарактеризовать как спокойные или тревожные.
- Например, с помощью анализа сантиментов можно оценить тревожность новостей news_t и использовать соответствующую переменную в качестве признака в реккурентной нейронной сети $x_t = \text{news}_{t-1}$ или в рамках классической модели, такой как AR(1):

$$y_t = \mu + \alpha y_{t-1} + \beta \text{news}_{t-1} + \varepsilon_t$$

- Представим, что у нас есть **тексты**, которые мы собираемся анализировать. Например, отзывы о фильмах. Совокупность этих текстов именуется **корпусом** (corpus).
- Обычно входящие в корпус тексты предварительно обрабатываются. Например, для облегчения последующего анализа часто убираются знаки препинания.
- Тексты корпуса **токенизируются**, то есть разбиваются на более мелкие части, например, буквы, слова, части слов или фразы. Эти части именуется **токенами** и обычно представляются в форме целых числен.
- Совокупность всех токенов именуется **словарем**.
- Для токенизации используются различные алгоритмы, например, кодирование пар байтов **Byte Pair Encoding (BPE)**.

- **Алгоритм BPE** – изначально все буквы (включая пробелы) воспринимаются как токены. Далее пары наиболее часто встречающихся токенов объединяются в новые токены. Процесс повторяется произвольное число раз.
- Представим, что наш корпус состоит из двух текстов:

Текст 1	Текст 2
карл у клары украл кораллы	клара у карла украла кларнет
кХлу клХыу крал кораллы	клХау кХлау крала клХнет
кХлYклХыYкрал кораллы	клХаYкХлаYкрала клХнет
кХлZлХыZрал кораллы	клХаZХлаZрала клХнет
кХлZRыZрал кораллы	кRaZХлаZрала кRнет

Словарь = (к, Х, л, Z, R, ы, р, а, о, R, н, е, т, ' '), где ' ' – пробел.

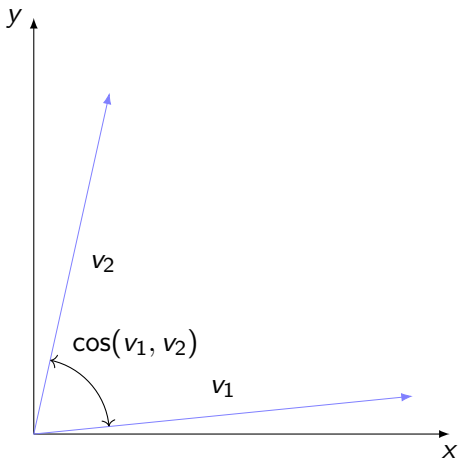
- Чем больше итераций совершает алгоритм, тем меньшим числом токенов описывается каждый текст, но тем больше общее число токенов. Поэтому выбор оптимальной длины словаря часто воспринимается как гиперпараметр.

- Для того, чтобы использовать токены в качестве признаков, они должны быть информативны, то есть содержать какую-то полезную информацию о текстах, для работы с которыми будут использоваться.
- **Проблема** – изначально токены, как правило, представляют между собой обычные целые числа.
- **Решение** – осуществить **эмбеддинг**, позволяющий закодировать токены в форме векторов.
- Близость между этими векторами отражает семантическое сходство между токенами.
- Близость между токенами, закодированными d-мерными векторами x и y, может быть посчитана, например, с помощью **косинусной близости**:

$$\text{cs}(x, y) = \sum_{i=1}^{d} x_i y_i / \sqrt{\sum_{i=1}^{d} x_i^2 \sum_{i=1}^{d} y_i^2} \in (-1, 1) \qquad \text{чем больше значение, тем выше близость}$$

- Например, токены кот, кошка и дверь могут быть закодированы векторами $x = (3, 4)$, $y = (4, 3)$ и $z = (5, 0)$ соответственно, откуда, в соответствии с косинусной близостью, токены кот и кошка более схожи, чем токены кот и дверь:

$$\text{cs(x, y)} = (3 \times 4 + 4 \times 3)/(\sqrt{(3^2 + 4^2) \times (4^2 + 3^2)}) = 0.96 \quad \text{cs}(x, z) = 0.6$$


y
v_2
$\cos(v_1, v_2)$
v_1
x

Анализ текстов

Непрерывный мешок со словами

- Группа популярных подходов к эмбеддингу именуется **word2vec**. Рассмотрим один из них, называемый **непрерывным мешком со словами (continuous bag of words)**.
- Обозначим через $T = \{t_1, ..., t_n\}$ словарь из n токенов t_i, каждому из которых сопоставлено векторное представление v_i (строка). Для простоты представим, что корпус состоит из одного текста $C = (w_1, ..., w_m)$, включающего m последовательных токенов w_i. Пусть $v(w_i) = v_k$, где k такое, что $w_i = t_k$.
- **Идея** – максимизируем по всем v_i следующую функцию правдоподобия:

$$\prod_{i \in \{q+1, \dots, m-q\}} P(w_i | w_{i-q}, ...w_{i-1}, w_{i+1}, .., w_{i+q}) = \prod_{i=1}^{m} \frac{e^{v(w_i)s(w_i)^T}}{\sum\limits_{j=1}^{m} e^{v(w_j)s(w_j)^T}}$$

$$s(w_i) = v(w_{i-q}) + ... + v(w_{i-1}) + v(w_{i+1}) + ... + v(w_{i+q})$$

- **Интуиция** – мы пытаемся спрогнозировать i-е слово, используя информацию о q словах слева от него и q словах справа.
- Например, если корпус имеет вид $C = (\text{Я}, \text{очень}, \text{сильно}, \text{люблю}, \text{математику})$, токены являются словами $V = \{\text{сильно}, \text{Я}, \text{математику}, \text{очень}, \text{люблю}\}$, то при $q = 1$ эти вероятности будут, например, иметь вид:

$$P(w_3 = \text{сильно} | w_2 = \text{очень}, w_4 = \text{люблю})$$

Вероятность, что третьим словом будет 'сильно', если ближайшие к нему (второе и четвертое) слова это 'очень' и 'люблю'.

- **Важно** – эти условные вероятности являются функциями от векторных представлений слов v_i. Максимизировав произведение условных вероятностей по всем v_i мы получаем эмбеддинг.