Прогнозирование временных рядов на примере цен на акции Google с помощью нейронной сети LSTM

Алексей Сек, БЭК182

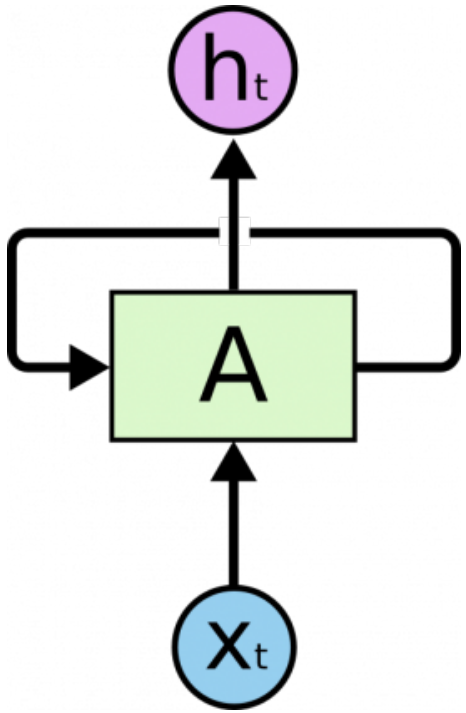
Рекуррентные нейронные сети (1/2)

Простые нейронные сети, как, например, MLP – хороши для многих задач, но для предсказания временных рядов как нельзя лучше подходят именно рекуррентные нейронные сети

Основная идея заключается в том, что НС может «запоминать», сохранять информацию о предыдущих этапах.

Это максимально близко к тому, как думают люди: данные, которые мы накапливаем в течение жизни – влияют на наши последующие размышления

Пример части рекуррентной НС



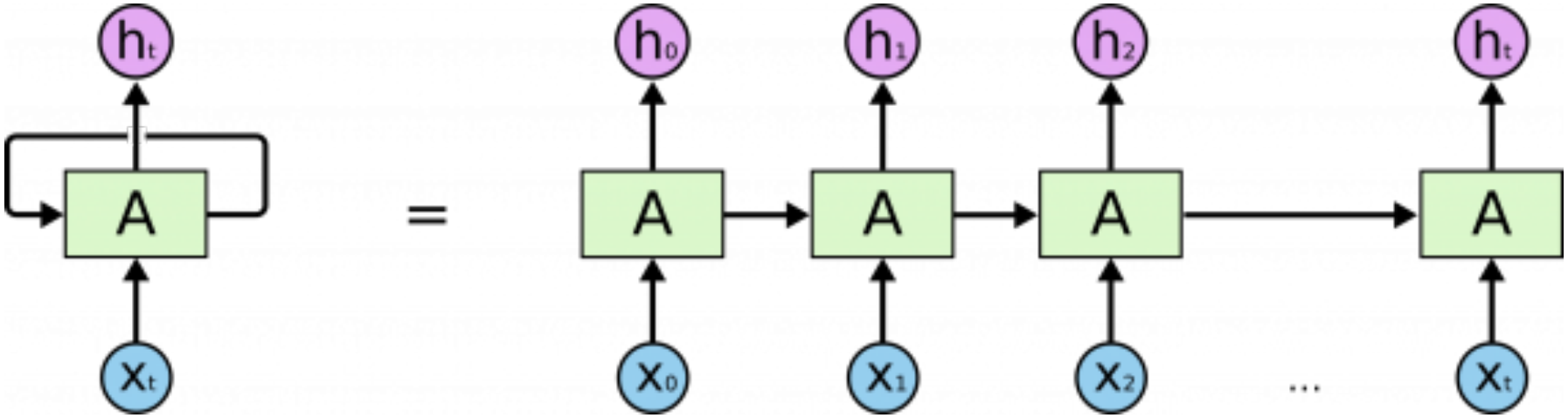
Выходной сигнал

Часть НС

Входной сигнал

Рекуррентные нейронные сети (2/2)

Если «развернуть» цикл, то понять устройство будет еще проще

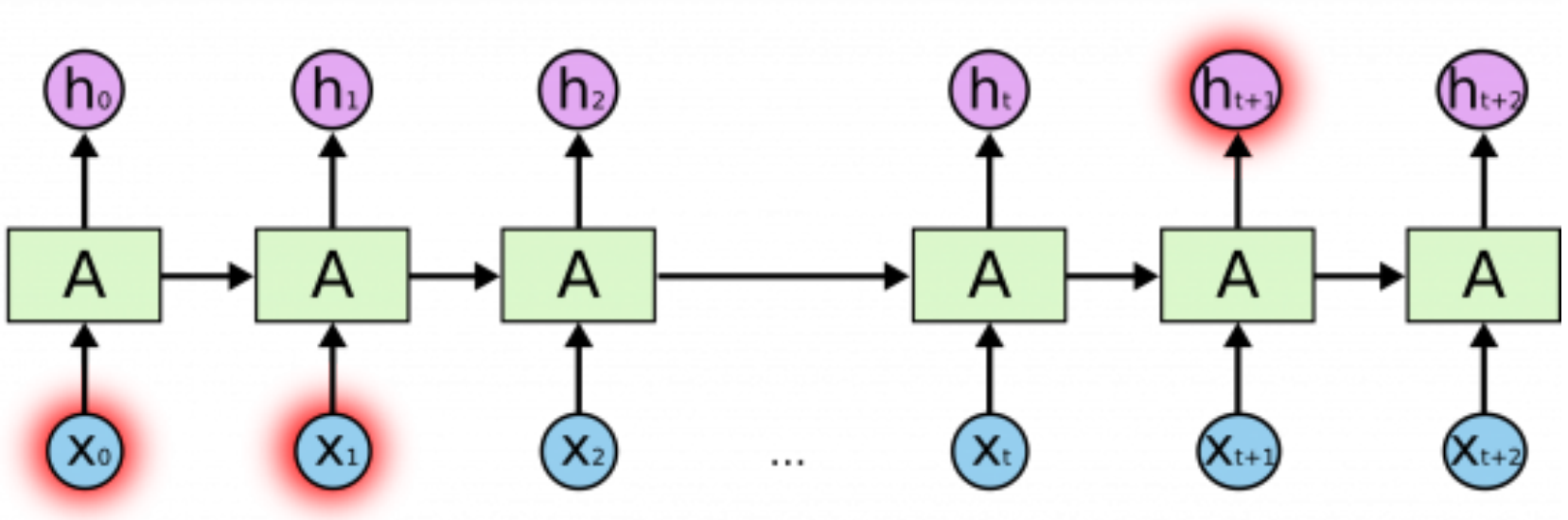


Получается, что эта нейронная сеть состоит из одинаковых повторяющихся частей, каждая из которых (кроме первой) – получает сигнал от предыдущей – это и есть та реализация использования информации, полученной на предыдущих этапах

Рекуррентная нейронная сеть используют полученную ранее информацию для решения последующих задач, например, уже полученные временные ряды для прогнозирования

Проблема рекуррентных нейронных сетей

Чем дальше оказывается следующий компонент – тем меньшее влияние на него оказывают компоненты, которые были в начале



Иными словами - по мере увеличения этого разрыва РНС теряют связь между информацией.

Почему это важно? Зачастую нам для обработки данных важно знать не только самые последние события, но также и те, которые были задолго до предсказываемого

В теории рекуррентные НС способны справиться с такими «долгосрочными зависимостями». Исследователь может тщательно подобрать параметры сети для устранения этой проблемы.

Но на деле такое практически невозможно, что было доказано в 90-е годы 20 века

Рекуррентная нейронная сеть LSTM

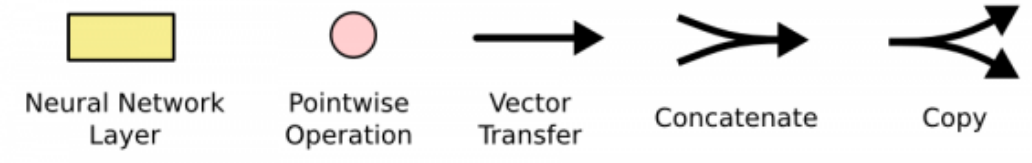
LSTM хороша тем, что у нее отсутствует описанная ранее проблема ☺

LSTM специально разработаны для устранения проблемы долгосрочной зависимости. Их специализация — запоминание информации в течение длительных периодов времени, поэтому их практически не нужно обучать!

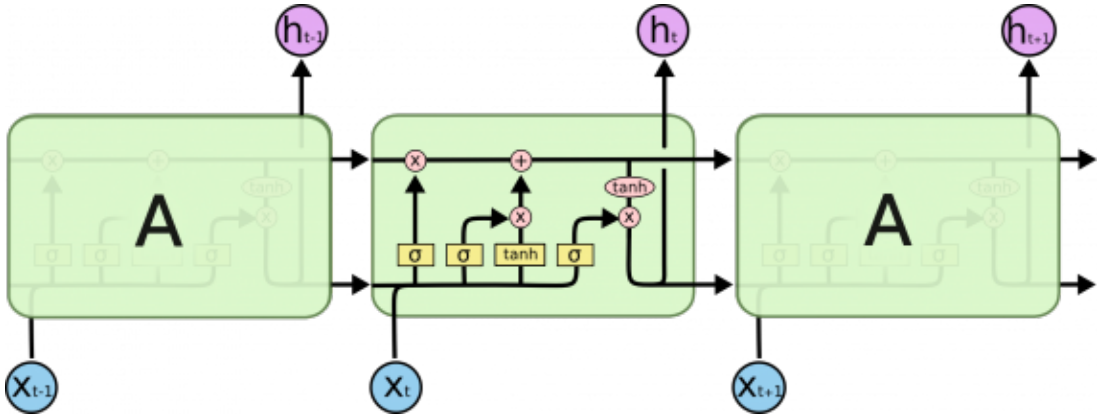
Все рекуррентные нейронные сети имеют форму цепочки повторяющихся модулей нейронной сети. В стандартных РНС этот повторяющийся модуль имеет простую структуру, например, один слой

Повторяющийся модуль LSTM в свою очередь устроен сложнее – он состоит из четырех взаимодействующих слоев

Условные обозначения



Структура LSTM



Принцип работы LSTM

Ключевым понятием LSTM является состояние ячейки: горизонтальная линия, проходящая через верхнюю часть диаграммы.

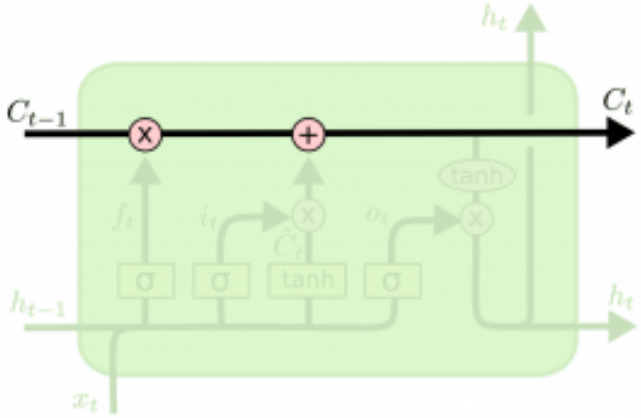
Состояние ячейки напоминает конвейерную ленту. Оно проходит через всю цепочку, подвергаясь незначительным линейным преобразованиям.

В LSTM уменьшает или увеличивает количество информации в состоянии ячейки, в зависимости от потребностей. Для этого используются тщательно настраиваемые структуры, называемые гейтами.

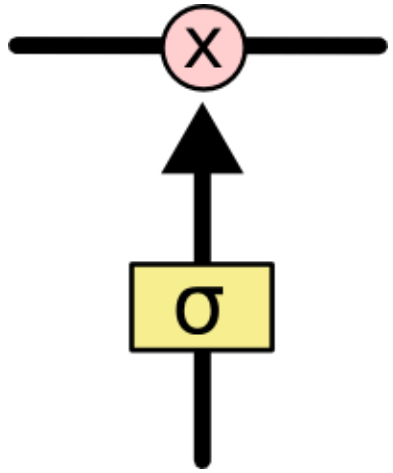
Гейт — это «ворота», пропускающие или не пропускающие информацию. Гейты состоят из сигмовидного слоя нейронной сети и операции поточечного умножения.

На выходе сигмовидного слоя выдаются числа от нуля до единицы, определяя, сколько процентов каждой единицы информации пропустить дальше. Значение «0» означает «не пропустить ничего», значение «1» — «пропустить все».

Состояние ячейки



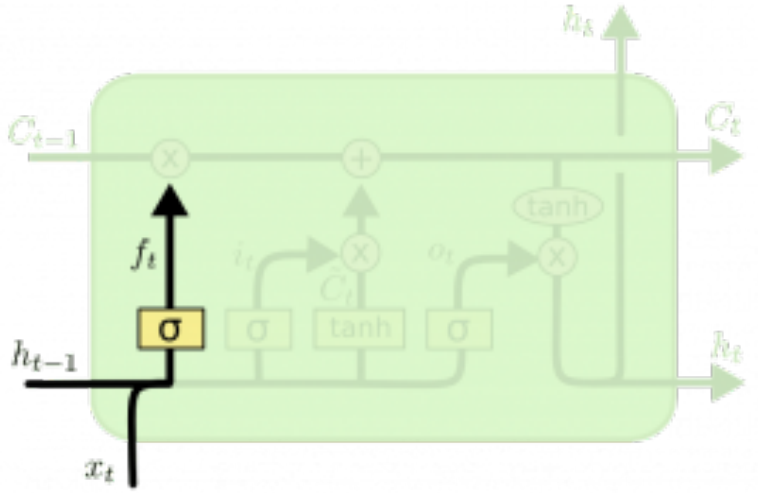
Гейт в LSTM



Этапы работы нейронной сети LSTM (1/3)

LSTM имеет три таких гейта для контроля состояния ячейки.

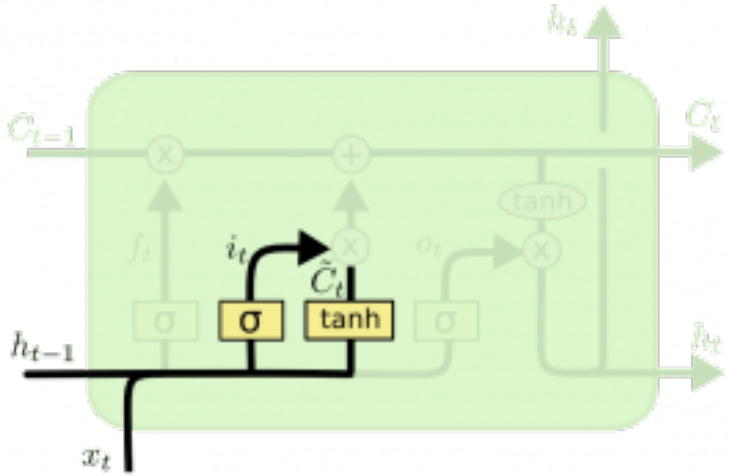
Слой утраты:
На первом этапе LSTM нужно решить, какую информацию мы собираемся выбросить из состояния ячейки. Это решение принимается сигмовидным слоем, называемым «слоем гейта утраты». Он получает на вход **h** и **x** и выдает число от 0 до 1 для каждого номера в состоянии ячейки **C**. **1** означает «полностью сохранить», а **0** — «полностью удалить».



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Этапы работы нейронной сети LSTM (2/3)

Слой сохранения:
На следующем шаге нужно решить, какую новую информацию сохранить в состоянии ячейки. Разобьем процесс на две части. Сначала сигмоидный слой, называемый «слоем гейта входа», решает, какие значения требуется обновить. Затем слой tanh создает вектор новых значений-кандидатов C, которые добавляются в состояние. На следующем шаге мы объединим эти два значения для обновления состояния.



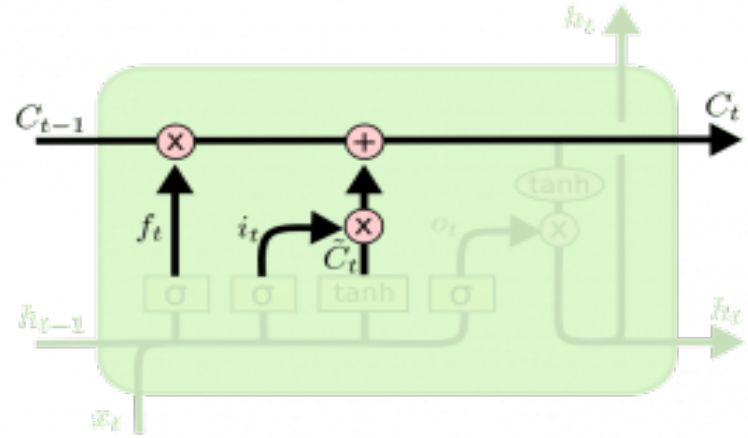
Идея: информация могла стать значимой, поэтому ее необходимо сохранить, для этого работает данный слой. Например, чтобы заменить ту информацию, которую «убрали» на слое утраты

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

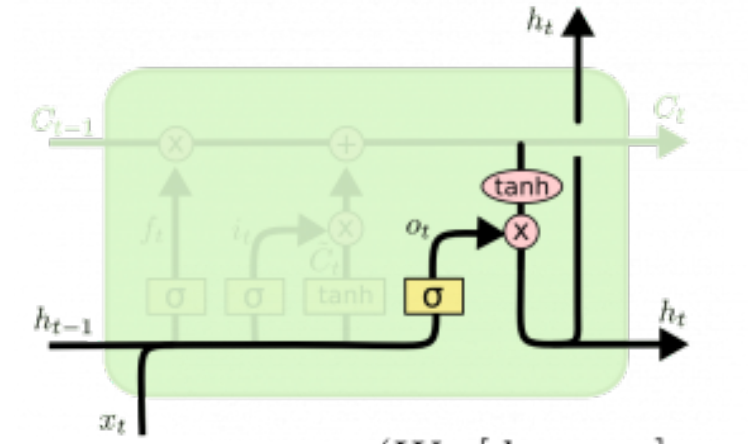
Этапы работы нейронной сети LSTM (3/3)

Новое состояние:
Теперь обновим предыдущее состояние ячейки для получения нового состояния C. Способ обновления выбран, теперь реализуем само обновление. Умножим старое состояние на f, теряя информацию, которую решили забыть. Затем добавляем i*C. Это новые значения кандидатов, масштабируемые в зависимости от того, как мы решили обновить каждое значение состояния.



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Вывод:
Наконец, нужно решить, что хотим получить на выходе. Результат будет являться отфильтрованным состоянием ячейки. Сначала запускаем сигмоидный слой, который решает, какие части состояния ячейки выводить. Затем пропускаем состояние ячейки через tanh (чтобы разместить все значения в интервале [-1, 1]) и умножаем его на выходной сигнал сигмовидного гейта.

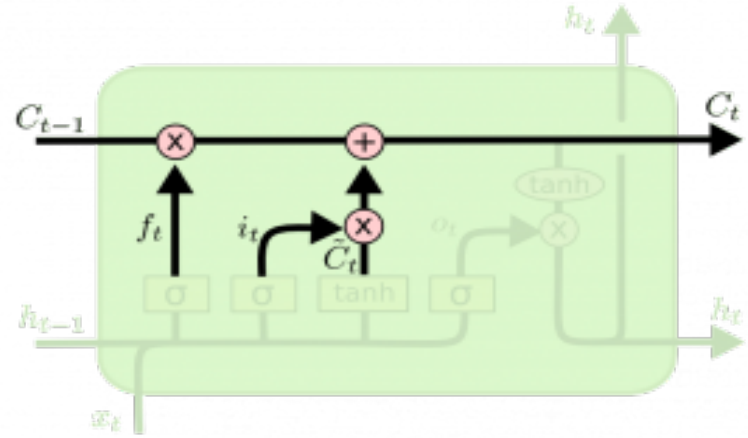


$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

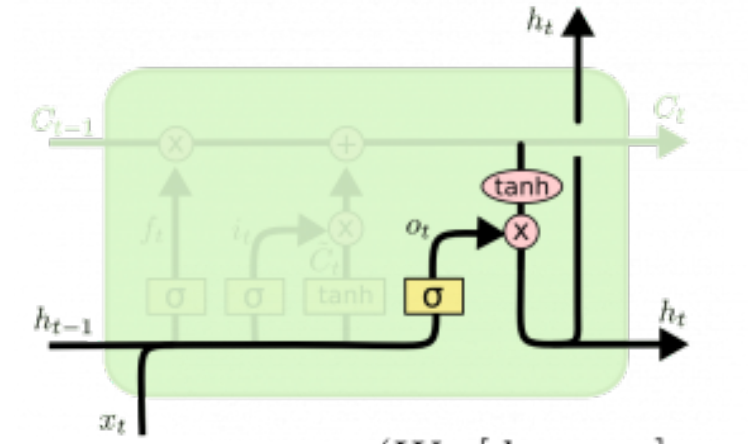
Этапы работы нейронной сети LSTM (3/3)

Новое состояние:
Теперь обновим предыдущее состояние ячейки для получения нового состояния C. Способ обновления выбран, теперь реализуем само обновление. Умножим старое состояние на f, теряя информацию, которую решили забыть. Затем добавляем i*C. Это новые значения кандидатов, масштабируемые в зависимости от того, как мы решили обновить каждое значение состояния.



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Вывод:
Наконец, нужно решить, что хотим получить на выходе. Результат будет являться отфильтрованным состоянием ячейки. Сначала запускаем сигмоидный слой, который решает, какие части состояния ячейки выводить. Затем пропускаем состояние ячейки через tanh (чтобы разместить все значения в интервале [-1, 1]) и умножаем его на выходной сигнал сигмовидного гейта.



$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

Пример решения задачи с помощью LSTM

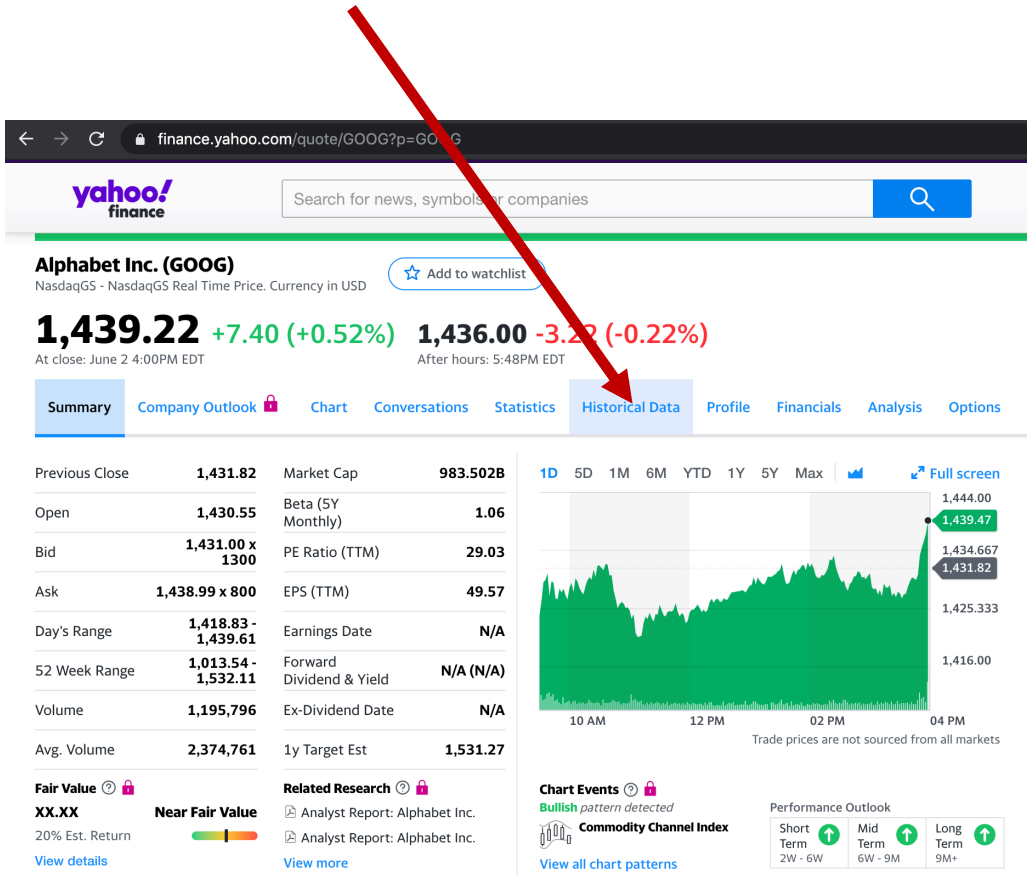
Задача:
Есть массив данных с котировками акций Alphabet (Google) за последние 5 лет

Необходимо на основе имеющихся данных предсказать цену следующего дня

Идея решения:

- Разбить данные на «окна» по 30 дней каждое, предсказываемая величина – цена закрытия 31 дня
- «Окно» мы сдвигаем каждый раз на один день
- В результате получаем отличный датасет для тренировки нашей рекуррентной НС LSTM
- Предсказывать будем не по нескольким признакам, а только по предыдущим значениям

Где взять данные:



Пример решения задачи с помощью LSTM

Основные библиотеки для решения:

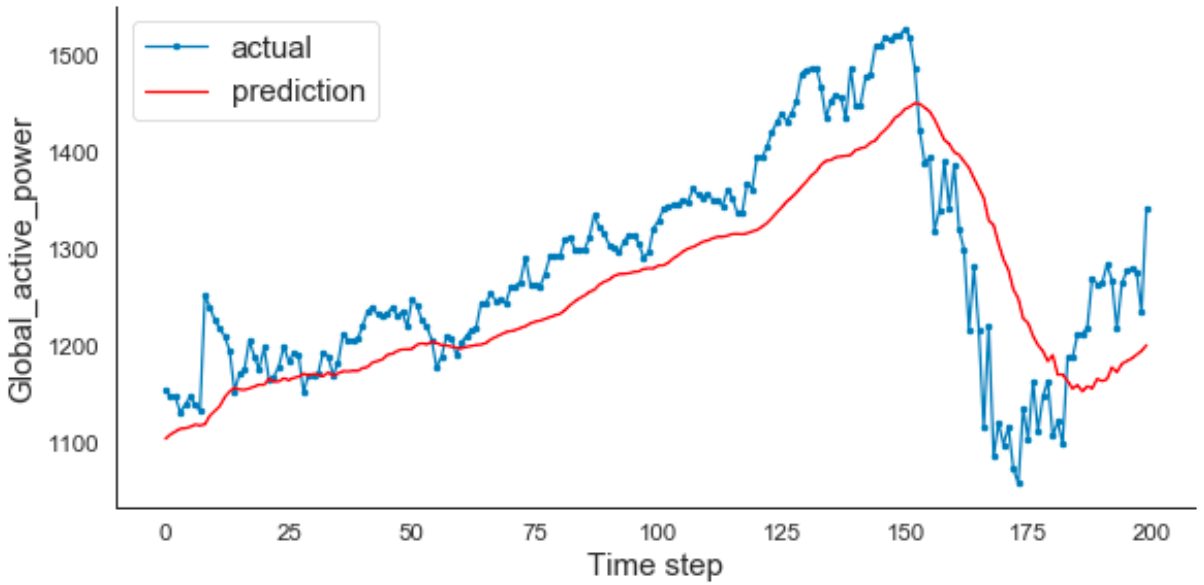
1. Tensorflow и Keras
2. Scickit-learn
3. Pandas, Numpy
4. Matplotlib, Seaborn

Ссылка на решение задачи в python:
https://github.com/AlexeySek/HW-Neural-Networks

Параметры сети:

- 30 нейронов на входе – прогноз на основе предыдущих 30 дней
- 1 нейрон на выходе – предсказываем только один 31-й день
- Функция ошибки – MSE
- Оптимизатор – ADAM (Более подробно: https://habr.com/en/post/318970/)
- 20 итераций (эпох) обучения

Результаты обучения



Вывод: PHC LSTM отлично подходит для решения задачи прогнозирования временных рядов

Мы рассмотрели простой случай, где используются только предыдущие значения параметра, возможно усложнить модель и делать предсказания на основе нескольких факторов, а не одного

Источники

- https://neurohive.io/ru/osnovy-data-science/lstm-nejronnaja-set/
- http://www.bioinf.jku.at/publications/older/2604.pdf
- https://www.youtube.com/watch?v=Kv4NyVW9IZ4
- https://www.youtube.com/watch?v=wYI7RZz4Rz0
- https://habr.com/ru/company/wunderfund/blog/331310/

Источники для примера указаны в ipynb, ссылка на репозиторий с работой и данными на Github:

https://github.com/AlexeySek/HW-Neural-Networks