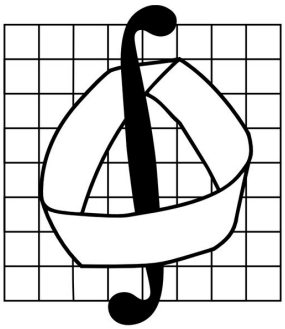
Курсовая работа защищена с оценкой _

Ученый секретарь кафедры доцент Валединский В. Д.

Московский государственный университет имени М.В. Ломоносова

Механико-математический факультет

Кафедра Вычислительной математики



КУРСОВАЯ РАБОТА

Прогнозирование финансовых временных рядов с использованием ансамбля нейронных сетей и анализа поведения рынка

Выполнена студентом 510 группы

Мартыновым Николаем Сергеевичем

Научный руководитель:

д.ф.-м.н, проф. каф.

Вычислительной математики

М.И. Кумсков

Москва, 2023

Аннотация

Исследован подход к решению задачи прогнозирования финансовых временных рядов и других нестационарных процессов. Предложен новый метод прогнозирования, основанный на обнаружении закономерностей, предсказывающих будущую динамику ряда, непосредственно внутри самого ряда, а также на получении вспомогательных данных для прогнозирования дальнейшего поведения ряда из анализа остального набора временных рядов.

Результатом решения этой задачи предложен метод прогнозирования финансовых временных рядов и методы оптимизации параметров разработанной архитектуры ансамбля нейронных сетей, проведены вычислительные эксперименты на примере предсказания движения цены акции, которые говорят о перспективности данного метода, а также определены дальнейшие направления исследований в обзорном виде.

Содержание

Введение

Цель работы

Целью данной работы является исследование архитектур ансамблей нейронных сетей и построение модели для предсказания тренда[1] временного ряда [1, 2][2] выделенного финансового инструмента, которая бы учитывала, что этот временной ряд зависит от множества других рыночных эмитентов, находящихся в той же подгруппе[3] финансовых инструментов, с последующим построением соответствующего вычислительного эксперимента и дальнейшим анализом полученных результатов для нахождения наилучшей архитектуры.

Ожидаемые и полученные результаты

Так как целью нашей работы является классификационная модель с элементами из $\mathbb{Z}_3$[4], то для успешности вычислительного эксперимента минимальный ожидаемый порог, хуже которого модель не должна работать в любом случае, есть модель, результатом работы которой является случайная величина имеющая дискретное равномерное распределение с тремя элементами [3].

Получена архитектура нейронной сети[5], пользуясь которой на тестовой выборке количество правильных ответов превышает количество правильных ответов модели, выдающей один из ответов случайным образом.

[1] Подробнее о том, что такое тренд и какие его разновидности рассматриваются в данной работе описано в разделе: Тренд.

[2] Временным рядам посвящён раздел: Временные ряды.

[3] Определение подгруппы финансовых инструментов, от которых мы ищем зависимость выделенного временного ряда, из всех доступных представляет собой отдельную задачу, о которой можно прочитать в разделе: Вспомогательные задачи.

[4] С подробностями постановки задачи можно ознакомиться в разделе: Постановка основной задачи.

[5] Подробнее см. Глава 2. Нейронные сети.

Вспомогательные задачи

Для корректного проведения вычислительных экспериментов и достижения основной цели в данной работе нам потребуется выполнить несколько вспомогательных работ:

1. Как было упомянуто в разделе Цель работы, после получения результатов вычислительного эксперимента следует подбор наилучшей архитектуры нейронной сети. Из этих соображений требуется каким-то образом определять качество полученной модели. «Мерой» для полученной модели и её работы является функция потерь[6], выбор которой и является одной из вспомогательных задач;

2. Изучение зависимости ключевого временного ряда от остальных и выбор подгруппы с наибольшей корреляцией из всех доступных для анализа временных рядов.

3. Обзор на нейронные сети и методы оптимизации, необходимые для обучения моделей.

4. Работа с финансовыми временными рядами требует базового введения в экономическую теорию, поэтому для понимания специфик финансовых временных рядов, о которых идёт речь в данной работе, важной задачей является предоставление обзора на эту тему.

5. Разобьём на основные задачи проведение непосредственно самого вычислительного эксперимента:

   (a) Выбор среды разработки и обзор на облачные вычисления[7].

   (b) Обзор используемых Python [4] библиотек.

[6] О ней отдельный раздел: Функция потерь.
[7] см. Облачные вычисления

(c) Загрузка данных с помощью программного обеспечения (Simplifying Finance(SimFin))[8].

(d) Предобработка [5] полученных данных в формат, лучшим образом подходящий для обучения нейросети.

(e) Построение модели.

(f) Тестирование модели на различных исходных параметрах, которые не входят в множество тех, на которых обучается нейронная сеть.

История работы

Схожие проекты

Краткое содержание

Глава 1 посвящена постановке задачи и её математически точной формулировке. В ней же изложены методы, которыми предлагается решать поставленную задачу, а также описание схожих проектов, в которых авторы занимаются этим вопросом.

Во второй главе речь идёт о нейронных сетях. Сначала определяется простейшая нейронная сеть, а затем добавляется её многослойность и нелинейность. Последняя достигается с помощью функций активации, о которых написано далее. Когда получено представление об устройстве нейросети, ставится вопрос о её обучении. Для понимания как обучать модель определяется функция потерь, благодаря которой можно заводить речь о поверхности ошибок. После этого описаны основные алгоритмы оптимизации, которыми обучают сеть. Также вводится понятие многоклассовой классификации.

[8]Подробнее в Сбор данных

Третья глава посвящена статистико-экономической составляющей данной работы. Для понимания метода, которым мы сводим исходную задачу предсказания непрерывного значения временного ряда по остальным к предсказанию дискретного значения (класса), необходимой частью работы является описание понятия тренда, которому посвящена подсекция в этой главе. Далее, приводится теория о финансовых временных рядах и их характерных особенностях, например, масштабируемости ряда вдоль временной оси. Пусть у нас имеется много финансовых временных рядов, тогда хотелось бы уметь сравнивать их между собой. Поэтому далее поднимается тема метрик схожести рядов между собой и её связи с функциями потерь, определёнными для векторов из $\mathbb{R}^n$. Если известны методы сравнения рядов по «схожести», то имеется возможность распределить множество временных рядов на подгруппы «близких» между собой по этой метрике. Для этого используются методы кластеризации.

Четвёртая глава полностью выделена под вычислительный эксперимент и описание всего, что необходимо пояснить до его реализации:

- Облачные вычисления, которые помогают коллективно с коллегами делать вычисления не на собственной вычислительной машине, а на вычислительном сервере с, вероятно, куда более мощными вычислительными характеристиками.
- Финансовый on-line сервис SimFin.
- Используемые библиотеки с описанием их применения.
- Метод построения модели и её обучения.

В заключении приводятся результаты вычислительного эксперимента с соответствующей визуализацией. После чего описан анализ полученных резуль-

татов, исследуя который делается вывод о данной работе и о её возможных направлениях развития.

Глава 1

Постановка основной задачи

Целью данной работы является построение модели для предсказания тренда[9] временного ряда $\{x_t^0\}_{t\in T}$ (где T - это множество всех временных тиков) выделенного финансового инструмента, которая бы учитывала, что этот временной ряд зависит от множества других рыночных эмитентов, находящихся в той же группе[10] финансовых инструментов $X := \{\{x_t^i\}_{t\in T}\}_{i=0}^n$, что и $\{x_t^0\}_{t\in T}$. Другими словами, нас интересует нахождение отображения $\phi : X|_{i=1,\dots,n} \mapsto \{x_t^0\}_{t\in T}$, действующее в момент времени $t = \hat{t} \in T$ по правилу $\phi(\hat{t}) : X|_{\substack{i=1,\dots,n \\ t=\hat{t}}} \mapsto \mathbb{Z}_3$, где элементы из $\mathbb{Z}_3$ соответствуют тренду по следующему закону:

- 0 - тренд является убывающим(нисходящим), т.е. медвежий тренд;
- 1 - тренд отсутствует;
- 2 - тренд является восходящим, т.е. бычий тренд.

Замечание 1: Определение группы финансовых инструментов X, от которых мы ищем зависимость выделенного временного ряда, представляет собой отдельную задачу, о которой можно прочитать в разделе Задача кластеризации.

Замечание 2: Отсутствующий тренд определяется тем свойством, что Daily Return [6], определяемый формулой $R^i(t) = \frac{x_t^i - x_{t-1}^i}{x_{t-1}^i}$, по модулю не превосходит некоторой константы $|R^i(t)| < bound$. Выбор константы $bound$ также представляет собой отдельную задачу.

[9] Подробнее о том, что такое тренд и какие его разновидности рассматриваются в данной работе описано в разделе: Тренд.

[10] Определение группы финансовых инструментов X, от которых мы ищем зависимость выделенного временного ряда, представляет собой отдельную задачу, о которой можно прочитать в разделе: Вспомогательные задачи.

Методы решения

Нейронные сети с памятью или её аналогиями справляются с поставленной задачей лучше, чем архитектуры без неё, так как у временного ряда финансовых котировок есть характерные периоды, которые должны учитываться целиком, чего практически невозможно добиться с помощью неглубоких архитектур без памяти.

Таким образом, стоит упомянуть рекуррентные нейронные сети (RNN) [7]. Безусловно, наиболее острой проблемой у RNN, за недопущением которой нужно внимательно следить, является проблема исчезающего градиента[11]. Эта проблема возникает из-за того, что очень глубокие нейронные сети, оптимизированные с помощью процедуры, называемой обратным распространением (backpropagation)[12], используют производные между каждым слоем для обучения. Эти производные по значению могут быть слишком велики или же, наоборот, слишком малы.

Одним из решений является применение специальных RNN с модулями памяти, например, таких как Long-Short-Term-Memory (LSTM) [8] или Gated Recurrent Units (GRU) [9].

LSTM является особой разновидностью архитектуры рекуррентных нейронных сетей, способной к обучению долговременным зависимостям. Запоминание информации – это её основная задача. Рекуррентная нейронная сеть имеет форму цепочки повторяющихся модулей нейронной сети. Повторяющийся модуль в стандартной RNN состоит из одного слоя. Структура LSTM также напоминает цепочку, но модули выглядят иначе. Вместо одного слоя нейронной сети они содержат целых четыре.

[11] Подробнее в подразделе: Проблема исчезающего градиента
[12] О ней можно почитать подробнее в Обратное распространение ошибки

Схожие проекты

1. В первую очередь хотелось бы упомянуть мою предыдущую работу на похожу тематику:

   У неё есть несколько существенных недостатков, первый из которых заключается в том, что модель не учитывает состояние фондового рынка целиком, которое в самом деле отражает реальное настроение инвесторов, влияющих на формирование цены эмитентов во всём рынке.

   Пример. Пусть некоторая компания публикует отчёт за предыдущий квартал, который по всем показателям лучше предыдущего отчёта, а также допустим, что случается негативное событие в той отрасли (или иным образом определённой подгруппе финансовых инструментов), к которой принадлежит эта компания. Результатом этих двух событий может стать понижение стоимости данной компании в виду того, что инвесторы опасаются покупать что-либо в данной отрасли (подгруппе).

   Вторым недостатком является выбранная метрика, которая не учитывает волатильность компании.(см. Рис. 1)

   Последний недостаток заключается в том, что технические показатели есть ни что иное, как некоторая относительно несложная функция $f(x)$ от исходных данных, а следовательно, даже простейшие нелинейные нейронные сети, обладающие достаточной глубиной, хорошо выражают признак $f(x)$ через x воссозданием функции f.

2. https://towardsdatascience.com/pytorch-lstms-for-time-series-data-cd16190929d7

3. Sima Siami-Namini, Neda Tavakoli, Akbar Siami Namin. A Comparative Analysis of Forecasting Financial Time Series Using ARIMA, LSTM, and BiLSTM. arXiv preprint arXiv:1911.09512v1, 21 Nov 2019.

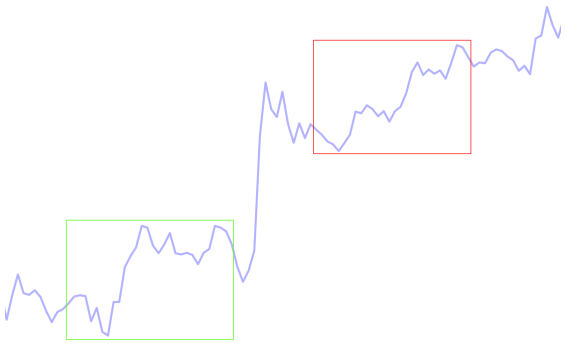


Рис. 1: График цены произвольного финансового инструмента за некоторый промежуток времени. Если модель работает корректно на данных из области, обведённой зелёным контуром, то она вполне может показывать совершенно неадекватные результаты на области, обведённой красным контуром, так как плохо обучена для отмасштабированных вдоль вертикальной оси данных.

4. https://chunliangli.github.io/docs/19iclrKLCPD.pdf,
https://github.com/OctoberChang/klcpd_code

Выводы

Математическая формализация помогает дать исходной задаче точное представление в известных математических терминах и работать с рядами как с математическими объектами, для которых существует обширная теоритически обоснованная область исследований.

Описанные методы прогнозирования позволяют отметить тот общий подход, который применяется для предсказания значений временных рядов современными учёными.

Глава 2. Нейронные сети

Введение

Нейронная сеть прямого распространения

- Нейронные сети состоят из узлов (можно представлять их как некоторые блоки), соединённых связями;
- Каждая связь имеет соответствующий вес и уровень активации;
- У каждого узла есть входная функция (обычно суммирование взвешенных входов), функция активации[13] и выходная функция.

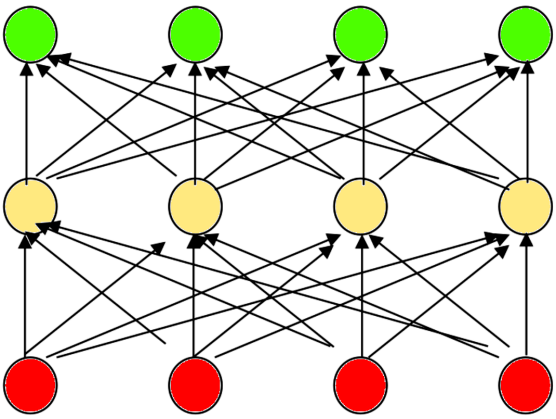


Рис. 2: Многослойная нейронная сеть прямого распространения. Общая схема.
Входные узлы покрашены в красный цвет, выходные узлы покрашены в зелёный цвет, а в оранжевый цвет покрашены узлы скрытого слоя. Скрытых слоёв обычно бывает несколько.

Определим входной вектор $x = (x_0, x_1, \ldots, x_n)^T$ и весовой вектор $\theta = (\theta_0, \theta_1, \ldots, \theta_n)^T$. Определим некоторую функцию активации $g(z)$. Тогда выходная функция $h_\theta(x) = g(\Theta^T x)$. (Сеть изображена на Рис.3)

[13] Подробнее о них см. в разделе Функция активации

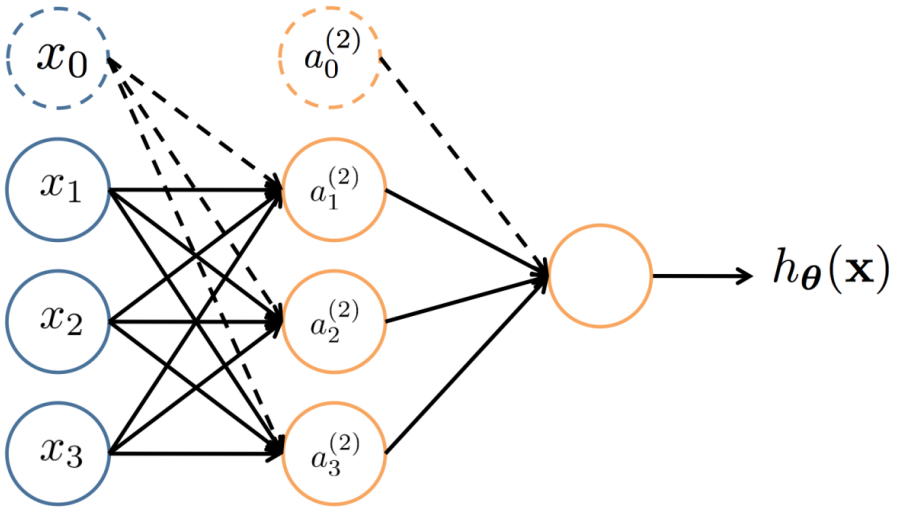


Рис. 3: Интерпретация для $n = 3$ с выходной функцией $h_\theta(x)$.
Первый слой - входной с вектором x, второй слой - скрытый с вектором $a = (a_0^{(2)}, a_1^{(2)}, \ldots, a_m^{(2)})^T$, здесь $m = 3$.

Пусть $a_i^{(j)}$ - активация узла i в слое j; $\Theta^{(j)}$ - матрица весов, отвечающая отображению с j-го слоя в $(j+1)$-ый.

Тогда сеть, изображённая на Рис.3 будет записана в виде следующей системы:

$$\begin{cases} a_1^{(2)} = g(\theta_{10}^{(1)} x_0 + \theta_{11}^{(1)} x_1 + \theta_{12}^{(1)} x_2 + \theta_{13}^{(1)} x_3), \\ a_2^{(2)} = g(\theta_{20}^{(1)} x_0 + \theta_{21}^{(1)} x_1 + \theta_{22}^{(1)} x_2 + \theta_{23}^{(1)} x_3), \\ a_3^{(2)} = g(\theta_{30}^{(1)} x_0 + \theta_{31}^{(1)} x_1 + \theta_{32}^{(1)} x_2 + \theta_{33}^{(1)} x_3). \end{cases} \tag{1}$$

И её выходом будет $h_\theta(x) = a_1^{(3)} = g(\theta_{10}^{(2)} a_0^{(2)} + \theta_{11}^{(2)} a_1^{(2)} + \theta_{12}^{(2)} a_2^{(2)} + \theta_{13}^{(2)} a_3^{(2)})$.

Пусть сеть имеет на j-ом слое s_j узлов, а на $(j+1)$-ом слое s_{j+1} узлов. Тогда Θ_j - это матрица размерности $s_{j+1} \times (s_j + 1)$.

Пусть H - количество слоёв сети. (Слой с номером 0 - входной, с номером H - выходной, остальные слои называются **скрытыми**.)

Тогда результат работы y некоторой (абстрактной) нейронной сети с H слоями

можно записать следующим образом:

$$y = qg(\Theta_H^T g(\Theta_{H-1}^T \dots g(\Theta_1^T x) \dots)) \text{ [10]}, \tag{2}$$

где $q = \sqrt{(s_0 s_1 \dots s_H)^{\frac{H-1}{2H}}}$ - коэффициент нормализации.

Функция потерь

Функция потерь [11] L: $\mathbb{R}^n \times \mathbb{R}^n \mapsto \mathbb{R}$ используется как метрика для оптимизации параметров нейронных сетей. Наша цель - минимизировать ошибку нейронной сети путём правильного изменения её параметров. Ошибка высчитываются с использованием функции потерь, а конкретнее с помощью сопоставления фактического значения и того значения, которое спрогнозировала нейронная сеть.

Пусть $f(x)$ такова, что $y = f(x)$ - истинное отображение, которое мы хотим приблизить нейронной сетью. Пусть $\hat{y} = \hat{f}(x, \theta)$ - полученная предсказательная модель, где Θ - множество параметров модели (веса).
Функция потерь в общем виде записывается как: $L(\Theta) = \int l(f(x), \hat{f}(x, \Theta)) d\mu(x)$, где $l(f(x), \hat{f}(x, \Theta))$ - мера несоответствия (расстояние) между f и $\hat{f}$, а μ - некоторая усредняющая мера. Например: $d\mu(x) = \frac{1}{N} \sum_{l=1}^{N} (\delta(x - x_l))$, где δ - дельта-функция Дирака [12]; или же $d\mu(x) = p(x)dx$, где $p(x)$ - некоторая функция распределения.
Процесс обучения нейронной сети описывается задачей минимизации:

$$L(\Theta) \to \min_{\Theta}$$

На практике обучение нейронных сетей происходит за несколько шагов, которые мы будем называть **эпохами**, в каждом из которых происходит обновление параметров обучаемой модели.

Точнее, **эпохой** (epoch) называется полный проход в обе стороны всего датасета через нейронную сеть один раз. Так как одна эпоха зачастую слишком велика для компьютера, то для того, чтобы не пропускать через нейронную сеть разом весь датасет, данные делятся на несколько частей, называемых **батчами** (batches). Такое разбиение позволяет легче обрабатывать данные и обучаться на конкретной выделенной их части. **Итерациями** называется число батчей, необходимых для завершения одной эпохи.

Рассмотрим распространённые метрики, которые используются в качестве функции потерь:

1. Средняя квадратическая ошибка (MSE) [13, 14] - распространённый показатель, используемый для оценки регрессии. Он измеряет среднеквадратичную ошибку наших прогнозов. Для каждой точки вычисляется квадратичная разница между прогнозами и целью, а затем усредняются эти значения.

   Формула:

   $$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \quad \text{, где:}$$

   y_i – фактический ожидаемый результат, а $\hat{y}_i$ – прогноз модели.

   Замечание: Если мы сделаем всего-лишь один очень плохой прогноз, то тот факт, что мы возводим в квадрат, сделает ошибку ещё больше, а это в свою очередь может исказить метрику в сторону ухудшения показателей модели. Поэтому MSE очень чувствительна к выбросам в данных.

2. Коэффициент детерминации (Coefficient of determination (R^2 regression score function) [15, 16]) хорошо вычисляется в регрессионных моделях и измеряет долю вариации в зависимой переменной. В определении R^2 требуется вычисление только функции среднего и дисперсии, что приме-

нимо к общим моделям. Это согласуется с классической мерой неопределенности с использованием дисперсии и сводится к классическому определению коэффициента детерминации при рассмотрении моделей линейной регрессии. Для одномерного случая рассчитывается по формуле:

$$R^2 = 1 - \frac{MSE(y, \hat{y})}{MSE(E(y), E(\hat{y}))}$$

.

3. Cross Entropy [17]. Пусть K - количество классов в задаче классификации, $y \in \mathbb{Z}_K$ - настоящий класс, к которому отнесён данный пример, а вектор $\hat{y} = (\hat{y}_1, \ldots, \hat{y}_{K-1})^T, \hat{y}_i \in [0, 1],\ \forall i = 0, \ldots, K-1$. Тогда функция потерь записывается следующей формулой:

$$L_{ce}(\hat{y}_0, \ldots, \hat{y}_{K-1}) = -\sum_{i=0}^{K-1} I\{y = i\} \log(\hat{y}_i)$$

Функция активации

В биологических нейронных сетях функция активации обычно является абстракцией, представляющей скорость возбуждения потенциала действия в клетке. Перенося этот принцип в вычислительные нейронные сети, также добавим активационную функцию. Она будет проверять произведённое узлом значение выхода на предмет того, должны ли связи, соединённые с этим узлом, рассматривать этот нейрон как активированный или же игнорировать его. В случае суммирования в сети координат входного вектора, умноженных на соответствующие веса активации функции активации добавляют нелинейность к сети, поэтому итоговое отображение $\hat{f}(x, \Theta)$ не обязательно будет линейным.

Виды различных функций активаций [18](см. Рис 4-7):

- Ступенчатая функция активации: имеет вид $g(z) = C\,\mathrm{sign}(z)$;
- Сигмоидная функция — это ограниченная неубывающая и дифференцируемая функция, имеющая ровно одну точку перегиба.
  Примерами сигмоидных функций являются:
  1. Логистическая функция: $g_{log} : \mathbb{R} \mapsto (0, 1)$ по правилу $z \mapsto \frac{1}{1+e^{-z}}$;
  2. Функция $\arctan(z)$;
  3. Функция $\tanh(z)$;
  4. softsign: $g_{soft} : \mathbb{R} \mapsto (-1, 1)$ по закону $z \mapsto \frac{1}{1+|z|}$.
- Линейная функция активации: $f(z) = z$;
- ReLU: $f_{ReLU} : R \mapsto (0, +\infty)$, $z \mapsto \max(z, 0)$;
- LeakyReLU: $f_{lReLU,\alpha} : R \mapsto (0, +\infty)$, $z \mapsto \max(z, 0) + \alpha \min(0, z)$;
- swish: $f_{swish,\alpha} : \mathbb{R} \mapsto (C_\alpha, +\infty)$, $z \mapsto z \cdot f_{log}(\alpha z) = \frac{z}{1+e^{-\alpha z}}$

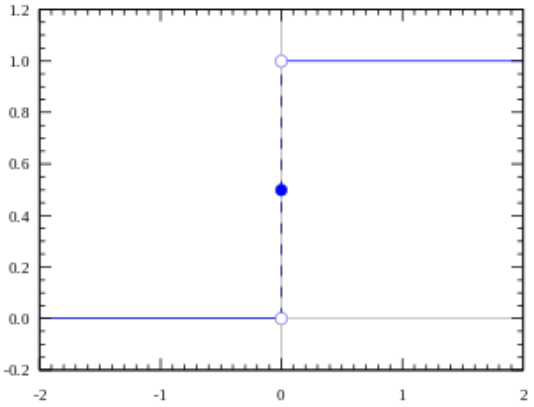


Рис. 4: $\mathrm{sign}(z)$.

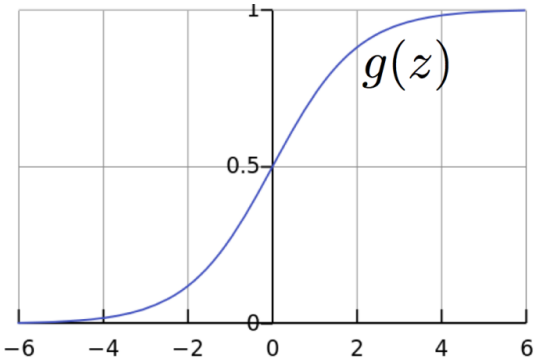


Рис. 5: Сигмоида.

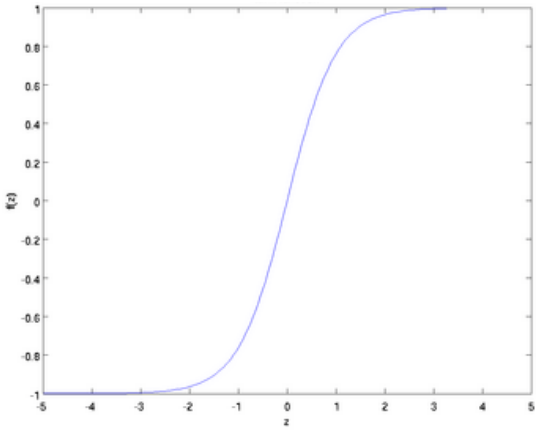


Рис. 6: $\tanh(z)$.

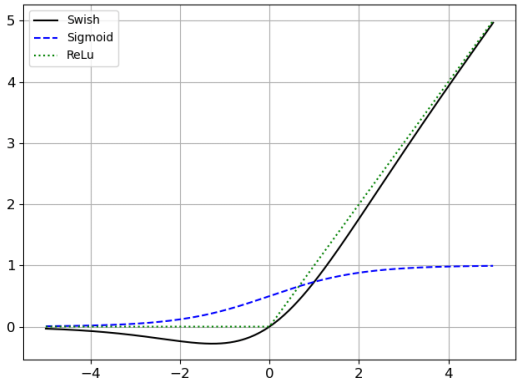


Рис. 7: swish, Сигмоида, ReLU.

Оптимизация параметров нейронных сетей

Обозначим W - пространство весов нейронной сети (Weights) вместо Θ.
Пусть у нас задана некоторая функция потерь $L(W)$.

Поверхность ошибок(loss surface)

Функция L образует поверхность в пространстве параметров W. Обычно для её исследования рассматривают проекцию на два выделенных вектора в пространстве параметров: пусть e_1 и e_2. Пусть x_i и y_i - координаты точки из входной выборки в плоскости $< e_1, e_2 >$. Тогда $f(x_i, y_i) = L(x_i e_1 + y_i e_2)$ вырисовывает в $\mathbb{R}^3$ поверхность с координатами точек $p_i = (x_i, y_i, f(x_i, y_i))^T$. Такая поверхность называется поверхностью ошибок или же поверхностью потерь.
Примеры иллюстраций поверхностей потерь для реальных вычислительных экспериментов(см Рис. 8-11) [19]:

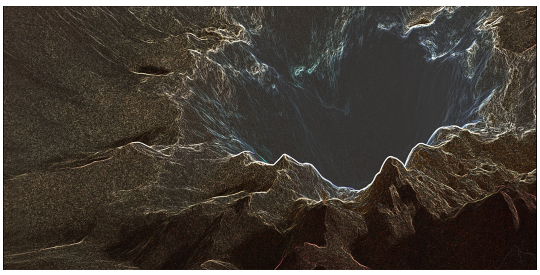


Рис. 8:

Рис. 9:

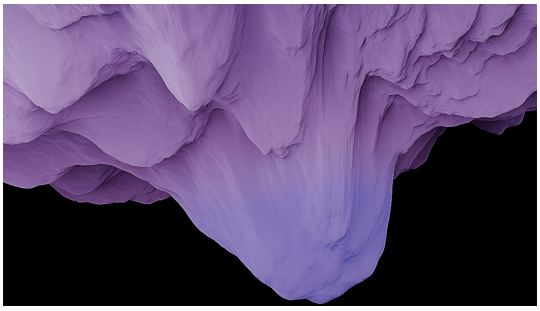


Рис. 10:

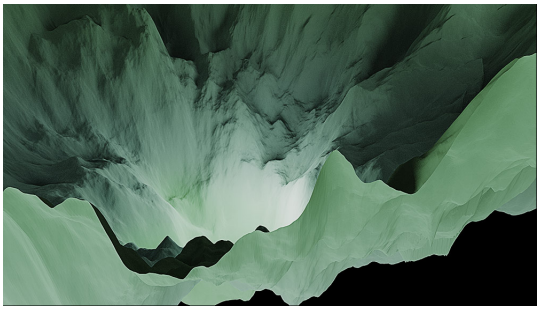


Рис. 11:

Методы оптимизации

Ознакомимся с некоторыми алгоритмами для оптимизации [20, 21] функции, определённой на некоторой поверхности параметров:

- Стандартный градиентный спуск (GD) с коэффициентом скорости обучения (learning rate(lr)) [22] $\alpha > 0$:

$$W^{n+1} = W^n - \alpha \nabla_W L(W^n)$$

  Его смысл заключается в сдвиге на поверхности ошибок вдоль вектора градиента. Зачастую этот параметр меняют непосредственно во время процесса оптимизации[14].

[14] Подробнее о динамическом изменении lr в нейронной сети в процессе обучения см. в разделе Построение модели

- Градиентный спуск с моментом [23] $\beta \in (0,1)$ и lr $\alpha > 0$:

$$\begin{cases} W^{n+1} = W^n - V^n, \\ V^{n+1} = \alpha \nabla_W L(W^n) + \beta V^n. \end{cases} \tag{3}$$

Смысл этого усовершенствованного метода в заведении дополнительной переменной, которая представляет из себя усреднённый градиент, который запоминает значения градиента на предыдущих шагах. Коэффициент β обязан быть строго меньшим единицы, иначе присутствует расходящаяся геометрическая зависимость V^n от $V^k, k < n$ при $n \to \infty$. Физический смысл момента β - это момент инерции. Он помогает выбираться из ситуаций, когда оптимизация остановилась бы в локальном оптимуме при наличие более глобального оптимума (см. Рис. 12). Это же можно наблюдать в седловых точках(см. Рис. 13, 14).

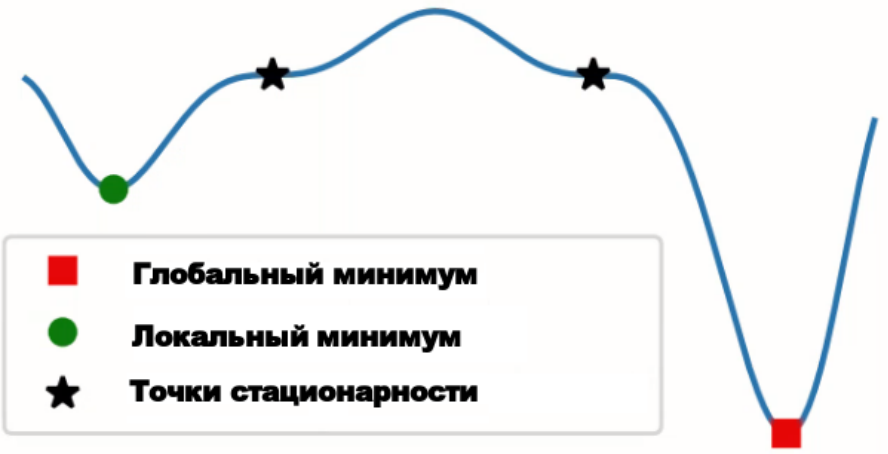


Рис. 12: Случай одномерной проекции на $\mathbb{R}$.

- Градиентный спуск с моментом Нестерова:

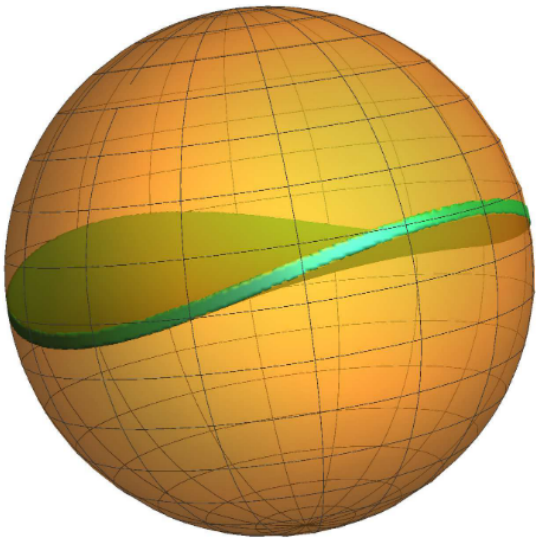


Рис. 13: Поверхность в $\mathbb{R}^3$ локально похожая на «седло», на которой происходит застревание в процессе градиентного спуска.

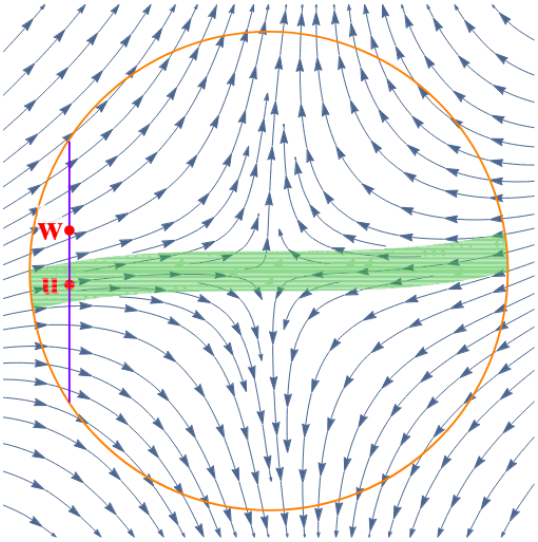


Рис. 14: Проекция «седловой» точки поверхности на $\mathbb{R}^2$, где присутствует «полоса застревания», обозначенная зелёным цветом. Таким образом, точка u - останется в седловой точке, а точка w продолжит процесс оптимизации к глобальному минимуму.

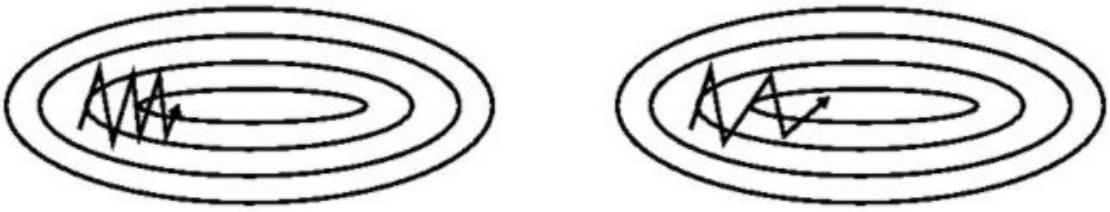


Рис. 15: Сравнение процессов оптимизации. Слева - GD без момента, справа - с моментом.

$$\begin{cases} W^{n+1} = W^n - V^n, \\ V^{n+1} = \alpha \nabla_W L(W^n - \beta V^n) + \beta V^n. \end{cases} \tag{4}$$

В отличии от предыдущего алгоритма, здесь аргумент функции потерь обусловлен аппроксимацией значения, которое параметр примет на следующем шаге. Таким образом при расчете градиента он будет находиться не в текущей позиции, а в положении следующего шага.

- Ещё несколько популярных методов оптимизации, которые сами по себе представляют отдельный интерес для изучения, однако достаточно сложны для краткого изложения: AdaGrad [24], RMSProp [25], Adam [26].

Обратное распространение ошибки

Обратное распространение ошибки [27] - это способ обучения нейронной сети. Он регулирует каждый вес пропорционально тому, насколько он способствует общей ошибке. Если мы будем итеративно уменьшать ошибку каждого веса, в конце концов у нас будет ряд весов, которые дают хорошие прогнозы. Смысл этого процесса в том, чтобы оценить влияние ошибки и распределить её между весами.

Вспомним, как записывается прямое распространение (уравнение 2). К тому же $y = \hat{f}(x, W)$. Обозначим $z_j := g(z_{j-1}, W_j),\ \forall j = 1, \ldots, H$.

Тогда формула для вычисление градиента по весу W_j выглядит следующим образом:

$$\nabla_{W_j}\hat{f}(x,W) = \frac{\partial g_H}{\partial z_{H-1}}(z_{H-1},W_H)\cdots\frac{\partial g_{j+1}}{\partial z_j}(z_j,W_{j+1})\cdot\frac{\partial g_j}{\partial W_j}(z_{j-1},W_j)$$

Замечание: Если в сети W весов и сеть совершает N элементарных операций, то для подсчёта $\nabla_W L$ требуется $O(N)$ операций. В то время как вычисление аппроксимации градиента с помощью конечных разностных схем требует $O(WN)$ операций.

Проблема исчезающего градиента

Проблемой исчезающего градиента [28] называют вычислительную трудность, которая возникает при обучении искусственных нейронных сетей с применением методов обучения на основе градиента и обратного распространения ошибки. Она заключается в том, что в некоторых случаях градиент становится чрезвычайно мал. Это предотвращает хоть какое-то изменение весов. В худшем случае может полностью остановиться дальнейшее обучение нейронной сети.

Задача классификации

Обозначим за H - количество слоёв нейросети. $s \in \mathbb{N}_+^H$ - вектор, содержащий количество узлов на каждом слое. Входной вектор x назовём вектором признаков, а $s_0 = n$ - размерностью входа или количеством признаков.
Выходной слой $\hat{y} \in \mathbb{R}^K$, где K - количество узлов в $\hat{y}$, т.е. $K := s_{H-1}$.
Задача мультиклассовой классификации (иллюстрация на Рис.16) заключается в определении класса, к которому принадлежит входной вектор x.
Другими словами, требуется найти отображение $y : \mathbb{R}^n \mapsto \mathbb{Z}_K$.

Замечание: Как правило классы представляются дискретными числами.

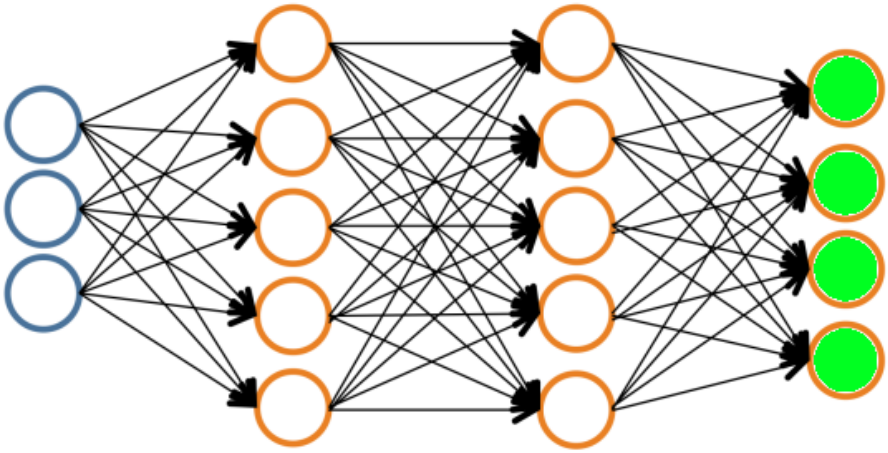


Рис. 16: Зелёным цветом помечены узлы выходного слоя, соответствующие $K = 4$ классам.

Например, вектор $\hat{y} = (0, 0, 1, 0)^T$ для сети с Рис.16 - соответствует классу под номером 3. Поэтому в силу того, что оптимизация[15] нейронных сетей требует вычисление градиентов, выходной слой в задаче классификации удобно делать непрерывным, переходя к так называемой «Мягкой(Soft)» классификации: $\hat{y}(x) \in [0, 1]^K$, где каждая компонента отвечает «степени уверенности» нейронной сети в том, что данный x принадлежит именно этому классу. (Можно представлять как аналог некоторой вероятностной меры)

Задача кластеризации

Постановка задачи

Кластером [37] называется группа однородных элементов, характеризуемых общим свойством.

Пусть задано пространство некоторых объектов X с функцией расстояния между двумя элементами: $\rho : X \times X \mapsto \mathbb{R}_+$. Пусть $X^n := \{x_i\}_{i=1}^n, x_i \in X$ -

[15] Подробнее будет изложено в разделе Оптимизация параметров нейронных сетей.

исходная выборка для обучения.

Задача кластеризации заключается в нахождении меток кластеров $\{y_j\}_{j=1}^m \in Y$ таким образом, чтобы в каждом кластере были только близкие друг к другу объекты, и, напротив, в разных кластерах объекты существенно различались, а также отображение $\psi : X \mapsto \mathbb{Z}_m$, сопоставляющее элементу заданной выборки его кластерную метку. Такое ψ называется алгоритмом кластеризации.

Замечание: Чаще всего, заранее неизвестно точное число кластеров $|Y|$, а если известно, то только из аналитических соображений на тему применимости алгоритма в той или иной конкретной предметной области.

Проиллюстрируем на Рис. 17 работу алгоритма кластеризации.

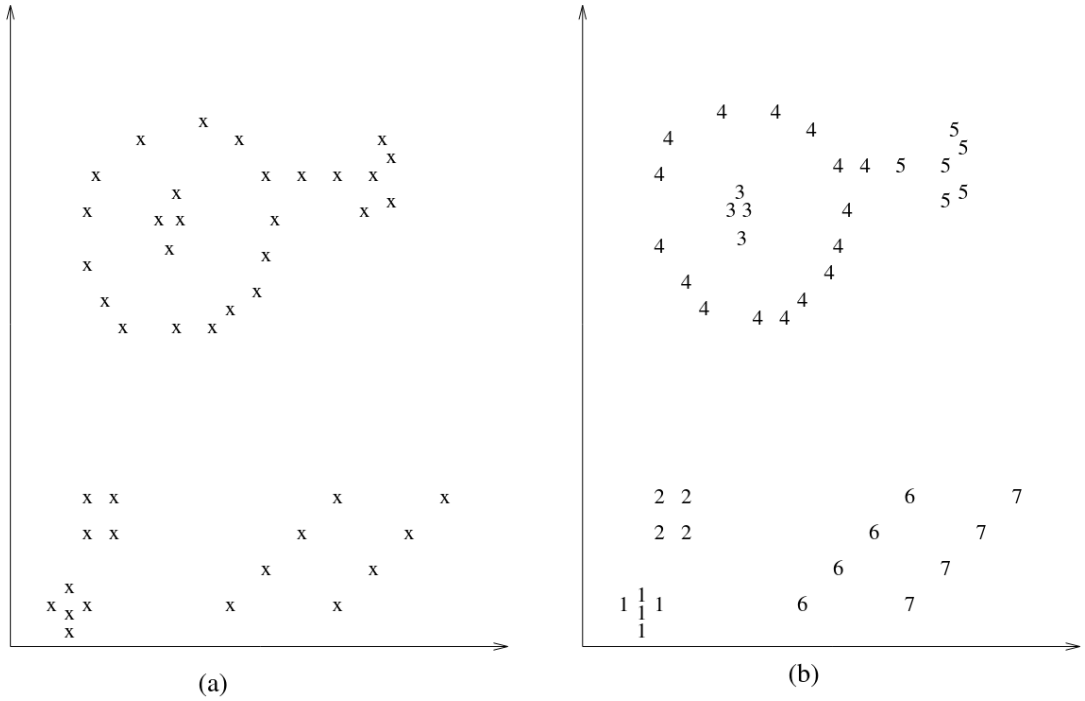


Рис. 17: Работа алгоритма кластеризации. (a) - исходная выборка X^n в проекции на $\mathbb{R}^2$, (b) - размеченные данные с помощью $m = 7$ кластерных меток.

Кластеризация применяется для решения следующих задач:

1. Задача предварительной обработки данных с целью упростить дальней-

шую работу с ними. Разбиваем исходную выборку для обучения на группы схожих объектов, чтобы работать с каждой группой в отдельности;

2. Задача сжатия объёма хранимых данных;

3. Задача об обнаружении объектов, которые уникальны, так как не лежат ни в одном из кластеров.

Для этой работы интереснее всего применимость именно в первой задаче.

Алгоритмы

Введём следующие обозначения:
Пусть μ_y - центры кластеров, $K_y = \{x_i \in X^n | y_i = y\}$ - сами кластеры, μ - центр масс всей выборки, $\Sigma_y = diag(\sigma^2_{y_1}, \ldots, \sigma^2_{y_n})$ - диагональная ковариационная матрица кластера.

Кратко опишем существующие методы [38]:

1. Эвристические графовые алгоритмы.

   Представим выборку в виде графа, где вершины - объекты обучающей выборки из X, а взвешенные рёбра между вершинами - попарные расстояния $\rho(x_1, x_2)$ между ними.

   (a) **Алгоритм выделения связных компонент.**

   Связной компонентой графа называется подмножество его вершин, в котором любые две вершины можно соединить путём, целиком лежащим в этом подмножестве.

   Задача найти число $R \in [\min\limits_{\rho_{ij}}, \max\limits_{\rho_{ij}}]$ такое, что если удалить рёбра, для которых $\rho_{ij} > R$, то граф распадётся на несколько связных компонент, которые и будут являться искомыми кластерами.

(b) **Алгоритм кратчайшего незамкнутого пути.**

Пусть дано число кластеров K. Задача состоит в построении графа с n вершинами и $n-1$ рёбрами так, чтобы рёбра соединяли все вершины и обладали минимальной суммарной длиной. Когда такой граф найден, удаляются $K-1$ самых длинных рёбер, и связный граф распадается на K кластеров. (см. Рис. 18 - 19)

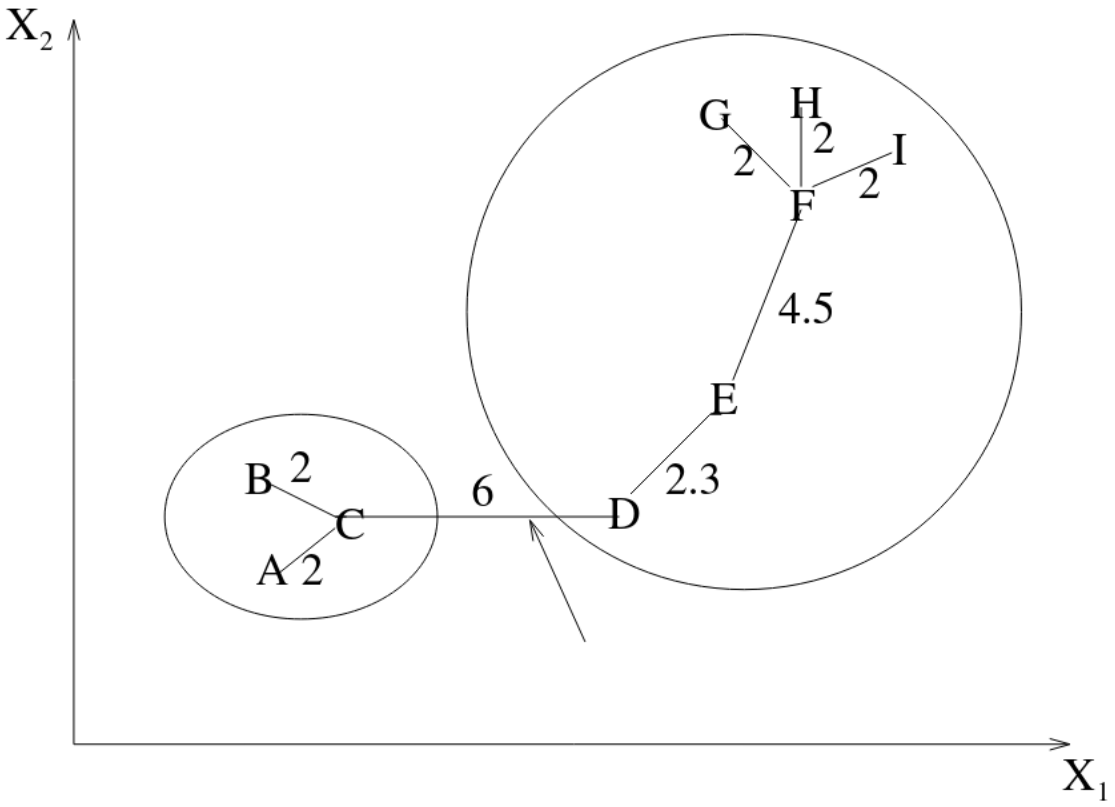


Рис. 18: Результат работы алгоритма выделения связных компонент для $K=2$. CD - ребро максимальной длины, по которому произошло разложение компонент.

Замечание: Построение кратчайшего незамкнутого пути производится за $O(n^3)$ операций.

(c) **Алгоритм FOREL.**

Имеется выделенный объект $x_0 \in X$ и задано число R. На каждом

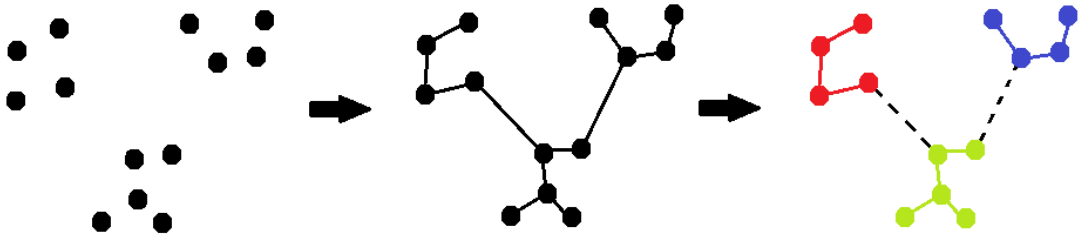


Рис. 19: Результат работы алгоритма выделения связных компонент. $K = 3$.

шаге выделяется подвыборка обучающей выборки, которая попадает в шар $\{x_i : \rho(x_0, x_i) \leq R\}$, после чего x_0 переопределяется как центр тяжести полученного множества. Процедура выполняется до выполнения критерия остановки, который заключается в достаточно малом изменении положения точки x_0.

2. Статистические алгоритмы.

   Пусть кластеры достаточно хорошо описываются некоторым семейством вероятностных распределений. Тогда задача кластеризации сводится к разделению смеси распределений по конечной выборке. Пусть для кластера

   (a) **EM-алгоритм** [39], который заключается в итерационном повторении двух шагов. На E-шаге с использованием формулы Байеса[40] вычисляются скрытые переменные $g_{iy} = P(y_i = y \in Y)$. На M-шаге с помощью полученных g_{iy} уточняются характеристики (μ_y, Σ_y) кластера.

   (b) **Метод k-средних(k-means)[41].** Для возможности использования этого метода необходимо иметь гипотезу о наиболее вероятном количестве кластеров, так как параметр K вводится заранее. Главная идея — минимизация расстояний $\rho(x_{iy}, x_{jy})$ между элементами кластера и максимизация расстояния между кластерами $dist(K_{y_i}, K_{y_j})$.

3. Иерархические алгоритмы кластеризации.

   Также называются алгоритмами таксономии. Они строят не одно разбиение изначальной выборки X^n на непересекающиеся кластеры $K_y, y \in Y$, а систему вложенных разбиений.

   Среди алгоритмов иерархической кластеризации различаются два основных типа. Дивизимные (нисходящие) - это алгоритмы, разбивающие выборку на всё более мелкие кластеры. Более распространены агломеративные(восходящие) алгоритмы, в которых объекты объединяются в более крупные кластеры с каждым шагом. Пример работы восходящего алгоритма см. на Рис.20.

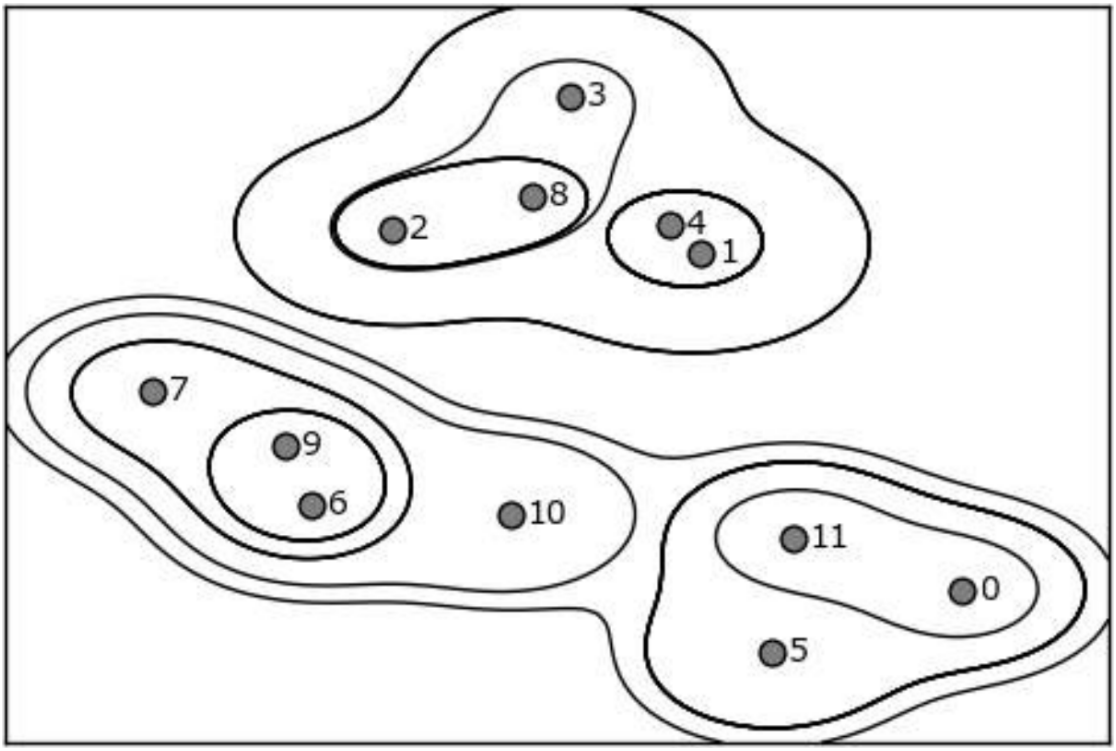


Рис. 20: Пример иерархического присвоения кластеров (показаны в виде линий), полученное с помощью алгоритма агломеративной кластеризации.

Результат таксономии обычно удобно иллюстрировать в виде таксоно-

мического дерева или же дендрограммы. (см. Рис 21)

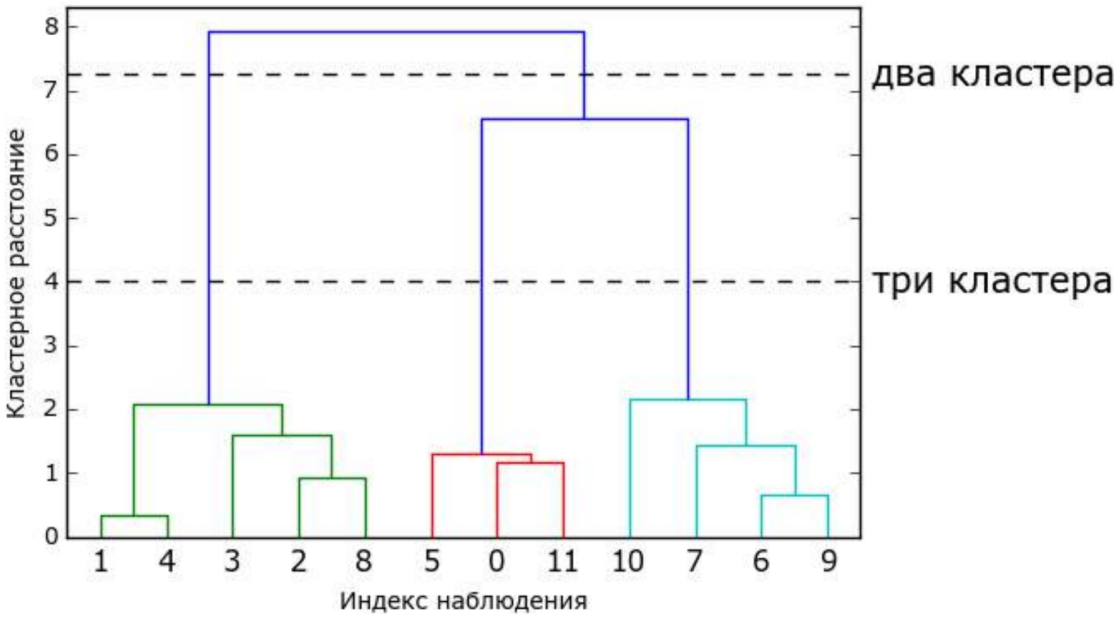


Рис. 21: Дендрограмма для примера, изображённого на Рис.20.

Функция потерь для кластеризации

Задача кластеризации сводится к задаче дискретной оптимизации по некоторому функционалу L. Его можно определить различными способами. Приведём несколько из них:

- Среднее внутрикластерное расстояние $F_0 := \frac{\sum\limits_{i<j} \rho(x_i,x_j) I_{y_i=y_j}}{\sum\limits_{i<j} I_{y_i=y_j}}$;
- Обратное среднее межкластерное расстояние: $\tilde{F}_1 := \frac{1}{F_1} := \frac{\sum\limits_{i<j} I_{y_i \neq y_j}}{\sum\limits_{i<j} \rho(x_i,x_j) I_{y_i \neq y_j}}$;
- Сумма средних внутрикластерных расстояний: $\Phi_0 := \sum\limits_{y \in Y} \frac{1}{K_y} \sum\limits_{i:y_i=y} \rho^2(x_i, \mu_i)$. Физический смысл Φ_0 есть инерция кластера K_y относительно его центра масс;
- Сумма межкластерных расстояний: $\tilde{\Phi}_1 := \frac{1}{\Phi_1} := \frac{1}{\sum\limits_{y \in Y} \rho^2(\mu,\mu_i)}$.

Для учёта одновременно как межкластерных, так и внутрикластерных расстояний вычисляют функцию потерь $\frac{F_0}{F_1}$ или $\frac{\Phi_0}{\Phi_1}$.

Выводы

Нейронные сети, даже самые простые их вариации, хорошо справляются с выражением настоящей зависимости в виде отображения $y = f(x)$ с помощью представления $\hat{y} = \hat{f}(x)$, описанного в данной главе, с добавлением нелинейности от функции активации. На тему выразимости есть следующие замечательные статьи: о выразимости с использованием ReLU [29, 30], о минимальной ширине нейронной сети при достижении достаточного условии выразимости с использованием различных функций активации [31]. К тому же если совсем немного сузить функциональное пространство, в котором ищется исходная функция $f(x)$, то эффективная аппроксимация достигается $\Leftrightarrow \exists$ эффективное приблизижение функции $f(x) = x^2$, что в свою очередь доказывается с использованием явных конструкций в виде функций под названием «sawtooth»(см. Рис.22) [32, 33].

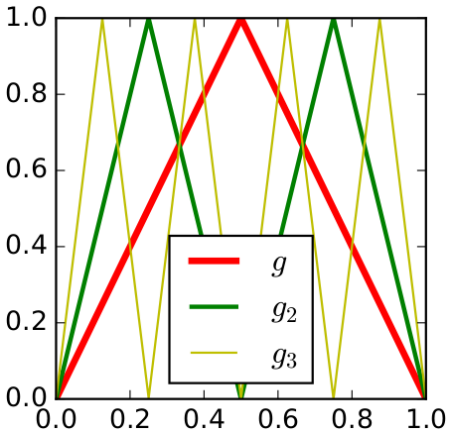


Рис. 22: Функции вида $g_m(x) = \underbrace{g(g(\ldots g}_{m}(x)\ldots))$, где $g(x) = \begin{cases} 2x, x < \frac{1}{2} \\ 2(1-x), x \geq \frac{1}{2}. \end{cases}$

Таким образом в распоряжении читателя есть различные функции потерь, активационные функции и архитектуры нейронных сетей, которые вообще говоря, не обязательно имеют последовательное соединение между слоями. Соединения между слоями могут выстраиваться в граф, имеющий циклы вычислений, или же образовывать дополнительную связную структуру, которая отвечает за сохранение старых результатов (так называемую «память сети»).

Глава 3. Анализ рынка

Экономический глоссарий

Тренд

Тренд представляет собой общую систематическую линейную или нелинейную характерную компоненту временного ряда, которая может изменяться во времени. Многие стратегии торговли основываются на локальных трендах (тенденциях), по-другому понимаемых как положительная автокорреляция в изменениях цен [34].

Обычно различают три вида трендов (см. Рис. 23-25).

1. Восходящий тренд: При восходящей тенденции каждый последующий локальный максимум цены выше предыдущего и каждый последующий локальный минимум также выше предыдущего.

2. Нисходящий тренд: При нисходящей тенденции каждый последующий локальный максимум ниже предыдущего и каждый последующий локальный минимум ниже, чем предыдущий.

3. Боковой тренд: При боковой тенденции каждый последующий локальный максимум и локальный минимум находятся примерно на том же уровне, что и предыдущие.

Замечание: У тренда обязательно есть его начало и конец. В нашей статье, у нас будут получены данные, которые обновлены за некоторый период времени $Period$. Поэтому сглаживая общее понятие тренда, мы можем определить его через модуль Daily Return: $|R^i(t)| < bound$.

Рис. 23: Восходящий тренд.

Рис. 24: Нисходящий тренд.

Рис. 25: Боковой тренд.

Временные ряды

Стационарные и **нестационарные** процессы:

В зависимости от того, изменяются или не изменяются во времени t параметры процессов, их подразделяют на стационарные и нестационарные. Стационарные ряды - это ряды, которые имеют постоянные по времени среднее, дисперсию и автокорреляции.

Ряды не являющиеся стационарными называются нестационарными.

Другими словами, если обозначать совокупность статистических характеристик, влияющих на процесс, через Q, то при стационарном процессе $\frac{\partial Q}{\partial t} = 0$ т.е. эти параметры могут изменяться в пространстве признаков X, но не изменяются по времени t. Однако для нестационарного процесса $\frac{\partial Q}{\partial t} \neq 0$, т.е. характеристики процесса изменяются не только в пространстве X, но и по времени t.

Финансовый временной ряд в базовом понимании (в общем случае) является нестационарным процессом. Однако для некоторых архитектур можно рассматривать ряды, у которых временные промежутки распределены неравномерно таким образом, чтобы временной ряд был стационарным. Рассматривая изменения временного промежутка между данными у временного ряда, мы приходим к понятию Multi-Scale моделей [35, 36]. Финансовый временной ряд обладает характерным свойством масштабируемости, поэтому работа мо-

дели на временном ряде, а также его тенденционное поведение очень сильно зависит от выбираемого временного промежутка между данными.

Метрики временных рядов

Если фиксирован временной интервал между данными, то функцией потерь между двумя временными рядами вполне могут послужить функции из раздела Функция потерь, если представлять ряд как вектор последовательных значений в пространстве, где каждый базисный вектор строго соответствует своему временному тику.
Также в качестве метрики можно рассматривать нормированные динамические характеристики временного ряда, такие как дисперсия, волатильность и т.п.

Выводы

Финансовые временные ряды в силу общего взаимодействия на одном финансово-экономическом поле определённо имеют взаимосвязи друг с другом, которые могут прослеживаться благодаря, например, сезонной популярности той или иной отрасли, ежегодным политическим факторам и т.д. Важно, что финансовый временной ряд при анализе тренда масштабируется на различные временные промежутки, в которых заключена какая-либо информация извне в смысле некоторой сезонности. Таким образом, имеет смысл проводить эксперименты на различных временных промежутках, чтобы учесть те факторы, которые влияют на тенденцию временного ряда.

Чтобы определить это финансово-экономическое поле, нужно либо рассматривать замкнутый фондовый рынок, либо провести кластеризацию временных рядов с помощью алгоритмов кластеризации по некоторой метрике и получить для ключевого финансового инструмента выделенный сектор, в

котором дальше выражать многомерную функцию зависимости с помощью нейронных сетей.

Глава 4. Описание алгоритма вычислительного эксперимента

Облачные вычисления

Облачные вычисления [42] (cloud computing) — это модель обеспечения удобного сетевого доступа по требованию к некоторому общему фонду конфигурируемых вычислительных ресурсов, которые могут быть оперативно предоставлены и освобождены с минимальными эксплуатационными затратами.

При использовании облачных вычислений потребители информационных технологий могут существенно снизить капитальные расходы — на построение центров обработки данных, закупку серверного и сетевого оборудования, аппаратных и программных решений по обеспечению непрерывности и работоспособности — так как эти расходы поглощаются провайдером облачных услуг. Кроме того, длительное время построения и ввода в эксплуатацию крупных объектов инфраструктуры информационных технологий и высокая их начальная стоимость ограничивают способность потребителей гибко реагировать на требования рынка, тогда как облачные технологии обеспечивают возможность практически мгновенно реагировать на увеличение спроса на вычислительные мощности.

Различные сервисы, предоставляющие услуги для облачных вычислений:

- Блокнот Google Colab (GC) — это бесплатная интерактивная облачная среда для работы с кодом от компании Google. Она позволяет одновременно с коллегами работать с данными с помощью написания кода на языке программирования Python прямо в браузере.

- Yandex DataSphere – в отличие от GC это платный блокнот, в котором

тарифицируется фактическое время вычислений. Особенности использования сервиса можно изучить в документации [43].

- Kaggle Kernels – кроме Python, сервис Kaggle поддерживает язык программирования R, интегрируется с Google Cloud Storage, BigQuery и AutoML.

- Azure Notebooks – поддерживает не только Python, но и другие языки (R, F#). Сервисы Microsoft Azure, как и Яндекса, тарифицируются за фактическое время использования.

В данной работе используется блокнот Google Colab для написания программной реализации вычислительного эксперимента.
Ссылка для просмотра кода:
https://github.com/kolmar7777/Forecasting-financial-time-series-using-a-neural-network-and-analysis-of-market-behavior/blob/home/Presentation_of_AI_for_financial_Time_Series.ipynb

Необходимые библиотеки

NumPy [44] (сокращенно от Numerical Python) — библиотека с открытым исходным кодом для языка программирования Python. Математические алгоритмы, реализованные на интерпретируемых языках, зачастую работают гораздо медленнее, чем алгоритмы, реализованные на компилируемых языках. Библиотека NumPy предоставляет реализации вычислительных алгоритмов (в виде функций и операторов), оптимизированные для работы с многомерными массивами.

Pandas [45] является программной библиотекой, написанной на языке программирования Python и используемой для обработки и анализа данных. Её

работа с данными строится поверх библиотеки NumPy, являющейся инструментом более низкого уровня. Pandas предоставляет специальные структуры данных и операции для манипулирования числовыми таблицами и временными рядами. Название библиотеки происходит от эконометрического термина «панельные данные», используемого для описания многомерных структурированных наборов информации.

DataFrame [46] - это объект библиотеки Pandas для работы с индексированными массивами двумерных данных (таблицами).

Matplotlib [47] — это обширная библиотека для создания статических, анимированных и интерактивных визуализаций на Python. Получаемые изображения зачастую используются в качестве иллюстраций в публикациях.

PyTorch [48] - фреймворк машинного обучения для языка Python с открытым исходным кодом. Разрабатывается преимущественно группой искусственного интеллекта Facebook. Также вокруг этого фреймворка выстроена экосистема, состоящая из различных библиотек, разрабатываемых сторонними командами:

- PyTorch Lightning и Fast.ai, упрощающие процесс обучения моделей,
- Pyro - модуль для вероятностного программирования от Uber,
- Flair - для обработки естественного языка,
- Catalyst - для обучения моделей глубокого обучения.

Разберёмся, что такое фреймворк глубокого обучения [49]. Под глубоким обучением как правило понимают обучение функции, представляющей собой композицию множества нелинейных преобразований. Такая сложная функция ещё называется потоком или графом вычислений. Фреймворк глубокого обучения должен уметь делать всего три вещи:

1. Определять граф вычислений;

2. Дифференцировать граф вычислений;

3. Вычислять его.

Сбор данных

Сбор данных производится с помощью сервиса SimFin (Simplifying Finance) [50].

В настоящее время SimFin предлагает онлайн-платформу для исследований, а также прямой доступ к данным через их массовую загрузку и API (Application Programming Interface — программный интерфейс приложения) [51][16].

Для работы с API пользователю требуется пройти регистрацию в SimFin и получить свой уникальный ключ (API Key). Для осуществления запросов и получения требуемых финансовых данных требуется скопировать его в код программной реализации заменив строку `sf.set_api_key('Your_Access_Key')`.

Получение временных рядов

Запрос, сформированный с помощью API SimFin, возвращает таблицу Pandas.DataFrame с требуемыми данными.

В полученной таблице содержутся значения запрошенного экономического показателя взаимнооднозначно соотнесённые с временными метками.

Далее выделяется ключевой эмитент, который будет предсказываться построенной архитектурой нейронной сети. Его значения вместе с временными метками сохраняются как два отдельных вектора значений, а из полученной таблицы вектор, соответствующий ключевому эмитенту, удаляется.

Замечание: В данной работе в целях приближения исходной задачи к прак-

[16] Подробнее с функциями и методами, которые предоставляет пользователям данное API, можно ознакомиться в разделе «Documentation» [51]

тическому применению вектор ключевых значений составляется из значений, соответствующих значениям на момент открытия следующего дня, а вся остальная таблица составлена из значений на момент закрытия текущего дня. Таким образом, нейронная сеть по известным значениям во время закрытия текущего дня делает прогноз значения ключевого элемента в момент открытия на следующий день.

После выделения требуемого ключевого вектора в оставшейся таблице требуется выделить подгруппу эмитентов, с помощью использования значения которых будет проведено дальнейшее обучение нейронной сети.[17]

С использованием методов библиотек, которые были описаны выше, данные, оставшиеся в таблице, сохраняются в виде временного ряда, где каждой временной метке соответствует вектор значений всех эмитентов в этот момент времени. Далее проводится проверка соответствия временных меток ключевого эмитента и полученного временного ряда всех остальных эмитентов подгруппы.

После проверки на совпадение имеет смысл говорить о векторе значений ключевого эмитента y и векторе векторов (т.е. матрице X) значений всех остальных элементов подгруппы. Полученные вектор y и матрица X с помощью библиотеки NumPy переводятся в одномерный и двумерный массивы соответственно.

[17] В данной работе в качестве искомой подгруппой выбраны все оставшиеся доступные эмитенты в таблице, но возможным направлением развития является кластеризация векторов значений эмитентов и последующий выбор одного из полученных кластеров в качестве предсказательной подгруппы. Другим направлением развития может послужить, напротив, выбор по одному представителю из каждого класса для того, чтобы они вносили существенно различную информацию о скрытом состоянии всего рынка в целом. Иными словами, центр кластера может послужить основным источником информации обо всём кластере целиком. К тому же грамотное использование кластеризации способно существенно снизить нагрузку на вычислительную машину. Подробнее см. в разделе Задача кластеризации.

Обработка данных

Заменим каждый элемент y_t вектора y на один из трёх классов, к которому принадлежит y_t, в соответствии с наличием и типом тренда в данной точке (0 - тренд убывающий, 1 - тренд отсутствует, 2 - тренд восходящий).
Таким образом $y_t \rightleftarrows z_t \in \mathbb{Z}_3,\ \forall t \in \{Time_t\}_{t=1}^{T}$.

Аналогичным образом поступаем с матрицей X. Тренд эмитента определеяется в точке t через значение эмитента на предыдущем шаге $t-1$ и на текущем t по формуле $|\frac{y_t - y_{t-1}}{y_{t-1}}| < bound$.

Затем разбиваем данные на батчи. Для этого нам нужно определить размер разбиения. Размер батча в этой работе предлагается сделать 32 точки данных, так как это позволяет спроецировать поиск закономерностей на ежемесячные потоки данных. Другим подходом может быть рассмотрение 128 точек данных, которые покрывают квартал. Для удобства в PyTorch предоставляется ряд утилит для загрузки датасетов, их предварительной обработки и взаимодействия с ними. Эти вспомогательные классы находятся в модуле torch.utils.data [52].

После выбора размера батча создаётся новый объект датасета, наследующий класс torch.utils.data.Dataset и являющийся источником данных. Обязательным требованием к его использованию является переопределение метода __len__ так, чтобы он возвращал количество образцов в датасете, а также метода __getitem__ для доступа к единичному значению по конкретному индексу. После создания датасета используется класс DataLoader из этого же модуля для автоматического разбиения данных на батчи и загрузки датасета в модель нейронной сети.

Пакетная нормализация [53] (batch-normalization) — метод, который позволяет повысить производительность и стабилизировать работу нейронных сетей. Суть данного метода заключается в том, что некоторым слоям

нейронной сети на вход подаются данные, предварительно обработанные и имеющие нулевое математическое ожидание и единичную дисперсию.

Построение модели

Если в модели обучение производится с использованием оптимизатора, в котором используется параметр lr[18], то для наибольшей эффективности процесса оптимизации применяют learning rate sheduler [54], который изменяет значение параметра lr прямо во время обучения в зависимости от заданного алгоритма[19]. В данной работе мы будем либо задавать фиксированный lr на всех этапах обучения, либо learning rate sheduler, привязанный к определённым этапам обучения (высчитываемым через количество эпох).

Параметры.

Для первичного вычислительного эксперимента зададим следующие начальные параметры:

$bound = 0.01$,

Функция потерь(Lf) - Cross Entropy(CE),

Размер батча (batch_size(bs)) = 32,

Количество эпох (epochs) = 50,

Оптимизатор (opt) - SGD,

lr = 0.1, а затем делится на 10 после $\lfloor (epochs/10) \rfloor$, $\lfloor (epochs/3) \rfloor$ эпох.

Модель.

Построим двухслойную нейронную сеть (с одним скрытым и одним выходным слоем), где оба слоя будут линейными, а между ними нелинейности

[18] см. Методы оптимизации

[19] Вообще говоря, параметр lr может изменяться как параметр нейронной сети и тоже подвергаться обучению. Об этом написано в работе [55]

добавит функция активации ReLU с последующей батч-нормализацией.

Размер скрытого слоя (hidden_dim_size(hds)) выберем 120 узлов.

Данные.

Тест проводится на датасете, в котором 1256 точек данных $x \in \mathbb{Z}_3^{2010}$, 942 из которых являются обучающей выборкой, а 314 тестовой.

Как показывает результат на Рис. 26 подобная классификация действитель-

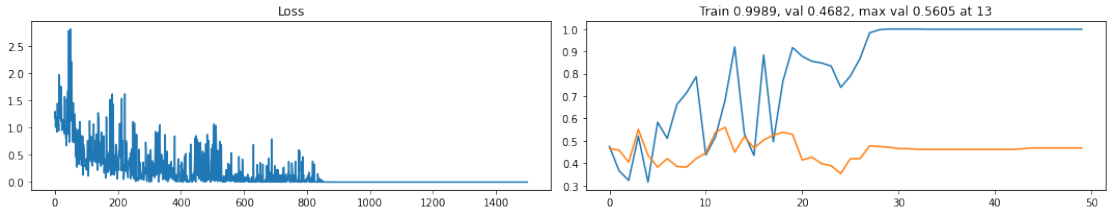


Рис. 26: Результат работы двуслойной нейросети.
Lf = CE, bs = 32, epochs = 50, opt = SGD, lr = 0.1|0.01|0.001, bound = 0.01, hds = 120. Левый график отображает значение функции потерь на каждой итерации. На правом графике изображена доля правильно классифицированных данных из обучающей (синяя линия) и тестовой (оранжевая линия) выборок на каждой из эпох.

но предсказывает гораздо лучше, чем модель, выдающая случайные данные с равномерным дискретным распределением ($\frac{100}{3}\%$ правильно распределённых данных), так как полученная на 13-ой эпохе модель нейронной сети правильно классифицировала 56.05% тестовой выборки, а глядя на правый график можно убедиться, что правильность на обучающей выборке ещё выше, так как синяя линяя находится над оранжевой. Даже после переобучения сети тесты на валидационной выборке выдают результат в 46.82%, а значит, что в данных действительно содержатся такие закономерности, очень строго придерживаясь которых (модель переобучена по входным данным), на выходе получается неплохая обобщающая способность.

Выводы

Проведение вычислительного эксперимента удобно осуществлять с помощью написания кода на интерпретируемом языке программирования в сервисе, предоставляющем облачные вычисления на своих серверах. Такая реализация снижает требования на локальную вычислительную систему.
Во время написания кода для удобства реализации следует пользоваться различными полезными библиотеками и модулями, имеющими понятный интерфейс взаимодействия и высокую скорость вычислений. Необходимыми библиотеками в данной работе являются: NumPy, Pandas, Matplotlib, PyTorch.

Сбор данных производится с помощью запроса от зарегистрированного пользователя на сервер сервиса SimFin через предоставляемое им API. Затем производятся представление полученных данных в требуемый формат, соответствующий финансовым временным рядам, и их проверка на корректное сопоставление по временным меткам. После этого переводим значения соответствующих временных рядов в метки тренда и формируем требуемые для оптимизации параметров нейронной сети валидационную и обучающую выборки.

Когда данные сохранены в надлежащем виде, строится модель для вычислительного эксперимента с заданными начальными параметрами архитектуры нейронной сети. Построенная модель обучается и полученный результат на тестовых данных во время обучения свидетельствует о том, что нейронная сеть хорошо выделяет закономерности, которые помогают получить хорошую обобщающую способность.

Заключение

Результаты

Визуализация результатов

Тесты для двухслойной линейной нейронной сети.

Функция активации - ReLU. Пусть bound = 0.01. Результаты построения архитектуры (см. Рис. 27- 30)

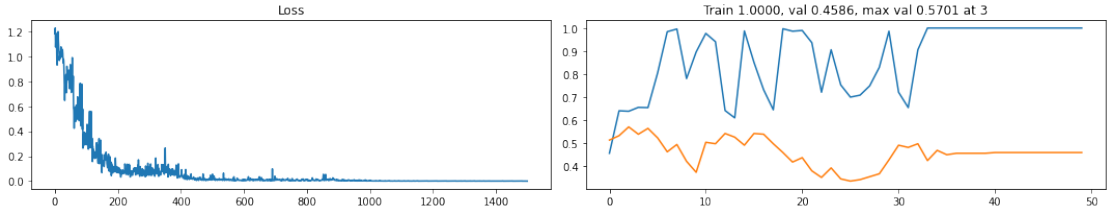


Рис. 27: Lf = CE, bs = 32, ep = 50, opt = Adam, lr = 0.1|0.01|0.001, hds = 120.

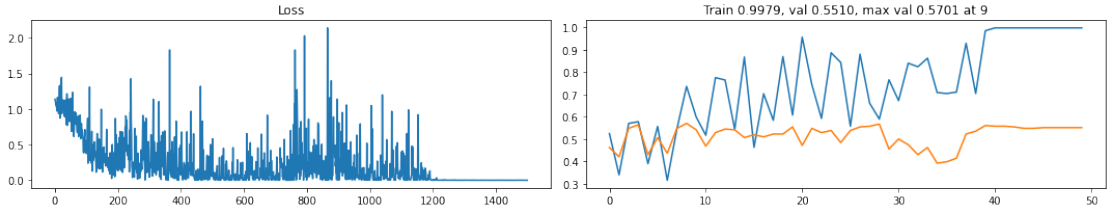


Рис. 28: Lf = CE, bs = 32, ep = 50, opt = SGD, lr = 0.1|0.01|0.001, hds = 32.

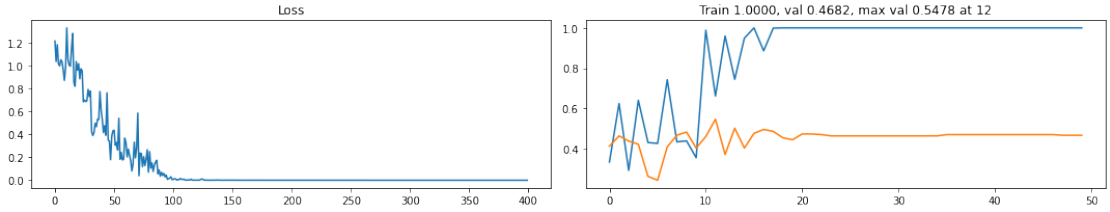


Рис. 29: Lf = CE, bs = 128, ep = 50, opt = SGD, lr = 0.1|0.01|0.001, hds = 32.

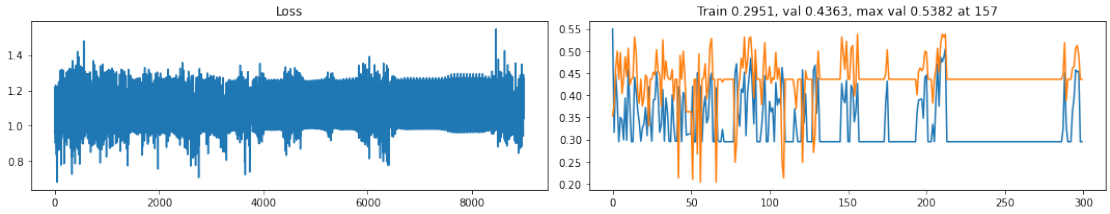


Рис. 30: Функция активации - логистическая.
Lf = CE, bs = 32, ep = 300, opt = SGD, lr = 0.1|0.01|0.001, hds = 32.

Тесты для четырёхслойной линейной нейронной сети.

Пусть bound = 0.01.

Нелинейность между всеми слоями с помощью ReLU и последующей батч-нормализации.

Размерности скрытых слоёв по порядку: 32, 128, 7, 3. (Результаты Рис. 31-33)

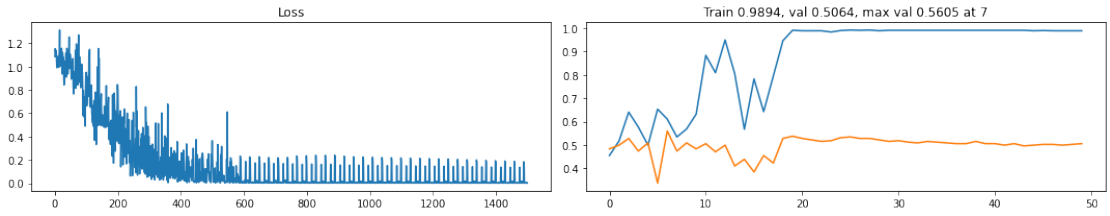


Рис. 31: Lf = CE, bs = 32, ep = 50, opt = SGD, lr = 0.1|0.01|0.001

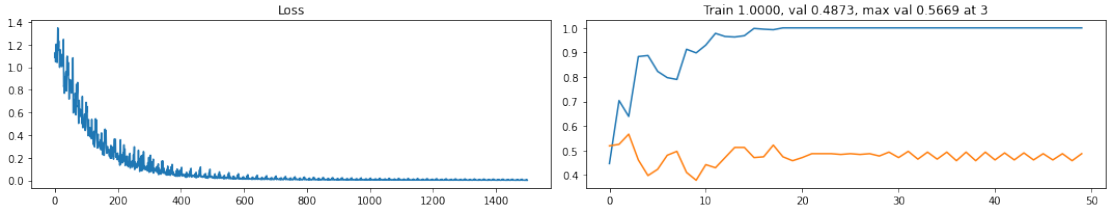


Рис. 32: Lf = CE, bs = 32, ep = 50, opt = Adam, lr = 0.1|0.01|0.001

Размерности скрытых слоёв по порядку: 32, 32, 32, 3. (Результаты см. Рис. 33)

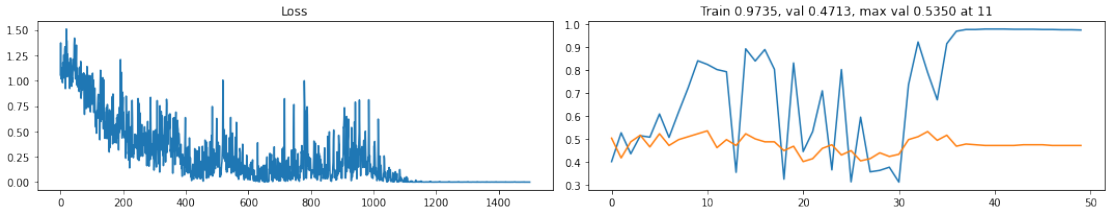


Рис. 33: Lf = CE, bs = 32, ep = 50, opt = SGD, lr = 0.1|0.01|0.001

Тесты (Результаты см. Рис. 34) для нейронной сети с 6 ***слоями, устроенными следующим образом (последовательно):***

- Первый слой - линейный, размерность - 120, нелинейность - ReLU, батч-нормализация после.
- Второй слой - линейный, размерность - 3.
- Третий слой - LSTM, скрытых слоёв - 64.
- Четвёртый слой - LSTM, скрытых слоёв - 32.
- Пятый слой - линейный, размерность - 120, нелинейность - ReLU, батч-нормализация после.
- Шестой слой - линейный, размерность - 3.

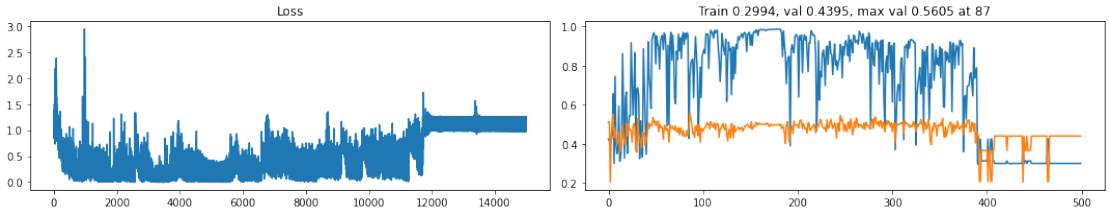


Рис. 34: Lf = CE, bs = 32, ep = 500, opt = SGD, lr = 0.1|0.01|0.001

Тесты (Результаты см. Рис. 35-38) для нелинейной нейронной сети с 6 ***слоями, устроенными следующим образом (последовательно):***

- Первый слой - линейный, размерность - 120, нелинейность - ReLU, батч-нормализация после.
- Второй слой - линейный, размерность - 120.
- Третий слой - RNN, размерность - 128.
- Четвёртый слой - RNN, размерность - 32.
- Пятый слой - линейный, размерность - 120, нелинейность - LeakyReLU, батч-нормализация после.
- Шестой слой - линейный, размерность - 3.

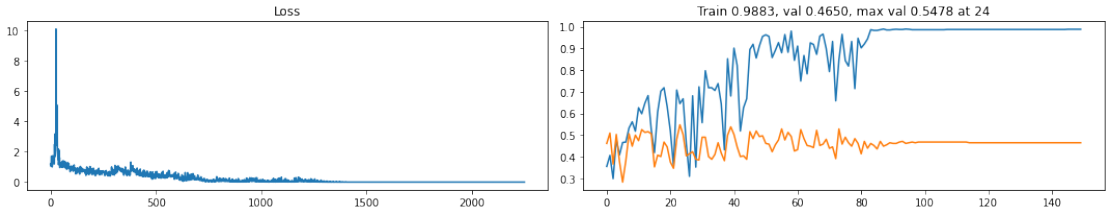


Рис. 35: Lf = CE, bs = 64, ep = 150, opt = SGD, lr = 0.1|0.01|0.001

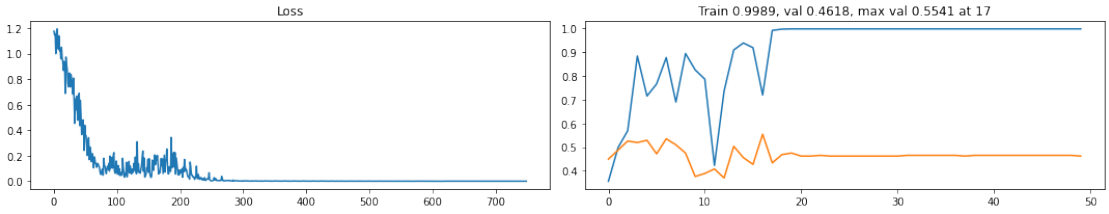


Рис. 36: Lf = CE, bs = 64, ep = 50, opt = Adam, lr = 0.1|0.01|0.001

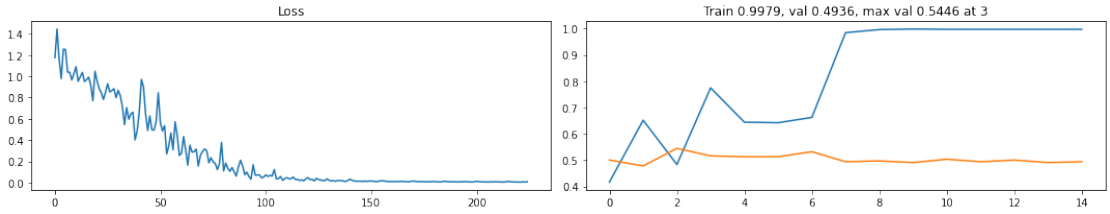


Рис. 37: Lf = CE, bs = 64, ep = 50, opt = Adagrad, lr = 0.1|0.01|0.001

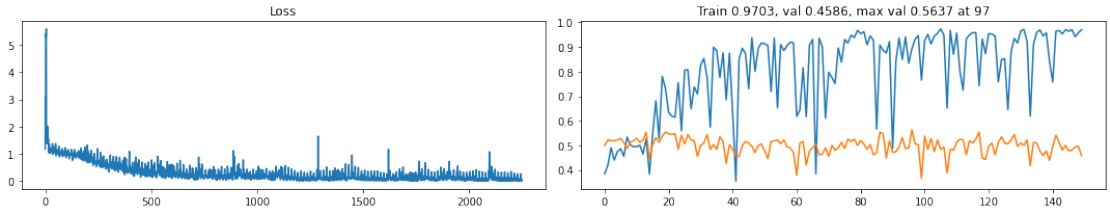


Рис. 38: Lf = CE, bs = 64, ep = 150, opt = RMSprop, lr = 0.1|0.01|0.001

Анализ полученных результатов

Для анализа результатов на объективность выбранной границы определения тренда bound рассмотрим распределение меток исходной выборки целиком и меток результатов, полученных в результате предсказания нейронной сетью на валидационной выборке: (см. Рис. 39-40)

Как можно заметить, распределение не перевешивает слишком сильно в какую-то сторону. На исходной выборке с такой границей данные разделены практически равномерно, а на валидационной выборке нейросеть не выдаёт преимущественно какое-то одно решение для всех входных данных. Перевес в сторону убывающего тренда в данном случае и правильность предсказаний 52.23% для трёх классов могут говорить, например, об убывающем состоянии на всём рынке в период времени соответствующий тестовой выборке. В действительности, так оно и есть, тестовая выборка на момент написания работы начинается с середины февраля 2020 года, когда во время пандемии коронавирусной инфекции COVID-19 произошёл крах фондового рынка, который

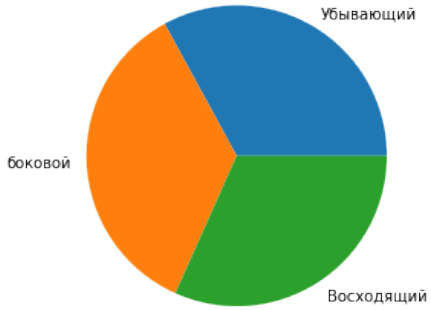


Рис. 39: Круговая диаграмма распределения меток всей исходной неразделённой выборки. Тренд размечен как: убывающий - 414, боковой - 444, восходящий - 398.

Рис. 40: Круговая диаграмма распределения меток результатов, полученных в результате предсказания нейронной сетью на валидационной части датасета на последней эпохе. Тренд размечен как: убывающий - 156, боковой - 90, восходящий - 68.

стал впоследствии одним из сильнейших обвалов мировых фондовых рынков в истории. [56]

Для сравнения процесса обучения на различных алгоритмах оптимизации рекуррентной нейросети были проведены эксперименты на схожих параметрах (Рис. 35-38). Из графиков видно, что алгоритмы Adam и AdaGrad (особенно) достаточно быстро переобучаются и застревают в локальном минимуме. Стоит отметить, что анализируя работу остальных алгоритмов, которые не переобучаются так быстро, читатель может заметить, что описанные выше локальные минимумы совсем не сильно отличаются от локальных минимумов на поверхности ошибок, в которые попадает процесс оптимизации для SGD и RMSprop. Подобное поведение подтверждается в работах [57, 58, 59].

Создание более глубокой сети с одинаковой шириной линейных слоёв не приводит к улучшению результата по сравнению с двумя внутренними линейными слоями. Это свидетельствует о том, что выразимости нейронной сети с двумя линейными слоями достаточно для таких закономерностей, которые

возникают на данных по всему фондовому рынку.

Замечание: Даже на переобученной нейронной сети, которая в своих адаптированных параметрах содержит практически полную информацию об обучающей выборке, результат на тестовой выборке достаточно высок и отличается в лучшую сторону от результата работы модели, расставляющей метки класса наугад равномерно. Это может означать, что закономерности на фондовом рынке цикличны, так как те же закономерности, что были на обучающей выборке «в прошлом», правильным образом описывают те, что на тестовой выборке «в будущем».

Выводы

Выбор граничного условия на тренд в 1% отклонения хорошо описывает движения цен и при этом не приводит к перевесу данных на сторону какого-либо класса.

Оптимумы, к которым сходится процесс оптимизации параметров нейронной сети, находятся приблизительно на одной горизонатльной гиперплоскости на поверхности ошибок.

Глубины двухслойной линейной нейронной сети хватает для выражения закономерностей. Некоторые архитектуры быстро переобучаются, но всё равно остаются конкурентны на тестовой выборке.

Дальнейшие направления развития

1. То, что определение границы изменения цены в 1% достаточно удачно задаёт разбиение обучающей выборки на классы, не гарантирует, что усреднённое определение тренда как изменение цены за день в ту или иную сторону (игнорирование движения цен внутри дня и сглаживание

реального поведения процесса) достаточно правильно описывает поведение рынка и является хорошей интерпретацией для учёта нейронной сетью во время обучения. Тренд можно определять не по предыдущему и текущему значениям, а по определённой последовательности предшествующих текущему значений, которые также могут выстраиваться в различные паттерны (похожие закономерные последовательные структуры), определив которые как отдельный класс, вероятно возможно добиться гораздо более разумных результатов. Внутри модели для предсказания трёхклассового движения цены (восходящий, нисходящий, боковой) можно сначала выполнять подзадачу классификации, где сеть обнаруживает, к какому из паттернов принадлежит данная точка данных, а затем с меткой паттерна определять один из трёх классов движения цены.

2. Важным направлением дальнейших исследований может послужить подзадача кластеризации рыночных эмитентов и дальнейшей обработкой в качестве входных векторов сети только некоторой подгруппы. (см. сноску в разделе Получение временных рядов.)

3. Использование более сложных архитектур могут поспособствовать более детальной обработке исходной информации, а также появлению отдельных слоёв, отвечающих за выявление осмысленных фрагментов информации внутри временного ряда. Например, какой-либо статистической динамической характеристики финансового временного ряда как нестационарного процесса.

4. Определение благоразумного расписания для параметра lr. Например, отдельное обучение по эпохам параметра lr, как параметра нейронной сети и последовательная адаптация. Манипуляции вокруг этого парамет-

ра помогут не оставаться в локальных оптимумах и быстрее сходиться к другим критическим точкам, которые могут быть горадо лучше.

Список литературы

[1] В.Н. Афанасьев, М.М. Юзбашев, “Анализ временных рядов и прогнозирование”, Учебник. — М.: Финансы и статистика, 2001. — 228 с.

[2] Анализ временных рядов, http://statsoft.ru/home/textbook/modules/sttimser.html#1general [04.03.2023]

[3] Discrete uniform distribution, https://en.wikipedia.org/wiki/Discrete_uniform_distribution [04.03.2023]

[4] “Python is a programming language that lets you work quickly and integrate systems more effectively”, https://www.python.org/ [04.03.2023]

[5] Andreas C. Müller & Sarah Guido, “Introduction to Machine Learning with Python”, Москва 2016-2017

[6] David W. Toth , Bruce Jones. Against the Norm: Modeling Daily Stock Returns with the Laplace Distribution. arXiv preprint arXiv:1906.10325, 25 Jun 2019.

[7] R. Pascanu, C. Gulcehre, K. Cho, and Y. Bengio. How to construct deep recurrent neural networks. arXiv preprint arXiv:1312.6026v5, 2014.

[8] Adam Balusik, Jared de Magalhaes, Rendani Mbuvha. Forecasting The JSE Top 40 Using Long Short-Term Memory Networks. arXiv preprint arXiv:2104.09855, 2021.

[9] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, Yoshua Bengio, “On the properties of neural machine translation: Encoder-decoder approaches”, arXiv preprint arXiv:1409.1259, 2014.

[10] Anna Choromanska, Mikael Henaff, Michael Mathieu, Gérard Ben Arous, Yann LeCun.The Loss Surfaces of Multilayer Networks. arXiv preprint arXiv:1412.0233v3, 21 Jan 2015.

[11] “Понимание различных функций потерь для нейронных сетей”, https://www.machinelearningmastery.ru/understanding-different-loss-functions-for-neural-networks-dd1ed0274718/ [04.03.2023]

[12] В.М. Федоров, “Курс функционального анализа”, Учебник. — СПб.: Издательство «Лань», 2005. — 352 с. — (Учебники для вузов. Специальная литература).

[13] “Understanding different Loss Functions for Neural Networks”, https://towardsdatascience.com/understanding-different-loss-functions-for-neural-networks-dd1ed0274718 [04.03.2023]

[14] Mean squared error, https://en.wikipedia.org/wiki/Mean_squared_error [04.03.2023]

[15] Hans-Peter Piepho, “A Coefficient of Determination (R2) for Linear Mixed Models”, arXiv preprint arXiv:1805.01124v1, 2018

[16] Dabao Zhang, “A Coefficient of Determination for Generalized Linear Models”, Department of Statistics, Purdue University, 2016

[17] Loss Functions in Machine Learning, https://medium.com/swlh/cross-entropy-loss-in-pytorch-c010faf97bab [04.03.2023]

[18] Johannes Lederer. Activation Functions in Artificial Neural Networks: A Systematic Overview. arXiv preprint arXiv:2101.09957, 25 Jan 2021.

[19] Exploring the Landscape, https://losslandscape.com/ [04.03.2023]

[20] Sebastian Ruder. An overview of gradient descent optimization algorithms. arXiv:1609.04747v2, 15 Jun 2017.

[21] Б.Т. Поляк, "Введение в оптимизацию": Наука. Главная редакция физико-математической литературы, 1983. — 384 с.

[22] Pedro Carvalho, Nuno Lourenço, Penousal Machado. Evolving Learning Rate Optimizers for Deep Neural Networks. arXiv preprint arXiv:2103.12623v1, 23 Mar 2021.

[23] "Why Momentum Really Works", https://distill.pub/2017/momentum/ [04.03.2023]

[24] Duchi, John, Hazan, Elad, and Singer, Yoram. Adaptive subgradient methods for online learning and stochastic optimization. The Journal of Machine Learning Research, 12:2121–2159, 2011.

[25] Tieleman, T. and Hinton, G. Lecture 6.5 - RMSProp, COURSERA: Neural Networks for Machine Learning. Technical report, 2012.

[26] Diederik P. Kingma, Jimmy Ba.Adam: A Method for Stochastic Optimization. arXiv preprint arXiv:1703.00887, 30 Jan 2017.

[27] https://neurohive.io/ru/osnovy-data-science/obratnoe-rasprostranenie/ [26.05.2022]

[28] “Разработка и применение нейронной сети для интеллектуального контент-анализа социальных сетей”, http://earchive.tpu.ru/bitstream/11683/49371/1/TPU574138.pdf [04.03.2023]

[29] Boris Hanin, Mark Sellke. Approximating Continuous Functions by ReLU Nets of Minimal Width. arXiv preprint arXiv:1710.11278v2, 10 Mar 2018.

[30] Shiyu Liang, R. Srikant. Why Deep Neural Networks for Function Approximation?. arXiv preprint arXiv:1610.04161v2, 3 Mar 2017.

[31] Patrick Kidger, Terry Lyons. Universal Approximation with Deep Narrow Networks. arXiv preprint arXiv:1905.08539v2, 8 Jun 2020.

[32] Matus Telgarsky. Benefits of depth in neural networks. arXiv preprint arXiv:1602.04485v2, 27 May 2016.

[33] Dmitry Yarotsky. Error bounds for approximations with deep ReLU networks. arXiv preprint arXiv:1610.01145v3, 1 May 2017.

[34] Paul Sooderlin.Prediction of stock return. University of St. Gallen, St. Gallen, Switzerland, 2001.

[35] Liu Guang, Wang Xiaojie, Li Ruifan. Multi-Scale RCNN Model for Financial Time-series Classification. arXiv preprint arXiv:1911.09359v1, 2019.

[36] Liu Guang, Wang Xiaojie, Li Ruifan. Multi-Scale RCNN Model for Financial Time-series Classification. arXiv preprint arXiv:1911.09359v1, 2019.

[37] М.Г. Семенов. Разработка генетического алгоритма для решения задачи кластеризации данных. Сибирский федеральный университет, 2016.

[38] Jain A., Murty M., Flynn P. Data clustering: A review. // ACM Computing Surveys, 1999. - Vol. 31, no. 3. - Pp. 264–323.

[39] К. В. Воронцов. Лекции по алгоритмам кластеризации и многомерного шкалирования. — М.: МГУ, 2007. — 18 с.

[40] М.Б. Лагутин. Наглядная математическая статистика.-М.: П-центр, 2003.

[41] И.Д. Мандель. Кластерный анализ. - М.: Финансы и Статистика, 1988.

[42] Shivaji P. Mirashe, N. V. Kalyankar. Cloud Computing. arXiv preprint arXiv1003.4074v1, 2010.

[43] Yandex Cloud, https://cloud.yandex.ru/docs/free-trial/ [04.03.2023]

[44] "NumPy - The fundamental package for scientific computing with Python", https://numpy.org/ [04.03.2023]

[45] pandas documentation, https://pandas.pydata.org/docs/index.html [04.03.2023]

[46] pandas.DataFrame class, https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html [04.03.2023]

[47] Matplotlib: Visualization with Python, https://matplotlib.org/ [04.03.2023]

[48] PyTorch, https://pytorch.org/ [04.03.2023]

[49] "PyTorch - ваш новый фреймворк глубокого обучения", https://habr.com/ru/post/334380/ [04.03.2023]

[50] Simplifying Finance, https://simfin.com/ [04.03.2023]

[51] Simple financial data API for Python, https://github.com/SimFin/simfin [04.03.2023]

[52] torch.utils.data.DataLoader class, https://pytorch.org/docs/stable/data.html [04.03.2023]

[53] Sergey Ioffe, Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. arXiv preprint arXiv:1502.03167v3, 2 Mar 2015.

[54] How to adjust learning rate, https://pytorch.org/docs/stable/optim.html#how-to-adjust-learning-rate [04.03.2023]

[55] Zhen Xu, Andrew M. Dai, Jonas Kemp, Luke Metz. Learning an Adaptive Learning Rate Schedule. arXiv preprint arXiv:1909.09712v1, 20 Sep 2019.

[56] 2020 stock market crash, https://en.wikipedia.org/wiki/2020_stock_market_crash [04.03.2023]

[57] Fort, Stanislav, Huiyi Hu, and Balaji Lakshminarayanan. "Deep ensembles: A loss landscape perspective."arXiv preprint arXiv:1912.02757 (2019).

[58] Izmailov, Pavel, et al. "Averaging weights leads to wider optima and better generalization."arXiv preprint arXiv:1803.05407 (2018).

[59] Anokhin, Ivan, and Dmitry Yarotsky. "Low-loss connection of weight vectors: distribution-based approaches."International Conference on Machine Learning. 2020.

Приложения

Ссылка на GitHub репозиторий с материалами по данной работе: https://github.com/kolmar7777/Forecasting-financial-time-series-using-a-neural-network-and-analysis-of-market-behavior

По всем вопросам касательно данной работы можно обращаться на почту:

nikolai.martynov@math.msu.ru