Санкт-Петербургский государственный университет

Избранные труды
весенней научно-практической
конференции по вопросам информатики,
математики, механики и астрономии

МАТ-МЕХ. НАУКА 2025

28 апреля – 3 мая 2025 г.
Санкт-Петербург

Санкт-Петербург
2025

УДК 004; 51; 531/534; 52
ББК 16; 22.1; 22.2; 22.6
М34

Печатается по рекомендации
кафедры системного программирования
Санкт-Петербургского государственного университета

Избранные труды весенней научно-практической конференции по вопросам информатики, математики, механики и астрономии «Мат-мех. Наука 2025». 28 апреля – 3 мая 2025 г. Санкт-Петербург — СПб.: Издательство ВВМ, 2025. — 186 с.
ISBN 978-5-9651-1633-1

Тематика сборника затрагивает широкий круг актуальных проблем теоретической и прикладной математики, информатики, механики и астрономии. Цели конференции: представление результатов и организация научного общения преподавателей, ведущих учёных и молодых исследователей, апробация результатов студентов и аспирантов, установление и укрепление научных связей между исследовательскими группами.

Для студентов и аспирантов естественно-научных специальностей.

Р е ц е н з е н т ы:
проф. д.ф.-м.н. А. Н. Терехов; доц., к.т.н. М. В. Абрамов;
проф., д.ф.-м.н. И. Г. Бурова; проф., д.ф.-м.н. Н. Э. Голяндина;
проф., д.ф.-м.н. С. М. Ермаков; проф., д.ф.-м.н. Н. К. Кривулин;
проф., д.ф.-м.н., чл.-корр. РАН Н. В. Кузнецов; проф., д.ф.-м.н. А. А. Макаров;
проф., д.ф.-м.н. Т.Н. Мокаев; проф., д.ф.-м.н., В. М. Рябов;
доц., к.ф.-м.н., Д. М. Лебединский; ст.преп., к.т.н. Ю. В. Литвинов;
доц., к.ф.-м.н. Д. В. Луцив; доц. к.ф.-м.н., А. Ю. Пономарева;
СТО, TheDenominator, Inc. к.т.н. С. Ю. Сартасов;
м.н.с. ФИЦ РАН, к.т.н. В. Ф. Столярова; доц., к.ф.-м.н. П. В. Шпилев;
стажёр-исследователь ИПМаш РАН Н. Л. Шанарова

Вероятностные графические модели, нечеткие системы, мягкие вычисления и социокомпьютинг, машинное обучение

Абрамов Максим Викторович
к.т.н., доцент кафедры информатики

Влияние гиперпараметров оптимизатора Lion на точность предсказания в классических датасетах

Баранов А.А., СПбГУ, Санкт-Петербург st107591@spbu.ru,
Михайлов Д.А., СПб ФИЦ РАН, Санкт-Петербург dam@dscs.pro

Аннотация

Оптимизатор Lion, разработанный в 2023 году, обладает уникальными механизмами обновления параметров, отличающимися от традиционных оптимизаторов SGD, Adam, что обуславливает актуальность его подробного изучения. В данной работе проводится исследование влияния гиперпараметров оптимизатора Lion на обучение нейронных сетей. Эксперимент ставится с использованием сеток гиперпараметров на датасетах Titanic, Mushroom, Spambase и модели двухслойной нейронной сети с ReLU и Dropout. Результаты эксперимента показали существенное влияние ряда гиперпараметров на итоговую модель. Полученные данные из исследования могут быть полезны при практическом применении оптимизатора Lion для глубокого обучения нейронных сетей.

Введение

Спрос на нейронные сети с каждым годом только растёт [1], и вместе с ним возрастают требования к их скорости обучения и эффективности [2] [5]. В свою очередь, алгоритмы оптимизации нейронных сетей позволяют достичь более высоких результатов в качестве ускорения обучения и повышения качества итоговой модели [3] [6].

Одним из таких алгоритмов оптимизации является Lion – оптимизатор, разработанный в 2023 году командой Google Brain с использованием автоматизированного поиска алгоритмов. Ключевое отличие Lion от более традиционных алгоритмов оптимизации, таких как Adam и SGD, заключается в использовании знака текущего градиента и момента в качестве обновления параметров, что позволяет достичь меньших затрат по памяти [4].

В рамках данной работы проводится исследование влияния гиперпараметров оптимизатора Lion на точность предсказания, а также сравнение Lion с популярным алгоритмом оптимизации AdamW, являющимся модификацией алгоритма Adam.

Сравнение Lion с AdamW

AdamW и Lion – это два современных оптимизатора, которые представляют разные подходы к обновлению параметров нейросети.

Оптимизатор AdamW является адаптивным методом обучения. Его ключевая идея лежит в вычислении первого момента m_t и второго момента v_t с целью последующего обновления направления и величины параметров алгоритма. В частности, вычисление параметра модели θ выглядит следующим образом:

$$\theta_t = \theta_{t-1} - \eta_t(\frac{m_t}{\sqrt{v_t} + \epsilon} + \lambda\theta_{t-1}),$$

где θ_{t-1} – параметр модели на предыдущем шаге; θ_t – параметр модели на текущем шаге; η_t – скорость обучения; m_t и v_t — соответственно первый и второй моменты градиента; ϵ – небольшое число для избежания деления на ноль; λ – скорость затухания весов.

В отличие от AdamW подход Lion является более эффективным и менее трудоёмким. Он заключается в отслеживании направления обновления. Его ключевое различие кроется в том, что он заменяет масштабированные значения градиента на их знак, следующим образом:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)\nabla_\theta f(\theta_t), \quad (1)$$

$$\theta_{t+1} = \theta_t - \eta_t(\text{sign}(m_t) - \lambda\theta_{t-1}), \quad (2)$$

где β_1 – коэффициент экспоненциального сглаживания градиента; $\text{sign}(m_t)$ — направление обновления (по знаку); f – функция потерь.

Характеристика	**Lion**	**AdamW**
Адаптивность	Нет(фиксированный шаг)	Да(используется второй момент)
Сглаживание	β_1, только знак	β_1, β_2
Затраты по памяти	Хранение β_1	Хранение β_1, β_2
Вычислительные нагрузки	Знак	Корень, деление

Таблица 1: Сравнение AdamW и Lion

Таким образом, алгоритм оптимизации Lion представляет собой эффективную альтернативу популярным алгоритмам оптимизации, таким как

AdamW и другие. Особый интерес вызывают его гиперпараметры: коэффициент экспоненциального сглаживания градиента(β_1), скорость обучения(η), скорость затухания весов(λ).

Экспериментальное исследование влияния гиперпараметров оптимизатора Lion на точность

В рамках работы был проведён эксперимент с целью оценки точности фиксированной модели, использующей оптимизатор Lion как с параметрами по умолчанию, так и с гиперпараметрами, подобранными с помощью перебора по сетке гиперпараметров.

В эксперименте была реализована нейросетевая модель на основе библиотеки PyTorch. Архитектура модели представляет собой простую полносвязную нейросеть, включающую следующие компоненты: входной слой; функция активации ReLU; слой Dropout; выходной слой.

Гиперпараметры модели были заданы заранее и оставались фиксированными на протяжении всех экспериментов. Единственными варьируемыми величинами были гиперпараметры оптимизатора Lion: β_1 – коэффициент экспоненциального сглаживания градиента; η_t – скорость обучения; λ – скорость затухания весов.

В исследовании использовались датасеты: Titanic, Spambase, Mushroom.

Titanic датасет состоит из 891 записи и 12 признаков. Ключевая цель модели, использующей данный датасет, определить, выжил ли конкретный человек или нет. Основные признаки датасета: пол человека; класс билета – значение данного поля один из вариантов: первый класс, второй класс, третий класс; возраст; стоимость билета; порт отправления – значение данного поля один из вариантов: Шербур, Квинстаун, Саутгемптон.

Spambase датасет состоит из 4601 записи и 57 признаков. Ключевая цель модели, использующей данный датасет, определить, является ли письмо спамом или нет. Основные признаки датасета делятся на несколько групп: частоты слов — частота возникновения наиболее характерных для спама слов в письме; частоты символов — содержание специальных символов в тексте; статистика по длине слов — различные длины последовательностей заглавных букв.

Mushroom датасет состоит из 8124 записей и 22 признаков. Ключевая цель модели, использующей данный датасет, определить, является ли гриб ядовитым. Все признаки датасета являются категориальными, они состоят из списка значений. Основные признаки: запах; цвет спор; цвет пластинок;

размер пластинок; образование синяков; тип ножки; цвет шляпки; среда обитания.

Экспериментальное исследование проводилось с использованием двух сеток гиперпараметров: одна – большего размера и предназначена для малого датасета, чтобы время обучения соответствовало обучению на больших датасетах с меньшими по размеру сетками.

Первая сетка использовалась для датасета Titanic, она состояла из 20 значений по каждому гиперпараметру оптимизатора Lion, что в сумме составляет 8000 комбинаций. Её значения варьировались следующим образом:

- коэффициент экспоненциального сглаживания градиента β_1: от 0.8 до 0.98;
- скорость обучения η_t: от $5 \cdot 10^{-5}$ до 10^{-2};
- скорость затухания весов λ: от 0 до 0.1.

Вторая сетка использовалась для датасетов Mushroom, Spambase, она состояла из 10 значений по каждому гиперпараметру оптимизатора Lion, что в сумме составляет 1000 комбинаций. Её значения варьировались следующим образом:

- коэффициент экспоненциального сглаживания градиента β_1: от 0.8 до 0.99;
- скорость обучения η_t: от 10^{-5} до 10^{-2};
- скорость затухания весов λ: от 0 до 0.1.

В рамках экспериментального исследования использовалась конфигурация оптимизатора Lion с гиперпараметрами, установленными по умолчанию:

- коэффициент экспоненциального сглаживания градиента β_1: 0.9;
- скорость обучения η_t: 10^{-4};
- скорость затухания весов λ: 0.

Также в качестве метрики качества была использована метрика Accuracy (точности) предсказания ключевой переменной в каждом из датасетов. Она подсчитывается с помощью следующей формулы:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN},$$

где TP — истинно положительные: модель правильно предсказала положительный класс, TN — истинно отрицательные: модель правильно предсказала отрицательный класс, FP — ложно положительные: модель ошибочно предсказала положительный класс, FN — ложно отрицательные: модель ошибочно предсказала отрицательный класс.

Эксперимент показал, что для средних и больших датасетов результаты, полученные с использованием гиперпараметров по умолчанию, отличаются от наилучшей комбинации, найденной в процессе поиска, не более чем на 0.5 процента. Так точность модели для датасета Spambase оказалась следующей:

- Lion с гиперпараметрами по умолчанию: 0.93268;
- Lion с лучшей комбинацией из сетки гиперпараметров: 0.93702;
- прирост точности: 0.00434.

Точность модели для датасета Mushroom, в отличие от датасета Spambase, снизилась, что, вероятно, обусловлено ограниченным размером используемой сетки гиперпараметров:

- Lion с гиперпараметрами по умолчанию: 0.98708;
- Lion с лучшей комбинацией из сетки гиперпараметров: 0.98154;
- прирост точности: -0.00554.

Однако в ходе эксперимента с использованием датасета Titanic, результаты улучшились более чем на 22.5 процента. Так точность модели на данном датасете оказалась следующей:

- Lion с гиперпараметрами по умолчанию: 0.67832;
- Lion с лучшей комбинацией из сетки гиперпараметров: 0.83217;
- прирост точности: 0.15385.

Заключение

Основной результат работы заключается в комплексном экспериментальном исследовании влияния ключевых гиперпараметров оптимизатора Lion на качество обучения нейронных сетей на прикладных задачах, что позволяет расширить понимание его практического потенциала в свете недавней публикации алгоритма.

Улучшение результатов в первую очередь обусловлено увеличением размера сетки гиперпараметров, и как следствие, увеличением количества возможных комбинаций гиперпараметров. На основании результатов эксперимента можно утверждать, что гиперпараметры оптимизатора Lion существенно влияют на точность модели, особенно в условиях ограниченного объёма данных. Это показывает важность настройки гиперпараметров оптимизатора Lion при работе с малыми датасетами.

Список литературы

[1] Stefan H., Conrad S., Sarvnaz K., Alexandra B., Claire N.: Artificial intelligence adoption in the physical sciences, natural sciences, life sciences, social sciences and the arts and humanities: A bibliometric analysis of research publications from 1960-2021 // Technology in Society. – 2023. – T. 74. – C. 102260.

[2] Alen J., Dubravko F., Juraj H., Hrvoje B.: Short-Term Photovoltaic Power Plant Output Forecasting Using Sky Images and Deep Learning // Energies. – 2023. – T. 16. – No. 14. – Pp. 5428.

[3] Pamadi V. N., Rastogi D. Optimizing Neural Network Performance Through Adaptive Learning Algorithms // International Journal of Research in all Subjects in Multi Languages [Subject: Computer Science]. 2025. Vol. 13. – Pp. 2321-2853.

[4] Xiangning C., Chen L., Da H. Symbolic Discovery of Optimization Algorithms. // 37th Conference on Neural Information Processing Systems (NeurIPS 2023). – 2023. – Vol. 36. – Pp. 49205-49233.

[5] Alexander D. G., Chris L., Matthew T.J.: Can Learning Optimization Make Reinforcement Learning Less Difficult? // 38th Conference on Neural Information Processing Systems (NeurIPS 2024). – 2024. – Vol. 37. – Pp. 5454-5497.

[6] Carlo A., Silvia S., Jakob N. F., Patrick R0., Yee W. T.: Meta-Learning Objectives for Preference Optimization // arXiv preprint arXiv:2411.06568. - 2024.

Обзор некоторых подходов к оптимизации нейронных сетей через эволюционные методы

Цветкова А.Ю., СПбГУ, Санкт-Петербург tsvetkova.anna.at6@gmail.com,
Михайлов Д.А., м.н.с. СПб ФИЦ РАН, Санкт-Петербург dam@dscs.pro

Аннотация

Оптимизация нейронных сетей остаётся одной из ключевых задач в машинном обучении. Помимо классических градиентных методов, активно развиваются альтернативные подходы, основанные на эволюционных принципах. Эволюционные алгоритмы позволяют эффективно подбирать веса нейросетей без использования производных, что делает их устойчивыми к шуму и локальным минимумам. В данной работе представлен обзор современных методов нейроэволюции, таких как NEAT, HyperNEAT, DeepNEAT, CoDeepNEAT и CoEGAN. Рассматриваются их принципы, архитектурные особенности и области применения.

Введение

Оптимизация нейронных сетей является активно развивающимся направлением в области машинного обучения: дополняются и совершенствуются существующие методы, разрабатываются новые. Наиболее широко используются классические градиентные методы, такие как стохастический градиентный спуск (SGD), метод импульсов (momentum), метод Нестерова, оптимизаторы Adam, AdaGrad, AdaDelta и другие [1]. Однако эти подходы обладают рядом ограничений: чувствительностью к шуму, склонностью к преждевременной сходимости, а также высокой зависимостью от настроек гиперпараметров. В качестве альтернативы были предложены эволюционные алгоритмы, которые осуществляют оптимизацию нейронных сетей без использования градиента. Такой подход менее подвержен локальным минимумам и обладает большей устойчивостью к нестабильным данным. Настоящая работа посвящена обзорному анализу методов, основанных на эволюционных принципах, и их применению в задачах построения и обучения нейросетей.

Нейроэволюция

Нейроэволюция [2] представляет собой метод обучения нейронных сетей, в основе которого лежат не классические методы оптимизации, такие как гра-

диентный спуск, а эволюционные алгоритмы. Нейроэволюционные алгоритмы можно условно разделить на две группы: алгоритмы, оптимизирующие и топологию сети и веса, и алгоритмы, которые работают только с оптимизацией весов. Например, более ранние вариации нейроэволюционных алгоритмов оптимизировали только веса, а топология сети оставалась фиксированной. Однако, структура сети сильно влияет на производительность, поэтому появилась потребность в методах, которые оптимизируют топологию. Структура нейронной сети определяет размер пространства поиска. Если топология сети слишком проста, есть риск не найти оптимальное решение. Методы, оптимизирующие и топологию и веса получили название топологические и весовые эволюционирующие искусственные нейронные сети (topology and weight evolving artificial neural network, TWEANN) [3].

NEAT

Нейроэволюционный алгоритм NEAT (NeuroEvolution of Augmenting Topologies) [2, 4, 5] был разработан в 2002 году Кеннетом Стэнли и Ристо Микулайненом. NEAT относится к тем алгоритмам, которые способны оптимизировать и топологию и веса нейронных сетей. Метод позволяет решать проблемы, которые возникают из-за особенностей нейроэволюции: благодаря историческим меткам NEAT обеспечивает корректную работу кроссинговера нейронных сетей с разной топологией, а механизм видообразования препятствует преждевременному вымиранию хороших вариантов конфигураций сетей. Важно также отметить, что NEAT начинает работать с простыми сетями, постепенно добавляя новые узлы и связи. Более подробно стоит разобрать каждую из особенностей алгоритма.

Историческая маркировка используется для решения проблемы конкурирующих конвенций - одной из проблем нейроэволюции. Так как структуры развиваются независимо друг от друга в разных сетях, существует более одного способа представления нейронных сетей. Потомство после скрещивания таких сетей может потерять функциональность. Во избежание такой ситуации, система должна уметь определять идентичные структуры. NEAT делает это при помощи исторических меток: каждому новому гену, добавленному в геном, присваивается уникальное инновационное число, которое увеличивается с каждой новой мутацией. Маркировки облегчают кроссинговер, идентифицируя гомологичные участки между различными сетями. Инновационное число каждого гена наследуется потомством, что делает возможным отслеживание эволюции сети [5].

Видообразованием называется процесс, при котором похожие сети груп-

пируются вместе и конкурируют только между собой, а не со всей популяцией. Это дает время новым сетям оптимизировать веса и улучшить производительность, прежде чем они будут сравнены с другими сетями. В NEAT сети группируются по видам на основе структуры их соединений. Топологическое сходство определяется с помощью выравнивания геномов на основе исторических меток и определения совпадающих, непересекающихся и избыточных генов среди индивидуумов [5].

- Совпадающие гены - это гены, которые одинаковые у двух сетей.
- Непересекающиеся гены - это гены, которые различаются между двумя сетями и расположены в середине генома.
- Избыточные гены - это гены, которые различаются между двумя сетями и расположены в конце генома.

Топологическое сходство вычисляется с помощью расстояния совместимости [5, 6]:

$$\delta = \frac{c_1 E}{N} + \frac{c_2 D}{N} + c_3 W$$

где E — число избыточных генов, D — число непересекающихся генов, W — средняя разница в весе совпадающих генов, N — число генов в большем геноме, а c1, c2, c3 — коэффициенты, определяющие важность трех факторов. На практике N принимается равным 1.

Более подробно рассмотрим процесс эволюции нейронной сети. NEAT начинает постепенно наращивать сложность сети, начиная с полностью связных сетей без скрытых узлов. Однако нейронная сеть модифицируется только в том случае, если добавление новых узлов и связей увеличит производительность сети. NEAT использует прямое кодирование: между фенотипом и генотипом существует взаимно-однозначное соответствие. Генотип состоит из двух типов генов: одни используются для кодирования узлов, другие - для кодирования связей. Гены узлов имеют поля, указывающие на тип узла: входной, выходной, скрытый. Гены связи содержат информацию об идентификаторах узлов, между которыми расположена связь, о весе соединения, а также о номере инновации, который позволяет найти соответствующие гены во время скрещивания.

HyperNEAT

HyperNEAT (Hypercube-based NeuroEvolution of Augmenting Topologies) [3] - одна из модификаций классического алгоритма NEAT, представленная

Кеннетом Стэнли в 2007 году. Особенностью метода HyperNEAT является проекция нейронной сети в гиперкубическое пространство, в котором каждому нейрону присваиваются координаты. Эти координаты подаются на вход в функцию CPPN (Compositional Pattern Producing Networks). CPPN реализует косвенное кодирование, которое не требует типичной стадии пошагового построения сети, в отличие от метода NEAT, в котором нейронная сеть постепенно эволюционирует.

Гиперкубическое пространство - это многомерное пространство, где каждая точка соответствует конкретной конфигурации нейронной сети. HyperNEAT хранит шаблон связей, где каждая пара кодирует соединение между двумя узлами и вычисляет четырехмерную функцию S = F(x1, y1, x2, y2), где (x1, y1) - координаты исходного узла, а (x2, y2) - координаты целевого узла [6].

Расстояние между узлами сети определяется с помощью линейной комбинации количества избыточных генов (I), непересекающихся генов (E), разницы в весе совпадающих генов W [5, 6]:

$$D = k_1 \cdot \frac{I}{Q} + k_2 \cdot \frac{E}{Q} + k_3 \cdot W$$

Благодаря HyperNEAT появилась возможность компактно кодировать большие нейронные сети маленькими геномами и изменять размер эволюционировавших нейронных сетей после завершения обучения.

DeepNEAT и CoDeepNEAT

Модель NEAT также была расширена для работы с большими пространствами поиска, такими как глубокие нейронные сети, такой расширенный метод - DeepNEAT и его улучшенная версия - CoDeepNEAT [7, 8].

DeepNEAT - это расширение NEAT, которое рассматривает целые слои как гены, а не отдельные нейроны. Основное внимание уделяется определению составов слоев, а не выбору нейронов. Таким образом, становится удобнее работать с более крупными и глубокими сетями.

DeepNEAT работает по стандартному генетическому алгоритму: создает начальную популяцию особей, каждая из которых представлена графом, и развивает их на протяжении поколений. В течение этих поколений индивидуумы мутируют путем добавления или удаления узлов и ребер из своих графов, изменения фиксируются при помощи исторической маркировки (аналогично NEAT). Хромосомы сравниваются в каждом поколении метрикой сходства, затем классифицируются по субпопуляциям - видам. Каждый

вид оценивается по общей приспособленности его особей, которая рассчитывается при помощи функции распределения приспособленности. Выжившие виды развиваются отдельно друг от друга посредством кроссинговера между особями одного вида, таким образом формируется следующее поколение.

В силу того, что узлы теперь представлены слоями, а не нейронами, необходимо хранить дополнительную информацию о слоях: тип (сверточный, плотный, рекуррентный), свойство (количество нейронов, размер ядра, размер шага, функция активации), тип соединения узлов. Для этого используют таблицу возможных гиперпараметров (хромосомную карту) для каждого узла и еще одну таблицу - таблицу глобальных параметров, применимых ко всей сети (скорость обучения, алгоритм обучения и предварительная обработка данных) [7].

CoDeepNEAT расширяет метод DeepNEAT. Он разделяет построение топологии на два разных уровня: module chromosomes и blueprint chromosomes.

- Module Chromosomes или модульные хромосомы - это графы, которые описывают небольшую структуру связанных слоев. Такие графы называют модулями
- Blueprint Chromosomes или хромосомы-чертежи - это графы, которые описывают, как модули соединяются между собой. На основе таких графов строится нейросеть, где в качестве узлов используются эти модули.

Таким образом, CoDeepNEAT работает с каждым видом хромосом по отдельности, а потом компонует их в сети.

В CoDeepNEAT эволюция осуществляется независимо для каждой популяции хромосом и требует индивидуальной оценки качества особей. Отличие генетического алгоритма в CoDeepNEAT от генетического алгоритма DeepNEAT состоит в том, что оценка присваивается и blueprint-хромосомам и модулям, которые в них используются. Затем эта оценка распределяется между соответствующими видами [7].

COEGAN

COEGAN является адаптацией алгоритма DeepNEAT для генеративно-состязательной сети (Generative adversarial network, GAN) [9]. В этой модели сочетаются нейроэволюционные и коэволюционные подходы для обучения. Модель создана для обеспечения более стабильного метода обучения и автоматического проектирования архитектур нейронных сетей.

GAN активно применяется в компьютерном зрении. В системе GAN одна из сетей генерирует образцы (генератор), а другая старается отличить правильные образцы от неправильных (дискриминатор). Дискриминатор и генератор являются глубокими нейронными сетями с заранее определенной архитектурой: структура сетей и гиперпараметры выбираются экспериментально, поэтому временные затраты увеличиваются. Генеративная сеть пытается создать новый образец, смешав несколько исходных образцов. Дискриминативная сеть обучается различать подлинные и поддельные образцы. Таким образом целью сети-генератора является повышение процента ошибок сети-дискриминатора, а целью сети-дискриминатора является улучшение точности распознавания. Существуют подходы, которые могут автоматизировать проектирование архитектур таких нейронных сетей.

Геном модели COEGAN может быть представлен в виде массива генов, где ген - описание слоя нейросети. Рассматриваются три типа слоев: линейные, сверточные и транспонированные сверточные. Такая последовательность генов преобразуется в архитектуру глубокой нейронной сети. Каждом гену сопоставляется случайным образом выбранная функция активации: ReLU, LeakyReLU, ELU, Sigmoid и Tanh [9]. Случайным параметром линейного слоя является количество входных параметров, а сверточного и транспонированного сверточного слоя - количество выходных каналов. Таким образом, гены имеют только функцию активации, выходные признаки и выходные каналы, подверженные мутации.

В COEGAN популяция состоит из двух субпопуляций генераторов Gi и дискриминаторов Di. Внутри каждой субпопуляции применяется механизм видообразования, используемый в NEAT, чтобы способствовать инновациям в каждой субпопуляции. В COEGAN оптимизируется только топология нейронной сети, а веса оптимизируются при помощи градиентного спуска. Это обусловлено тем, что количество внутренних параметров нейронной сети (в частности, к ним относятся и веса) достаточно велико и эволюция такой сети приведет к сильному увеличению вычислительной сложности [9].

Заключение

В данной работе были исследованы несколько методов оптимизации нейронных сетей, основанных на эволюционных принципах. Среди них методы NEAT, HyperNEAT, DeepNEAT, CoDeepNEAT и метод COEGAN. Одним из первых эволюционных методов оптимизации был NEAT, оптимизирующий топологию и веса нейронных сетей, на основе которого впоследствии были разработаны некоторые другие методы. К таким расширениям NEAT отно-

сится, например, HyperNEAT, способный работать с более сложными сетями и использующий вместо прямой эволюции метод косвенного кодирования. Другими вариациями NEAT стали алгоритмы DeepNEAT, CoDeepNEAT, способные работать с глубокими нейронными сетями. Отдельно был рассмотрен метод COEGAN, оптимизирующий генеративно-состязательные сети, используя сочетание коэволюционных и нейроэволюционных подходов.

Список литературы

[1] Kingma D. P., Ba J. Adam: A method for stochastic optimization // arXiv preprint arXiv:1412.6980. – 2014.

[2] Stanley K. O., Miikkulainen R. Evolving neural networks through augmenting topologies // Evolutionary Computation. – 2002. – Т. 10. – № 2. – С. 99–127.

[3] D'Ambrosio D. B., Gauci J., Stanley K. O. HyperNEAT: The first five years // Growing Adaptive Machines: Combining Development and Learning in Artificial Neural Networks. – 2014. – С. 159–185.

[4] Ibrahim M. Y., Babu G. S., Manogaran G., Chilamkurti N. Advances in neuroevolution through augmenting topologies – a case study // 2019 11th International Conference on Advanced Computing (ICoAC). – IEEE, 2019. – С. 111–116.

[5] Stanley K. O., Miikkulainen R. Efficient evolution of neural networks through complexification. – Texas: The University of Texas at Austin, 2004.

[6] Zhukabayeva T., Omarov B., Yessengaliyev A., Seidakhmetov B., Rakhmetov A., Fattakhov R. Network attack detection using neuroevolution of augmenting topologies (NEAT) algorithm // JOIV: International Journal on Informatics Visualization. – 2024. – Т. 8. – № 1. – С. 387–394.

[7] Bohrer J. S., Grisci B. I., Dorn M. Neuroevolution of neural network architectures using CoDeepNEAT and Keras // arXiv preprint arXiv:2002.04634. – 2020.

[8] Miikkulainen R., Liang J., Meyerson E., Rawal A., Fink D., Francon O., Raju B., Shahrzad H., Navruzyan A., Duffy N., Hodjat B. Evolving deep neural networks // Artificial Intelligence in the Age of Neural Networks and Brain Computing. – Academic Press, 2024. – С. 269–287.

[9] Costa V., Oliveira L. S., Bittencourt G. COEGAN: evaluating the coevolution effect in generative adversarial networks // Proceedings of the Genetic and Evolutionary Computation Conference. – 2019. – С. 374–382.

Вычислительная стохастика и статистические модели

Голяндина Нина Эдуардовна

д.ф.-м.н., доцент, заведующая кафедрой статистического моделирования

Ермаков Сергей Михайлович

д.ф.-м.н., профессор, профессор кафедры статистического моделирования

Исследование одной стохастической оценки центра симметрии в многомерном пространстве

Филатова А.А., СПбГУ, Санкт-Петербург rii.filatova@gmail.com,
Ермаков М.С., СПбГУ, Санкт-Петербург erm2512@mail.ru

Аннотация

Рассматривается новый робастный стохастический алгоритм для оценки центра симметрии многомерных распределений с выпуклыми центрально-симметричными поверхностями уровня плотности. Исследованы его временная сложность, скорость сходимости и робастность. Также рассмотрена модификация алгоритма, повышающая точность оценки. Представлены сравнительные численные эксперименты с существующими алгоритмами оценки многомерного параметра положения, включая медиану Тьюки, Оджа и геометрическую медиану. Поскольку в рассматриваемом классе распределений эти алгоритмы оценивают центр симметрии, сравнение является объективным.

Введение

В одномерном случае естественные оценки параметра положения (медиана, математическое ожидание) известны и их робастные версии легко определяются. Однако в многомерном случае при построении робастных версий оценок параметра положения статистики сталкиваются с серьезными трудностями. Известно, что не существует естественного обобщения одномерной медианы для многомерных данных [2, 3]. Это связано с тем, что в многомерном пространстве, в отличие от $\mathbb{R}^1$, нет естественного упорядочивания точек, что делает прямое перенесение одномерных концепций невозможным. В результате в литературе предлагаются различные подходы к определению многомерной медианы, такие как геометрическая медиана [5], медиана Оджа [6], концепции на основе глубины данных, в частности, медиана Тьюки [1] и другие [3].

Хотя единого общепринятого обобщения одномерной медианы на многомерный случай не существует, есть определенные свойства, которыми оно должно обладать. Например, для одномерных симметричных распределений медиана совпадает с центром симметрии. Естественно предположить, что то же должно быть верно и в многомерном пространстве. В этом случае мы в качестве обобщения понятия симметричного распределения рассмотрим распределения с выпуклыми центрально-симметричными поверхностями уровня плотности. Поэтому основой предлагаемого алгоритма является сообра-

жение, что для подобных распределений он должен оценивать центр симметрии.

Важным практическим ограничением многих существующих методов является их высокая вычислительная сложность [6, 7, 8], что существенно сужает область их применения. Для преодоления этого ограничения в работе предлагается стохастический подход, потенциально обеспечивающий как робастность, так и снижение вычислительных затрат.

Новый стохастический алгоритм для оценки многомерного параметра положения

Предположения алгоритма

Алгоритм 1 оценивает параметр положения для распределений, поверхности уровня плотности f которых являются выпуклыми центрально-симметричными, то есть для любого $\mathbf{x} \in \mathbb{R}^d$:

- $f(\mathbf{x} + \mathbf{x}_0) = \text{const}$ – выпуклое множество;
- $f(\mathbf{x}_0 + \mathbf{x}) = f(\mathbf{x}_0 - \mathbf{x})$, где $\mathbf{x}_0$ – центр симметрии.

В этом случае параметр положения определяется как значение, совпадающее с центром симметрии распределения $\mathbf{x}_0 \in \mathbb{R}^d$.

Из предположений следует, что алгоритм 1 может быть рассмотрен как метод оценки центра симметрии. Величина, которая получается в результате работы алгоритма, далее будет называться **стохастической медианой**.

Формулировка алгоритма

Алгоритм 1 сводит многомерную задачу к последовательности одномерных подзадач. На каждом шаге итерационного процесса выбирается случайное направление прямой, затем наблюдения проецируются на нее, и вычисляется медиана для одномерных точек-проекций. Такой подход позволяет последовательно приближаться к центру симметрии, при этом используя простые вычисления и обеспечивая эффективное решение задачи оценки параметра положения в многомерном случае.

Алгоритм 1 оценки многомерной медианы

1. Рассматриваем выборку $\mathbf{x}_1, \ldots, \mathbf{x}_n$ из распределения с функцией плотности f, которая удовлетворяет предположениям алгоритма.

2. Произвольным образом выбираем точку $\widehat{\mathbf{m}}_1$ — начальное приближение. Задаем погрешность вычисления ε.

3. Моделируем случайный вектор $\mathbf{u}_i$, равномерно распределенный на единичной сфере. Проводим прямую

$$l_i = \{\widehat{\mathbf{m}}_i + \lambda \mathbf{u}_i,\ \lambda \in \mathbb{R}\}.$$

4. Проецируем наблюдения $\mathbf{x}_1, \ldots, \mathbf{x}_n$ на прямую l_i, получаем точки-проекции $y_{i1}, \ldots, y_{in}$.

5. Находим медиану $\widehat{\mathbf{m}}_{i+1}$ точек $y_{i1}, \ldots, y_{in}$.

6. Алгоритм завершается, когда выполняется условие $\|\widehat{\mathbf{m}}_i - \widehat{\mathbf{m}}_{i+1}\| < \varepsilon$. Иначе увеличиваем счётчик шагов i на 1 и возвращаемся к шагу 3.

Чтобы уменьшить влияние случайности на финальную оценку, рассмотрим **модификацию** алгоритма 1. Вместо того чтобы использовать результат последней итерации, предлагается после достижения заданной точности ε сделать еще k итераций и в качестве финальной оценки взять среднее по этим k итерациям. Пусть до достижения заданной точности алгоритм 1 выполнялся N шагов, тогда

$$\widehat{\mathbf{m}}_{\text{avg}_k} = \frac{1}{k} \sum_{i=N+1}^{N+k} \widehat{\mathbf{m}}_i.$$

Усреднение последних итераций позволяет снизить разброс оценки и делает её более устойчивой.

Анализ временной сложности алгоритма

Сложность одного шага алгоритма 1 для выборки объемом n в пространстве размерности d выглядит следующим образом:

$$O(d) + O(nd) + O(n) + O(d) + O(d) = O(nd). \tag{1}$$

Это означает, что время выполнения одного шага алгоритма 1 линейно зависит от количества точек в выборке n и размерности пространства d. Если алгоритм 1 выполняется k итераций до сходимости, общая временная сложность алгоритма будет $O(knd)$, где k — число итераций.

Сравнение с некоторыми известными алгоритмами оценки медианы

В таблице 1 представлена сравнительная информация о временной сложности алгоритмов оценки различных обобщений медианы в пространстве размерности d.

Медиана	**Временная сложность**	**Источник**
Тьюки*	$O((d+k)n^2 + n^2 \log n)$	[4]
Оджа	$O(kdn^d \log n)$	[6]
Геометрическая	$O(knd)$	[5]
Стохастическая	$O(knd)$	(1)

Таблица 1: Временная сложность вычисления многомерной медианы

Как видно из таблицы 1, алгоритмы оценки геометрической и стохастической медианы обладают наилучшей асимптотикой, что делает их более предпочтительными для работы с большими выборками ($n \gg 1$) и в высокоразмерных пространствах ($d \gg 1$).

Также отметим, что в таблице 1 указана временная сложность аппроксимационного алгоритма для оценки медианы Тьюки (ABCDepth) [4], так как точный алгоритм для больших n и d является NP-сложной задачей [10]. Однако, используя технику рандомизированной оптимизации [9], можно добиться сложности $O(n^{d-1})$ для точного вычисления медианы Тьюки.

Для центрально-симметричных выпуклых распределений все рассмотренные медианы совпадают с центром симметрии. Это следует из их определений:

- **Медиана Тьюки** максимизирует глубину Тьюки — минимальное число точек по одну сторону от любой гиперплоскости, проходящей через нее [1]. В случае центральной симметрии максимальная глубина достигается в центре симметрии, так как любое смещение от него её уменьшает.

- **Медиана Оджа** минимизирует сумму объёмов симплексов по всем комбинациям возможных d точек выборки [6]. При центральной симметрии и выпуклости эта сумма минимизируется в центре симметрии, так как эти свойства гарантируют, что любое смещение увеличивает сумму объёмов.

- **Геометрическая медиана** минимизирует сумму евклидовых расстояний до всех точек выборки [5]. Для центрально-симметричного распре-

деления минимум достигается в центре симметрии, так как симметрия означает, что любое смещение увеличивает сумму расстояний.

Сходимость алгоритма для случая равномерного распределения в шаре

Предложение 1. Пусть $d \geqslant 2$ – размерность пространства, T_ε – среднее количество шагов алгоритма до попадания в ε-окрестность, r_0 – расстояние между начальным приближением и нулем.

Алгоритм 1 в случае равномерного распределения в шаре сходится, причем

$$\frac{T_\varepsilon}{\ln(r_0/\varepsilon)} \xrightarrow{\mathrm{P}} \frac{1}{-\mathrm{E}_d} \text{ при } \varepsilon \to 0,$$

$$\text{где } \mathrm{E}_d = \begin{cases} -\sqrt{\pi} \cdot \left[\ln 2 - \sum\limits_{i=1}^{d-2} \dfrac{(-1)^{i+1}}{i} \right], & d - \text{четное}, \\ \sqrt{2} \cdot \left[\ln 2 - \sum\limits_{i=1}^{d-2} \dfrac{(-1)^{i+1}}{i} \right], & d - \text{нечетное}. \end{cases}$$

Идея доказательства.

Рассмотрим идеальную ситуацию, когда наблюдений очень много и приближения к центру симметрии не зависят от объема и расположения точек выборки.

Пусть $r(\widehat{\mathbf{m}}_i) = r_i$ — расстояние между i-м приближением к центру симметрии $\widehat{\mathbf{m}}_i$ и истинным значением центра симметрии.

Тогда с помощью геометрических рассуждений можно показать, что последовательность из $-\ln r_i$ является процессом восстановления. Сходимость этого процесса может быть доказана с помощью элементарной теоремы восстановления [11]. Также из геометрических соображений вычисляется средняя величина шага этого процесса.

Таким образом, имеет место сходимость по вероятности нормированного числа шагов алгоритма к константе.

Из значения константы E_d следует, что с увеличением размерности пространства d алгоритм будет сходиться за большее количество шагов, что ожидаемо.

Исследование сходимости и робастности

Сходимость

На рисунке 1 представлены зависимости нормы ошибки от количества итераций для равномерного распределения в шаре и для стандартного многомерного нормального в случае размерностей 2-6.

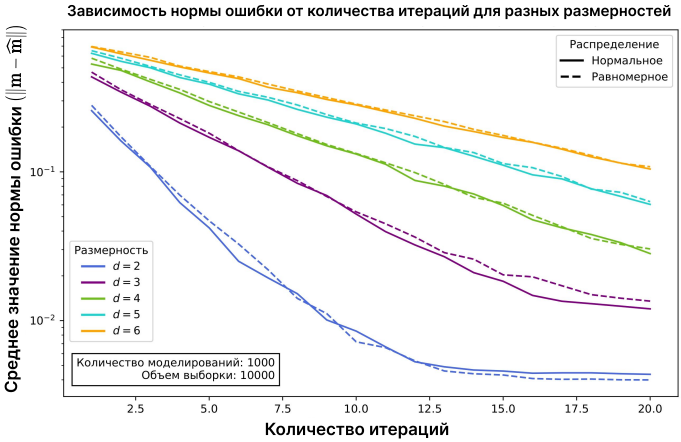


Рис. 1: Зависимость нормы ошибки от количества итераций. Делается 1000 моделирований, чтобы получить среднее значение нормы ошибки.

Результаты моделирования показывают, что при увеличении размерности скорость сходимости алгоритма замедляется и приближение к истинному значению происходит за большее количество итераций. Причем это характерно для обоих рассматриваемых распределений в равной степени. Это может говорить о том, что тип распределения оказывает незначительное влияние на общую тенденцию сходимости алгоритма.

Как видно из рисунка 1, на начальных итерациях скорость сходимости подобна линейной, после чего наблюдается выполаживание. Наиболее отчётливо этот эффект виден для размерностей 2 и 3, тогда как в случае больших размерностей выполаживание происходит за большее число итераций. На основании этого можно предположить, что скорость сходимости остаётся линейной при любой размерности, а выполаживание связано с тем, что выборка имеет конечный объем: на больших итерациях алгоритм может прыгать по элементам выборки, лежащим рядом с центром симметрии.

Робастность

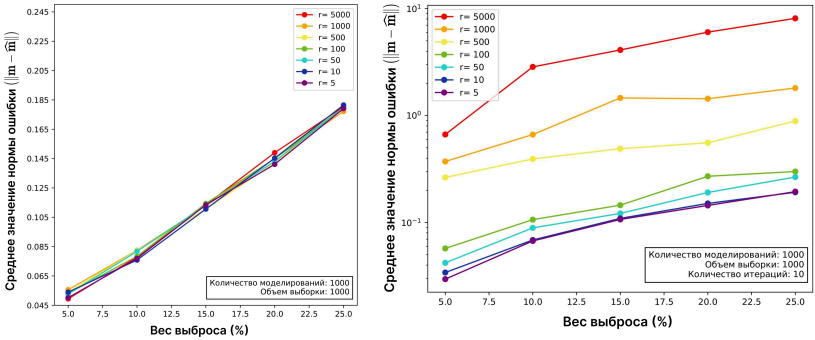
Случай фиксированной точности

Рассмотрим зависимость нормы ошибки от веса выброса для различных расстояний от истинного значения центра симметрии в случае фиксированной точности для двумерного стандартного нормального распределения. Вес выброса — это доля точек в выборке, которые заменяются на точку-выброс. Результаты моделирования представлены на рисунке 2 (a). Видно, что точность оценки медианы уменьшается практически линейно с увеличением веса выброса. Это означает, что точность пропорциональна весу выброса, и при более значимых выбросах оценка становится менее точной. Причем от удаленности от истинного значения это не зависит, так как для всех r результаты примерно одинаковы при заданной точности ε.

Случай фиксированного количества итераций

Изменим критерий остановки алгоритма 1: совершим ровно k итераций вне зависимости от того, достигнута ли точность ε, и запустим алгоритм вновь на тех же данных. Результаты моделирования представлены на рисунке 2 (b).

Из результатов эксперимента можно предположить, что удаленность выброса влияет на скорость сходимости алгоритма 1: для меньших значений r алгоритм сходится быстрее, чем для больших значений r.



(a) Фиксирована точность (b) Фиксировано количество итераций

Рис. 2: Зависимость нормы ошибки от веса выброса и удаленности от истинного значения. Часть выборки заменяется на точку с заданным весом на расстоянии $r \in \{5, 10, 50, 100, 500, 1000, 5000\}$ от истинного значения. Рассматриваются веса 5%, 10%, 15%, 20%, 25% от объема выборки. Алгоритм 1 запускается для $\varepsilon = 0{,}01$ (a) и для фиксированного количества итераций $k = 10$ (b).

Сравнение результатов работы с другими алгоритмами

Сравним работу предложенного алгоритма с популярными алгоритмами оценивания многомерного параметра положения.

Рассмотрим изменение ошибки оценки центра симметрии для разных объемов выборки в случае многомерного стандартного нормального распределения размерности 4. Результаты моделирования представлены на рисунке 3.

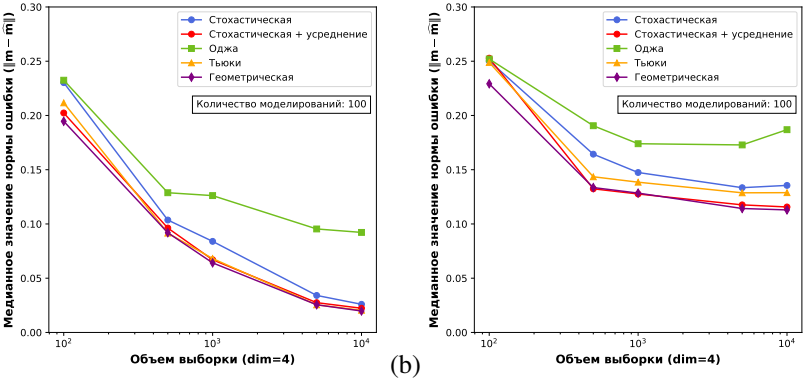


Рис. 3: Медианные значения нормы ошибок для выборок объема 100, 500, 1000, 5000 и 10000 из нормального распределения с (b) выбросами и (a) без них.

Исходя из вида рисунка 3 (a) можно предположить, что алгоритм 1 при заданных объемах выборки сопоставим по точности с геометрической медианой — конкурентом по временной сложности, а также с медианой Тьюки.

Заменим в выборке 5% наблюдений на выброс — точку $(8,\ 6,\ 5,\ 10)^{\mathbf{T}}$ и запустим алгоритмы вновь. Результаты представлены на рисунке 3 (b). Видно, что предложенный алгоритм 1 уже немного хуже оценивает параметр положения, однако точность все равно близка к геометрической медиане и медиане Тьюки.

Таким образом, результаты сравнения показывают, что новый алгоритм 1 при более подробном исследовании может конкурировать с известными алгоритмами оценки многомерного параметра положения.

Заключение

В рамках работы был реализован новый стохастический алгоритм робастного оценивания центра симметрии в многомерном пространстве. Исследо-

вана на модельном примере и строго доказана его скорость сходимости, а также временная сложность. Проведено сравнение с распространёнными алгоритмами, которое показало, что предложенный алгоритм и его модификация, повышающая точность оценки, дают сопоставимые результаты и не уступают по точности. Алгоритм эффективен для больших выборок и высоких размерностей, что делает его перспективной альтернативой существующим методам.

Список литературы

[1] Rousseeuw P., Struyf A. Computation of robust statistics: depth, median, and related measures // Handbook of Discrete and Computational Geometry. Second ed. – Chapman & Hall/CRC. – 2004. – P. 1279–1292.

[2] Huber P. J. Robust Statistics // Wiley. – 1981. – P. 308.

[3] Small C. G. A Survey of Multidimensional Medians // International Statistical Review. – 1990. – Vol. 58, No. 3. — P. 263–277.

[4] Bogićević M., Merkle M. Approximate Calculation of Tukey's Depth and Median With High-dimensional Data // Yugoslav Journal of Operations Research. – 2018. – Vol. 28, No. 4. – P. 475–499.

[5] Cohen M. B., Lee Y. T., Miller G., Pachocki J., Sidford A. Geometric median in nearly linear time // Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing. – 2016. – P. 9–21.

[6] Ronkainen T., Oja H., Orponen P. Computation of the multivariate Oja median // Developments in robust statistics: International Conference on Robust Statistics ICORS '01, Stift Vorau, Itävalta, heinäkuu 2001. – 2002. – P. 344–359.

[7] Langerman S., Steiger W. Optimization in arrangements // Proc. 20th Sympos. Theor. Aspects Comp. Sci. Berlin: Springer. – 2003. – Vol. 2607. P. 50–61.

[8] Aloupis G., Langerman S., Soss M., Toussaint G. Algorithms for bivariate medians and a Fermat-Torricelli problem for lines // Comput. Geom. – 2003. – Vol. 26. P. 69–79.

[9] Chan T.M. An optimal randomized algorithm for maximum Tukey depth // Proc. 15th Annu. ACM-SIAM Sympos. Discrete Algorithms (SODA '04). – 2004. – P. 430–436.

[10] Dyckerhoff R., Mozharovskyi P. Exact computation of the halfspace depth // Computational Statistics & Data Analysis. – 2016. – Vol. 98. – P. 19–30.

[11] Боровков А. А. Теория вероятностей. — М.: Эдиториал УРСС. – 1999. – 472 с.

О зависимости ошибки восстановления сигнала от ранга при возмущении в виде выброса

Храмов А.П., СПбГУ, Санкт-Петербург alexsandr.khramov@gmail.com,
Голяндина Н.Э., СПбГУ, Санкт-Петербург n.golyandina@spbu.ru

Аннотация

Рассматривается временной ряд состоящий из суммы сигнала и возмущения. Исследуется ошибка восстановления сигнала, полученная методом SSA. Для этого применяется теория возмущений, позволяющая исследовать ошибки первого порядка по величине возмущения. При этом численно показано, что результаты, полученные таким образом, асимптотически описывают поведение полной ошибки. Для возмущения в виде шума и выброса показано, что смещение оценки сигнала зависит только от выброса, а дисперсия — только от шума с нулевым математическим ожиданием. Получены явные формулы для первого порядка ошибки в случае экспоненциального сигнала и выброса (до диагонального усреднения). Численное моделирование показало, что MSE оценки сигнала пропорционально рангу сигнала при той же амплитудной модуляции.

Введение

Исследуется временной ряд $\mathsf{X} = (x_1, \ldots, x_N)$, $x_i \in \mathbb{R}$, длины N, состоящий из двух компонент: сигнала S и возмущения E. Возмущение E зачастую представляется в виде выброса или шума. Важной задачей является получение оценки сигнала $\widetilde{\mathsf{S}}$. Для этого может применяться метод SSA, представленный, например, в [1], и описанный нами в первой части работы. Интерес представляет исследование ошибки восстановления сигнала $\widetilde{\mathsf{S}} - \mathsf{S}$.

В работе применяются теория возмущений [2] и результаты, полученные в [3] для исследования первого порядка ошибки. В [4], применяя этот же подход, были представлены явные формулы для ошибки первого порядка оценки константного сигнала при возмущении в виде выброса. В данной работе получен явный вид матрицы первого порядка ошибки для экспоненциального сигнала $\mathsf{S} = \left(e^{mb}\right)_{m=1}^{N}$ и возмущения в виде выброса. Существующие робастные варианты метода SSA [5, 6] являются итеративными, в силу этого трудоемкими и также менее точными при отсутствии выбросов по сравнением с базовым вариантом. Поэтому исследование влияния выбросов на результат восстановления сигнала с помощью базового метода SSA представляет значительный интерес.

В последней части статьи представлены результаты численного моделирования, демонстрирующие поведение полной ошибки и ошибки первого порядка. В статье [4] была показана пропорциональность MSE оценки сигнала рангу для сигнала в виде суммы комплексных экспонент, имеющих постоянный модуль. В данной работе показан аналогичный результат для вещественного экспоненциально-гармонического сигнала.

Алгоритм SSA

Кратко опишем алгоритм SSA для получения оценки сигнала.

Входные параметры: временной ряд $\mathsf{X} = (x_1, \dots, x_N)$, длина окна L, ранг сигнала r.

Результат: восстановленный сигнал $\widetilde{\mathsf{S}}$.

1. **Вложение.** Построим $\mathbf{X} \in \mathbb{R}^{L \times K}$ — L-траекторную матрицу временного ряда X: $\mathbf{X} = \mathcal{T}_L\mathsf{X} = [X_1 : \dots : X_K]$, где $K = N - L + 1$, $X_i = (x_i, \dots, x_{i+L-1})^{\mathrm{T}} \in \mathbb{R}^L$.

2. **Разложение.** Построим сингулярное разложение матрицы $\mathbf{X}$:

$$\mathbf{X} = \sum_{k=1}^{\operatorname{rank}\mathbf{X}} \sqrt{\lambda_k} U_k V_k^{\mathrm{T}} = \sum_{k=1}^{\operatorname{rank}\mathbf{X}} \widehat{\mathbf{X}}_k,$$

   где U_k и V_k — левые и правые сингулярные векторы $\mathbf{X}$, $\sqrt{\lambda_k}$ — сингулярные числа в порядке убывания.

3. **Группировка.** Группируем слагаемые $\widehat{\mathbf{X}}_k$ из разложения, относящиеся к сигналу: $\widehat{\mathbf{S}} = \sum_{k=1}^{r} \widehat{\mathbf{X}}_k$.

4. **Диагональное усреднение.** Применяем проекцию на пространство ганкелевых матриц: $\widetilde{\mathbf{S}} = \Pi_{\mathcal{H}}\widehat{\mathbf{S}}$, и переходим к форме ряда: $\widetilde{\mathsf{S}} = \mathcal{T}_L^{-1}\widetilde{\mathbf{S}}$.

Ошибкой восстановления будем называть ряд $\mathsf{F} = \widetilde{\mathsf{S}} - \mathsf{S}$.

L-рангом временного ряда называют ранг его L-траекторной матрицы. Интерес представляют сигналы конечного ранга, т.е. сигналы, L-ранг которых не зависит от длины окна. В [1] подробнее описаны свойства рядов конечного ранга. Там же приведены примеры сигналов конечного ранга, в частности, экспоненциально-гармонический сигнал $\mathsf{S} = (s_m)_{m=1}^N$ с

$$s_m = \sum_{j=1}^{p} A_j e^{\alpha_j m} \cos(2\pi\omega_j m + \varphi_j),$$

где $A_j, \alpha_j \in \mathbb{R}$, $\omega_j \in [0, 1/2]$, $\varphi_j \in [0, 2\pi)$. Для суммы из одного слагаемого ранг равен 1, если $\omega_1 = 0$ или $\omega_1 = 0.5$, и равен 2 иначе.

Исследование ошибки первого порядка

Рассмотрим ряд $\mathsf{X} = \mathsf{S}(\delta)$, где $\mathsf{S}(\delta) = \mathsf{S} + \delta\mathsf{E}$; обозначим $\widetilde{\mathsf{S}}$ оценку сигнала методом SSA. Из [3] известно следующее представление (полной) ошибки восстановления:

$$\mathsf{F} = \mathsf{F}(\mathsf{S}, \delta\mathsf{E}) = \widetilde{\mathsf{S}} - \mathsf{S} = \mathcal{T}^{-1}\Pi_{\mathcal{H}}(\delta\mathbf{S}^{(1)} + \delta^2\mathbf{S}^{(2)}(\delta)).$$

Назовём ошибкой восстановления первого порядка часть полной ошибки, линейно зависящую от δ: $\delta\mathsf{F}^{(1)} = \mathcal{T}_L^{-1}\Pi_{\mathcal{H}}(\delta\mathbf{S}^{(1)})$. Примем $\delta = 1$. Теорема 2.1 из [3] даёт следующую формулу для $\mathbf{S}^{(1)} \in \mathbb{R}^{L\times K}$:

$$\mathbf{S}^{(1)} = \mathbf{S}^{(1)}(\mathsf{S}, \mathbf{E}) = -\mathbf{P}_0^{\perp}\mathbf{E}\mathbf{Q}_0^{\perp} + \mathbf{P}_0^{\perp}\mathbf{E} + \mathbf{E}\mathbf{Q}_0^{\perp}, \tag{1}$$

где $\mathbf{P}_0^{\perp}$ — проектор на пространство столбцов $\mathbf{S}$, $\mathbf{Q}_0^{\perp}$ — проектор на пространство строк $\mathbf{S}$.

Для сигналов ранга 1 формула (1) принимает вид:

$$\mathbf{S}^{(1)} = \mathbf{S}^{(1)}(\mathbf{E}, \mathsf{S}) = -(U^{\mathrm{T}}\mathbf{E}V)UV^{\mathrm{T}} + UU^{\mathrm{T}}\mathbf{E} + \mathbf{E}VV^{\mathrm{T}}. \tag{2}$$

Смещение и дисперсия оценки сигнала при возмущении в виде шума и выброса

Пусть $\mathsf{E} = \mathsf{E}_n + \mathsf{E}_o$, где E_n — случайный стационарный процесс с нулевым математическим ожиданием и достаточно малой дисперсией, E_o состоит из нулей за исключением значения $a \in \mathbb{R}$ на позиции k. Ряд E_n будем называть шумом, E_o — выбросом.

Утверждение 1. *Рассмотрим сигнал S, возмущение $\mathsf{E} = \mathsf{E}_n + \mathsf{E}_o$, где E_n — шум, E_o — выброс. Тогда*

$$\mathbb{E}\mathsf{F}^{(1)}(\mathsf{S}, \mathbf{E}) = \mathsf{F}^{(1)}(\mathsf{S}, \mathbf{E}_o),$$
$$\mathbb{D}\mathsf{F}^{(1)}(\mathsf{S}, \mathbf{E}) = \mathbb{D}\mathsf{F}^{(1)}(\mathsf{S}, \mathbf{E}_n),$$

где $\mathbf{E}_n = \mathcal{T}_L\mathsf{E}_n$, $\mathbf{E}_o = \mathcal{T}_L\mathsf{E}_o$.

Доказательство. Из линейности $\mathcal{T}_L$ следует, что $\mathbf{E} = \mathbf{E}_n + \mathbf{E}_o$. Из линейности формулы (1) по $\mathbf{E}$ получаем

$$\mathbf{S}^{(1)}(\mathsf{S}, \mathbf{E}) = \mathbf{S}^{(1)}(\mathsf{S}, \mathbf{E}_n) + \mathbf{S}^{(1)}(\mathsf{S}, \mathbf{E}_o),$$

а из линейности $\Pi_{\mathcal{H}}$ и $\mathcal{T}_L^{-1}$ следует, что

$$\mathsf{F}^{(1)}(\mathsf{S}, \mathbf{E}) = \mathsf{F}^{(1)}(\mathsf{S}, \mathbf{E}_n) + \mathsf{F}^{(1)}(\mathsf{S}, \mathbf{E}_o).$$

Так как слагаемое $\mathsf{F}^{(1)}(\mathsf{S}, \mathbf{E}_o)$ не случайно,

$$\mathbb{D}\mathsf{F}^{(1)}(\mathsf{S}, \mathbf{E}) = \mathbb{D}\mathsf{F}^{(1)}(\mathsf{S}, \mathbf{E}_n),$$

а пользуясь линейностью математического ожидания и тем, что $\mathbb{E}\mathsf{E}_n = \mathbf{0}$, получаем, что

$$\mathbb{E}\mathsf{F}^{(1)}(\mathsf{S}, \mathbf{E}) = \mathsf{F}^{(1)}(\mathsf{S}, \mathbf{E}_o).$$

□

Далее будут рассматриваться сигналы, для которых первый порядок ошибки адекватно описывает полную ошибку при росте N. Для таких сигналов утверждение 1 позволяет сделать следующий вывод: при возмущении в виде шума и выброса, шум отвечает за дисперсию оценки сигнала, а выброс — за смещение.

Утверждение 2. *Пусть E — возмущение в виде выброса a на позиции k. Тогда ошибка первого порядка $\mathsf{F}^{(1)}$ пропорциональна a.*

Доказательство. Определим матрицу $\mathbf{E}_1 \in \mathbb{R}^{L \times K}$ следующим образом:

$$\mathbf{E}_1[p, q] = \begin{cases} 1, & p + q - 1 = k; \\ 0, & \text{иначе.} \end{cases} \tag{3}$$

Тогда $a\mathbf{E}_1$ — траекторная матрица возмущения в виде рассматриваемого выброса. Подставляя её в формулу (1), получаем

$$\mathbf{S}^{(1)}(\mathsf{S}, a\mathbf{E}_1) = a\mathbf{S}^{(1)}(\mathsf{S}, \mathbf{E}_1),$$

откуда

$$\mathsf{F}^{(1)}(\mathsf{S}, a\mathbf{E}_1) = a\mathsf{F}^{(1)}(\mathsf{S}, \mathbf{E}_1).$$

□

В случае сигнала, для которого первый порядок ошибки адекватно описывает полную, и возмущения в виде шума и выброса, утверждение 2 даёт следующий результат: смещение оценки сигнала пропорционально значению выброса a. Значит достаточно рассматривать смещение в случае $a = 1$.

Вещественный экспоненциальный сигнал

Рассмотрим сигнал в виде вещественной экспоненты $\mathsf{S} = \left(e^{mb}\right)_{m=1}^{N}$, $b \in \mathbb{R}$ и возмущение E в виде выброса $a = 1$, на позиции k. Получим явный вид матрицы первого порядка ошибки восстановления сигнала S.

Сингулярные векторы этого сигнала имеют следующий вид

$$U = \frac{1}{v_L}\left(e^b, e^{2b}, \ldots, e^{Lb}\right)^{\mathrm{T}},\ V = \frac{1}{v_K}\left(e^b, e^{2b}, \ldots, e^{Kb}\right)^{\mathrm{T}}, \tag{4}$$

где

$$v_j^2 = e^{2b} + e^{4b} + \ldots + e^{2jb} = e^{2b}\,\frac{e^{2jb}-1}{e^{2b}-1},\ j \in \mathbb{N}.$$

Траекторная матрица возмущения имеет вид (3).

В формуле (2) обозначим слагаемые следующим образом

$$\mathbf{A} = U^{\mathrm{T}}\mathbf{E}VUV^{\mathrm{T}},\ \mathbf{B} = UU^{\mathrm{T}}\mathbf{E},\ \mathbf{C} = \mathbf{E}VV^{\mathrm{T}}$$

и подставим в них векторы U и V из (4) и матрицу $\mathbf{E} = \mathbf{E}_1$ из (3). Найдём элементы этих матриц. Имеем:

$$\mathbf{A}[p,q] = U^{\mathrm{T}}\mathbf{E}V(UV^{\mathrm{T}})[p,q] = \left(\frac{1}{v_L v_K}\sum_{m_1,m_2} e^{(m_1+m_2)b}\mathbf{E}[p,q]\right)\cdot\frac{e^{(p+q)b}}{v_L v_K} =$$

$$= \frac{e^{(p+q)b}}{v_L^2 v_K^2}\sum_{m_1+m_2-1=k} e^{(m_1+m_2)b} = \frac{w_k}{v_L^2 v_K^2}e^{(p+q+k+1)b},$$

где w_k — число элементов на k-ой диагонали, перпендикулярной главной, значит

$$w_k = \begin{cases} k, & k \le L; \\ L, & L < k < K; \\ N-k+1, & k \ge K. \end{cases}$$

Далее,

$$\mathbf{B}[p,q] = \sum_{m=1}^{L}(UU^{\mathrm{T}})[p,m]\mathbf{E}[m,q] = \begin{cases} \frac{1}{v_L^2}e^{(p-q+k+1)b}, & k+1-L \le q \le k; \\ 0, & \text{иначе}; \end{cases}$$

и

$$\mathbf{C}[p,q] = \sum_{m=1}^{K}\mathbf{E}[p,m](VV^{\mathrm{T}})[m,q] = \begin{cases} \frac{1}{v_K^2}e^{(-p+q+k+1)b}, & k+1-K \le p \le k; \\ 0, & \text{иначе}. \end{cases}$$

Обозначим $A = e^{(p+q+k+1)b}/(v_L^2 v_K^2)$. Итого получаем: если $k \le L$,

$$\mathbf{S}^{(1)}[p,q] = A\left(-k + v_L^2 e^{-2pb}\mathbb{1}_{[1,k]}(p) + v_K^2 e^{-2qb}\mathbb{1}_{[1,k]}(q)\right); \tag{5}$$

если $L < k < K$,

$$\mathbf{S}^{(1)}[p,q] = A\left(-L + v_L^2 e^{-2pb} + v_K^2 e^{-2qb}\mathbb{1}_{[k+1-L,k]}(q)\right); \tag{6}$$

если $k \ge K$, пусть $l = N - k + 1$, тогда

$$\mathbf{S}^{(1)}[p,q] = A\left(-l + v_L^2 e^{-2pb}\mathbb{1}_{[k+1-K,L]}(p) + v_K^2 e^{-2qb}\mathbb{1}_{[k+1-L,K]}(q)\right). \tag{7}$$

Сформулируем этот результат в виде теоремы.

Теорема 1. *Пусть сигнал* $\mathsf{S} = \left(e^{mb}\right)_{m=1}^{N}$, $b \in \mathbb{R}$, *возмущение* E *состоит из выброса* a *на позиции* k, $\mathbf{S}^{(1)}$ *— матрица первого порядка ошибки. Тогда* $\mathbf{S}^{(1)}/a$ *имеет вид:* (5)*, если* $k \le L$*;* (6)*, если* $L < k < K$*;* (7)*, если* $k \ge K$.

Численное моделирование

Численное сравнение первого порядка ошибки и полной ошибки восстановления сигнала

При исследовании вещественного экспоненциального сигнала с длиной ряда $N \to +\infty$, рассматривается последовательность сигналов $\mathsf{S}_N = \left(e^{mb_N}\right)_{m=1}^{N}$, где $b_N = C/N, C \in \mathbb{R}$. Для такой последовательности сигналов, последний элемент ряда $e^{Nb_N} = e^C = \text{const}$, а первый элемент ряда $e^{b_N} = e^{C/N} \longrightarrow 1$, при $N \longrightarrow +\infty$.

В Таблице 1 продемонстрированы результаты для сигналов $\mathsf{S}_N = \left(e^{mb_N}\right)_{m=1}^{N}$, $b_N = \ln(2)/N$, и выброса $a = 15$ на позиции k. Столбцы таблицы отвечают за разные длины N; строки — за различные позиции выброса k и длины окна L. Для каждой пары k, L указан максимальный модуль полной ошибки восстановления и модуль максимальной разницы между полной ошибкой и первым порядком ошибки. Как можно заметить, разность стремится к нулю при любых значениях L и k, но полная ошибка стремится к нулю только, если L пропорционально N.

Численное сравнение ошибки для больших рангов

Продемонстрируем поведение ошибки восстановления при возмущении в виде выброса в случае сигналов больших рангов.

N	100	1000	10000
$k = N/2 - 1, L = N/2$, полная	3.63×10^{-1}	3.98×10^{-2}	4.06×10^{-3}
$k = N/2 - 1, L = N/2$, разница	5.58×10^{-2}	7.38×10^{-4}	7.61×10^{-6}
$k = 19, L = 20$, полная	7.72×10^{-1}	7.51×10^{-1}	7.50×10^{-1}
$k = 19, L = 20$, разница	6.29×10^{-2}	9.67×10^{-3}	9.86×10^{-4}
$k = 19, L = N/2$, полная	4.79×10^{-1}	4.28×10^{-2}	4.17×10^{-3}
$k = 19, L = N/2$, разница	7.51×10^{-2}	1.27×10^{-3}	1.29×10^{-5}
$k = N/2, L = 20$, полная	7.51×10^{-1}	7.50×10^{-1}	7.50×10^{-1}
$k = N/2, L = 20$, разница	5.74×10^{-3}	5.12×10^{-6}	5.07×10^{-9}

Таблица 1: Максимальная разница между ошибкой I порядка и полной для выброса.

На Рис. 1 изображено поведение модуля ошибки восстановления сигнала при разных длинах окон L. На левом рисунке изображён случай вещественного экспоненциального сигнала $\mathsf{S}_N = \left(e^{mb_N}\right)_{m=1}^{N}$, $b_N = \ln(2)/N$, $N = 3999$, имеющего ранг 1, и выброса $a = 1$, на позиции $k = N/3$. Справа изображена ошибка для сигнала $s_m = e^{mb_N}(\cos(2\pi m/40) + \cos(2\pi m/20 + \pi/3))$, имеющего ранг 4. В Таблице 2 показано значение MSE оценки этих сигналов, умноженное на длину ряда, а также отношение MSE для ранга 4 к MSE для ранга 1. Как можно заметить, это отношение близко к 4.

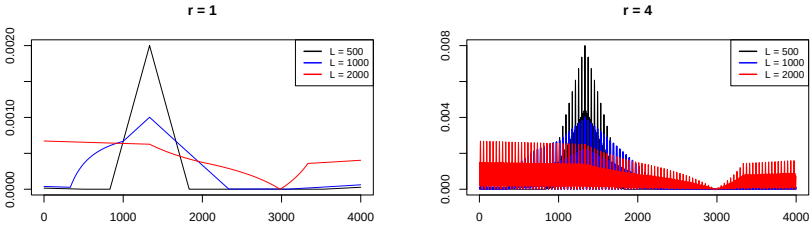


Рис. 1: Поведение ошибки восстановления при разных рангах.

Заключение

Получен явный вид матрицы первого порядка ошибки для экспоненциального сигнала при возмущении в виде выброса. Численно показано, что для такого сигнала первый порядок по величине выброса адекватно описыва-

L	$\mathrm{rank} = 1$	$\mathrm{rank} = 4$	Отношение
500	1.33e-03	5.34e-03	4.002
1000	7.35e-04	2.94e-03	4.001
2000	8.90e-04	3.56e-03	4.004

Таблица 2: MSE оценки сигнала ранга 1 и 4.

ет полную ошибку. Также численно показана зависимость MSE от ранга для сигнала в виде суммы экспоненциально-модулированных гармоник с одинаковым показателем экспонент, что делает теоретический результат потенциально применимым для гораздо более широкого класса сигналов, чем просто вещественные экспоненты.

Так как доказано, что при возмущении в виде суммы шума и выброса смещение первого порядка оценки сигнала зависит только от значения выброса, полученные результаты дают полезную информацию и для зашумленных рядов.

Список литературы

[1] Golyandina N., Nekrutkin V., Zhigljavsky A. Analysis of Time Series Structure: SSA and Related Techniques. — New York : Chapman&Hall/CRC, 2001.

[2] Kato T. Perturbation theory for linear operators. — Springer-Verlag, Berlin-Heidelberg-New York, 1966.

[3] Nekrutkin V. Perturbation expansions of signal subspaces for long signals // Statistics and Its Interface. — 2010. — Vol. 3. — P. 297–319.

[4] Golyandina N., Senov M., Khramov A. On errors of signal estimation using complex singular spectrum analysis // Engineering Proceedings. — 2025. — to be published.

[5] Trickett S., Burroughs L., Milton A. Robust Rank-Reduction Filtering for Erratic Noise // SEG International Exposition and Annual Meeting. — 2012. — Vol. All Days. — P. SEG–2012–0129.

[6] Kalantari Mahdi, Yarmohammadi Masoud, Hassani Hossein. Singular Spectrum Analysis Based on L1-Norm // Fluctuation and Noise Letters. — 2016. — Vol. 15, no. 01. — P. 1650009.

Пакет для численного обращения характеристических функций cfinversion

Имамутдинова Л. Р., Национальный исследовательский университет «Высшая школа экономики», Санкт-Петербург lrimamutdinova@edu.hse.ru,
Михайлов М.Д., Санкт-Петербургский государственный университет
mikhailovmikhaild@yandex.ru[1]

Аннотация

Данная работа посвящена пакету `cfinversion`, предоставляющему набор различных методов численного обращения характеристических функций для абсолютно непрерывных распределений, а также анализу этих методов применительно к некоторым распределениям.

Введение

Характеристические функции (х.ф.) являются важным инструментом для работы с вероятностными распределениями. Для случайной величины X с функцией распределения F_X её характеристическая функция определяется как

$$\varphi_X(t) = \mathbb{E}\left[e^{\mathrm{i}tX}\right] = \int_{-\infty}^{+\infty} e^{\mathrm{i}tx} dF_X(x).$$

Характеристическая функция всегда существует и однозначно задает распределение случайной величины. В случае, когда X имеет абсолютно непрерывное распределение с плотностью $f_X(x)$, характеристическая функция равна преобразованию Фурье плотности $f_X(x)$.

В ряде задач необходимо уметь восстанавливать плотность или функцию распределения по заданной характеристической функции. Примерами таких задач служат вычисление распределений времени ожидания в различных моделях теории очередей [1], расчет хвостовых вероятностей в распределениях из финансовой математики [3] и расчет p-value в некоторых статистических тестах [4].

В абсолютно непрерывном случае существует несколько вариаций формулы обращения характеристической функции. Общая формула обращения для восстановления плотности $f_X(x)$ по характеристической функции $\varphi_X(t)$ приведена в [6]. Для достаточно широкого класса абсолютно непрерывных

[1]Работа выполнена при поддержке СПБГУ (Pure ID 116636233)

распределений имеют место формулы обращения (1) и (2):

$$F_X(x) = \frac{1}{2} - \frac{1}{2\pi} \cdot \text{v.p.} \int_{-\infty}^{+\infty} \frac{e^{-\mathrm{i}tx}\varphi_X(t)}{\mathrm{i}t} dt, \tag{1}$$

$$f_X(x) = \frac{1}{2\pi} \cdot \text{v.p.} \int_{-\infty}^{+\infty} e^{-\mathrm{i}tx}\varphi_X(t) dt. \tag{2}$$

Формула (1) верна для всех а.н. распределений [8], формула (2) верна только когда соответствующее главное значение существует.[2]

Не всегда возможно провести обращение х.ф. аналитически, поэтому возникает потребность в применении численных методов. На данный момент существует несколько специализированных инструментов с открытым исходным кодом, которые позволяют решать задачу обращения х.ф.:

- библиотека `CharFunTool`, написанная на `R`. Также имеет реализацию на `MATLAB`. В этой библиотеке реализованы только два метода для обращений характеристических функций абсолютно непрерывных распределений: на основе обычного численного интегрирования и на основе DFT;

- библиотеки для дискретного преобразования Фурье (DFT) или его вариантов, которые можно использовать для аппроксимации интеграла в формуле обращения;

- библиотеки для численного интегрирования, которые содержат специальные алгоритмы для вычисления интегралов возникающих в формуле обращения (например `QAWF` в `QUADPACK`).

Обращение преобразования Фурье доступно также в проприетарных математических пакетах `MATLAB` и `Wolfram`, а библиотеки `ISML` и `NAG` предлагают специальные квадратуры для вычисления интегралов вида (1) и (2). В пакете `SAS` обращение х.ф. недоступно в качестве самостоятельной процедуры, но используется, например, при проверке временных рядов на стационарность.

Несмотря на то, что процедура `QAWF` и её аналоги подходят для численного обращения характеристической функции, более продвинутые методы обращения х.ф., предложенные в работах [1] и [2], на данный момент не имеют реализации. Следует также отметить, что процедура `QAWF` и её аналоги

[2] В общей формуле обращения из [6] интеграл в (2) понимается в смысле суммирования по Чезаро.

не оптимизированы для восстановления значений плотности/функции распределения сразу в нескольких точках. Поэтому разработка инструмента для обращения характеристических функций является актуальной задачей.

Описание пакета cfinversion

Пакет `cfinversion` написан на `Python` в рамках проекта `PySATL`. В пакете реализованы методы из пакета `CharFunTool` в соответствии с [1] («наивное» численное интегрирование) и [5] (на основе DFT). Также реализованы пять методов расчета функции распределения, предложенные в работе Г. Бохмана [2], которые базируются на техниках из теории аппроксимации. Для вычисления плотности с помощью этих методов в пакете используется численное дифференцирование.

Работу методов можно понимать как вычисление интегралов в (1) и (2) с помощью численных методов, заменяя интегралы по $\mathbb{R}$ на интегралы по отрезку $[-T;T]$. Метод «наивного» численного интегрирования применяет правило трапеции к этим формулам с $2N+1$ точкой и шагом δ. В [1] отмечается, что для интегралов такого вида правило трапеции подходит лучше, чем другие правила.

Методы Бохмана используются для вычисления функции распределения, плотность вычисляется через численное дифференцирование функции распределения. Все реализованные методы предполагают, что $\varphi''(0)$ существует. Для простоты реализации внутренних вычислений характеристические функции стандартизируются с помощью преобразования

$$\varphi_X(t) \mapsto \exp\left(-\mathrm{i}t\frac{\mu}{\sigma}\right)\varphi_X\left(\frac{t}{\sigma}\right),$$

где $\mu = \mathrm{i}\varphi'(0)$ и $\sigma = \sqrt{-\varphi''(0) - \mu^2}$. Величины μ и σ вычисляются с помощью численного дифференцирования на основе приближений, представленных в работе [7] или могут быть заданы пользователем при инициализации метода. Заметим, что хотя формально для корректной работы методов требуется существование $\varphi''(0)$, в [2] указывается, что эти методы также могут работать с распределениями с более тяжелыми хвостами.

В предположении, что $\varphi'_X(0) = 0$ и $\varphi''_X(0) = 1$, методы Бохмана работают следующим образом (запись $\sum'$ означает, что слагаемое, отвечающее нулевому индексу, отсутствует).

A. Вычисление суммы Римана (3) для оценки интеграла в формуле (1)

$$F_X(x) \approx \frac{1}{2} + \frac{\delta x}{2\pi} - \sum_{n=-N}^{N}{}' \frac{\varphi_X(\delta n)}{2\pi \mathrm{i} n} e^{-\mathrm{i}\delta n x}. \tag{3}$$

B. Обращение х.ф. $\varphi_X(t)C(t/T)$, где $C(t)$ — специальная «срезающая» характеристическая функция с носителем на $[-1;1]$:

$$F_X(x) \approx \frac{1}{2} + \frac{\delta x}{2\pi} - \sum_{n=-N}^{N}{}' C\Big(\frac{n}{N}\Big) \frac{\varphi_X(\delta n)}{2\pi \mathrm{i} n} e^{-\mathrm{i}\delta n x}.$$

C. Вычислять разность функции распределения $F_X(x)$ и функции распределения стандартного нормального закона $\Phi(x)$. В этом методе функция распределения вычисляется как $F_X(x) \approx \Phi(x) + H_N(x,\delta)$, где

$$H_N(x,\delta) = \sum_{n=-N}^{N}{}' \frac{\exp\big(-\frac{1}{2}\delta^2 n^2\big) - \varphi_X(\delta n)}{2\pi \mathrm{i} n} e^{-\mathrm{i}\delta n x}.$$

D. Использовать метод C с дополнительной поправкой на алиасинг:

$$F_X(x) \approx \Phi(x) + H_N(x,\delta) - \sum_{k=1}^{K-1} H_N(x + k d_1, \delta_1),$$

где $K, N \vdots K$ — свободный параметр и величины d_1, δ_1 определяются через N, K и δ.

E. Комбинация методов B и D:

$$F_X(x) \approx \Phi(x) + G_N(x,\delta) - \sum_{k=1}^{K-1} G_N(x + k d_1, \delta_1),$$

где, как и ранее, $K, N \vdots K$ — свободный параметр, числа d_1, δ_1 определяются через N, K и δ, и

$$G_N(x,\delta) = \sum_{n=-N}^{N}{}' C\Big(\frac{n}{N}\Big) \frac{\exp\big(-\frac{1}{2}\delta^2 n^2\big) - \varphi_X(\delta n)}{2\pi \mathrm{i} n} e^{-\mathrm{i}\delta n x}.$$

DFT-метод ускоряет вычисления в методе A, используя прием, описанный например в [5]. Полагая $\delta' = \frac{\delta}{2\pi}, x_m = \frac{m}{\delta'(2N+1)}, m = -N \ldots, N$ в (3), получаем следующее:

$$F_X(x_m) \approx \frac{1}{2} + x\delta' - \sum_{n=-N}^{n=N}{}' \frac{\varphi_X(2\pi\delta n)}{2\pi \mathrm{i} n} e^{-2\pi \mathrm{i} \frac{nm}{2N+1}}. \tag{4}$$

Для вычисления суммы (4) в точках x_m используется быстрое преобразование Фурье, а в остальных точках значение интерполируется. В отличие от реализации в `CharFunTool`, для DFT-метода в `cfinversion` используется кусочно-линейная интерполяция вместо PHCIP-интерполяции при вычислении функции распределения. Процедура вычисления плотности полностью аналогична процедуре вычисления функции распределения.

Для возможности добавления новых методов в будущем, реализации методов следуют общему интерфейсу `ContinuousInverter`, который имеет методы `fit`, `pdf` и `cdf`. Метод `fit` принимает на вход характеристическую функцию и выполняет необходимые предварительные вычисления. Методы `pdf` и `cdf` возвращают значения плотности и функции распределения соответственно в заданных точках.

Асимптотическая сложность реализованных методов представлена в таблице 1. В таблице C_{cf} обозначает сложность вычисления х.ф. в одной точке, асимптотики методов `cdf` и `pdf` приведены также для вычисления в одной точке.

Метод	`fit`(φ)	`cdf`(x) / `pdf`(x)
Методы Бохмана A-C	$\mathcal{O}(N \cdot C_{\mathrm{cf}})$	$\mathcal{O}(N)$
Методы Бохмана D, E	$\mathcal{O}(K \cdot N \cdot C_{\mathrm{cf}})$	$\mathcal{O}(K \cdot N)$
Naive Integration	$\mathcal{O}(N \cdot C_{\mathrm{cf}})$	$\mathcal{O}(N)$
DFT-метод	$\mathcal{O}\big(N(\log N + C_{\mathrm{cf}})\big)$	$\mathcal{O}(\log N)$

Таблица 1: Асимптотическая сложность методов

Численное исследование реализованных методов

Для исследования точности методов было проведено два численных эксперимента на четырех распределениях: стандартное нормальное $\mathcal{N}(0,1)$, Лапласа $\mathrm{Laplace}(0,1)$, равномерное на отрезке $U(0,1)$ и распределение квадра-

та величины, имеющей равномерное распределение $U^2(0,1)$. Эти распределения были выбраны, так как обладают рядом свойств, влияющими на поведение методов: быстро убывающая плотность, плотность с разрывом производной в 0, плотность с разрывами и плотность с сингулярностью соответственно.

Дизайн численных экспериментов

Первый эксперимент изучает, какие значения NLRE (negative log relative error) получаются в результате применения реализованных методов. NLRE определяется как

$$\mathrm{NLRE}(y_{\mathrm{estim}}(x_i), y_{\mathrm{true}}(x_i)) = -\frac{\log_{10}|y_{\mathrm{estim}}(x_i) - y_{\mathrm{true}}(x_i)|}{\log_{10}|y_{\mathrm{true}}(x_i)|},$$

где $x_1, \ldots, x_m$ — точки, в которых вычисляется плотность (функция распределения), $y_{\mathrm{estim}}(x)$ — значение плотности (функции распределения) в точке x, вычисленное через численное обращение х.ф., $y_{\mathrm{true}}(x)$ — теоретическое значение плотности (функции распределения) в точке x. Целая часть NLRE равна количеству корректно вычисленных значащих цифр. Для изучения поведения NLRE были выбраны следующие параметры методов: размер сетки дискретизации $N = 10^5$, шаг дискретизации $\delta = 10^{-2}$.

Во втором эксперименте изучается поведение ℓ^∞-нормы ошибки с увеличением размера N сетки дискретизации, где под ℓ^∞-нормой понимается величина

$$||y_{\mathrm{estim}} - y_{\mathrm{true}}|| = \max_{i=1,\ldots,m} |y_{\mathrm{estim}}(x_i) - y_{\mathrm{true}}(x_i)|.$$

Ожидается, что ℓ^∞-норма ошибки будет уменьшаться, так как она мажорируется sup-нормой функции ошибки $\mathrm{err}(x) = y_{\mathrm{true}}(x) - y_{\mathrm{estim}}(x)$. Отметим, что в общем случае sup-норма ошибки может сходиться к 0, в то время как минимум NLRE будет оставаться постоянным.

В этом эксперименте размер N сетки дискретизации последовательно принимает значения:

$$10^3;\ 2.5 \cdot 10^3;\ 5 \cdot 10^3;\ 7.5 \cdot 10^3;\ 10^4;\ 2.5 \cdot 10^4;\ 5 \cdot 10^4;\ 7.5 \cdot 10^4;\ 10^5.$$

При всех N шаг дискретизации составляет $\delta = 10^{-2}$.

В обоих экспериментах $m = 500$ и точки x_i образуют равномерную сетку либо на отрезке $[-2; 2]$ в случае распределений $\mathcal{N}(0,1)$ и $\mathrm{Laplace}(0,1)$, либо на отрезке $[0.01; 0.99]$ в случае распределений $U(0;1)$ и $U^2(0;1)$.

Результаты численного исследования

На рис. 1, 2, 3 и 4 для каждого метода изображены значения NLRE при вычислении плотности и функции распределения в точках x_i.

Для нормального распределения (рис. 1), наивное численное интегрирование позволяет достичь точности порядка 16 значащих цифр, что соответствует погрешности машинной арифметики. При этом методы Бохмана дают такой же результат для функции распределения, но для плотности распределения значительно уступают наивному интегрированию, видимо, из-за численного дифференцирования. Для распределения Лапласа (рис.2) при вычислении плотности разница между методами незначительна, однако для вычисления функции распределения метод на основе DFT, а также методы Бохмана B и E оказываются хуже на несколько значащих цифр. С другой стороны для равномерного распределения и его квадрата (рис.3 и рис.4 соответственно) методы B и E, наоборот, демонстрируют более высокую точность. При этом отсутствие сильных осцилляций на графиках NLRE для методов B и E указывает на то, что эти методы выдают менее осциллирующие результаты. Отметим также, что точность метода на основе DFT при вычислении функции распределения для $U^2(0,1)$ сравнима с точностью методов B и E, а для распределения $U(0,1)$ метод на основе DFT уступает методу E на одну значащую цифру в среднем.

Поведение ℓ^∞-норм ошибок при росте количества точек дискретизации изображено на рис. 5, 6, 7 и 8 в логарифмических осях. Результаты согласуются с первым экспериментом: для восстановления плотности распределения, в случае когда она непрерывна, методы Бохмана B и E, которые демонстрируют меньшую точность. С другой стороны, для распределений с разрывами и сингулярностями в плотности методы B и E могут давать выигрыш на один-два порядка при вычислении плотности при достаточно большом размере сетки дискретизации (см. рис. 7 и 8).

В целом можно заключить, что метод на основе DFT, предложенный в [5], хоть и асимптотически быстрее, но может показывать меньшую точность по сравнению с методами Бохмана. С другой стороны, метод на основе прямого численного интегрирования, рассмотренный в [1], проще в реализации и дает наилучшие результаты в случае, когда плотность непрерывна. Однако если в плотности есть разрывы или сингулярности, то наилучшую точность можно получить, используя методы Бохмана B и E для вычисления плотности. Для вычисления же функции распределения лучше также использовать эти методы или метод на основе DFT.

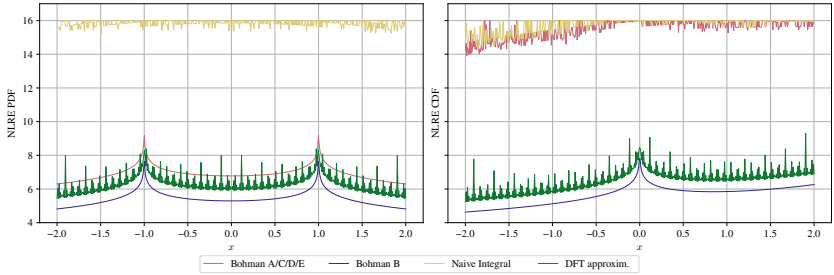


Рис. 1: Разброс NLRE при вычислении плотности и ф.р. для $\mathcal{N}(0,1)$ на $(-2;2)$.

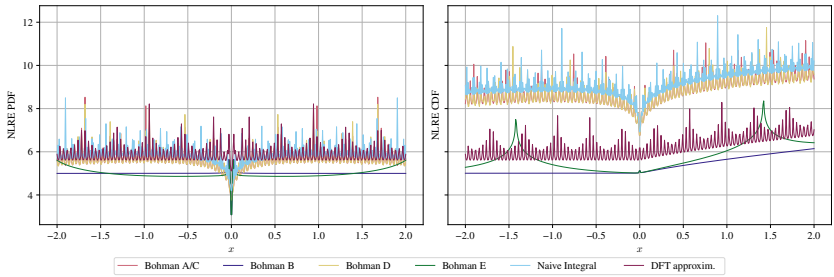


Рис. 2: Разброс NLRE при вычислении плотности и ф.р. для $\mathrm{Laplace}(0,1)$ на $(-2;2)$.

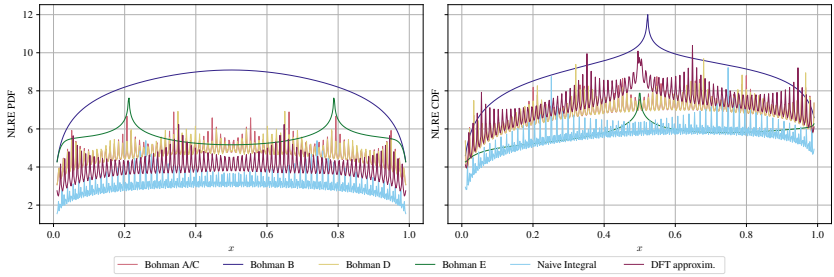


Рис. 3: Разброс NLRE при вычислении плотности и ф.р. для $U(0,1)$ на $(0.01;0.99)$.

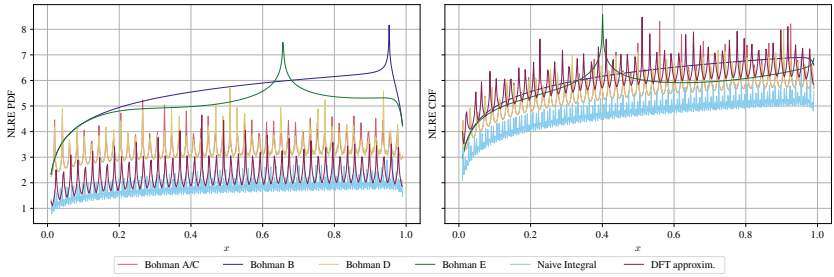


Рис. 4: Разброс NLRE при вычислении плотности и ф.р. для $U^2(0,1)$ на $(0.01;0.99)$.

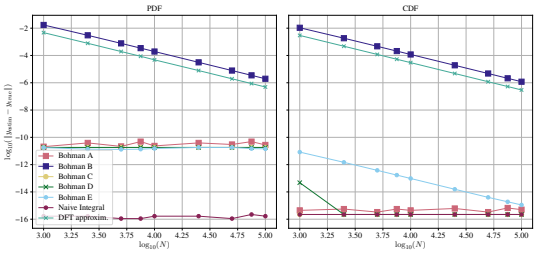


Рис. 5: Поведение ℓ^∞-нормы ошибки при увеличении N для $\mathcal{N}(0,1)$.

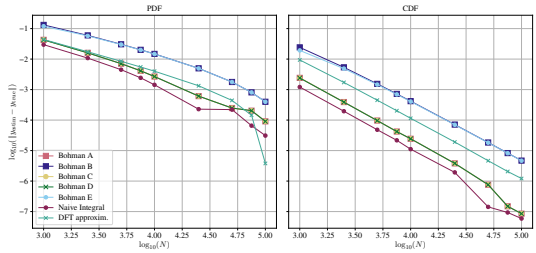


Рис. 6: Поведение ℓ^∞-нормы ошибки при увеличении N для $\mathrm{Laplace}(0,1)$.

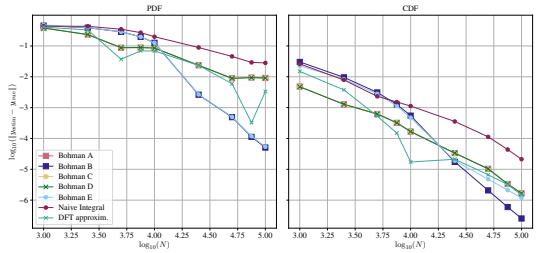


Рис. 7: Поведение ℓ^∞-нормы ошибки при увеличении N для $U(0,1)$.

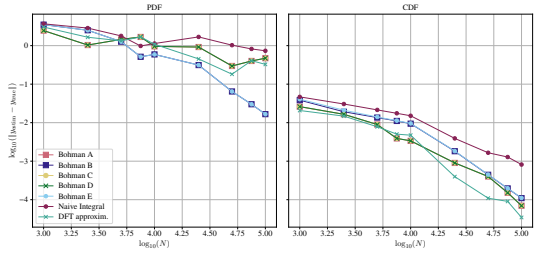


Рис. 8: Поведение ℓ^∞-нормы ошибки при увеличении N для $U^2(0,1)$.

Заключение

В рамках работы был реализован и опубликован[3] пакет `cfinversion`, включающий в себя методы для численного обращения характеристических функций абсолютно непрерывных распределений, предложенные в [2] и [5]. Для реализованных методов проведены численные эксперименты, показавшие, что методы B и E, предложенные в [2], дают более точные результаты по сравнению с методами на основе дискретного преобразования Фурье и простым численным интегрированием при вычислении значений плотности, имеющей разрывы или точки сингулярности.

Список литературы

[1] Abate J., Whitt W. The Fourier-series method for inverting transforms of probability distributions // Queueing Syst. 1992. Т. 10. № 1–2. С. 5–87.

[2] Bohman H. Numerical inversions of characteristic functions // Scandinavian Actuarial Journal. Informa UK Limited, 1975. Т. 1975, № 2. С. 121–124.

[3] Feng L., Lin X. Inverting Analytic Characteristic Functions and Financial Applications // SIAM J. Finan. Math. 2013. Т. 4. № 1. С. 372–398.

[4] Witkovský V. Computing the exact distribution of the Bartlett's test statistic by numerical inversion of its characteristic function // Journal of Applied Statistics. 2019. Т. 47. № 13–15. С. 2749–2764.

[5] Witkovsky V. Numerical inversion of a characteristic function: An alternative tool to form the probability distribution of output quantity in linear measurment models //Acta IMEKO. – 2016. – Т. 5. – №. 3. – С. 32-44.

[6] Lukacs E. Inversion formulae for characteristic functions of absolutely continuous distributions // The American Mathematical Monthly. – 1964. – Т. 71. – №. 1. – С. 44-47.

[7] Witkovský V. On the exact computation of the density and of the quantiles of linear combinations of t and F random variables // Journal of Statistical Planning and Inference. – 2001. – Т. 94. – №. 1. – С. 1-13.

[8] Gil-Pelaez J. Note on the inversion theorem // Biometrika. – 1951. – Т. 38. – №. 3-4. – С. 481-482.

[3] https://github.com/PySATL/pysatl-cfinversion, дата обращения 2025-06-05

О способах выделения сигнала на основе критерия Monte Carlo SSA

Потешкин Е.П., СПбГУ, Санкт-Петербург egor.poteshkin@yandex.ru,
Голяндина Н.Э., СПбГУ, Санкт-Петербург n.golyandina@spbu.ru

Аннотация

Рассматриваются методы анализа сингулярного спектра (SSA) и Monte Carlo SSA (MC-SSA) для решения задач обнаружения и выделения сигналов во временных рядах. Предложены три подхода к восстановлению сигнала: адаптивный, полуадаптивный и метод с фиксированной проекцией. Для оценки частоты сигнала используется метод MC-SSA. Проведен численный эксперимент, сравнивающий точность восстановления при различных уровнях шума, типах сигнала и значениях параметра δ, определяющего длину частотного интервала при отборе компонент. Результаты показывают, что полуадаптивный вариант является универсальным выбором, наиболее устойчивым к наличию умеренной амплитудной модуляции.

Введение

Рассмотрим следующую модель: $\mathsf{X} = \mathsf{S} + \mathsf{R}$, где X — наблюдаемый временной ряд, S — сигнал, R — шум, т.е. реализация некоторого стационарного процесса. В работе рассматривается две проблемы: проблема обнаружения сигнала S и проблема выделения сигнала при его наличии.

Для решения первой проблемы используется метод Monte Carlo SSA (MC-SSA) [1], проверяющий гипотезу $H_0 : \mathsf{S} = 0$, а для решения второй — метод анализа сингулярного спектра (singular spectrum analysis, SSA) [2, 3]. Один из шагов SSA подразумевает визуальный анализ для определения компонент сигнала, поэтому возникает потребность в автоматизации этого шага, этой проблеме посвящены, например, работы [4, 5, 6, 7]. Целью работы является определение подходов к автоматическому выделению слабых сигналов, обнаруживаемых критерием MC-SSA, и их сравнение по точности их выделения.

Метод SSA

Базовый алгоритм

Пусть $\mathsf{X} = (x_1, \ldots, x_N)$, $x_i \in \mathbb{R}$, — временной ряд длины N. Зафиксируем параметр $L, 1 < L < N$, называемый длиной окна и построим так называемую траекторную матрицу $\mathbf{X} = [X_1 : \ldots : X_K]$, состоящую из $K = N - L + 1$ векторов вложения $X_i = (x_i, \ldots, x_{i+L-1})^{\mathrm{T}} \in \mathbb{R}^L$.

Следующий шаг — разложение в сумму матриц единичного ранга $\mathbf{X} = \sum_{i=1}^{d} \mathbf{X}_i$. В базовом SSA используется сингулярное разложение матрицы $\mathbf{X}$, где столбцы $\mathbf{X}_i$ состоят из проекций столбцов матрицы $\mathbf{X}$ на порождаемые ею самой левые сингулярные векторы.

Далее компоненты полученного матричного разложения группируются на основе свойств левых сингулярных векторов, и каждая сгруппированная матрица преобразуется во временной ряд. Таким образом, результатом SSA является разложение временного ряда.

SSA с проекцией

Метод SSA использует адаптивный базис, но существует возможность зафиксировать некоторые компоненты разложения. Пусть $\mathbf{D} \in \mathbb{R}^{L \times m}$ — матрица, проекцию на столбцы которой мы хотим зафиксировать в разложении $\mathbf{X}$. Тогда SSA с проекцией отличается от базового алгоритма только шагом разложения:

1. В случае, если столбцы матрицы $\mathbf{D}$ не ортонормированны, $\mathbf{D}$ приводится к нужному виду путем ортогонализации Грамма-Шмидта.
2. Вычисляется матрица $\mathbf{C} = \mathbf{D}\mathbf{D}^{\mathrm{T}}\mathbf{X}$.
3. Вычисляется матрица $\mathbf{X}^{\star} = \mathbf{X} - \mathbf{C}$.
4. Матрица $\mathbf{X}^{\star}$ раскладывается в сумму матриц ранга 1.

Метод Monte Carlo SSA

Рассмотрим задачу поиска сигнала во временном ряде. Модель временного ряда имеет вид

$$\mathsf{X} = \mathsf{S} + \boldsymbol{\xi},$$

где S — сигнал, $\boldsymbol{\xi}$ — стационарный процесс с нулевым средним. Тогда нулевая гипотеза $H_0 : \mathsf{S} = 0$ и альтернатива $H_1 : \mathsf{S} \neq 0$.

Зафиксируем длину окна L и обозначим траекторную матрицу ряда $\boldsymbol{\xi}$ как $\boldsymbol{\Xi}$. Рассмотрим вектор $W \in \mathbb{R}^L$ единичной длины, называемый проекционным вектором. Введем величину

$$p = \left\| \boldsymbol{\Xi}^{\mathrm{T}} W \right\|^2 .$$

Статистикой критерия является величина

$$\widehat{p} = \left\| \mathbf{X}^{\mathrm{T}} W \right\|^2 .$$

Распределение статистики критерия оценивается с помощью моделирования согласно нулевой гипотезе, отсюда и название метода.

Если вектор W — синусоида с частотой ω, то $\widehat{p}$ отражает вклад частоты ω в исходный ряд. Так как частота ожидаемого сигнала неизвестна, то необходимо рассматривать несколько векторов W_k, $k = 1, \ldots, H$. Решение возникающей при этом проблемы множественного тестирования рассматривается в [8]. Гипотеза об отсутствии сигнала отвергается, если хотя бы для одного вектора $W = W_k$ значение $\widehat{p}$ оказывается значимым.

Важной частью метода MC-SSA является способ выбора векторов W_k. В данной работе в качестве векторов для проекции берутся косинусы с равноотстоящими частотами $\omega_k = k/(2L)$, $k = 1, \ldots, L$. В этом случае можно говорить о значимых частотах, присутствующих в сигнале.

Подходы к выделению сигнала

Для ряда X длины N и $0 \leqslant \omega_1 \leqslant \omega_2 \leqslant 0.5$ определим меру, следуя [4]

$$T(\mathsf{X}; \omega_1, \omega_2) = \frac{1}{\|\mathsf{X}\|^2} \sum_{k:\omega_1 \leqslant k/N \leqslant \omega_2} I_N(k/N),$$

где I_N — периодограмма ряда X. Величину $T(\mathsf{X}, \omega_1, \omega_2)$ можно рассматривать как долю вклада частот, содержащегося в интервале $[\omega_1, \omega_2]$.

В данной работе будем считать, что сигнал представляет из себя экспоненциально-модулированную гармонику:

$$\mathsf{S} = \{ A e^{\alpha n} \cos(2\pi\omega n) \}_{n=1}^{N} ,$$

где $\omega \in (0, 0.5)$. Пусть $\hat{\omega}$ — оценка ω. Обозначим

$$D_1 = \begin{pmatrix} \cos(2\pi\hat{\omega}1) \\ \dots \\ \cos(2\pi\hat{\omega}L) \end{pmatrix}, \; D_2 = \begin{pmatrix} \sin(2\pi\hat{\omega}1) \\ \dots \\ \sin(2\pi\hat{\omega}L) \end{pmatrix} \in \mathbb{R}^L.$$

Рассмотрим следующие варианты выделения сигнала S по частоте $\hat{\omega}$:

1. «adaptive»: применить SSA и выбрать первые две компоненты разложения, у которых мера T на интервале $[\hat{\omega} - \delta, \hat{\omega} + \delta]$, $\delta > 0$, больше некоторого порога $T_0 \in [0, 1]$;

2. «semi-adaptive»: применить SSA с проекцией с $\mathbf{D} = D_1 \in \mathbb{R}^{L\times 1}$ и выбрать, помимо компоненты, соответствующей вектору D_1, первую компоненту разложения, у которой мера T на интервале $[\hat{\omega}-\delta, \hat{\omega}+\delta]$, $\delta > 0$, больше некоторого порога $T_0 \in [0, 1]$;

3. «fixed»: применить SSA с проекцией с $\mathbf{D} = [D_1 : D_2] \in \mathbb{R}^{L\times 2}$ и выбрать компоненты разложения, соответсвующие векторам D_1, D_2.

Оценивать частоту ω будем с помощью MC-SSA:

1. Найти индекс наиболее значимой частоты, т.е. $k = \operatorname{argmax}_i(\widehat{p}_i - c_i)$, где c_i — верхняя граница доверительного интервала для $\widehat{p}_i$;

2. Вычислить значение $\hat{\omega}$ как взвешенное среднее частот $\omega_{k-1}, \omega_k, \omega_{k+1}$ с весами $w_i = \max(0, \widehat{p}_i - c_i)$;

Такой способ оценки позволяет получить более точную оценку ω в случае, когда она не попадает в решетку $k/(2L)$.

Численное сравнение подходов

Проведем численный эксперимент с целью понять, какой из предложенных способов восстановления сигнала наиболее точен. Пусть $N = 99$, процесс $\boldsymbol{\xi}$ — модель AR(1) с параметрами $\phi = 0.7$, $\sigma^2 \in \{0.2, 0.4, 0.6, 0.8, 1\}$. Для SSA $L = 50$, для MC-SSA $L = \widetilde{L} = 40$ (выводы устойчивы к выбору длины окна). В вариантах «adaptive» и «semi-adaptive» $\delta = 0.025$ и $T_0 = 0.5$. Рассмотрим два типа сигнала S, один из которых является частным случаем другого:

1. $\alpha = 0$, $A = 1$ — гармоника с постоянной амплитудой.

2. $\alpha = 0.05$, $A = 0.025$ — экспоненциально-модулированная гармоника.

Возьмем $\omega = 0.115$. Заметим, что при таком выборе частоты сигнала $\widetilde{L}\omega$ не целое, а значит ω не попадает в решетку $k/(2\widetilde{L})$.

На рис. 1 изображена зависимость MSE восстановления сигнала от дисперсии белого шума σ^2. По графикам видно, что в случае постоянной амплитуды ($\alpha = 0$) выигрывает вариант «fixed», однако в случае непостоянной амплитуды фиксированный базис оказывается наихудшим. Полуадаптивный базис, являясь неким компромиссом между адаптивным и фиксированным базисами, оказывается вторым по точности в случае $\alpha = 0$ и сравнимым с адаптивным в рассмотренном случае $\alpha \neq 0$. При увеличении $|\alpha|$, начиная с какого-то момента, фиксированная половина базиса ухудшает восстановление сигнала.

Теперь посмотрим, как будут изменяться ошибки при уменьшении/увеличении δ для фиксированного $\widetilde{L}$. На рис. 2 δ уменьшена, а на рис. 3 увеличена в два раза ($\delta = 0.0125$ и 0.05 соответственно). Из этих графиков видно, что слишком маленькое δ приводит к ухудшению точности адаптивного и полуадаптивного вариантов в случае $\alpha \neq 0$. Связано это с тем, что частота экспоненциально-модулированной гармоники, в отличие от гармоники с постоянной амплитудой, всегда растекается по спектру и чем больше абсолютное значение показателя экспоненты α, тем сильнее это растекание. Увеличение δ в два раза не привело к значительному изменению точности методов, однако, если и дальше увеличивать δ, ошибки, как и в случае слишком маленького δ, опять возрастут.

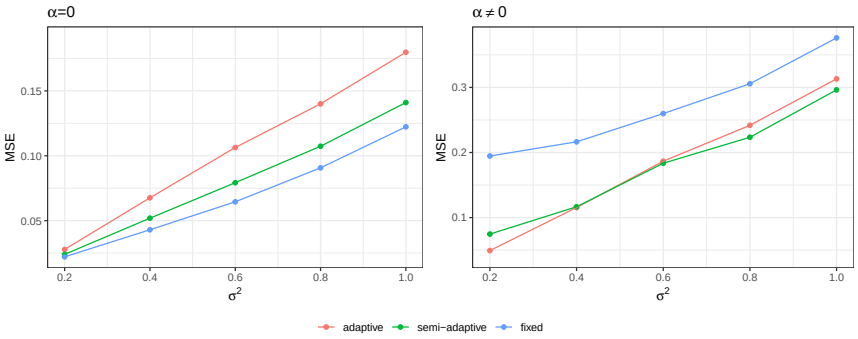


Рис. 1: MSE восстановления сигнала ($\delta = 0.025$)

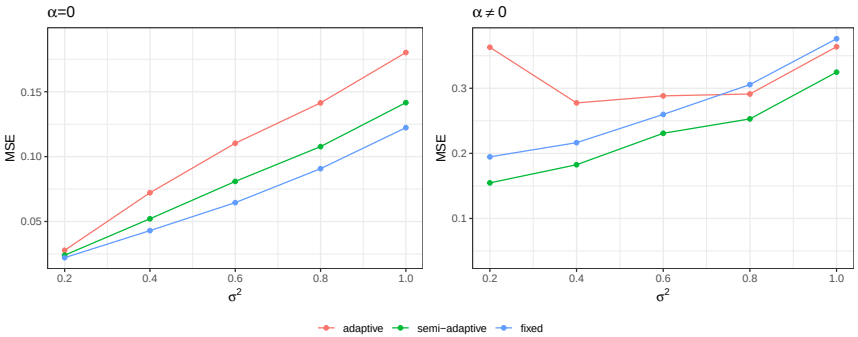


Рис. 2: MSE восстановления сигнала ($\delta = 0.0125$)

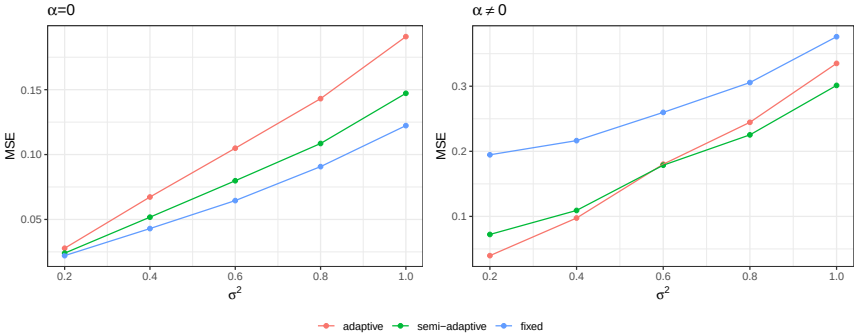


Рис. 3: MSE восстановления сигнала ($\delta = 0.05$)

Заключение

В статье рассмотрены три подхода к автоматическому выделению сигнала во временных рядах с использованием критерия MC-SSA и метода анализа сингулярного спектра. Исследование показало, что полуадаптивный вариант может быть использован в качестве базового в ситуации, когда неизвестно наличие или отсутствие амплитудной модуляции, при этом модуляция не очень сильная. Для выбора параметра δ нужно делать предположения о силе модуляции. Заметим, что при выборе длины окна для MC-SSA нужно учитывать сочетание мощности критерия и точности оценивания частоты.

Список литературы

[1] Allen M., Smith L. Monte Carlo SSA: detecting irregular oscillations in the presence of coloured noise // Journal of Climate. — 1996. — Vol. 9. — P. 3373–3404.

[2] Broomhead D., King G. Extracting qualitative dynamics from experimental data // Physica D: Nonlinear Phenomena. — 1986. — Vol. 20, no. 2–3. — P. 217–236.

[3] Golyandina N., Nekrutkin V., Zhigljavsky A. Analysis of Time Series Structure: SSA and Related Techniques. — Chapman and Hall/CRC, 2001.

[4] Александров Ф., Голяндина Н. Автоматизация выделения трендовых и периодических составляющих временного ряда в рамках метода «Гусеница»-SSA // Exponenta Pro (Математика в приложениях). — 2004. — Vol. 7-80. — P. 54–61.

[5] Kalantari M., Hassani. H. Automatic grouping in singular spectrum analysis // Forecasting. — 2019. — Vol. 1, no 1. — P. 189–204.

[6] Bogalo J., Poncela P., Senra E. Circulant singular spectrum analysis: A new automated procedure for signal extraction // Signal Processing. — 2021. — Vol. 179.

[7] Golyandina N., Dudnik P., Shlemov A. Intelligent Identification of Trend Components in Singular Spectrum Analysis // Algorithms. — 2023. — Vol. 16. — ID 353.

[8] Golyandina N. Detection of signals by Monte Carlo singular spectrum analysis: multiple testing // Statistics and Its Interface. — 2023. — Vol. 16. no 1. — P. 147–157.

Байесовский подход к обнаружению разладки

Татьяненко А.Д., СПбГУ, Санкт-Петербург alexdtat@gmail.com,
Гориховский В.И., СПбГУ, Санкт-Петербург Gorihovskyvyacheslav@gmail.com,
Кутуев В.А., Лаборатория технологий программирования инфраструктурных решений СПбГУ, Санкт-Петербург v.kutuev@spbu.ru

Аннотация

Рассматривается байесовский подход к обнаружению разладки. Описываются его преимущества для онлайн-задачи, позволяющие использовать алгоритм итеративно и локализовывать разладку по распределению длины пробега (количества шагов с последней разладки). Рассматриваются гауссовская и экспоненциальная предсказательные модели. Преимущество этих моделей на основе сопряжённых распределений — наличие явных формул для получения апостериорных гиперпараметров, что оказывается вычислительно простым решением. Предлагается эвристическая предсказательная модель, выбирающая на основе обучающей подвыборки более подходящую функцию правдоподобия из двух рассматриваемых. Описывается исследование зависимости этой эвристики от размера порога детектора и размера обучающей подвыборки, проведённое на различных распределениях. Основное внимание уделяется качеству локализации и задержкам.

Введение

Задача обнаружения разладки

Разладка — один из видов аномалий во временных рядах. Существуют различные определения, но в контексте данной работы под ней подразумевается точка смены одного распределения данных на другое (наблюдения на сегментах между ближайшими разладками независимы и одинаково распределены). Задача обнаружения разладки [3]:

$$\{y_i\}, i = 1, \ldots, n$$
$$H_0 \colon y_i \sim F_0, i = 1, \ldots, n$$
$$H_a \colon \exists \tau : 1 \leq \tau < n, y_i \sim \begin{cases} F_1, & i > \tau, \\ F_0 & \text{иначе.} \end{cases}$$

Тогда τ — точка разладки. На практике также часто полезно не только обнаруживать разладку, но и определять момент времени, в который она произошла (при этом возможно некоторое отклонение от её настоящего местоположения), что далее будет называться *локализацией*. Эта задача возникает во многих контекстах, например, при анализе зашумлённых бенчмарков с целью определения момента падения производительности [4].

В данной работе рассматривается случай с размером данных $n = 500$ и разладкой (в случае её наличия) $\tau = 250$. Зависимость результатов работы алгоритма и подобранных значений параметров от размера данных и положения разладки здесь не рассматривается, являясь темой для отдельного исследования.

В связи с предоставлением небольшого набора алгоритмов в существующих Python-инструментах для обнаружения разладки и с отсутствием инструментов для сравнения соответствующих алгоритмов ведётся разработка PySATL-CPD. Рассматриваемый в данной статье алгоритм также входит в него.

Байесовский подход

В байесовском подходе к обнаружению разладки количество шагов, прошедших с последней разладки, рассматривается как случайная величина r_t (она называется длиной пробега) [2]. Её распределение можно представить как $P(r_t \mid \boldsymbol{x}_{1:t}) = P(r_t, \boldsymbol{x}_{1:t})/P(\boldsymbol{x}_{1:t})$, где $\boldsymbol{x}_{1:t}$ — вектор наблюдений за время $1 : t$. Тогда, с учётом независимости данных, совместную вероятность для указанного выше распределения можно разложить следующим образом:

$$\begin{aligned} &P(r_t, \boldsymbol{x}_{1:t}) = \textstyle\sum_{r_{t-1}} P(r_t, r_{t-1}, \boldsymbol{x}_{1:t}) = \\ &= \textstyle\sum_{r_{t-1}} P(r_t, x_t \mid r_{t-1}, \boldsymbol{x}_{1:t-1}) P(r_{t-1}, \boldsymbol{x}_{1:t-1}) = \\ &= \textstyle\sum_{r_{t-1}} P(r_t \mid r_{t-1}) P(x_t \mid r_{t-1}, \boldsymbol{x}_t^{(r)}) P(r_{t-1}, \boldsymbol{x}_{1:t-1}). \end{aligned} \tag{1}$$

В формуле 1 $\boldsymbol{x}_t^{(r)}$ означает, соответственно, наблюдения, относящиеся лишь к данным после последней зафиксированной разладки. Первый множитель (1) — это предсказательная вероятность, вычисляемая с помощью предсказательной модели. Более подробно они рассматриваются далее. Второй множитель (1) определяется функцией выживаемости. Она представляет априорное распределение на появление разладки, не зависящее непосредственно от наблюдений:

$$H(\tau) = \frac{P_{\text{gap}}(g = \tau)}{\sum_{t=\tau}^{\infty} P_{\text{gap}}(g = t)}, \tag{2}$$

где $P_{\text{gap}}(g)$ — дискретное априорное распределение на промежутке между разладками.

В зависимости от длины пробега указанная вероятность вычисляется следующим образом:

$$P(r_t \mid r_{t-1}) = \begin{cases} H(r_{t-1}+1) & \text{если } r_t = 0 \\ 1 - H(r_{t-1}+1) & \text{если } r_t = r_{t-1} + 1 \\ 0 & \text{иначе} \end{cases}.$$

Как частный случай, если $P_{\text{gap}}(g)$ — геометрическое, то функция выживаемости (2) — константа: $H(\tau) = 1/\lambda$.

Последний множитель (1) оказывается рекуррентным, начиная с единственной возможной длины пробега, имеющей вероятность 1.0.

Адаптация алгоритма к онлайн-задаче

Онлайн-задача обнаружения разладки требует от алгоритма возможности обрабатывать данные по мере их поступления на протяжении долгого времени и по мере работы принимать решения об обнаружении разладки (алгоритм имеет доступ лишь к данным до текущего момента, а не ко всем сразу). Описанный выше подход хорош как математический метод, но требует адаптации для практического применения в случае решения онлайн-задачи.

Во-первых, требуется добавление явного критерия наличия разладки, поскольку в исходной работе [2] описано лишь получение распределения длины пробега. Для этого мной были выделены детектор, определяющий наличие разладки, и локализатор, определяющий её местоположение. В качестве детектора рассматривается порог вероятности максимальной длины пробега, поскольку именно она указывает на последнюю найденную разладку. В качестве локализатора предлагается выбор наиболее вероятной длины пробега из остальных.

Во-вторых, требуется возможность автоматически подбирать параметры предсказательной модели. Для этого в алгоритм был добавлен этап обучения предсказательной модели, на котором выделяется следующая за последней разладкой (или находящаяся в начале данных) подвыборка для последующей оценки на ней априорных гиперпараметров. Обучение после каждой разладки позволяет получать для каждого сегмента наблюдений более точную предсказательную модель.

В-третьих, требуется возможность работать достаточно долго. Исходный алгоритм имеет линейный рост требующихся на каждом шаге памяти и времени. Однако в случае принятия решения о наличии разладки можно удалять

все данные, относящиеся к сегменту ранее неё, поскольку распределения до и после разладки отличаются. В случае отсутствия разладки ситуация оказывается сложнее, поскольку приходится моделировать все возможные длины пробега, растущие линейно. Тем не менее, проанализировав характерную задержку при обнаружении разладки, можно предложить эвристику, основанную на параллельном запуске (с задержкой) и подготовке вспомогательного алгоритма, на который происходит переключение с основного по прошествии некоторого времени. Это будет приводить к потере информации, но позволит снизить сложность алгоритма до линейной вместо квадратичной на больших данных.

Предсказательные модели

Предсказательные модели позволяют оценивать правдоподобие принадлежности данных к распределению с оценёнными параметрами. В общем случае это требует интегрирования и решения оптимизационных задач, причём это оказывается необходимым для каждой длины пробега, поэтому количество таких операций на каждом шаге растёт линейно со временем. Общий вид предсказательной вероятности таков: $P(x_t \mid r_{t-1}, \boldsymbol{x}_t^{(r)})$, где $\boldsymbol{x}_t^{(r)}$ — вектор наблюдений для текущего пробега.

Хорошим частным случаем предсказательных моделей оказываются функции правдоподобия с сопряжёнными априорным/апостериорным распределениями. Они позволяют из обучающей подвыборки (обозначим её как $\boldsymbol{x}^{learn}$) оценивать априорные значения гиперпараметров, в явном виде вычислять их апостериорные значения из априорных с учётом полученного наблюдения (обозначим его как x^{obs}) и вычислять предсказательную вероятность с учётом прошлых наблюдений. Однако такие предсказательные модели покрывают достаточно небольшой класс распределений и потому требуют дополнительной оценки качества работы на данных, для которых не дают теоретических гарантий. Разные обозначения наблюдений из обучающей подвыборки и наблюдений, используемых при моделировании распределения длины пробега, вводятся для удобного разграничения разных этапов работы алгоритма. С формальной точки зрения они могут быть заменены просто наблюдениями с соответствующими временны̀ми индексами. Далее будут рассматриваться именно модели со свойством сопряжённости или эвристики на их основе.

Гауссовская предсказательная модель

При реализации базовой версии алгоритма (технически предназначенной для работы с данными фиксированного размера) была добавлена гауссовская предсказательная модель. С использованием сопряжённого нормального-обратного-гамма распределения она позволяет оценивать и среднее, и дисперсию [5].

Экспоненциальная предсказательная модель

Гауссовская предсказательная модель показала себя заметно хуже на данных с использованием экспоненциального распределения или распределения Вейбулла. Хотя распределение Вейбулла и является обобщением экспоненциального распределения, для него не существует функции правдоподобия со свойством сопряжённости [6], поэтому было решено добавить экспоненциальную предсказательную модель. С использованием сопряжённого гамма-распределения она позволяет оценивать параметр экспоненциального распределения.

Эвристическая адаптивная предсказательная модель

Несмотря на то, что рассмотренные выше модели способны сами определять априорные значения параметров, на пользователе всё ещё остаётся ответственность за выбор самой модели. Кроме того, если рассматриваемые данные состоят из сегментов с распределениями из разных семейств, зафиксированная модель будет ограничивать алгоритм в гибкости. Поэтому было решено добавить эвристическую адаптивную модель, выбирающую на этапе обучения одну из функций правдоподобия (гауссовскую или экспоненциальную).

В качестве наивной эвристики было выбрано сравнение вероятностей получить обучающую подвыборку с учётом оценённых априорных гиперпараметров. Поскольку измерения предполагаются независимыми, при оценке вероятности для каждого измерения перемножаются. Стоит отметить, что корректный выбор предсказательной модели (например, с использованием критериев согласия) — отдельная большая задача, выходящая за пределы данной работы, но потенциально перспективная.

Численное исследование

Подготовка к исследованию

Для исследования алгоритма были сгенерированы синтетические данные с зафиксированными параметрами из следующих семейств: нормальное (до разладки $E[Y] = 0.0$, $\sigma = 1.0$, после разладки $E[Y] = 10.0$, $\sigma = 5.0$), равномерное ($E[Y] = 2.5$, $\sigma = 0.87$), бета ($E[Y] = 0.5$, $\sigma = 0.15$), экспоненциальное ($E[Y] = 0.2$, $\sigma = 0.2$) и Вейбулла ($E[Y] = 5.0$, $\sigma = 5.0$). Размер одного массива данных составлял 500, разладка (в случае её наличия) находилась в точке 250. Допустимое отклонение от размеченной (настоящей) разладки было выбрано $k = 25$ (параметр точности обнаружения разладки). Перебирались все возможные упорядоченные пары, а также случаи без разладки. Для каждой из конфигураций было сгенерировано по 1000 массивов данных.

На обоих этапах использовались постоянная функция выживаемости, пороговый детектор и локализатор, выбирающий наиболее вероятную (за исключением наибольшей, поскольку она соответствует уже найденной ранее разладке) длину пробега. Для функции выживаемости частота рассчитывалась так, чтобы априорная вероятность появления хотя бы одной разладки на 500 наблюдений составляла 0.5.

Гауссовская предсказательная модель

Было проведено численное исследование гауссовской предсказательной модели [1]. Сначала были определены значения порогов, соответствующие уровням значимости 0.1, 0.05, 0.01 и 0.005 на нормально распределённых данных (в Таблице 1 обозначено как $\alpha_{\mathcal{N}}$). На данном этапе исследования вычислялись распределения длины пробега на каждом шаге алгоритма, после чего проверялось первое пересечение порога. Это вело к тому, что ложно положительные (FP) срабатывания алгоритма исключали истинно положительные (TP). Далее были вычислены вероятности того, что при первом пересечении порога удавалось локализовать разладку в пределах допустимого отклонения от настоящей. В Таблице 1 эти вероятности обозначается как $P_{Y_0 - Y_1}$, где Y_0 — распределение до разладки, Y_1 — после.

В более чем 97% случаев на всём отрезке длины 500 где-либо была обнаружена разладка. При этом гауссовская предсказательная модель в ходе локализации разладки показала себя хуже при работе с экспоненциальным распределением, с распределением Вейбулла и с бета-распределением по сравнению с остальными случаями (Таблица 1). В случае экспоненциального распределения и распределения Вейбулла это может быть связано с их асиммет-

$\alpha_{\mathcal{N}}$	0.1	0.05	0.01	0.005
P\Порог	0.27	0.18	0.07	0.04
$P_{\mathcal{N}-\mathcal{B}}$	0.794	0.824	0.864	0.878
$P_{\mathcal{N}-Exp}$	0.779	0.805	0.84	0.863
$P_{\mathcal{N}-U}$	0.87	0.893	0.917	0.929
$P_{\mathcal{N}-W}$	0.783	0.811	0.853	0.863
$P_{\mathcal{N}-\mathcal{N}}$	0.968	0.975	0.981	0.981

Таблица 1: Локализация с гауссовской предсказательной моделью

ричностью и существенно отличающимся носителем, из-за чего с точки зрения гауссовского правдоподобия многие наблюдения оцениваются как выбросы.

Эвристическая адаптивная предсказательная модель

При исследовании эвристической адаптивной предсказательной модели уже анализировалось поведение онлайн-алгоритма, а не базового, поэтому вместо оценки положения первой разладки оценивалось положение всех и попадание хотя бы одной из них в искомую окрестность. Были выбраны для оценки вычисленные ранее значения порогов, а также установлены размеры обучающих подвыборок 50, 20, 10 и 5 для предсказательных моделей.

Рассматривались данные такого же размера и с таким же положением разладки. Были выбраны следующие распределения с параметрами до разладки и после (если она была) соответственно:

- нормальное распределение $\mathcal{N}$ с параметрами $\mu_0 = 1.0$, $\sigma_0 = 1.0$ и $\mu_1 = 10.0$; $\sigma_1 = 5.0$;
- равномерное распределение U с параметрами $a_0 = 0.0$, $b_0 = 1.0$ и $a_1 = 1.0$, $b_1 = 4.0$;
- бета-распределение $\mathcal{B}$ с параметрами $\alpha_0 = 0.5$, $\beta_0 = 0.5$ и $\alpha_1 = 5.0$, $\beta_1 = 5.0$;
- экспоненциальное распределение Exp с параметрами $\lambda_0 = 1.0$ и $\lambda_1 = 5.0$;
- распределение Вейбулла W с параметрами $k_0 = 1.0$, $\lambda_0 = 0.5$ и $k_1 = 1.0$, $\lambda_1 = 5.0$.

Вероятность FP результата на данных без разладки оказалась не более 0.1 (случай нормального распределения и максимального порога, тогда как остальные вероятности FP оказались не более 0.06). В результате обнаружилось лишь 2 случая, где вероятность обнаружить разладку в пределах отклонения от настоящей (TP) была менее 0.79: $\mathcal{B} - \mathcal{B}$ и $W - U$. В обоих из них каждый раз на этапе обучения выбиралась экспоненциальная модель, тогда как для бета-распределения и для равномерного распределения лучше отрабатывала гауссовская модель. Предположительно, это связано с тем, что экспоненциальная модель выдавала большее правдоподобие для наблюдений из носителя $[0;\ 1]$, но хуже улавливала изменения плотности внутри него. Для решения этой проблемы, возможно, стоит рассмотреть ренормировку на этапе оценки предсказательной модели на обучающей подвыборке.

Также было замечено появление *дублирующих* ложно положительных (FP) разладок: на данных без разладок, как было уже упомянуто ранее, оказывалось сравнительно мало FP срабатываний, однако в случае наличия разладок количество FP срабатываний росло. При детальном рассмотрении оказалось, что это связано с некорректным выбором предсказательной модели после первой разладки, когда в обучающую подвыборку попадал сегмент данных до настоящей разладки, что мешало корректному выбору функции правдоподобия. Для устранения этого недостатка возможно использование эвристик, проводящих дополнительную обработку обучающей подвыборки (например, опуская первые m наблюдений).

Анализ задержек локализации показал, что, за вычетом оговоренных ранее плохих случаев, максимальная средняя из них составила 117, а максимальная медиана составила 110. Такие задержки, в совокупности с указанной выше вероятностью локализовать разладку с искомой точностью, указывают на возможность применения эвристики с параллельным вспомогательным алгоритмом, если выбирать общее рабочее время около 500 и время подготовки около 250.

Заключение

В работе была представлена адаптация байесовского алгоритма под решение онлайн-задачи обнаружения разладки[1]. Далее были рассмотрены различные варианты предсказательных моделей, основанных на функциях правдоподобия со свойством сопряжённости: гауссовская, экспоненциальная и эвристическая адаптивная. Численное исследование на синтетических данных для гауссовской предсказательной модели показало работоспособность ал-

[1]Исходный код (nick: alexdtat): `https://github.com/PySATL/pysatl-cpd` — 05.06.2025

горитма в случае использования гауссовской предсказательной модели. Эвристическая адаптивная модель частично позволила расширить область применения алгоритма, однако также появилось 2 случая, в которых она показывала себя плохо. Это связано с недостатками наивной эвристики, для борьбы с которыми в будущем можно рассматривать более продвинутые подходы к корректному выбору предсказательной модели. Также был приведён анализ вызванных эвристической моделью FP срабатываний алгоритма на данных с разладкой и предложен вариант решения проблемы попадания в обучающую подвыборку сегмента данных до настоящей разладки. Анализ задержек локализации указывает на возможность использования эвристики поддерживающего алгоритма для устранения роста затрат памяти и времени на каждом новом шаге алгоритма.

Список литературы

[1] Татьяненко А. Адаптация и исследование байесовского подхода к обнаружению разладки для PySATL-CPD-Module. 2024. URL: https://se.math.spbu.ru/thesis/texts/Tat'janenko_Aleksej_Dmitrievich_Spring_practice_3rd_year_2024_text.pdf — дата обращения 05.06.2025.

[2] Adams R., MacKay D. Bayesian Online Changepoint Detection // Arxiv preprint. 2007. arXiv:0710.3742.

[3] Aminikhanghahi S., Cook D. J. A survey of methods for time series change point detection // Knowl. Inf. Syst. 2017. Т. 51. С. 339 –367.

[4] Fleming M., Kolaczkowski P., Kumar I. et al. Hunter: Using Change Point Detection to Hunt for Performance Regressions // Proc. of the 2023 ACM/SPEC International Conference on Performance Engineering (ICPE '23). 2023. С. 199 –206.

[5] Murphy K. Conjugate Bayesian analysis of the Gaussian distribution. 2007.

[6] Soland R. Bayesian Analysis of the Weibull Process With Unknown Scale and Shape Parameters // IEEE Trans. Reliab. 1969. Т. R –18, № 4. С. 181 –184.

Модели, методы и приложения тропической математики

Кривулин Николай Кимович
д.ф.-м.н., профессор, профессор кафедры статистического моделирования

Решение многокритериальной задачи определения приоритетов дорожных работ

Яковлев Д. М., СПбГУ, Санкт-Петербург st095998@student.spbu.ru,
Кривулин Н. К., СПбГУ, Санкт-Петербург nkk@math.spbu.ru

Аннотация

Рассматривается многокритериальная задача парных сравнений для определения приоритетов выполнения дорожных работ по развитию транспортной инфраструктуры. Приводится решение задачи на основе log-чебышевской аппроксимации матриц парных сравнений с помощью тропической алгебры.

Многокритериальная задача парных сравнений

Многокритериальная задача парных сравнений [1, 2, 3] позволяет определить значимость альтернатив, например, при планировании развития дорожно-транспортной инфраструктуры [4]. Задача состоит в оценке рейтингов N альтернатив с K критериями, которые сравниваются попарно. Задана матрица $\mathbf{C} = (c_{ij})$ парных сравнений критериев размера $K \times K$, $c_{ij} > 0$ определяет степень значимости критерия i по сравнению с критерием j (во сколько раз критерий i предпочтительнее j). Для каждого критерия $k = 1, \ldots, K$ известны матрицы $\mathbf{A}_k = (a_{ij}^{(k)})$ сравнений альтернатив размера $N \times N$, $a_{ij}^{(k)} > 0$ определяет степень значимости критерия i в сравнении с j относительно критерия k. Вектор $\mathbf{x} = (x_i)$ рейтингов (приоритетов) альтернатив размера $N \times 1$, x_i показывает абсолютный рейтинг альтернативы i. Общая задача — на основе $\mathbf{C}, \mathbf{A}_1, \ldots, \mathbf{A}_K$ определить $\mathbf{x}$.

Метод log-чебышёвской аппроксимации

Рассмотрим метод решения [1, 3, 5], который заключается в аппроксимации в метрике Чебышёва матриц парных сравнений согласованной матрицей с применением результатов тропической алгебры, обеспечивающих аналитическое решение. Матрица $\mathbf{A} = (a_{ij})$ называется *согласованной*, если $a_{ij} = a_{ik}a_{kj}$, $i, k, j = 1, \ldots, N$. Решение, полученное с помощью указанного метода, может быть не единственным, поэтому вводятся наилучший и наихудший дифференцирующие векторы $\mathbf{x}$ и $\mathbf{y}$ соответственно. Наилучшее решение характеризуется максимальным, а наихудшее решение - минимальным значением отношения между элементами вектора решений по всему множеству полученных решений.

Для аналитического решения задачи она формулируется в терминах max-алгебры, которая представляет собой множество неотрицательных вещественных чисел с операцией сложения $\oplus$, заданной по правилу $x \oplus y = \max(x, y)$, и операцией умножения, заданной как обычно [2, 6]. Сложение и умножение матриц, согласованных по размеру, а также умножение на число выполняются по обычным правилам с заменой операции + на $\oplus$. Единичная матрица имеет обычный вид и обозначается символом I. Степень матрицы определяется в смысле тропического умножения матриц.

След квадратной матрицы $\mathbf{A} = (a_{ij})$ порядка N находится как сумма

$$\operatorname{tr} \mathbf{A} = a_{11} \oplus a_{22} \oplus \cdots \oplus a_{NN} = \bigoplus_{i=1}^{N} a_{ii}.$$

Тропический спектральный радиус λ матрицы $\mathbf{A}$ вычисляется по формуле

$$\lambda = \operatorname{tr} \mathbf{A} \oplus \operatorname{tr}^{1/2}(\mathbf{A}^2) \oplus \cdots \oplus \operatorname{tr}^{1/N}(\mathbf{A}^N) = \bigoplus_{k=1}^{N} \operatorname{tr}^{1/k}(\mathbf{A}^k).$$

Оператор Клини для матрицы $\mathbf{A}$ при условии $\lambda \leqslant 1$ определён в виде

$$\mathbf{A}^* = \mathbf{I} \oplus \mathbf{A} \oplus \cdots \oplus \mathbf{A}^{N-1} = \bigoplus_{k=0}^{N-1} \mathbf{A}^k.$$

Для вектор-столбца $\mathbf{x} = (x_i)$ без нулевых элементов определён сопряжённый вектор-строка $\mathbf{x}^- = (x_i^{-1})$. Вектор-столбец, состоящий из единиц, обозначается $\mathbf{1} = (1, \ldots, 1)$. Норма вектора $\mathbf{x}$ порядка N имеет вид

$$\|\mathbf{x}\| = x_1 \oplus x_2 \oplus \cdots \oplus x_N = \bigoplus_{i=1}^{N} x_i = \mathbf{1}^{\mathrm{T}} \mathbf{x}.$$

Решение задачи парных сравнений в терминах max-алгебры ищется выполнением следующих шагов [1, 3] .

1. Вычисление наилучшего и наихудшего дифференцирующих векторов весов для матрицы $\mathbf{C}$ парных сравнений критериев.

   1.1 Вычисление матрицы Клини для определения весов критериев

$$\mathbf{D} = (\lambda^{-1}\mathbf{C})^* = \bigoplus_{k=0}^{K-1} (\lambda^{-1}\mathbf{C})^k, \quad \lambda = \bigoplus_{k=1}^{K} \operatorname{tr}^{1/k}(\mathbf{C}^k). \qquad (1)$$

1.2 Определение по столбцам матрицы $\mathbf{D} = (\mathbf{d}_k)$ наилучшего дифференцирующего вектора весов

$$\mathbf{w} = \mathbf{d}_i \|\mathbf{d}_i\|^{-1}, \quad i = \arg \max_{1 \leqslant k \leqslant K} \|\mathbf{d}_k\| \|\mathbf{d}_k^-\|. \tag{2}$$

В случае неединственного вектора $\mathbf{w}$ выбираются векторы, которые покомпонетно меньше других векторов.

1.3 Вычисление наихудшего дифференцирующего вектора весов

$$\mathbf{v} = (\mathbf{1}^{\mathrm{T}} \mathbf{D})^-. \tag{3}$$

2. Вычисление на основе матриц $\mathbf{A}_1, \ldots, \mathbf{A}_K$ и вектора $\mathbf{w} = (w_k)$ наилучшего дифференцирующего вектора рейтингов альтернатив.

2.1 Определение взвешенной суммы матриц парных сравнений

$$\mathbf{P} = \bigoplus_{k=1}^{K} w_k \mathbf{A}_k. \tag{4}$$

2.2 Вычисление матрицы Клини для рейтингов альтернатив

$$\mathbf{Q} = (\mu^{-1}\mathbf{P})^* = \bigoplus_{n=0}^{N-1} (\mu^{-1}\mathbf{P})^n, \quad \mu = \bigoplus_{n=1}^{N} \mathrm{tr}^{1/n}(\mathbf{P}^n). \tag{5}$$

2.3 Определение по столбцам матрицы $\mathbf{Q} = (\mathbf{q}_n)$ наилучшего вектора рейтингов альтернатив

$$\mathbf{x} = \mathbf{q}_j \|\mathbf{q}_j\|^{-1}, \quad j = \arg \max_{1 \leqslant n \leqslant N} \|\mathbf{q}_n\| \|\mathbf{q}_n^-\|. \tag{6}$$

В случае неединственного вектора $\mathbf{x}$ выбираются векторы, которые покомпонетно меньше других векторов.

3. Вычисление наихудшего дифференцирующего вектора рейтингов альтернатив для матриц $\mathbf{A}_1, \ldots, \mathbf{A}_K$ и вектора $\mathbf{v} = (v_k)$.

3.1 Определение взвешенной суммы матриц парных сравнений

$$\mathbf{R} = \bigoplus_{k=1}^{K} v_k \mathbf{A}_k. \tag{7}$$

3.2 Вычисление матрицы Клини для рейтингов альтернатив

$$\mathbf{S} = (\nu^{-1}\mathbf{R})^* = \bigoplus_{n=0}^{N-1} (\nu^{-1}\mathbf{R})^n, \quad \nu = \bigoplus_{n=1}^{N} \mathrm{tr}^{1/n}(\mathbf{R}^n). \tag{8}$$

3.3 Вычисление наихудшего вектора рейтингов альтернатив

$$\mathbf{y} = (\mathbf{1}^{\mathrm{T}} \mathbf{S})^-. \tag{9}$$

Определение приоритетов дорожных работ

В статье [4] на основе парных сравнений альтернатив выбираются виды дорожных работ для развития дорожно-транспортной инфраструктуры в Восточном Тиморе. Заметим, что в этой статье не все используемые матрицы являются обратно симметричными. Это приводит к проблеме решения задач с исходно ошибочными данными. Рассматриваемый метод log-чебышёвской аппроксимации не зависит от ошибок в данных, что позволяет применить его в этих условиях.

Рассмотрим критерии оценки способов развития дорожно-транспортной инфраструктуры: 1) прочность дорожной покрытий, 2) надёжность дорожных покрытий, 3) безопасность автомобильных дорог, 4) работоспособность дорожных покрытий, 5) содержание дорожных покрытий. Матрица парных сравнений критериев имеет вид:

$$\mathbf{C} = \begin{pmatrix} 1 & 3 & 5 & 7 & 2 \\ 1/3 & 1 & 3 & 3 & 1/5 \\ 1/5 & 1/3 & 1 & 2 & 1/4 \\ 1/7 & 1/3 & 1/2 & 1 & 1/3 \\ 1/2 & 5 & 4 & 3 & 1 \end{pmatrix}.$$

Имеются следующие альтернативные виды работ: 1) монтаж и установка подпорных стен, 2) монтаж и установка дренажных систем дорожного покрытия, 3) монтаж водопроводных труб (Culverts), 4) выравнивание грунта и 5) вырубка деревьев и кустарников на полосах отвода. Матрицы парных сравнений альтернатив по каждому критерию имеют вид:

$$\mathbf{A}_1 = \begin{pmatrix} 1 & 2 & 3 & 7 & 4 \\ 1/2 & 1 & 4 & 1/4 & 1/3 \\ 1/3 & 1/4 & 1 & 1/5 & 1/2 \\ 1/7 & 1/4 & 5 & 1 & 1/3 \\ 1/4 & 3 & 2 & 3 & 1 \end{pmatrix}, \quad \mathbf{A}_2 = \begin{pmatrix} 1 & 2 & 3 & 6 & 4 \\ 1/3 & 1 & 6 & 6 & 6 \\ 1/5 & 1/6 & 1 & 3 & 4 \\ 1/6 & 1/6 & 1/3 & 1 & 3 \\ 1/4 & 1/6 & 1/4 & 1/3 & 1 \end{pmatrix},$$

$$\mathbf{A}_3 = \begin{pmatrix} 1 & 2 & 3 & 1/2 & 1/5 \\ 1/2 & 1 & 2 & 1/4 & 1/4 \\ 1/3 & 1/2 & 1 & 1/6 & 1/6 \\ 2 & 4 & 6 & 1 & 1/2 \\ 5 & 4 & 6 & 2 & 1 \end{pmatrix}, \quad \mathbf{A}_4 = \begin{pmatrix} 1 & 3 & 5 & 7 & 9 \\ 1/3 & 1 & 2 & 5 & 5 \\ 1/5 & 1/3 & 1 & 3 & 3 \\ 1/7 & 1/5 & 1/3 & 1 & 2 \\ 1/9 & 1/5 & 1/3 & 1/2 & 1 \end{pmatrix},$$

$$\mathbf{A}_5 = \begin{pmatrix} 1 & 2 & 3 & 2 & 3 \\ 1/2 & 1 & 3 & 4 & 5 \\ 1/3 & 1/3 & 1 & 2 & 2 \\ 1/3 & 1/4 & 1/2 & 1 & 3 \\ 1/2 & 1/5 & 1/2 & 1/3 & 1 \end{pmatrix}.$$

Подчеркнём, что матрицы $\mathbf{A}_2, \mathbf{A}_4, \mathbf{A}_5$ не обратно симметричные.

Решение задачи

Использование описанной выше процедуры решения задачи на основе log-чебышёвской аппроксимации приводит к следующим результатам.

По формулам (1) находим матрицу для определения весов критериев

$$\mathbf{D} = \begin{pmatrix} 1 & \lambda^2 & 3\lambda & 6 & 2/\lambda \\ 3/5\lambda^2 & 1 & 3/\lambda & 6/\lambda^2 & \lambda/5 \\ 1/5\lambda & \lambda/3 & 1 & 2/\lambda & 2/3\lambda^2 \\ 1/10 & \lambda^2/6 & \lambda/2 & 1 & 1/3\lambda \\ 3\lambda/10 & 5/\lambda & 15/\lambda^2 & 3\lambda & 1 \end{pmatrix}, \quad \lambda = 10^{1/4} \approx 1{,}7783.$$

Чтобы применить формулы (2), сначала найдем

$$\|\mathbf{d}_1\|\|\mathbf{d}_1^-\| = 10,$$
$$\|\mathbf{d}_2\|\|\mathbf{d}_2^-\| = \|\mathbf{d}_3\|\|\mathbf{d}_3^-\| = \|\mathbf{d}_4\|\|\mathbf{d}_4^-\| = \|\mathbf{d}_5\|\|\mathbf{d}_5^-\| = 6.$$

Выбрав $i = 1$, найдем наилучший дифференцирующий вектор весов

$$\mathbf{w} = \mathbf{d}_1\|\mathbf{d}_1\|^{-1} = (1 \quad 3/5\lambda^2 \quad 1/5\lambda \quad 1/10 \quad 3/\lambda^3)^{\mathrm{T}}.$$

По формуле (3) вычислим наихудший дифференцирующий вектор весов

$$\mathbf{v} = (\mathbf{1}^{\mathrm{T}}\mathbf{D})^- = (1 \quad 1/\lambda^2 \quad 1/3\lambda \quad 1/6 \quad \lambda/2)^{\mathrm{T}}.$$

С помощью (4) найдем взвешенную сумму матриц

$$\mathbf{P} = \bigoplus_{k=1}^{5} w_k\mathbf{A}_k = \begin{pmatrix} 1 & 2 & 3 & 7 & 4 \\ 1/2 & 1 & 4 & 6\lambda/5 & 3\lambda/2 \\ 1/3 & 1/4 & 1 & 3\lambda/5 & 3\lambda/5 \\ 2/5\lambda & 4/5\lambda & 5 & 1 & 9/\lambda^3 \\ 1/\lambda & 3 & 2 & 3 & 1 \end{pmatrix}.$$

Применяя формулы (5), построим матрицу Клини для определения наилучшего дифференцирующего вектора рейтингов альтернатив

$$\mathbf{Q} = \begin{pmatrix} 1 & 14/\mu^2 & 35/\mu^2 & 7/\mu & 14/3\mu \\ 5/3\mu^2 & 1 & 5/\mu & 1 & \mu/3 \\ 1/3\mu & 2/5 & 1 & 2/5 & 2\mu/15 \\ 5/3\mu^2 & 2/\mu & 5/\mu & 1 & 2/3 \\ 5/\mu^3 & 3/\mu & 15/\mu^2 & 3/\mu & 1 \end{pmatrix}, \quad \mu = 3(\lambda/2)^{1/2}.$$

Для использования формулы (6) сначала вычислим

$$\begin{aligned} &\|\mathbf{q}_1\|\|\mathbf{q}_1^-\| = 3\mu \approx 8{,}486, \\ &\|\mathbf{q}_2\|\|\mathbf{q}_2^-\| = \|\mathbf{q}_3\|\|\mathbf{q}_3^-\| = \|\mathbf{q}_5\|\|\mathbf{q}_5^-\| = 35/\mu^2 \approx 4{,}374, \\ &\|\mathbf{q}_4\|\|\mathbf{q}_4^-\| = 35/2\mu \approx 6{,}186. \end{aligned}$$

Выберем $i = 1$ и найдём наилучшее дифференцирующее решение

$$\mathbf{x} = \mathbf{q}_1\|\mathbf{q}_1\|^{-1} \approx (1{,}0000 \quad 0{,}2083 \quad 0{,}1178 \quad 0{,}2083 \quad 0{,}2209)^{\mathrm{T}}.$$

Полученный вектор определяет следующий порядок альтернатив

$$\mathbf{I} \succ \mathbf{V} \succ \mathbf{II} \equiv \mathbf{IV} \succ \mathbf{III}.$$

Применяя (7), найдём взвешенную сумму матриц

$$\mathbf{R} = \bigoplus_{k=1}^{5} v_k \mathbf{A}_k = \begin{pmatrix} 1 & 2 & 3 & 7 & 4 \\ 1/2 & 1 & 4 & 2\lambda & 5\lambda/2 \\ 1/3 & \lambda/6 & 1 & \lambda & \lambda \\ 2/3\lambda & 4/3\lambda & 5 & 1 & 3\lambda/2 \\ 5/3\lambda & 3 & 2 & 3 & 1 \end{pmatrix}.$$

По формулам (8) находим матрицу для определения рейтингов альтернатив

$$\mathbf{S} = \begin{pmatrix} 1 & 21/5\nu & 35/\nu^2 & 7/\nu & 7/5 \\ 25/6\nu^2 & 1 & 5/\nu & 1 & \nu/3 \\ 5/3\nu^2 & 2/5 & 1 & 2\nu/15 & 2\nu/15 \\ 5/2\nu^2 & 3/5 & 5/\nu & 1 & \nu/5 \\ 25/2\nu^3 & 3/\nu & 15/\nu^2 & 3/\nu & 1 \end{pmatrix}, \quad \nu = (15\lambda/2)^{1/2}.$$

По формуле (9) вычислим наихудшее дифференцирующее решение

$$\mathbf{y} = (\mathbf{1}^{\mathrm{T}}\mathbf{S})^{-} \approx (1{,}0000 \quad 0{,}8695 \quad 0{,}3811 \quad 0{,}5217 \quad 0{,}7143)^{\mathrm{T}}.$$

Полученный вектор определяет следующий порядок альтернатив

$$\mathbf{I} \succ \mathbf{II} \succ \mathbf{V} \succ \mathbf{IV} \succ \mathbf{III}.$$

Заключение

В статье рассмотрена многокритериальная задача парных сравнений и применен метод log-чебышёвской аппроксимации для определения приоритетов дорожных работ. Заметим, что оба полученных решения согласуются в определении наиболее и наименее предпочтительных альтернатив.

Авторы благодарят рецензента за полезные замечания и комментарии.

Список литературы

[1] Krivulin N. K. Application of tropical optimization for solving multicriteria problems of pairwise comparisons using log-Chebyshev approximation // Int. J. Approx. Reason. — 2024. — Vol. 169. — P. 109–168.

[2] Кривулин Н.К. Методы идемпотентной алгебры в задачах моделирования и анализа сложных систем. — СПб.: Изд-во СПбГУ, 2009.

[3] Кривулин Н. К., Агеев В. А. Методы тропической оптимизации в многокритериальных задачах оценки альтернатив на основе парных сравнений // Вестник СПбГУ. Прикладная математика. Информатика. Процессы управления. — 2019. — №4. — Р. 472–488.

[4] Da Costa X. H. Comprehensive evaluation of video data by using the Analytic Hierarchy Process (AHP) method // Timorese Academic Journal of Science and Technology. — 2018. — Vol. 1. — P. 87–95.

[5] Решение многокритериальных задач оценки альтернатив на основе парных сравнений / Кривулин Н. К., Булгакова Д. С., Григорьев Д. А., Нагуманова К. И., Приньков А. С., Салова Я. А., Филатова А. А. // Компьютерные инструменты в образовании. — 2024. — №2. — Р. 5–29.

[6] Маслов В. П., Колокольцов В. Н. Идемпотентный анализ и его применение в оптимальном управлении. М.: Наука, 1994.

Параллельные алгоритмы, вэйвлетная обработка числовых потоков

Макаров Антон Александрович
д.ф.-м.н., профессор, заведующий кафедрой параллельных алгоритмов

О структуре учебных планов образовательных программ в сфере IT

Евдокимова Т.О., СПбГУ, Санкт-Петербург t.evdokimova@spbu.ru,
Иванцова О.Н., СПбГУ, Санкт-Петербург o.ivancova@spbu.ru,
Пономарев Н.А., СПбГУ, Санкт-Петербург n.ponomarev@spbu.ru

Аннотация

Учебные планы образовательных программ направлений подготовки программистов, связанных с информатикой или компьютерными технологиями, в разных вузах отличаются своей структурой соотношения объемов математических и программистских дисциплин. Целью работы является выборочное сравнение структур учебных планов подобных программ для выявления имеющих наибольшую математическую и наибольшую программистскую обязательные части и описания характерных для таких программ свойств.

Введение

Объемы обязательной математической подготовки и подготовки в области программирования не всегда одинаковы даже для образовательных программ одного направления. Тем более они могут различаться для относящихся к области IT программ разных направлений. Для сравнения структур учебных планов (УП) у «программистских» образовательных программ (ОП) были рассмотрены следующие характеристики:

- суммарное количество аудиторных часов и часов на самостоятельную работу;
- соотношение часов на обязательные математические дисциплины (ОМД), на обязательные программистские дисциплины (ОПД), на элективы (Э), на другие предметы (ДП), включающие в себя иностранные языки, физическую культуру, предметы гуманитарного цикла и т.п.;
- указанные соотношения часов по семестрам.

Анализируемые материалы

В работе проводится сравнение некоторых программ бакалавриата следующих классических университетов и некоторых профильных вузов (в скобках указаны использующиеся в дальнейшем аббревиатуры):

- (СПбГУ) Санкт-Петербургский государственный университет;
- (МГУ) Московский государственный университет;
- (КПФУ) Казанский (Приволжский) государственный университет;
- (НГУ) Новосибирский государственный университет;
- (ННГУ) Национальный исследовательский Нижегородский государственный университет;
- (СибГУТИ) Сибирский государственный университет телекоммуникаций и информатики;
- (ИТМО) Национальный исследовательский университет ИТМО.

Отметим, что не проводилось сравнение программ магистратуры, так как они являются надстройкой над базовым бакалаврским образованием; не рассматривались программы университета Иннополис, специализирующегося в высшем образовании на подготовке специалистов в области искусственного интеллекта.

В выбранных вузах были рассмотрены универсальные, то есть не являющиеся специализированными для какой-либо конкретной области, образовательные программы:

- СПбГУ, «Технологии программирования» (ТП), «Программная инженерия» (ПИ), «Программирование и информационные технологии» (ПиИТ) [1].
- МГУ, «Системное программирование и компьютерные науки» (СПиКН), «Фундаментальная информатика и информационные технологии» (ФИиИТ) [2].
- КПФУ, «Фундаментальная информатика и информационные технологии» (ФИиИТ), «Программная инженерия (Современная разработка программного обеспечения)» (СРПИ) [3].
- НГУ, «Программная инженерия и компьютерные науки» (ПИиКТ) [4].
- ННГУ, «Фундаментальная информатика и информационные технологии» (ФИиИТ), «Разработка программно-информационных систем» (РПИС) [5].

- СибГУТИ, «Программное обеспечение средств вычислительной техники и автоматизированных систем» (ПОСВТиАС), «Системное программное обеспечение» (СПО), «Программное обеспечение систем мобильной связи» (ПОСМС) [6].
- ИТМО, «Компьютерные системы и технологии» (КСиТ), «Разработка программного обеспечения» (РПО), «Системное и прикладное программное обеспечение» (СиППО) [7].

Результаты

Структура образовательных программ в целом

Сводная информация о типах дисциплин в рассмотренных образовательных программах приведена в Таблице 1. Для каждой программы указано суммарное количество часов (аудиторных и на самостоятельную работу) по дисциплинам каждого типа. Звездочкой (*) отмечены программы группы «Информатика»[1] (И). Не имеют отметки программы группы «Компьютерные технологии»[2] (КТ). Нижним индексом указано количество процентов, которое данный объем часов составляет от общей трудоемкости программы.

Цветом выделены имеющие наибольшие количества часов по математическим/программистским/элективным дисциплинам и по общей трудоемкости программы информатики и программы компьютерных технологий.

Структура некоторых образовательных программ по семестрам

Информация о доле часов на дисциплины каждого типа в каждом из семестров для некоторых программ группы КТ приведена в Таблице 2, а для некоторых программ группы И — в Таблице 3.

В каждом семестре проценты часов по каждому из типов дисциплин по отношению к общему количеству часов в данном семестре расположены по невозрастанию. Составляемый общим количеством часов в семестре процент от количества часов по образовательной программе в целом указан для каждой образовательной программы в верхней строке. Цветовые обозначения: ОМД, ОПД, Э, ДП.

По программам компьютерных технологий можно отметить следующее:

[1] Соответствует УГСН 09.00.00

[2] Примерно соответствует УГСН 02.00.00

Вуз, программа	ОМД	ОПД	Э	ДП	Итого
СПбГУ, ТП	$3288_{37\%}$	$1656_{19\%}$	$1654_{19\%}$	$2200_{25\%}$	8798
СПбГУ, ПИ (*)	$2296_{29\%}$	$3208_{41\%}$	$792_{10\%}$	$1518_{19\%}$	7814
СПбГУ, ПиИТ	$2292_{30\%}$	$2870_{38\%}$	$540_{7\%}$	$1872_{25\%}$	7574
МГУ, СПиКН	$2844_{35\%}$	$1872_{23\%}$	$1512_{19\%}$	$1908_{23\%}$	8136
МГУ, ФИиИТ	$2592_{33\%}$	$2052_{26\%}$	$1296_{17\%}$	$1872_{24\%}$	7812
КПФУ, ФИиИТ	$2664_{29\%}$	$3024_{33\%}$	$900_{10\%}$	$2532_{28\%}$	9120
КПФУ, СРПО (*)	$1404_{15\%}$	$2916_{40\%}$	$1620_{17\%}$	$3504_{37\%}$	9444
НГУ, ПИиКТ (*)	$1764_{20\%}$	$2916_{33\%}$	$1620_{18\%}$	$2470_{28\%}$	8770
ННГУ, ФИиИТ	$2880_{35\%}$	$2664_{33\%}$	$1116_{14\%}$	$1480_{18\%}$	8140
ННГУ, РПИС (*)	$2412_{31\%}$	$3276_{42\%}$	$288_{4\%}$	$1804_{23\%}$	7780
СибГУТИ, ПОСВТиАС (*)	$1862_{22\%}$	$3566_{42\%}$	$648_{8\%}$	$2338_{28\%}$	8414
СибГУТИ, СПО	$1836_{22\%}$	$4354_{51\%}$	$360_{4\%}$	$1913_{23\%}$	8463
СибГУТИ, ПОСМС (*)	$1512_{18\%}$	$4176_{50\%}$	$432_{5\%}$	$2162_{26\%}$	8282
ИТМО, КСиТ (*)	$1368_{17\%}$	$2844_{35\%}$	$1710_{21\%}$	$2124_{26\%}$	8046
ИТМО, РПО (*)	$1512_{17\%}$	$5148_{58\%}$	$216_{2\%}$	$2052_{23\%}$	8928
ИТМО, СиППО (*)	$1692_{18\%}$	$2844_{31\%}$	$2808_{30\%}$	$1980_{21\%}$	9324

Таблица 1: Количество часов выбранных ОП по типам дисциплин

- Программа МГУ СПиКН, относящаяся к направлению «Прикладная математика и информатика», содержит значительный объем обязательных дисциплин по программированию.
- Программы классических университетов имеют значительный объем обязательной математической подготовки (30% и более от общего количества часов).
- В профильном СибГУТИ доля ОПД растет от начала к концу обучения и в целом составляет половину общего объема часов. В классических университетах ситуация по семестрам различная, а общая доля часов на такие дисциплины не превышает трети.

ОП ╲ Семестр	1	2	3	4	5	6	7	8	Итого
	14%	13%	15%	14%	12%	12%	12%	8%	100%
	57	55	55	36	38	37	60	56	37
СПбГУ,	26	27	34	32	31	33	33	25	25
ТП	17	18	11	32	24	17	7	19	19
	0	0	0	0	7	13	0	0	19
	13%	15%	15%	14%	14%	11%	12%	6%	100%
	46	54	50	43	42	60	49	39	35
МГУ,	36	30	34	39	29	28	40	31	23
СПиКТ	18	16	16	18	16	8	11	15	23
	0	0	0	0	13	4	0	15	19
	15%	17%	14%	14%	14%	11%	8%	7%	100%
	46	41	40	43	39	48	64	52	33
МГУ,	34	35	34	35	26	32	12	16	26
ФИиИТ	20	24	26	22	22	16	12	16	24
	0	0	0	0	13	4	12	16	17
	15%	15%	15%	16%	11%	9%	12%	7%	100%
	44	54	38	46	36	33	37	41	35
ННГУ,	26	24	36	27	28	29	37	33	33
ФИиИТ	24	16	20	22	24	19	26	13	18
	6	6	6	5	12	19	0	13	14
	14%	14%	13%	13%	14%	11%	13%	8%	100%
	55	41	41	52	62	64	80	74	51
СибГУТИ,	24	38	42	40	23	20	10	26	23
СПО	21	21	17	8	15	16	10	0	22
	0	0	0	0	0	0	0	0	4

Таблица 2: Процентное распределение часов на дисциплины разных типов по каждому из семестров для некоторых образовательных программ группы КТ

- Элективные дисциплины обычно начинаются с третьего курса, за исключением ННГУ, где дисциплины этого типа есть в каждом семестре.
- В ННГУ наименьший размер доли дисциплин, не связанных с математикой, программированием или элективами (18%), и ни в каком семестре на них не тратится наибольшее количество часов. Последнее характерно и для МГУ.

По программам информатики можно отметить следующее:

- Доля ОПД на всех рассмотренных программах, с учетом большого количества элективов в СПбГУ, НГУ и ИТМО СиППО, составляет при-

ОП ╲ Семестр	1	2	3	4	5	6	7	8	Итого
	14%	14%	15%	19%	12%	10%	10%	6%	100%
	60	49	47	42	68	52	41	62	41
СПбГУ,	20	35	41	37	12	22	32	23	29
ПИ	20	16	12	21	12	22	18	15	19
	0	0	0	0	8	4	9	0	10
	13%	14%	12%	14%	16%	16%	10%	5%	100%
	44	41	46	46	43	45	45	46	33
НГУ,	28	38	30	36	27	28	42	36	28
ПИиКТ	22	15	24	18	20	19	13	18	20
	6	6	0	0	10	8	0	0	18
	16%	16%	15%	14%	12%	10%	12%	5%	100%
	40	39	53	66	58	38	40	66	42
ННГУ,	35	34	28	14	42	33	40	17	31
РПИС	25	27	19	13	0	19	12	17	23
	0	0	0	7	0	10	8	0	4
	17%	14%	14%	13%	13%	11%	12%	6%	100%
	47	39	58	63	44	50	67	62	50
СибГУТИ,	33	38	24	20	31	33	20	38	26
ПОСМС	20	23	18	17	25	17	13	0	18
	0	0	0	0	0	0	0	0	5
	14%	13%	12%	13%	14%	12%	13%	9%	100%
	42	43	50	39	74	79	90	89	58
ИТМО,	32	38	27	38	26	21	10	11	23
РПО	26	19	23	23	0	0	0	0	17
	0	0	0	0	0	0	0	0	2
	11%	10%	10%	11%	20%	20%	18%	0%	100%
	35	37	34	38	49	50	60	0	31
ИТМО,	34	33	33	31	31	21	33	0	30
СиППО	31	30	33	31	10	21	7	0	21
	0	0	0	0	10	8	0	0	18

Таблица 3: Процентное распределение часов на дисциплины разных типов по каждому из семестров для некоторых образовательных программ группы «Информатика»

мерно половину всех часов. При этом в двух последних вузах именно элективы занимают наибольшую долю часов на старших курсах.

- Доля ОМД обычно убывает от семестра к семестру, а такие дисциплины присутствуют в УП вплоть до 5–6 семестров (исключениями являются программа ННГУ, где ОМД есть во всех семестрах, и программа ИТМО РПО, где ОМД заканчиваются на 2-м курсе). В программах СПбГУ и ННГУ объем ОМД значительный — около 30%. В программах ИТМО доля ОМД наименьшая среди дисциплин разных типов.

- В СПбГУ наименьший размер доли дисциплины не по специальности и ни в каком семестре, как и в ИТМО СиППО, на них не тратится наибольшее количество часов.

Заключение

- Значительный объем часов по математическим или программистским, или элективным дисциплинам содержат каждая из рассмотренных программ. Но наибольшие значения в двух из трех категорий среди программ компьютерных технологий имеют СПбГУ ТП и МГУ СПиКН, а среди программ информатики — СибГУТИ ПОСВТиАС.

- Доля часов на дисциплины, не являющимися математическими, программистскими или элективными, обычно не превышает четверти от общего объема часов, но распределение таких дисциплин по семестрам в вузах различно и не имеет выраженной тенденции.

- Программы информатики в классических университетах по структуре — доли часов на дисциплины разных типов, соотношению этих долей и тенденции распределения по семестрам — более единообразны, чем программы компьютерных технологий.

Список литературы

[1] Учебные планы образовательных программ СПбГУ // СПбГУ URL: https://spbu.ru/sveden/education (дата обращения: 05.05.2025).

[2] Учебные планы образовательных программ МГУ // МГУ URL: https://msu.ru/sveden/education (дата обращения: 05.05.2025).

[3] Учебные планы образовательных программ КПФУ // КПФУ URL: https://kpfu.ru/do/uchebnyj-process/uchebnye-plany (дата обращения: 05.05.2025).

[4] Учебные планы образовательных программ НГУ // НГУ URL: https://www.nsu.ru/n/information-technologies-department/education_fit/programs/ (дата обращения: 05.05.2025).

[5] Учебные планы образовательных программ ННГУ // ННГУ URL: http://www.unn.ru/sveden/education/edu-op.php (дата обращения: 05.05.2025).

[6] Учебные планы образовательных программ СибГУТИ // СибГУТИ URL: https://sibsutis.ru/sveden/education/ (дата обращения: 05.05.2025).

[7] Учебные планы образовательных программ ИТМО // ИТМО URL:https://itmo.ru/sveden/education#5 (дата обращения: 05.05.2025).

Расширяемый язык программирования

Лебединская Н. А., СПбГУ, Санкт-Петербург n.lebedinskaya@spbu.ru,
Лебединский Д. М., СПбГУ, Санкт-Петербург d.lebedinsky@spbu.ru,
Смирнов А. А., ВА МТО имени генерала армии А. В. Хрулева, Санкт-Петербург
smirnov89119750282@yandex.ru

Аннотация

Краткое изложение вопросов, связанных с разработкой нового расширяемого языка программирования, в части структур данных, модели вычислений, организации таблиц, содержащих, в том числе, реализации команд и преобразования типов, и некоторых других.

Введение

Данный проект посвящен разработке нового расширяемого языка программирования. В связи с этим представляется разумным создание для этой цели двух языков: низкого уровня, о котором в основном и пойдет здесь речь, и высокого уровня, которому будут посвящены следующие работы. Цель работы состоит в том, чтобы изложить ряд основных идей и проектных решений, принятых при разработке языка низкого уровня.

Сама по себе идея создания расширяемого языка программирования определенно не нова. Существуют очень многие проекты, имеющие похожие цели. Упомянем здесь только некоторые из них; данный список определенно не может претендовать на полноту изложения данного вопроса.

Для начала упомянем поздне-советскую разработку, язык УТОПИСТ [1]. В книге, излагающей его описание, сказано, что он позволяет добавлять в систему новые понятия; однако, под понятиями там понимаются наборы соотношений, описывающие определенные физические явления, и позволяющие по имеющимся данным (значениям некоторых из переменных) восстанавливать значения остальных. Таким образом, речь идет о комбинации такого рода соотношений, позволяющей многоступенчато вычислять неизвестные значения параметров для «составных» явлений. Такой подход определенно сильно отличается от нашего, в котором под понятиями, добавляемыми в систему, имеются ввиду понятия собственно программирования, например новые парадигмы, т. е., скажем, добавление объектно-ориентированности, которая изначально в системе реализована не была.

Далее, упомянем систему katahdin [2]. Она действительно позволяет менять грамматику языка на ходу, и это уже гораздо ближе к тому, что происходит здесь, однако, и с этой системой имеется определенные отличия. Как

сказано в документе, ее описывающем, там используются PEG-грамматики и packrat-парсер. С одной стороны, такая реализация имеет то неоспоримое достоинство, что все работает очень быстро, но со стороны богатства выразительности, увы, здесь имеются определенные ограничения. Как вскользь замечено в руководстве, такой подход реализует, в частности, синтаксис фортрана частично. На наш взгляд, это выглядит не очень многообещающе.

Существует также язык XL [3]. В нем, однако, насколько нам известно, для расширяемости реализована перегрузка выражений. Конечно, необходимость чего-то большего в ближайшем будущем представляется нам маловероятной, но с нашей точки зрения, это серьезное ограничение.

Следующая система, которая будет здесь упомянута, — это язык seed7 [4]. В его документации написано, что в самом языке нет ключевых слов — все они могут быть определены как часть стандартной библиотеки в порядке расширения языка. Однако, и здесь есть небольшие проблемы — там нет автоматического преобразования типов, и для вычисления произведения целого числа на вещественное целый сомножитель должен быть явно преобразован к вещественному типу. Ввиду привычки к программированию на традиционных языках вроде C или C++, где такое преобразование происходит прозрачно, такая необходимость несколько раздражает.

Наконец, упомянем здесь диссертацию Bravenboer'a [5], в которой также описана система расширяемого языка. В качестве базового языка для расширения выбран Java, а инструментом для расширения служат возможности подключения дополнительных модулей, обеспечивающих понимание системой расширенного синтаксиса (и перевода фрагментов, его использующих, на базовый язык, что задает семантику таких фрагментов). Однако, насколько мы поняли, такое подключение выполняется глобально, на уровне всей программы, и запретить использование расширенного синтаксиса в определенных местах программы очень непросто, если вообще возможно.

Языки низкого и высокого уровня

Как уже было сказано в начале введения, для построения расширяемого языка программирования представляется полезным иметь на самом деле два языка — низкого уровня (как можно проще, с намеренно упрощенным синтаксисом и семантикой, но с реализацией средств расширения), который собственно и подлежит расширению, и язык высокого уровня, который является его расширением и может расширяться далее.

В качестве языка низкого уровня был выбран императивный язык, данные и программы которого выглядят как у LISP'a (с тем отличием, что в соответ-

ствующем дереве атомы, т. е. строки, стоят в каждой вершине дерева, а не только в листьях). Также отличие данной структуры от LISP'а состоит в том, что, кроме строки данных и указателей на сына и правого брата, в каждом узле стоит указатель назад (на левого брата, если он есть, на отца, если нет, и на переменную, содержащую данное дерево, если речь идет о корневом узле; разница между корневым и прочими узлами опознается по специальному флагу, также имеющемуся в каждом узле) и рекурсивный разделяемый мьютекс, предназначенный для организации транзакционной семантики.

Если возникает необходимость реализации дополнительной семантики, например, указания таких дополнительных свойств объекта, как тип, или внутри списка дополнительного указания ключей элементов, превращая список в словарь, используется механизм частных случаев: определенные частные случаи списка закрепляются для этих целей, но при этом сохраняется возможность трактовки такого рода частных случаев непосредственно как они написаны, для чего они должны встречаться внутри других частных случаев такого же вида.

Например, мы можем для указания дополнительных свойств объекта зафиксировать списки, первым элементом которых является строка «свойства», но при этом в таком списке есть данные с ключом «объект», представляющие собой тот объект, свойства которого добавляются, и содержимое этих данных трактуется, как оно написано, даже если это список, первым элементом которых является строка «свойства».

Важным элементом языка низкого уровня является так называемая таблица. Это ассоциативный массив, индексами и данными в котором являются объекты, т. е. деревья со строками в вершинах. При этом смысл данного соответствия определяется типом ключа, например, ситуации вызова функции может соответствовать реализация данной функции, а имени переменной — ее адрес. Такой подход хорош тем, что позволяет иметь все необходимые для разбора и выполнения программы таблицы в рамках одной структуры, и, более того, впоследствии, в качестве расширения, добавлять в систему таблицы, о которых на момент проектирования данной системы вовсе ничего известно не было.

Важным элементом таблицы являются наборы реализаций. Они представляют собой все те же деревья, некоторые части которых помечены особым образом и поэтому воспринимаются как реализации определенных процедур, представленных либо указаниями функций из динамических библиотек, либо списками команд. При этом важно, что сначала запускается самая левая реализация, которая затем может вызывать другие реализации той же процедуры из этого дерева. Такой механизм хорош тем, что позволяет решить относительно просто две проблемы.

Первая из них состоит в обеспечении возможности добавлять новые реализации «поверх» старых (чтобы вызов новой реализации включал в себя, в том числе, функциональность старых), а затем удалять добавленные таким образом реализации в произвольном порядке (при этом при удалении реализации из середины вызов удаленной реализации через обращение к предыдущей реализации из следующей за удаленной просто переключается на ту, которая стояла до удаленной). Вторая проблема состоит в том, чтобы можно было добавлять новые реализации той же функции, расширяющие ее область определения (если выполняемая реализация понимает свою недостаточность для выполнения необходимых действий, она может вызвать следующую, и так до тех пор, пока не будет вызвана последняя, играющая роль реализации по умолчанию).

Таблица, вообще говоря, не монолитна, а состоит из блоков, организованных в список, которые можно добавлять или убирать. Такая организация позволяет таблице иметь «локальные» части, что очень полезно для поддержки механизма локальных понятий для считывания и выполнения.

Модель вычислений

Единственный регистр виртуальной машины, используемой для выполнения программы на языке низкого уровня, хранит указатель на активационную запись работающей в данный момент процедуры. Вся остальная необходимая для ее работы информация, включающая указатели на выполняемый блок и текущую команду в нем, стек данных для вычислений, а также указатели на активационные записи вызвавших ее процедур и вызванных ею, приостановленных, но не законченных, хранятся в этой активационной записи.

В качестве начального механизма синтаксического разбора используется обобщенный GLL-парсер. В нем правыми частями правил грамматики выступают процедуры, код которых линеаризован, т. е. преобразован к такому виду, в котором выражения состоят из вызова одной операции, и нет вложенных управляющих структур (они реализуются через механизм безусловного и условного переходов).

На момент запуска программы, набор понятий, которые могут быть считаны, ограничивается тем самым расширением LISP, о котором шла речь в предыдущем разделе, с той лишь разницей, что поддерживается многостадийное программирование, позволяющее выполнять только что считанные списки команд (они определяются метками, т. е. теми строками, которые стоят в корневых узлах соответствующих деревьев), и если при считывании какого-либо изначально имевшегося понятия возникает ошибка, запускается

определенная для каждого типа ошибки функция, реализация которой берется из таблицы. Это позволяет эффективно бороться с бесконечной рекурсией: если для считывания и выполнения программы достаточно стандартных средств, она просто выполняется, а если нет — запускаются механизмы, отвечающие за расширения.

Одно из неоспоримых достоинств используемой структуры данных состоит в возможности строить большие структуры, содержащие другие подструктуры как свои части, поэтому вполне достаточно для любых функций иметь сигнатуру с одним параметром такого типа по значению и результатом такого же типа. Однако для удобства допускаются функции с большим числом аргументов. Большинство функций, как встроенных, так и реализованных, имеют в качестве параметра указатель на активационную запись, которая строится непосредственно перед первым запуском данной функции и определяет состояние ее выполнения (как, впрочем, и состояние выполнения функций, вызванных из данной, если такие были), так что если функция встроенная, она должна начинать свое выполнение с восстановления состояния, соответствующего содержимому активационной записи. Такой подход превращает функции в сопрограммы. Дополнительные аргументы при передаче их в качестве параметров помещаются в стек данных, имеющийся в активационной записи.

Важным также понятием является понятие типа. В этом проекте тип — это просто одно из дополнительных свойств, некий ярлык, навешиваемый на объект, для правильного выбора реализаций функций обработки данного объекта. Двойственным по отношению к понятию типа является понятие роли — дополнительном свойстве контекста, в который помещается объект. Информация об автоматическом преобразовании типов и ролей, а также соответствия между теми и другими хранится в таблице.

Такие преобразования, в частности, удобны для указания приоритетов операций; такое их использование, впрочем, показывает, что не для всякой роли годится всякое преобразование типов. Например, выражение может быть преобразовано к его значению при помощи вычисления, но если сумма сначала приводится к ее значению, а затем полученное значение используется в роли сомножителя, то получается обход правила о том, что сумма не может быть сомножителем (это правило означает, что приоритет сложения меньше, чем умножения).

Команды

Встроенная команда выглядит как определенного типа список, содержащий имя команды, отвечающее за то, что она должна делать, и параметры, набор которых меняется в зависимости от имени.

Встроены арифметические, логические команды, команды сравнения, простые структурные команды базовой обработки списков, простейшие управляющие команды.

Список встроенных команд может быть вычислен при помощи соответствующей функции, написанной на C++ и помещенной в стандартную библиотеку.

Наряду со встроенными, возможны и реализованные команды, как функции из динамических библиотек или списки команд, помещенные в соответствующий раздел таблицы и вызываемые оттуда, если при выполнении команды как встроенной возникают ошибки (или тип или имя нестандартные, или аргументы неподходящие). Также, возможны списки команд, выполняемые другими функциями, информация о чем должна храниться в дополнительных свойствах соответствующих списков.

Допустимыми также являются команды, ”размножающие” потоки выполнения. Вызов таких команд приводит к тому, что вместо обычных объектов из процедур, использующих такие команды, возвращаются ”суперсписки” результатов. Для реализации такого поведения можно в активационной записи процедур иметь как суперсписок уже накопленных результатов, так и суперсписок уже порожденных, но еще не довычисленных активационных записей.

Указатели

Существует два варианта указателей: традиционные (адреса в памяти) и человеко-читаемые (состоят из последовательности шагов, на каждом из которых происходит уточнение указываемого объекта, как правило, в пределах предыдущего). Наличие у каждого узла дерева указателя назад позволяет легко и быстро переходить от одной формы указателя к другой. Впрочем, шаги указателя могут иметь и менее тривиальный смысл, например, такой, как операция разыменования (указываемый до этого шага объект сам воспринимается как указатель, и мы переходим к тому объекту, на который он указывает). Также, как и в случае с разбором программы, если шаг указателя стандартный (т. е. разновидность шага указателя встроенная и необходимая обработка его данных тоже), его переход обслуживается встроенной функцией, но если во время этого перехода встречаются ошибки, запускается со-

ответствующая функция из таблицы.

Ссылки

Среди специальных видов списков, имеющихся в данной системе, есть еще и ссылки. Встроенными вариантами ссылок являются непосредственные, прямые и косвенные, с очевидным смыслом. Однако, допустимы и пользовательские варианты, для которых в таблице определяются процедуры чтения и записи. Такие ссылки могут определять как положение объектов, на которые они ссылаются, так и дополнительную структуру.

В обычной активационной записи выполняемой процедуры, как правило, имеется стек данных, с которым работает большинство встроенных команд, в частности, команды чтения и записи по ссылкам. Такой подход позволяет разделить одну операцию присваивания на две операции чтения и записи, для которых и определяются пользовательские процедуры по-отдельности.

Список литературы

[1] Тыугу Э. Х. Концептуальное программирование. — М.: Наука, 1984.

[2] Christopher G. S. A Programming Language Where the Syntax and Semantics Are Mutable at Runtime. — Bristol: University of Bristol, 2007.

[3] Lambert M Surhone (ed.), Miriam T Timpledon (ed.), Susan F Marseken (ed.) XL (Programming Language): Programming Language, Concept Programming, Imperative Programming, Derivative, Computer Programming. — Beau Bassin, Mauritius: Betascript Publishing, 2010.

[4] Mertes T. Entwurf einer erweiterbaren höheren Programmiersprache. — Vienna: Vienna University of Technology, 1984.

[5] Bravenboer M. Exercises in Free Syntax Syntax Definition, Parsing, and Assimilation of Language Conglomerates. — Utrecht: Universiteit Utrecht, 2008.

О подходах к идентификации экземпляров мусора на цифровых изображениях

Макаров А.А., СПбГУ, Санкт-Петербург a.a.makarov@spbu.ru,
Макарова С.В., СПбГУ, Санкт-Петербург sdrobot@mail.ru,
Шабунин А.Н, СПбГУ, Санкт-Петербург shandr52@gmail.com

Аннотация

В работе анализируются современные методы идентификации экземпляров мусора на цифровых изображениях с использованием технологий искусственного интеллекта. Рассмотрены подходы компьютерного зрения, глубокого обучения и гибридные системы, а также их применение для классификации, детекции и сегментации отходов. Выделены ключевые проблемы: вариативность объектов, дисбаланс данных и требования к инфраструктуре.

Введение

Рост объемов отходов и ужесточение экологических стандартов требуют принципиально новых методов обработки мусора. Ключевым этапом в цепочке переработки является точная идентификация каждого экземпляра отходов – от пластиковой бутылки до электронного лома. Автоматизация сортировки отходов становится критически важной задачей в контексте глобальных экологических вызовов. Рост объемов мусора, ограниченность ресурсов и необходимость сокращения полигонных отходов требуют внедрения инновационных технологий. Современные подходы на базе искусственного интеллекта, в частности методы компьютерного зрения, позволяют не только точно распознавать и классифицировать экземпляры вторичного сырья на цифровых изображениях, но и определять их границы, материал и степень загрязнения, что критично для автоматизации сортировки. Это не только повышает эффективность переработки, но и снижает затраты, заменяя трудоемкую ручную сортировку автоматизированными системами. Экономический аспект здесь очевиден: автоматизация сокращает операционные расходы и минимизирует человеческие ошибки. Кроме того, стремительное развитие глубокого обучения и увеличение вычислительных мощностей открывают новые горизонты для создания адаптивных и высокоточных моделей.

Глобальный объем твердых отходов превышает 2 млрд. тонн в год, при этом эффективность их переработки остается крайне низкой [1]. Основным

барьером является несовершенство методов сортировки: ручной труд демонстрирует до 30% ошибок неправильной классификации. Важной задачей для нашей страны является обеспечение перехода к созданию замкнутого цикла вторичной переработки отходов, в том числе через внедрение технологий искусственного интеллекта и развитие перерабатывающих мощностей, для возврата ресурсов в производство. Как показывает опыт внедрения системы AMP Robotics, автоматизация на базе технологий искусственного интеллекта, включая компьютерное зрение и глубокое обучение, снижает операционные расходы и увеличивает как скорость обработки, так и точность сортировки [2, 3]. Развитие нейросетевых архитектур (трансформеры, сверточные сети) и появление специализированных датасетов (TACO, WasteNet, TrashCan и т.п.) сделали возможной идентификацию даже мелких или деформированных объектов. Актуальность работы обусловлена экологическими (сокращение полигонных отходов) и экономическими (оптимизация переработки) факторами.

Цель работы – систематизировать современные подходы, оценить их эффективность и определить направления развития. В работе анализируются современные методы идентификации экземпляров мусора на цифровых изображениях с использованием технологий искусственного интеллекта. Рассмотрены подходы компьютерного зрения, глубокого обучения и гибридные системы, а также их применение для классификации, детекции и сегментации отходов. Приведены примеры внедрения в промышленности и бытовом секторе. Обозначены перспективы интеграции с мультимодальными сенсорами и открытыми стандартами. Проведенное исследование подтверждает, что автоматизация идентификации мусора является критическим элементом перехода к циркулярной экономике.

Методы идентификации

Классические алгоритмы компьютерного зрения

Ранние методы основывались на анализе низкоуровневых признаков. Например, пороговые методы, опираясь на цветовые модели (RGB, HSV), использовались для отделения стекла (высокая яркость) от пластика. Однако такие методы демонстрировали низкую точность при сложном освещении [4]. Другой подход заключался в анализе текстуры и формы – детекция на основе HOG (Histogram of Oriented Gradients) и SIFT (Scale-Invariant Feature Transform) – и позволял выявлять ключевые точки для идентификации формы объектов, но уступал нейросетевым методам в скорости и точности [5].

Глубокое обучение

- **Детекция объектов.** Современные архитектуры, такие как YOLO (You Only Look Once), обрабатывают изображения в реальном времени (до 150 FPS) с точностью mAP 70% для датасета TACO [6]. Модель Faster R-CNN, использующая прогноз регионов кандидатов (Region Proposal Network), достигает mAP 85%, но требует значительных вычислительных ресурсов [7].

- **Сегментация.** Для сегментации объектов применяются архитектура U-Net, изначально разработанная для медицинских изображений, позволяющая выделять контуры мусора на сложном фоне [8], и архитектура Mask R-CNN, совмещающая детекцию и выделение контуров, эффективная для работы с перекрывающимися объектами, такими как смятая упаковка, смятая бутылка [9].

- **Трансформеры.** Трансформеры, такие как Vision Transformers (ViT), анализируют глобальные зависимости в изображениях, улучшая идентификацию фрагментированных объектов (осколки стекла) [10]. Иерархическая структура трансформера Swin Transformer снижает вычислительные затраты при работе с высоким разрешением [11].

Гибридные подходы

Комбинация сверточных нейронных сетей с данными от лидаров или спектрометров (multispectral imaging) повышает точность распознавания (см., например, [12, 13]). Применение тепловизоров (thermal imaging) может использоваться для идентификации органических отходов по тепловому следу.

Данные и вычислительная инфраструктура

Наборы данных и аугментация

Публичные датасеты, такие как TACO [14], WasteNet [15], TrashCan [16] и т.п., могут предоставить базовую основу для обучения. Однако они недостаточно покрывают редкие классы (батарейки, электроника). Для решения этой проблемы применяется генерация синтетических данных, например, через GAN (StyleGAN3) или 3D-рендеринг (Blender), моделируя деформированные объекты, такие как смятые бутылки с вариациями освещения [17]. Разметка

данных выполняется с помощью полуавтоматических инструментов (CVAT, Supervisely), а аугментация включает геометрические преобразования (поворот, масштабирование) и наложение артефактов (блики, пыль, капли воды) через библиотеку Albumentations [18]. Микс-аугментация (MixUp) улучшает обобщающую способность моделей, особенно для наборов несбалансированных данных [19].

Проблемы идентификации

Вариативность объектов внутри одного класса (например, ПЭТ бутылки разного цвета, объема, некоторые из которых плохо перерабатываются) и их динамические деформации (смятие, разрывы) усложняют обучение моделей. Фоновый шум, включая наложение объектов друг на друга, блики и погодные условия, требует дополнительной обработки изображений.

Вычислительные мощности

Модели, обученные на синтетических данных, могут некорректно работать в реальных условиях. Обучение больших моделей требует значительных ресурсов. Для работы современных моделей необходимы промышленные системы GPU с поддержкой TensorRT. Мобильные решения (с применением дронов и роботов) могут использовать компактные архитектуры типа MobileNet [20].

Практическое применение

- **Роботизированные линии (AmpRobotics, ZenRobotics).** Интеграция технологий искусственного интеллекта с роботами-манипуляторами позволяет повысить скорость сортировки [2, 3].
- **Мобильные приложения.** On-device модели, например на базе TensorFlow Lite, помогают идентифицировать отходы, используя мобильные устройства [21].
- **Умные контейнеры (Bin-e).** Датчики и камеры анализируют наполнение контейнеров и оптимизируют логистику вывоза [22].

Перспективные технологические направления

Совершенствование методов идентификации мусора требует внедрения инновационных технологий и интеграции смежных подходов. Одним из подходов является использование NeRF (Neural Radiance Fields) [23], позволяющего синтезировать объемные структуры объектов из изображений с известными положениями камеры. Эта технология применима для идентификации деформированных объектов, таких как смятые банки и бутылки, за счет анализа их объемной геометрии. Дополняют этот подход диффузионные модели (Stable Diffusion, DALL-E), которые могут генерировать синтетические изображения мусора в сложных условиях, например, с имитацией загрязнений или частичного перекрытия. Такие модели помогают расширять наборы данных, компенсируя нехватку реальных данных. Для оптимизации процесса разметки можно использовать активное обучение (Active Learning), позволяющее сократить затраты на подготовку данных.

Комбинация мультимодальных сенсоров (RGB-камеры, гиперспектральные датчики, тактильные сенсоры и т. п.) повышает точность распознавания. Технология блокчейна внедряется для прозрачного трекинга жизненного цикла отходов [24]. Пилотные проекты, такие как IBM Waste Tracker, уже демонстрируют эффективность этой методики для отслеживания переработки пластика.

Заключение

Идентификация экземпляров мусора на цифровых изображениях – комплексная задача, требующая сочетания методов компьютерного зрения, глубокого обучения и мультимодальных данных. Современные подходы демонстрируют высокую эффективность, однако их внедрение требует учета вариативности объектов, ограничений данных и инфраструктуры. Перспективы связаны с гибридными системами, сочетающими компьютерное зрение, спектроскопию и робототехнику.

Список литературы

[1] Kaza, S., et al. (2018). *What a Waste 2.0: A Global Snapshot of Solid Waste Management to 2050*. Urban Development. DOI: 10.1596/978-1-4648-1329-0.

[2] *AMP Robotics: Case Studies*. URL: https://ampsortation.com/resources/case-studies.

[3] *ZenRobotics: Case Studies*. URL: https://www.terex.com/zenrobotics/case-studies.

[4] Krizhevsky, A., et al. (2012). "ImageNet Classification with Deep Convolutional Neural Networks." *Advances in Neural Information Processing Systems*, 25. URL: NIPS 2012.

[5] Lowe, D. G. (2004). "Distinctive Image Features from Scale-Invariant Keypoints." *International Journal of Computer Vision*, 60(2), 91-110. DOI: 10.1023/B:VISI.0000029664.99615.94.

[6] Redmon, J., et al. (2016). "You Only Look Once: Unified, Real-Time Object Detection." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. DOI: 10.1109/CVPR.2016.91.

[7] Ren, S., et al. (2015). "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks." *Advances in Neural Information Processing Systems*, 28. URL: NeurIPS 2015.

[8] Ronneberger, O., et al. (2015). "U-Net: Convolutional Networks for Biomedical Image Segmentation." *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. DOI: 10.1007/978-3-319-24574-4_28.

[9] He, K., et al. (2017). "Mask R-CNN." *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. DOI: 10.1109/ICCV.2017.322.

[10] Dosovitskiy, A., et al. (2020). "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale." *International Conference on Learning Representations (ICLR)*. URL: ICLR 2020.

[11] Liu, Z., et al. (2021). "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows." *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. DOI: 10.1109/ICCV48922.2021.00986.

[12] Lu, G., et al. (2022). "Deep multimodal learning for municipal solid waste sorting." *Sci. China Technol. Sci.*, 65, 324–335. DOI: https://link.springer.com/article/10.1007/s11431-021-1927-9.

[13] Shiddiq, M. et al. (2023). "Plastic and organic waste identification using multispectral imaging."

Materials Today: Proceedings, 87 (2), 338-344. DOI: https://www.sciencedirect.com/science/article/pii/S2214785323014724.

[14] Proença, P. F., et al. (2020). *TACO: Trash Annotations in Context Dataset*. arXiv:2003.06975. URL: arXiv:2003.06975.

[15] *WasteNet: The world's largest dataset for waste.* URL: https://recycleye.com/wastenet/.

[16] Hong, J., et al. (2020). *Trashcan: a semantically-segmented dataset towards visual detection of marine debris*. arXiv:2007.08097. URL: arXiv:2007.08097.

[17] Karras, T., et al. (2020). "Analyzing and Improving the Image Quality of StyleGAN." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. DOI: 10.1109/CVPR42600.2020.00813.

[18] Buslaev, A., et al. (2020). "Albumentations: Fast and Flexible Image Augmentations." *Information*, 11(2), 125. DOI: 10.3390/info11020125.

[19] Zhang, H., et al. (2017). "Mixup: Beyond Empirical Risk Minimization." arXiv:1710.09412. URL: arXiv:1710.09412.

[20] Howard, A. G., et al. (2017). "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications." arXiv:1704.04861. URL: arXiv:1704.04861.

[21] Handhayani, T., Hendryli, J. (2023). "Leboh: An Android Mobile Application for Waste Classification Using TensorFlow Lite." *Lecture Notes in Networks and Systems*, 544. DOI: 10.1007/978-3-031-16075-2_4

[22] *Bin-e: recycling solutions*. URL: https://www.bine.world/solutions.

[23] Mildenhall, B., et al. (2020). "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis." *European Conference on Computer Vision (ECCV)*. DOI: 10.1007/978-3-030-58452-8_24.

[24] Nakamoto, S. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*. URL: https://bitcoin.org/bitcoin.pdf.

Заметки о методах преподавания технологий обработки больших данных в прикладных областях

Мирошниченко И.Д., СПбГУ, Санкт-Петербург i.miroshnichenko@spbu.ru,
Султонов А.Ш., СПбГУ, Санкт-Петербург st138466@student.spbu.ru

Аннотация

Преподавание технологий обработки больших данных на математико-механическом факультете и факультете социологии, который представляет прикладную область с точки зрения специалистов разработки программного обеспечения, безусловно, отличается.

Статья посвящена особенностям подачи материала обучающимся старших курсов бакалавров и магистров факультета социологии или других направлений образовательных программ, цель которых — прикладное использование инструментов программного обеспечения, главным образом, изучение, исследование, применение специализированных пакетов, фреймворков и языков R и/или Python для решения реальной задачи моделирования, таким образом, чтобы компенсировать недостатки базовых системных знаний и понятий о структурах данных и параллельных процессах, происходящих при обработке больших данных и плохо формализованных подходах.

Введение

На протяжении почти двадцати лет значимость технологий, опирающихся на трансформацию Больших данных, всё больше увеличивается, эти технологии развиваются и усложняются, всё более широко проникают в самые различные области применения. В настоящее время методы обработки информации Больших данных, методы машинного обучения, нейронных сетей применяют при разработке проектов не только студенты естественно-научных факультетов, но и гуманитарных, решая задачи в прикладных областях цифровой экономики. Однако студентам гуманитарных факультетов или кафедр для получения полноценных профессиональных компетенций в области IT-технологий требуются более глубокие знания о структурах данных, принципах взаимодействия и особенностях функционирования потоков при распараллеливании в различных средах (с общей памятью и распределённых), способах передачи данных, платформах и интерфейсах, библиотеках, инструментах, видах облачных хранилищ и механизмах хранения или обработки

данных в них. Рассмотрение особенностей преподавания с этой точки зрения IT-дисциплин и является целью данной работы.

Различия в подходах к обучению

Понятно, что на гуманитарных факультетах преподаватели-специалисты, главным образом, обучают навыкам *применения* конкретных *методов* для решения конкретных задач проблемной области, а также изучают функции языков R и Python с точки зрения использования.

Это главное отличие направленности изучения методов обработки информации и структуры и функционирования языков программирования на естественно-научных факультетах (в частности на математико-механическом) и гуманитарных. На математико-механическом факультете как изучение основ программирования, так и методик, технологий, строится на детальном рассмотрении базовых структур, процессов, связей на всем процессе изучения различных дисциплин, на гуманитарных факультетах внимание обращается на возможности и набор средств (функций, пакетов), применяемых для конкретных целей в решении задач предметной области. Поэтому приходится обращать внимание на основы: на вид и представление данных в памяти, на особенности построения и функционирования конструкций, исключительные и критические состояния, на архитектуру аппаратных средств и влияние их конфигурации на ускорение вычислений или процесса взаимодействий, на необходимость, значимость и суть стандартов на всех уровнях от элементов архитектуры до построения инфраструктуры программного обеспечения.

Первый этап в освоении Big Data — изучение способов сбора, очистки, проверки на валидность больших данных и подготовки информации к дальнейшим преобразованиям — в основном, в изучении сложности не представляет, здесь, скорее, требуется исследование соответствующих пакетов, применяемых в конкретной прикладной области. Следующие этапы: способы и методы хранения и обработки — наоборот, включают в себя понимание принципов функционирования *распределенных систем хранения данных* и обработки запросов, а также основ механизмов работы с облачными платформами, которые предоставляют необходимую масштабируемость и гибкость для работы с большими данными. Кроме технических аспектов, важно не только применять, но и понимать суть методов анализа данных, включая *статистический анализ*, *машинное обучение* и методы визуализации данных, чтобы извлекать ценную информацию из больших объемов данных.

Как уже упоминалось, основные языки, используемые в работе с Big Data, — это Python и R. На разных платформах в интернет-пространстве орга-

низованы мощные библиотеки (в том числе большое количество авторских свободного доступа) для анализа данных и машинного обучения. Обычно в учебном процессе прикладной области используется достаточно большой набор известных пакетов (кроме базовых, предлагаемых средой моделирования), а в них рассматриваются конкретные команды (например, визуализации или анализа) с конкретными, наиболее часто используемыми, параметрами. В курсах, касающихся больших данных, приходится посвящать время тому, чтобы научить анализировать не только информацию, предоставленную автором в документации о методе, но и предварительно исследовать правильность его работы на разных типах данных, его сходимость, обусловленность и другие характеристики прежде, чем применить в собственном проекте.

Для того, чтобы разработать эффективный проект, используют такие инструменты, как *Apache Hadoop* и *Spark*, *NoSQL* базы данных, которые представляют собой основу для обработки, интеграции и анализа больших объемов данных, обеспечивая необходимую масштабируемость и гибкость. Также важным аспектом обучения является овладение навыками визуализации данных с помощью инструментов, таких как *Tableau* и *PowerBI*, что позволяет эффективно представлять и интерпретировать сложные наборы данных. Самостоятельная работа над проектом позволяет студентам применять изученные инструменты и методы в контексте реальных задач, развивает критическое мышление и аналитические навыки. Функция преподавателя в описываемых условиях – научить ориентироваться в обилии предоставляемой информации, критически отбирая наиболее подходящие инструменты к прикладной области, а также умению проводить сравнительный анализ их применимости к реальной задаче, профессиональному отбору параметров.

Необходимость фундаментальных знаний

Восполнение пробелов в базовых знаниях

Но для самостоятельной реализации поставленной задачи, оптимальной разработки структуры проекта, грамотного написания сценария нужны структурные основы базовых знаний. Восполнить эти пробелы — задача преподавателей математико-механического факультета, ведущих дисциплины, главным образом, на старших курсах. В частности, это касается, например, технологий, включающих понятия *распараллеливания* и/или *взаимодействия потоков* или детального объяснение понятий и особенностей моделирования и функционирования процессов, без которых вряд ли можно полноценно объяснить особенности структуры и функционирования *ETL* и *ELT* (основные механизмы обработки больших данных в процессе извлечения зна-

ний (*Data Mining*), машинного обучения, видов глубокого обучения.

Наконец, следует отметить, что ограничения в емкости устройств хранения информации сейчас уже не играют определяющей роли, значительно важнее становятся временные характеристиками вычислителей, и это требует внедрения в практику многопроцессорных распределенных систем. Возникает необходимость изучения специальных методов обработки информации, учитывающих эти обстоятельства.

Корректировка учебного плана: основные направления

Отметим ниже то новое, что пришлось ввести в текущий процесс (не меняя содержательную часть РПД дисциплин) при подготовке материала, касающегося изучения среды программирования R и Python и работы с большими данными:

1. В преподавании дисциплин, связанных с языками программирования R и Python, популярных в обработке больших данных и предоставляемых на бесплатных платформах (например, в тематике «среда R и доступ к данным»), нужно обращать внимание на то, что эти языки имеют свойства как *функциональных языков, так и объектно-ориентированных*, и показать, в чём это проявляется. Кроме того, часто приходится объяснять свойства этих парадигм на примерах, так как, в общей массе, обучающиеся эти понятия плохо представляют. В частности, важно отметить, что, ввиду свойства функциональности, переменной, как таковой, в языке нет. Есть объект, который строится из одномерного вектора с единственной компонентой, и объект не изменяется. При конструировании нового объекта из старого объекта, у объекта меняется ссылка на новое значение.

   Написание грамотных структурных сценариев – немаловажный пункт обучения программированию (и в прикладной области). Студенты не придают этому должного внимания, однако это важный момент, ускоряющий процесс написания, понимания и отладки сценария. Особенность парадигмы функционального программирования, так называемые, *ленивые вычисления*, неявно используются в реализациях сценариев на языках R и Python, но позволяют написать грамотные сценарии, реализующие сложные итеративные бесконечные методы. Таким образом, знание особенностей структур языка позволяет написать эффективный сценарий.

2. При изучении технологии работы с большими данными также оказывается полезным напомнить (или заново объяснить) структуру таких объ-

ектов как *хранилище, витрина, озеро*, по каким принципам строятся различные типы баз данных, каковы их свойства. Обычно обучающиеся приходят с навыками работы с реляционными базами данных, но с другими типами баз данных знакомы плохо. А для понимания построения, функционирования хранилищ, витрин, озер важно иметь представление:

- о многомерных базах данных, гибридных структурах,
- базах данных NoSQL, их функциональных особенностях,
- структурированных и неструктурированных,
- какие инструменты работают с теми или иными типами данных.

3. Важная тема, на которую надо обратить внимание — это *открытые системы*. Хотя в учебных планах явно могут не присутствовать пункты, касающиеся базовой трехуровневой модели открытых систем, оказывается полезным напомнить или объяснить:

   - что представляет собой протокол и интерфейс, а также семиуровневая модель передачи данных,
   - понятия масштабирования и балансировки в многопроцессорных и распределённых системах.

   которые весьма важны для понимания и общего представления современной инфраструктуры программного обеспечения. Этих понятий практически не касаются или недостаточно подробно изучают в профильных дисциплинах гуманитарных факультетов.

4. Хотя в профильных дисциплинах изучаются (или ранее в предыдущих курсах, возможно, изучались более подробно) плохо формализуемые подходы к решению задач, а именно, нейронные системы, генетические алгоритмы и нечеткие подходы, тем не менее, весьма полезно детально *проработать методику решения задач* с подробным разбором примеров, в особенности, *генетические алгоритмы*, необходимые в исследовании Интернет-пространства, и *нечеткие системы*, нужные для исследования систем с плохо формализуемыми входными и/или выходными данными.

Таким образом, в современных реалиях в лекциях по дисциплинам, связанным с большими данными, для гуманитариев требуется подробно объяснять студентам необходимость более детально (хотя в плане и содержательной части дисциплин предполагается прикладная составляющая, например, «доступ к данным из сети Интернет»)

- изучать технологии *MapReduce* и *Data Mining* (модель распределённых вычислений, представленная компанией Google, используемая для параллельных вычислений над очень большими наборами данных в компьютерных кластерах).
- анализировать функциональную структуру пакета утилит и библиотек *Hadoop* (используемого для построения систем, обрабатывающих, хранящих и анализирующих большие массивы нереляционных данных, а именно, данные датчиков, интернет-трафика, объектов метаданных, файлов журналов, изображений и сообщений в соцсетях).
- использовать возможности фреймворка *Big Data Storm* (программной платформы, определяющей структуру программной системы, иначе, программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта и созданный для работы с информацией в режиме реального времени).
- облачную технологию *DataLake* (инструмент, который позволяет хранить любые данные: `csv`, `xml`, `json`, `parquet`, `jpg`, `png`, `mov`, `mp3`, `pdf` и другие). В него можно загружать таблицы, у которых нет чёткой структуры, то есть периодически меняется количество и названия колонок и строк. Все эти данные можно загружать в озеро без обработки, то есть практически мгновенно. включающую в себя программную платформу, источники и методы пополнения данных, кластеры узлов хранения и обработки информации, управления, инструментов обучения). DataLake при необходимости масштабируется до многих сотен узлов без прекращения работы кластера.

Педагогические выводы и значение самообразования

Методы, инструменты, технологии, отмеченных выше подходов и интеграция Big Data с передовыми технологиями такими, как искусственный интеллект и машинное обучение позволяют более эффективно анализировать и использовать большие объемы данных, последние разработки и тенденции в этой быстро меняющейся цифровой инфраструктуре программного обеспечения.

Кроме того, непрерывное обучение и самообразование, чтение актуальной литературы и последних исследований в области обработки и анализа данных, играют важную роль в поддержании актуальности знаний и оставаться в курсе последних тенденций и инноваций в этой динамично развивающейся области. Реализация этого принципа в преподавании – в обязательной подготовке докладов на семинарах по современным технологиям.

Таким образом, материал дисциплин для факультета социологии и гуманитарных факультетов был серьезно переработан с учетом всех выше указанных пунктов в отличие от содержания материала для одноименных естественно-научных факультетов.

Заключение

В работе рассмотрены особенности преподавания дисциплин обработки и преобразования больших данных (актуальных сегодня на факультетах гуманитарных направлений для разработки проектов в различных прикладных областях) с точки зрения углубленности понимания внутренних процессов и инфраструктуры программного обеспечения, необходимых для повышения профессионализма специалиста. Для грамотной профессиональной подготовки студентов любой образовательной программы необходимо, *не ограничиваться прикладными наборами инструментов для разработки студенческих проектов*, необходимо иметь более глубокие знания в области как *структуры программного обеспечения, так и функциональности его отдельных инструментов и комплексного взаимодействия.*

Результаты могли бы быть полезны для построения соответствующих курсов гуманитарных направлений для магистров или бакалавров, что позволяло бы

- учитывать более профессионально особенности современного развития в области больших данных (и их роста),
- учитывать необходимость иметь представление о распределенных вычислениях, киберугрозах,
- разрабатывать проекты с учетом требований высокой доступности, адаптируемости к изменяющимся данным и их объёму, масштабируемости продуктов,
- иметь представление о подходах контейнеризации, микросервисных архитектур и облачных технологий.

Всё это могло бы помочь в выборе конкретного программного обеспечения для направлений разработки студенческих проектов.

С введением повсеместно в учебные планы дисциплин, связанных с обработкой больших данных, методов машинного обучения, алгоритмов исследования естественного языка с помощью искусственного интеллекта (не только как применение методов) становится еще более важно *грамотно и глубоко* изучать технические особенности рассматриваемых процессов.

Список литературы

[1] Боровская Е.В., Давыдова Н.А. Основы искусственного интеллекта: учебное пособие. — М.: БИНОМ Лаборатория знаний, 2020. — 127 с.

[2] IBM (2023). ELT vs. ETL: Similarities and Differences. https://www.ibm.com/think/topics/elt-vs-etl

[3] AWS (2023). ETL and ELT design patterns for lake house architecture using Amazon Redshift: Part 1. AWS Big Data Blog. https://aws.amazon.com/blogs/big-data/etl-and-elt-design-patterns-for-lake-house-architecture-using-amazon-redshift-part-1

[4] Освоение Big Data: стратегии и методы обучения для современных технологий // ЦифроОбраз. Образовательный портал про современное образование. — 20.12.2023. https://iit-bsuir.by/tehnologii-budushhego/osvoenie-big-data-strategii-i-metody-obuchenija-dlja-sovremennyh-tehnologij/

[5] Степанов А.Г., Плотников Г.А., Васильева В.С. Подходы к определению средств для построения методики обучения работе с большими данными // Информатика и образование. — 2021. — № 4 (323). — С. 54–61. https://info.infojournal.ru/jour/article/view/681/538

[6] «А можно быстрее?»: разбираем методы ускорения обучения нейронных сетей // PVSM.RU. — 05.09.2024. https://www.pvsm.ru/date/2024/09/05

[7] Зайнидинов Х.Н., Маллаев О.У., Зулунов Р.М., Нурмуродов Ж.Н. Методы *распараллеливания* процессов вычисления больших объемов данных с использованием технологий параллельного программирования // Автоматика и программная инженерия. — 2019. — № 4 (30). http://www.jurnal.nips.ru/

[8] Мирошниченко И.Д., Загоревская Н.С. Задачи обучения особенностям параллельного программирования в среде MPI на базе кластерного компьютерного класса. — 2019. https://pureportal.spbu.ru/files/116708592/_2019_.pdf

Прикладная кибернетика и искусственный интеллект

Кузнецов Николай Владимирович
член-корреспондент РАН, д.ф.-м.н., профессор, заведующий кафедрой прикладной кибернетики

Использование текстурных признаков Харалика для анализа изображений щитовидной железы

Соловьев И.П., СПбГУ, Санкт-Петербург i.soloviev@spbu.ru,
Лямин В.А., СПбГУ, Санкт-Петербург st067898@student.spbu.ru

Аннотация

В данной работе рассматривается применение текстурных признаков Харалика для распознавания узлов на ультразвуковых изображениях щитовидной железы. Ее основными целями являются извлечение численных признаков, характеризующих текстуру изображения, и последующее обучение нейросетевой модели. Эксперименты показывают, что оптимальный размер исследуемой области составляет 32×32 пикселя, а использование цветовых моделей RGB и HSV обеспечивает точность распознавания не менее 90%. Полученные результаты демонстрируют перспективность данного подхода для повышения эффективности медицинской диагностики.

Введение

Изображения ультразвукового исследования (УЗИ) являются дешевым, простым и эффективным способом для выявления аномалий щитовидной железы человека, в частности, онкологического заболевания. Одним из признаков рака является появление в щитовидной железе так называемых «узлов». Узлами принято называть все новообразования, которые отличаются по эхогенности (способности отражать ультразвуковой сигнал) от окружающей ткани щитовидной железы. Они делятся на несколько видов:

- изоэхогенные – узлы совпадают по плотности с тканью щитовидной железы;
- гиперэхогенные – узлы с повышенной эхогенностью (более плотная ткань); на изображении УЗИ выглядят более светлыми;
- анэхогенные – узлы с пониженной эхогенностью (менее плотная ткань); на изображении УЗИ выглядят более темными.

При просмотре большого числа изображений в медицинской клинике важно иметь программный инструмент, позволяющий быстро выявлять новообразования в щитовидной железе, а также помочь медикам определить пациентов, требующих срочной госпитализации. Существуют много различных

методов анализа цифровых изображений для извлечения характерных признаков для последующего обучения классифицирующих нейронных сетей. В данной статье мы используем для анализа избранные текстурные признаки Харалика [1], а также метод бинарной логики [2], и сравниваем полученные результаты.

Данные и методы

Обработка изображений

Для проведения исследования были взяты изображения УЗИ щитовидной железы, полученные в Клинике высоких медицинских технологий им. Н.И. Пирогова. Далее на заранее обученной модели для распознавания объектов была произведена разметка изображений в системе Supervisely [3]. Данная программа используется специалистами клиники и предназначена для разметки изображений вручную или при помощи нейронных сетей. На изображениях была выделена нормальная ткань щитовидной железы и узлы. На следующем этапе были уточнены границы разметки, а также проведена верификация полученных изображений специалистом клиники. Для корректного обучения нейронной сети необходимо обеспечить баланс между классами, извлекая равное количество областей с нормальной тканью щитовидной железы и с узловыми образованиями. Это позволяет избежать смещения модели в сторону одного из классов. На рисунке 1 показан пример получения данных областей. Они выбирались таким образом, чтобы в итоге получилось максимально возможное множество непересекающихся областей.

Применение текстурных признаков Харалика

Применение текстурных признаков Харалика основано на предположении, что текстурная информация изображения определяется пространственными взаимосвязями уровней серого. Эти взаимосвязи представляются в виде матриц, отражающих различные типы отношений между пикселями в заданной окрестности [4]. Далее из этих матриц извлекаются численные признаки для последующего анализа. Для корректной работы данного метода необходимо выбрать область на изображении в виде прямоугольника, где N_x – количество пикселей по горизонтали и N_y – количество пикселей по вертикали, а N_g – количество уровней серого. Далее вводим горизонтальную пространственную область $L_x = \{1, ..., N_x\}$ и вертикальную пространственную область $L_y = \{1, ..., N_y\}$, а также множество уровней серого $G = \{1, ..., N_g\}$.

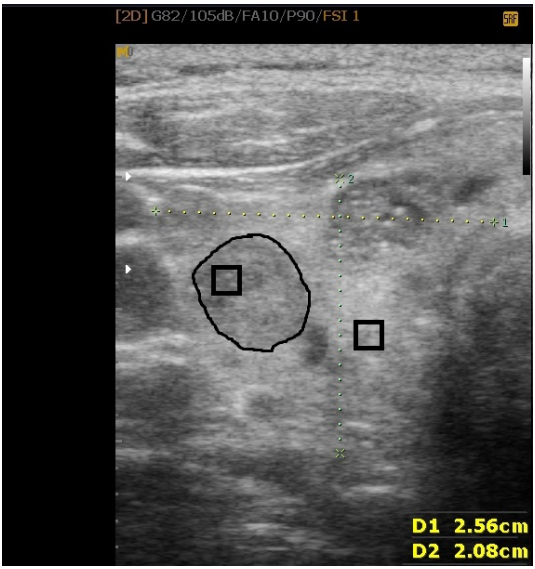


Рис. 1: Изображение УЗИ щитовидной железы с одним очерченным узлом и двумя квадратными блоками, выбранными из нормальной и узловой ткани щитовидной железы (внутри выделенной области)

Основываясь на введенных понятиях, определим функцию изображения I следующим образом: $I : L_y \times L_x \to G$.

Основным этапом реализации данного метода является вычисление четырех мер, представляющие собой матрицы частот P, в которых два соседних пикселя находятся на расстоянии d. Элементами матрицы P являются ненормализованные отношения уровней серого, которые записываются следующим образом:

- $P_{d,0^\circ}(i,j) = \#\{((k,l),(m,n)) \in (L_y \times L_x) \times (L_y \times L_x) | k-m = 0, |l-n| = d, I(k,l) = i, I(m,n) = j\}$;
- $P_{d,45^\circ}(i,j) = \#\{((k,l),(m,n)) \in (L_y \times L_x) \times (L_y \times L_x) | (k-m = d, l-n = -d) or (k-m = -d, l-n = d), I(k,l) = i, I(m,n) = j\}$;
- $P_{d,90^\circ}(i,j) = \#\{((k,l),(m,n)) \in (L_y \times L_x) \times (L_y \times L_x) | |k-m| = d, l-n = 0, I(k,l) = i, I(m,n) = j\}$;
- $P_{d,135^\circ}(i,j) = \#\{((k,l),(m,n)) \in (L_y \times L_x) \times (L_y \times L_x) | (k-m = d, l-n = d) or (k-m = -d, l-n = -d), I(k,l) = i, I(m,n) = j\}$,

где # обозначает мощность множества. На рисунке 2 показан пример вычисления данных матриц для изображения четыре на четыре пикселя при d равном одному.

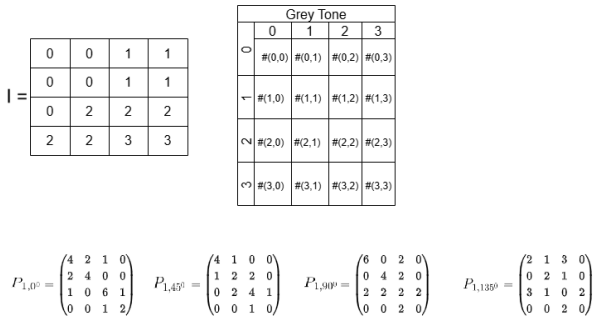


Рис. 2: Пример вычисления матриц. В качестве примера возьмем изображение I размером 4x4 пикселя с четырьмя уровнями серого. Затем построим таблицу отношений уровней серого, по которой вычисляем четыре матрицы вхождения P при d равном единице.

Следующим этапом необходимо нормализовать получившиеся матрицы, таким образом, чтобы сумма всех элементов равнялось единице. При помощи нормализованной матрицы p вычисляем математическое ожидание по строкам и столбцам (μ_x, μ_y), а так же их дисперсии(σ_x, σ_y) [5]. Формулы для расчета данных величин приведены ниже (так как для всех последующих вычислений выполняются одни и те же действия, то индексы расстояния между пикселями и угла опускаем):

- $\mu_x = \sum_{i=1} i \sum_{j=1} p(i,j)$;
- $\mu_y = \sum_{i=j} j \sum_{i=1} p(i,j)$;
- $\sigma_x = \sum_{i=1}(1-\mu_x)^2 \sum_{j=1} p(i,j)$;
- $\sigma_y = \sum_{j=1}(1-\mu_y)^2 \sum_{i=1} p(i,j)$.

При помощи ранее полученных величин далее вычисляются 15 признаков, которые можно использовать при дальнейшем исследовании изображения. В наших экспериментах наилучшие результаты получаются при использовании следующих признаков: однородность, контраст, корреляция, дисперсия, гомогенность.

- $f_1 = \sum_i \sum_j p(i,j)^2$ – однородность. Это текстурный признак, который характеризует однородность текстуры изображения. Близкое к единице значение данного признака говорит о гладкой структуре, т.е. значения пикселей мало вариативны и часто повторяются.

- $f_2 = \sum_{n=0}^{N_g-1} n^2 (\sum_{|i-j|=n}^{N_g} p(i,j))$ – контраст. Данный признак показывает насколько сильно отличаются интенсивности соседних пикселей.

- $f_3 = \frac{\sum_i \sum_j (ij) p(i,j) - \mu_x \mu_y}{\sigma_x \sigma_y}$ – корреляция. Она измеряет степень линейной зависимости между значениями интенсивностей пикселей, расположенных рядом. Например, низкая корреляция указывает на наличие резких границ и высокочастотных изменений в изображении.

- $f_4 = -\sum_i \sum_j p(i,j) \log(p(i,j))$ – энтропия Шеннона. Данный признак измеряет степень беспорядка или сложности текстуры на изображении.

- $f_5 = -\sum_i \sum_j \frac{1}{1+(i-j)^2} p(i,j)$ – гомогенность. Это признак измеряет, насколько близки значения пикселей друг к другу. В ходе экспериментов было выявлено, что он наиболее сильно влияет на точность модели (до 70%).

Обучение нейронных сетей

В качестве модели машинного обучения был выбран метод опорных векторов (SVM) библиотеки scikit-learn [6]. Перед обучением признаки были масштабированы, чтобы привести их к единому масштабу и избежать доминирования одних признаков над другими, а также данные были разделены на две группы. Обучающая выборка составляла 80% данных, тестовая — 20%.

Результаты

На первом этапе экспериментов был определен оптимальный размер извлекаемой области. В серии экспериментов рассматривались квадратные области размером 20x20, 25x25, 32x32 и 35x35 пикселей. Результаты приведены в таблице 1. На них видно, что наибольшей точности (доля правильных предсказаний среди всех предсказаний модели) модель достигает при размере области 32 на 32 пикселя.

На втором этапе была исследована зависимость точности модели от цветовой палитры. Использовались палитры RGB, LAB и HSV. Результаты приведены в таблице 2. Наибольшей точности удалось достичь на моделях RGB и HSV.

Размер области	Точность модели
20x20	0.68
25x25	0.76
32x32	0.91
35x35	0.89

Таблица 1: Точность модели в зависимости от размера области

Цветовая модель	Точность модели
RGB	0.91
HSV	0.9
LAB	0.82

Таблица 2: Точность модели в зависимости от цветовой модели

Для сравнения с другими методами извлечения признаков из изображения был реализован метод локального бинарного шаблона (LBP) [7]. Но модель, обученная на признаках, полученных при помощи данного метода, показала точность в 60%, что значительно хуже, чем при использовании текстурных признаков Харалика.

Заключение

В данной статье представлен подход к обработке размеченных изображений УЗИ щитовидной железы для последующего извлечения численных характеристик при помощи текстурных признаков Харалика. Также приведены серии экспериментов, по которым можно сделать следующие выводы:

- наиболее оптимальным размером области для разделения узлов от нормальной ткани щитовидной железы составляет 32 на 32 пикселя;
- использование цветовых моделей RGB и HSV дает лучший результат, чем LAB.

Благодарности

Авторы благодарят Клинику высоких медицинских технологий им. Н.И. Пирогова за предоставленные образцы изображений.

Список литературы

[1] Haralick R.M., Shanmugam K., Dinstein I. Textural features for image classification // IEEE Transactions on systems, man and cybernetics. 1973. Vol. SMC 3. № 6. P. 610-621.

[2] Stamos Katsigiannis Eystratios Keramidas Dimitris Maroulis Local Binary Patterns: New Variants and Applications of Ultrasound Images and Videos January 2014 149–175.

[3] Supervisely. Supervisely: AI-powered platform for computer vision [Электронный ресурс]. – Режим доступа: https://supervisely.com (дата обращения: 11.05.2025).

[4] Haralick R.M. Statistical and structural approaches to texture // Proceedings of the IEEE. 1979. Vol. 67, №.5. P. 786 - 804.

[5] Белов Н. И., Ермак М. А., Дубинич Е. А., Кузнецов А. Ю. Распознавание изображений на основе текстурных признаков Харалика и искусственных нейронных сетей // Научно-технический вестник информационных технологий, механики и оптики. — 2022. — Т. 22, № 2. — С. 279–286.

[6] Pedregosa, F., Varoquaux, G., Gramfort, A. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825–2830

[7] Dimitris K. Iakovidis a, Eystratios G. Keramidas b, Dimitris Maroulis Fusion of fuzzy statistical distributions for classification of thyroid ultrasound patterns // Artificial Intelligence in Medicine 50 (2010) 33–41.

Управление рисками и большие данные в промышленной безопасности киберфизических систем

Валеев С.С., УУНиТ, Уфа vss2000@mail.ru,
Кондратьева Н.В., УУНиТ, Уфа knv24@ymail.ru

Аннотация

Рассматривается задача обеспечения безопасности технологических процессов в киберфизических системах на базе внедрения современных информационных технологий, таких как анализ больших объемов данных и методы искусственного интеллекта. Отмечается, что использование технологий больших данных для обеспечения безопасности технологических процессов требует высокой культуры данных в компании. Самое главное в формировании этой культуры – бережное отношение к информационным активам организации. Сбор данных требует определенных усилий и инвестиций. Однако не всегда ясно, какие данные полезны, как долго данные следует хранить, и, самое главное, как данные следует использовать позже. Этот долгий путь от сбора данных до их использования необходимо пройти, и время все расставит по своим местам. Обсуждаются основные этапы внедрения технологий больших данных для обеспечения технологической безопасности на предприятии.

Введение

Киберфизические системы относятся к классу нелинейных распределенных систем управления, функционирующих в условиях воздействия различных факторов неопределенности, которые могут привести к нарушению законов управления и возникновению критических ситуаций [1]. В настоящее время для решения задач сопровождения подобных систем активно внедряются технологии больших данных с целью повышения технологической безопасности (управление на основе данных) [2], [3], [4].

При реализации технологических процессов на различных этапах формируется большой массив данных, отражающий текущее состояние киберфизической системы. Сложность технологических процессов и необходимость их постоянного мониторинга приводят к тому, что формируется большой массив данных (Big data), который необходимо хранить в течение определенного времени. Часто данные хранятся в разных несвязанных информационных

системах, что в некоторых случаях не позволяет проанализировать возникновение критической ситуации с учетом всех имеющихся данных.

Тем самым, формирование культуры работы с данными на предприятии является актуальной, и позволяет минимизировать риски возникновения критических ситуаций в условиях влияния различных факторов неопределенности.

Основные этапы внедрения технологий больших данных

Рассмотрим основные этапы внедрения технологий больших данных, используемых для решения задач обеспечения технологической безопасности в киберфизических системах.

На первом этапе необходимо провести аудит данных. Основная цель: провести аудит имеющихся данных, необходимый для выполнения функций технологической безопасности в соответствии со стандартами и регламентами. Основные процедуры реализации: составление перечня проблем обеспечения технологической безопасности объектов и инфраструктуры и возможных решений, необходимых для их реализации.

Отметим, что обычно данные собираются для формирования стандартных отчетов, т. е. целью в данном случае является формирование отчета. После этого неясно, что делать с массивами данных. Их можно оставить на хранение, а можно удалить. При хранении не всегда понятно, как долго необходимо хранить данные. Процесс аудита вызывает много вопросов, связанных со сроками хранения, ценностью информации и ее старением.

В настоящее время хранение информации стало частью отрасли обработки данных, и эти вопросы решаются в рамках облачных хранилищ компаний, относящихся к информационной отрасли. При хранении данных в системах хранения подрядчиков, в свою очередь, возникает задача сохранения конфиденциальности информационных ресурсов.

На втором этапе подготовки данных необходимо выполнить классификацию данных. Основная цель: классифицировать имеющиеся данные и их источники. Основные процедуры реализации: определение наличия данных из сертифицированных источников информации. Далее следует сформировать список необходимых данных, которые утеряны или могут быть утеряны. Необходимо также сформировать список недостающих требуемых данных из сертифицированных источников информации. Наконец, требуется сформировать список данных, которые нужны, но их источники недоступны или неизвестны.

Для интеллектуального анализа данных требуются дополнительные ре-

сурсы организации. Если это датчики, определяющие концентрацию вредных веществ, то при выборе типа, количества и мест их установки необходимо помнить о жизненном цикле данных, как упоминалось ранее. Поддержание жизненного цикла данных требует соблюдения метрологических стандартов, сертификации датчиков и программного обеспечения, сертификации каналов связи и систем защиты информации. В ряде случаев в рамках концепции больших данных могут использоваться информационные ресурсы других компаний. В этом случае возникает вопрос о качестве данных, предоставляемых сторонней компанией.

На третьем этапе проводится классификация данных по приоритету. Основная цель: расставить приоритеты и выделить наиболее важные ресурсы данных. Основные процедуры реализации: оценка сложности получения данных, формирование списка приоритетов при формировании набора данных и оценка затрат на их сбор.

Необходимо учитывать динамику приоритетов на различных этапах жизненного цикла киберфизических систем. Отметим, что следует помнить о противоречиях, которые могут возникнуть при выборе шкалы приоритетов. Выбор приоритетов является сложной оптимизационной задачей в рамках теории систем поддержки принятия решений, например, на основе многозначной логики.

Стоимость получения данных и формирование набора данных во многом зависит от типа киберфизической системы и ее особенностей. Стоимость может варьироваться за счет оптимизации затрат на датчики и контроллеры и формирования цифровых двойников объектов управления.

На четвертом этапе проводится анализ методов сбора данных. Основная цель: повышение эффективности используемых данных. В трудовой функции персонала, отвечающего за промышленную безопасность, необходимо описать алгоритм, как собирать данные о технологических процессах и инцидентах.

Несмотря на активное внедрение автоматизированных систем сбора данных, сохраняются ручные операции по контролю состояния конструкций, например, внешний осмотр установки, трубопровода, сварных швов и т. д. Также к ручным операциям относятся формирование отчетов и заполнение журналов. Следует отметить, что трудоемкие операции, связанные с обработкой журналов, отчетов и занесением этих данных в информационную систему, должны выполняться персоналом, ответственным за обработку данных.

При этом в информационной системе необходимо использовать алгоритмы анализа корректности сформированных наборов данных [5]. В рамках внедрения культуры данных в организации важно понимать, что сбор данных требует участия квалифицированных специалистов разных уровней [6].

На пятом этапе определяется способ и места хранения набора данных. Основная цель: обеспечение надежного хранения данных и возможности доступа к ним по требованию с использование информационных систем. Основные средства, используемые для эффективного хранения данных, включают в себя распределенные хранилища данных на базе облачных технологий [7].

При этом необходимо обеспечить эффективные меры защиты хранящихся данных на основе, например, распределения прав доступа, а также необходимо контролировать изменения, вносимые в наборы данных и методы управления версиями.

Процессы сбора и обработки данных для управления рисками

На Рис. 1 представлен процесс формирования потока данных о состоянии технологических процессов, анализе отказов и инцидентов, где DBP - база данных результатов мониторинга персоналом хода технологических процессов; DBS - база данных результатов измерений датчиками характеристик технологических процессов; DBf – база данных с результатами анализа отказов и неисправностей оборудования, а также нарушений в ходе технологических процессов; DBh - база данных, содержащая информацию о критических ситуациях и их последствиях.

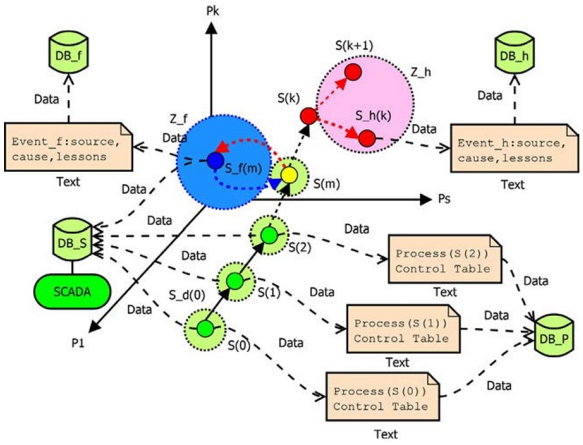


Рис. 1: Процессы сбора и обработки больших данных о состоянии киберфизической системы

Часть данных отражается в различных формах, например в виде отчетов, представленных в бумажных документах. Затем эта информация вводится в соответствующие информационные системы и хранится в базах данных. Например, отчеты по управлению технологическим процессом содержат информацию об отклонениях параметров и их причинах. Эти отчеты формируются двумя способами: технологом и с использованием автоматизированной системы управления и мониторинга (SCADA). Собранные данные позволяют анализировать состояние киберфизической системы и в случае необходимости влиять на развитие критической ситуации.

Заключение

Как следует из представленных основных шагов работы с данными (формирование культуры данных), это достаточно трудоемкий процесс. В настоящее время в современных компаниях вводятся должности ответственных за данные. Отметим, что это важные этапы на пути к применению больших данных при решении задач промышленной безопасности.

Важную роль на всех рассмотренных этапах играют процессы сбора и обработки данных. При этом основными проблемами являются синхронизация, верификация и необходимость интеграции разнородных информационных систем, что является основой культуры работы с данными.

Список литературы

[1] Kuznetsov N.V., Mokaev T.N. Numerical analysis of dynamical systems: unstable periodic orbits, hidden transient chaotic sets, hidden attractors, and finite-time Lyapunov dimension // Journal of Physics: Conference Series, 2019. — С. 012034. DOI:10.1088/1742-6596/1205/1/012034.

[2] Nguyen T., Gosine R.G., Warrian P. A systematic review of big data analytics for oil and gas Industry 4.0 // IEEE Access, 2020. 8:6118361201. DOI:10.1109/ACCESS.2020.2979678.

[3] Sircar A., Yadav K., Rayavarapu K., Bist N., Oza H. Application of machine learning and artificial intelligence in oil and gas industry // Petrol Res, 2021. 6:379-391. DOI:10.1016/j.ptlrs.2021.05.009.

[4] Tariq Z., Aljawad M.S., Hasan A., et al. A systematic review of data science and machine learning applications to the oil and gas industry. // J Petrol Explor Prod Technol. — 2021. 11:4339-4374. DOI:10.1007/s13202-021-01302-2.

[5] Луцив Д.В., Кознов Д.В., Шелиховский А.А., Романовский К.Ю., Чернышев Г.А., Терехов А.Н., Григорьев Д.А., Смирнова А.Н., Боровков Д.В., Васенина А.Н. Interactive duplicate search in software documentation // Статья в открытом архиве. — 2019. № arXiv:1908.08266 22.08.2019.

[6] Кондратьева Н.В., Янгирова А.Ф., Валеев С.С. Информационная система управления эвакуацией людей в критических ситуациях // В сборнике: XII всероссийское совещание по проблемам управления ВСПУ-2014. Институт проблем управления им. В.А. Трапезникова РАН. — 2014. — С. 8173–8179.

[7] Valeev S., Kondratyeva N. Large scale system management based on Markov decision process and big data concept // В сборнике: Application of Information and Communication Technologies, AICT 2016 - Conference Proceedings. — 2016. — С. 7991829. DOI:10.1109/ICAICT.2016.7991829.

Сравнительный анализ использования открытых моделей нейронных сетей в тестировании программного обеспечения

Латанов К.В., СПбГУ, Санкт-Петербург st134091@student.spbu.ru

Аннотация

В статье рассматривается использование открытых больших языковых моделей (LLM) нейронных сетей и оценивается эффективность их применения в тестировании REST API. Решена задача создания программного кода для автоматизированных проверок тестовых сценариев и проведен сравнительный анализ эффективности выполнения поставленной задачи на основании показателей выбранных метрик.

Введение

В процессе разработки ПО обычно выделяют несколько этапов [1], таких как планирование, разработка, тестирование и др. На практике возникает необходимость эффективного выполнения каждого из этих этапов. В данной статье речь пойдет об этапе тестирования, проводящемся при помощи автоматизации, в том числе с использованием средств искусственного интеллекта. Это позволяет уменьшить трудозатраты на выполнение рутинных задач по ручному тестированию программного обеспечения [2]. Ими, например, являются составление чек-листов, их прохождение с целью выявления дефектов объекта, адаптация существующих кейсов и разработка новых в случае изменения функционала.

Постановка задачи

При тестировании REST API возникает задача проверки работоспособности API-методов, поскольку именно они обеспечивают взаимодействие между компонентами системы, а также обмен данными между конечным пользователем и сервером. Проверка работоспособности API-методов начинается с подготовки чек-листа, состоящего из небольших кейсов и основанного на описанных в документации требованиях к рассматриваемому API. Сами кейсы должны быть атомарными и содержать следующие проверки:

- соответствие требованиям документации;

- корректность получаемых данных: код статуса, заголовок, тело ответа и другое;
- граничные значения;
- обработка невалидного ввода;
- отсутствие неожиданных артефактов и ошибок (например, 5xx) при открытии страницы.

Объектом тестирования будет являться публичное API *Petstore*, содержащее запросы разных типов: GET, PUT, POST и DELETE. Генерация автотестов будет осуществляться для раздела API под названием *Pet* [3].

Критериями сравнительного анализа будем считать:

1. Среднее покрытие эндпоинтов раздела Pet – среднее арифметическое отношений количества протестированных статус-кодов одного эндпоинта к количеству его возможных статус-кодов;
2. Общее покрытие статус-кодов раздела Pet – отношение количества всех протестированных статус-кодов раздела Pet к количеству всех возможных (неуникальных) статус-кодов раздела Pet;
3. Общее покрытие эндпоинтов API – отношение полностью протестированных эндпоинтов (у которых были протестированы все заявленные статус-коды) к количеству всех эндпоинтов API;
4. Общее покрытие статус-кодов API – отношение всех протестированных статус-кодов всего API к количеству всех возможных (неуникальных) статус-кодов API.

Цель работы – опираясь на подход, описанный в исследовании [4], оценить и сравнить эффективность применения современных языковых моделей для разработки тестовых сценариев. Будем генерировать тестовые сценарии при помощи больших языковых моделей и оценивать качество результатов. Нейронные сети получат заготовленный промпт с описанием тестовых сценариев, критериями приёмки и аспектами тестирования [5].

Ожидаемым результатом являются скрипты на языке Python, использующие библиотеки pytest и requests и содержащие автоматизированные тесты.

Сценариями тестирования будем считать тесты отдельно взятых модулей (unit-тесты) и глобальные сквозные проверки функционала (end-to-end-тесты). Для первой группы каждый вид рассматриваемого HTTP-запроса подвергнется тестированию основного функционала (прямое использование, по-

зитивные кейсы) и валидации отправляемых данных (неверное использование, негативные кейсы). Во втором случае целью проверок является поиск ошибок при совместном использовании каждой конечной точки в API (endpoint-а), изучение влияния endpoint-ов друг на друга, а также поиск неожиданных артефактов, противоречащих техническому заданию.

Создание автоматизированных тестовых сценариев

Был подготовлен единый промпт следующего вида:
***Дано**:*

1. *API - https://petstore3.swagger.io/*
2. *openapi.json этого API с описанием эндпоинтов и статус-кодов - https://petstore3.swagger.io/api/v3/openapi.json*

***Цель**: Написать автоматизированные проверки на Python с использованием библиотеки pytest (файл main.py), обеспечив максимальное покрытие тестами раздела Pet из рассматриваемого API.*
***Задачи**:*

- *провалидировать эндпоинты по имеющейся документации (получить для каждого все описанные в openapi.json статус-коды)*
- *провести один end-to-end кейс с использованием CRUD-операций раздела Pet (если есть несколько запросов одинакового типа, то последовательно вызвать каждый)*
- *добавить несколько тестов на негативные проверки (невалидные данные в body/header и т.д.)*
- *общее количество проверок должно быть не меньше 10*
- *обеспечить интеграцию создаваемого файла main.py (генерируемый) и готового файла metrics.py (задан ниже) для анализа покрытия API сгенерированными тестами. Результаты метрик должны выводиться после выполнения тестов в консоль в том же порядке, что в документации (сначала раздел Pet, затем все остальные).*

С полным текстом файла metrics.py можно ознакомиться в [5].

Тестировались такие нейросети, как Chat GPT-4o, Deepseek (базовый и R1), Gemini (2.0 Flash и 2.5 Pro), Grok 3 и Qwen2.5-Max. Результат сравнительного анализа работы нейронных сетей с точки зрения приведённых метрик охарактеризован в таблицах 1 и 2.

	Deepseek	Gemini 2.5 Pro	Grok 3
Кол-во итераций до успеха	3	4	3
Кол-во сгенерированных тестов	10	25	24
Процент верно выполненных	90.0%	84.0%	91.7%
Среднее покрытие эндпоинтов Pet	71.9%	75.0%	86.5%
Покрытие статус-кодов Pet	71.4%	71.4%	85.7%
Общее покрытие эндпоинтов API	15.8%	21.1%	26.3%
Общее покрытие статус-кодов API	33.3%	33.3%	40.0%

Таблица 1: Сравнительный анализ результатов работы нейронных сетей (успешное выполнение).

	ChatGPT -4o	Gemini 2.0 Flash	Qwen2.5-Max	DeepThink (R1)
Кол-во итераций до успеха	3	5	5	2
Кол-во сгенерированных тестов	12	13	16	15
Процент верно выполненных	100.0%	46.2%	100.0%	66.7%
Среднее покрытие эндпоинтов Pet	38.1%	36.5%	33.3%	44.8%
Покрытие статус-кодов Pet	–	33.3%	33.3%	42.9%
Общее покрытие эндпоинтов API	0.0%	10.5%	5.3%	10.5%
Общее покрытие статус-кодов API	–	15.6%	15.6%	20.0%

Таблица 2: Сравнительный анализ результатов работы нейронных сетей (выполнено с ошибками).

Итоги работы каждой из нейронных сетей:

- **Grok 3**: Дал хорошее первоначальное покрытие (∼50% для статус-кодов и эндпоинтов) и качественные тесты (∼85%, выполнилось успешно 11/13).

  На второй итерации значительно увеличил количество тестов (почти в

два раза), тем самым повысив покрытие до 86%. Хорошо структурировал тесты по группам и методам – e2e, позитивные, негативные;

- **Gemini 2.5 Pro**: На первой же итерации показал достойные результаты – среднее покрытие 75%, покрытие статус-кодов 71%. За несколько следующих итераций удалось сохранив эти показатели увеличить процент успешных тестов до 84%.

  Код содержит большое количество качественных проверок, а также четко разделен по группам тестов (позитивные, негативные, e2e). Отдельно стоит отметить высокую детализацию сценариев внутри многих тестов;

- **Gemini 2.0 Flash**: Изначально маленькое покрытие (11.5% в среднем при тестировании эндпоинтов) и низкое качество тестов (23.1% успешных). Но после нескольких итераций результат удалось в несколько раз улучшить (до 53% и 34% соответственно);

- **Deepseek**: Плохо начал с точки зрения успешности тестов (40%), но в итоге выполнил поставленную задачу, пройдя минимальный порог по некоторым метрикам.

  На выходе получен структурированный код с короткими комментариями, который четко выполняет заданные инструкции (атомарные проверки, e2e-сценарий, вариативные негативные тесты);

- **DeepThink (R1)**: В начале выдал достаточно низкий процент успешных тестов (40%) и покрытия (30% для эндпоинтов и 28% для статус-кодов). На второй итерации несколько улучшил показатели, однако все следующие попытки значительно ухудшали результат (сократилось число проверок, покрытие по нескольким метрикам стало меньше 10%).

  Код структурирован по виду проверок (позитивные, негативные и e2e), а также снабжен тезисными комментариями;

- **ChatGPT-4o**: Дал неплохое первоначальное покрытие, но неверно выполнил поставленную задачу – проверял входимость полученного статус-кода в список описанных вместо того, чтобы тестировать каждый из них. На следующих итерациях нелегитимно изменил структуру файла, из-за чего часть метрик перестала отображаться;

- **Qwen2.5-Max**: Изначально не справился с поставленной задачей, т.к. на протяжении нескольких итераций не выводил метрики. После явного

исправления этого дефекта метрики отобразились, однако за несколько итераций минимальные пороговые значения так и не были достигнуты (остались в пределах 30%).

Заключение

При том, что некоторые LLM лишь частично реализовали обозначенные требования, каждая модель смогла сгенерировать минимальный набор скриптов с автоматизированными тестами данного API. Полностью с поставленной задачей справился только Grok 3. С незначительной погрешностью, но тоже успешно справились Gemini 2.5 Pro и DeepSeek. Остальные модели не смогли исправить возникшие в ходе запусков ошибки или повысить показатели метрик до минимальных уровней.

Список литературы

[1] Hossain M. I. Software Development Life Cycle (SDLC) Methodologies for Information Systems Project Management // International Journal For Multidisciplinary Research. 2023. №5. https://www.ijfmr.com/papers/2023/5/6223.pdf

[2] Khankhoje R. An intelligent API-testing: unleashing the power of AI // The International Journal of Software Engineering & Applications (IJSEA). - 2024. - №1.

https://aircconline.com/ijsea/V15N1/15124ijsea01.pdf

[3] Swagger Petstore // swagger.io (дата обращения: 28.04.2025).

https://petstore.swagger.io/

[4] Pereira A., Lima B., Faria J.P. APITestGenie: Automated API Test Generation through Generative AI (дата обращения: 09.04.2025). https://arxiv.org/abs/2409.03838

[5] Репозиторий статьи // GitHub (дата обращения: 28.04.2025). https://github.com/kir64/math-mech-2025_publication/tree/main

[6] DeepSeek Chat // deepSeek.com (дата обращения: 28.04.2025).

https://chat.deepseek.com/

[7] ChatGPT chat // chatGPT.com (дата обращения: 28.04.2025).
https://chatgpt.com/?ref=dotcom

[8] DeepSeek-R1-Distill-Qwen-7B // huggingFace.com (дата обращения: 28.04.2025).
https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Qwen-7B

[9] Grok chat // grok.com (дата обращения: 28.04.2025).
https://grok.com/?referrer=website

[10] Gemini Chat // gemini.google.com (дата обращения: 28.04.2025).
https://gemini.google.com/app

[11] Qwen3-235B-A22B Chat // chat.qwen.ai (дата обращения: 28.04.2025).
https://chat.qwen.ai/

Распараллеливание в OpenMP, сплайновые аппроксимации и другие математические задачи

Бурова Ирина Герасимовна
д.ф.-м.н., профессор, профессор кафедры вычислительной математики

Применение сплайнов к численному решению нелинейных интегральных уравнений Вольтерра

Бурова И.Г., СПбГУ, Санкт-Петербург i.g.burova@spbu.ru,
Овинников Н.Д., СПбГУ, Санкт-Петербург st091083@student.spbu.ru,
Алцыбеев Г.О., СПбГУ, Санкт-Петербург gleb.alcybeev@spbu.ru

Аннотация

В работе рассматривается применение сплайновых аппроксимаций второго и пятого порядка для численного моделирования решения нелинейных интегральных уравнений Вольтерра. Приведены численные эксперименты.

Введение

Математические модели традиционно описываются с помощью дифференциальных, интегро-дифференциальных и интегральных уравнений. Известно множество различных численных методов для решения таких задач [1], [2], [3]. Для описания поведения сред разрабатываются математические модели, учитывающие как упругие, так и вязкие компоненты деформации. Такие модели позволяют предсказать отклик материала на различные внешние воздействия. Как было показано в работе [4], свойство вязкоупругости подразумевает, что, начиная с ненулевого момента времени, тело имеет ярко выраженные упругие свойства на коротких интервалах времени и ярко выраженные вязкие свойства на больших интервалах времени. На длительных интервалах времени T вязкоупругие среды можно считать изменяющимися медленно. Основной закон нелинейной деформации часто записывается либо в дифференциальной форме через реологические константы материала, либо в интегральной форме. При изучении среды рассматривают поведение ее математической модели при разных входных данных.

Интегральное уравнение Вольтерра

Рассмотрим интегральное уравнение Вольтерра

$$E\varepsilon\left(x\right)=\sigma\left(x\right)+\int_{0}^{x}K(x,s)f\left(\sigma\left(s\right)\right)ds,\quad x\in\left[0,T\right],\tag{1}$$

здесь ε — деформация, E — мгновенный модуль упругости, $T>0$, а K — ядро Ржаницына, см. [5], имеет следующий вид

$$K\left(x,s\right)=Ae^{-\beta(x-s)}\left(x-s\right)^{\alpha-1},\quad 0<\alpha<1,\quad A>0,\quad \beta>0.\tag{2}$$

В статье [4] предложена численная обработка реологических моделей в контексте нелинейной наследуемой теории ползучести, представлен численный метод для нелинейных слабосингулярных интегральных уравнений Вольтерра с ядром Ржаницына.

Рассмотрим пример модельного нелинейного интегрального уравнения Вольтерра второго рода из работы [4]

$$\begin{gathered}\sigma(x)+\int_0^x \frac{a\sigma(s)+b\sigma^2(s)}{\sqrt{x-s}}ds= \\ =2\left(a+b^2\right)\sqrt{x}-\frac{(a+2b)\pi x}{2}+\frac{4b}{3}x^{3/2},\quad x\in[0,1],\end{gathered} \tag{3}$$

точным решением которого является функция $\sigma(x)=1-\sqrt{x}$. Полагаем $a=0.015$, $b=0.399$.

Воспользуемся сплайнами второго порядка аппроксимации [6] для моделирования решения уравнения (3). Решение будем строить на сетке равноотстоящих узлов $\{x_j\}$, $j=0,1,\dots n$, $n\geqslant 2$, с шагом h. На промежутке $[x_j,x_{j+1}]$ аппроксимируем $\sigma(x)$ выражением

$$\tilde{U}(x)=\sigma(x_j)\,\omega_j(x)+\sigma(x_{j+1})\,\omega_{j+1}(x),\quad x\in[x_j,x_{j+1}], \tag{4}$$

где базисные сплайны $\omega_j(x)$ и $\omega_{j+1}(x)$ могут быть полиномиальными или неполиномиальными базисными функциями.

Базисные функции можно получить следующим образом. Введем непрерывные функции $\varphi_0(x)$ и $\varphi_1(x)$, образующие систему Чебышева. Тогда базисные сплайны $\omega_j(x)$, $\omega_{j+1}(x)$ на промежутке $[x_j,x_{j+1}]$ находим из условий

$$\tilde{U}(x)=\sigma(x),\sigma(x)=\varphi_0(x),\varphi_1(x).$$

В случае $\varphi_0(x)=1$ и $\varphi_1(x)=x$ получаем полиномиальные базисные сплайны

$$\omega_j(x)=\frac{x-x_{j+1}}{x_j-x_{j+1}},\quad x\in[x_j,x_{j+1}], \tag{5}$$

$$\omega_{j+1}(x)=\frac{x-x_j}{x_{j+1}-x_j},\quad x\in[x_j,x_{j+1}]. \tag{6}$$

Выражение (4) используем в интегральной части уравнения (3). В итоге, применяя теорию и практику построения квадратурных формул гауссова типа, получаем систему нелинейных алгебраических уравнений относительно $\sigma(x_j)$, $j=0,1,\dots n$. Решая систему, получаем каркас приближенного решения.

Приведем погрешность решения в случае применения кусочно-линейных сплайнов второго порядка аппроксимации при количестве узлов сетки $n = 50$. На рисунке 1 представлен график погрешностей решения в узлах сетки.

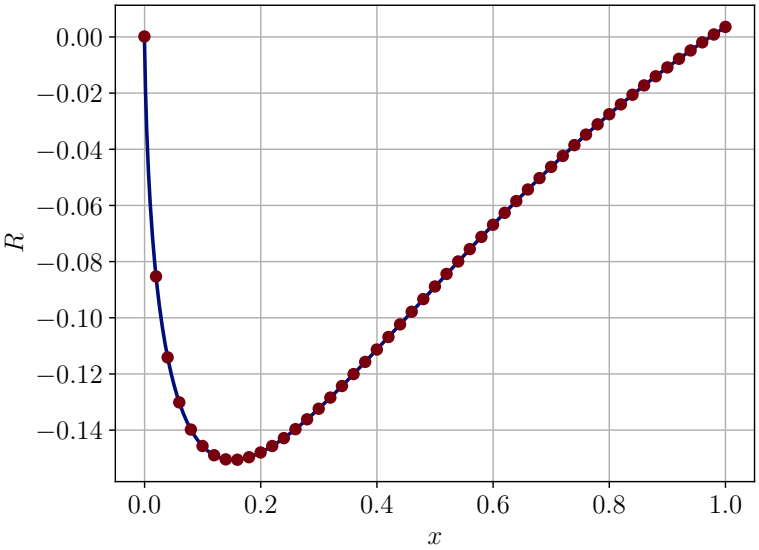


Рис. 1: График погрешностей приближенного решения при использовании кусочно-линейных сплайнов второго порядка аппроксимации, $n = 50$

Аналогично можно построить полиномиальную аппроксимацию пятого порядка. В начале, середине и конце промежутка $[0, 1]$ используем разные аппроксимации пятого порядка. В начале промежутка $[0, 1]$ используем

$$\begin{gathered}\tilde{U}(x) = \sigma(x_j)\,\omega_j(x) + \sigma(x_{j+1})\,\omega_{j+1}(x) + \\ +\sigma(x_{j+2})\,\omega_{j+2}(x) + \sigma(x_{j+3})\,\omega_{j+3}(x) + \sigma(x_{j+4})\,\omega_{j+4}(x), \\ x \in [x_j, x_{j+1}], \quad j = 0, 1, 2.\end{gathered} \tag{7}$$

В середине промежутка $[0, 1]$ используем

$$\begin{gathered}\tilde{U}(x) = \sigma(x_{j-2})\,\omega_{j-2}(x) + \sigma(x_{j-1})\,\omega_{j-1}(x) + \\ +\sigma(x_j)\,\omega_j(x) + \sigma(x_{j+1})\,\omega_{j+1}(x) + \sigma(x_{j+2})\,\omega_{j+2}(x), \\ x \in [x_j, x_{j+1}], \quad j = 3, \ldots, n-2.\end{gathered} \tag{8}$$

В конце промежутка $[0, 1]$ используем

$$\tilde{U}(x) = \sigma(x_{j-3})\,\omega_{j-3}(x) + \sigma(x_{j-2})\,\omega_{j-2}(x) + \sigma(x_{j-1})\,\omega_{j-1}(x) + \sigma(x_j)\,\omega_j(x) + \sigma(x_{j+1})\,\omega_{j+1}(x), \quad x \in [x_j, x_{j+1}], \quad j = n-3, n-2, n-1. \tag{9}$$

Таким образом, получаем базисные сплайны подходящие для аппроксимации в начале интервала интерполяции (правые базисные сплайны), в середине интервала интерполяции (серединные базисные сплайны) и в конце интервала интерполяции (левые базисные сплайны).

Воспользуемся сплайнами пятого порядка аппроксимации для моделирования решения уравнения (3). Решение будем строить на сетке равноотстоящих узлов $\{x_j\}$, $j = 0, 1, \ldots n$, $n \geqslant 6$, с шагом h. На рисунке 2 показаны графики точного и приближенного решений при $n = 7$. Красный график — точное решение, синий график — приближенное решение. На рисунке 3 представлен график погрешностей решения интегрального уравнения при $n = 7$. Таким образом, при большем количестве узлов сетки кусочно-линейные сплайны второго порядка аппроксимации все же дают большую погрешность, чем сплайны пятого порядка аппроксимации.

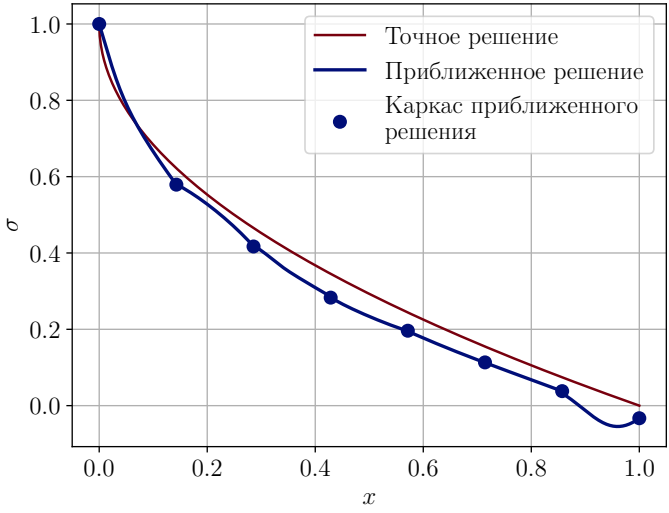


Рис. 2: График точного решения (красный) и приближенного решения при использовании сплайнов второго порядка аппроксимации (синий), $n = 7$

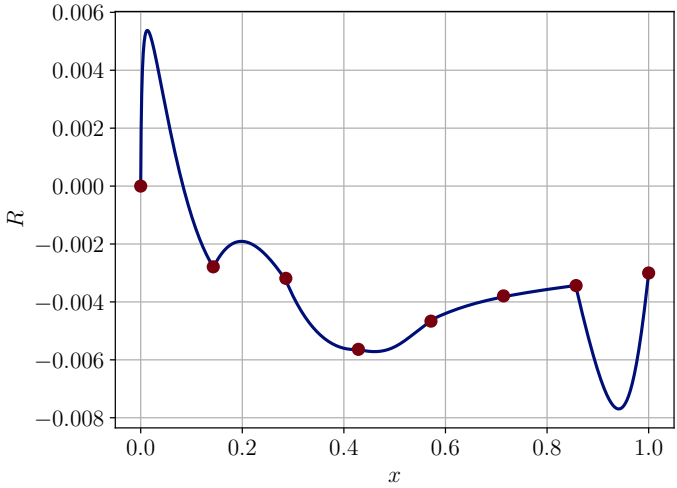


Рис. 3: График погрешностей приближенного решения при использовании сплайнов пятого порядка аппроксимации, $n = 7$

Заключение

Новизна данной работы заключается в том, что для численного решения нелинейного интегрального уравнения используются локальные сплайновые аппроксимации. Можно показать, что методы, основанные на использовании сплайновых аппроксимациях второго и пятого порядка, как правило, дают меньшую погрешность, чем использование других методов решения интегральных уравнений.

Список литературы

[1] Михлин С. Г. Приложения интегральных уравнений к некоторым проблемам механики, математической физики и техники. М.: ОГИЗ, 1947. 304 с.

[2] Михлин С. Г. Вариационные методы в математической физике. М.: Физматгиз, 1970. 512 с.

[3] Михлин С. Г., Морозов Н. Ф., Паукшто М. В. Интегральные уравнения в теории упругости. СПб.: Издательство Санкт-Петербургского государственного университета, 1994. 271 с.

[4] Тында А. Н., Романов А. Е. Численное решение нелинейных интегральных уравнений Вольтерра с дробно-экспоненциальными ядрами реологических моделей вязкоупругой среды // ИЗВЕСТИЯ Иркутского государственного университета. Серия «Математика». — 2012. — Т. 2. № 2. — С. 7–12.

[5] Ржаницын А. Р. Некоторые вопросы механики систем, деформирующихся во времени. М.: Гостехиздат, 1949.

[6] Михлин С. Г. Вариационно-сеточная аппроксимация // Записки научных семинаров ЛОМИ АН СССР. 1974. Т. 48. С. 32–188.

Извлечение признаков из магнитно-резонансной томографии с использованием атласа мозга для задач классификации

Неверов В.А., СПбГУ, Санкт-Петербург vasenkaneverov@mail.ru,
Липкович М.М., ИПМаш РАН, Санкт-Петербург lipkovich.michail@gmail.com

Аннотация

Современные исследования в области нейронаук требуют глубокого понимания функциональных связей между регионами мозга, что важно для диагностики и лечения заболеваний. Функциональная магнитно-резонансная томография является одним из инструментов для анализа этих связей. В данной работе проведено исследование с использованием методов машинного обучения для анализа связей между различными регионами.

Введение

Существующие подходы исследований в области нейронаук, такие как корреляционный анализ и трактография [1], обеспечивают определенные результаты в понимании функциональных и структурных связей между регионами мозга, однако остаются нерешенные вопросы оптимизации анализа и интерпретации данных. В данной работе рассматривается построение графа мозговых связей на основе функциональной магнитно-резонансной томографии (фМРТ) с использованием атласа Mutli-Modal Brain Atlas (MSDL) [2].

Основное направление исследования заключается в извлечении значимых признаков фМРТ, что позволит более точно оценить взаимодействия между различными регионами мозга [3]. В ходе работы была проведена интеграции методов машинного обучения для анализа данных, что может углубить понимание функциональной архитектуры мозга [4].

Задача состоит в применении фМРТ для анализа связей в мозге с использованием атласа, обучении модели для классификации данных испытуемых на предмет различий между здоровыми и больными пациентами. Цель работы заключается в построении графа мозговых связей, что позволит оценить взаимодействия между различными регионами мозга и выявить их особенности.

Ход работы

В ходе данного исследовательского проекта была проведена комплексная работа, направленная на анализ и обработку функциональных магнитно-резонансных томографических (фМРТ) данных с применением современных методов нейровизуализации и машинного обучения. Для реализации проекта использовался язык Python, а также ряд библиотек, включая nilearn для работы с фМРТ данными [6], numpy и pandas для манипуляций с массивами данных, networkx для построения графов и scikit-learn для задач машинного обучения.

Для анализа была загружена карта мозговых регионов из атласа MSDL с помощью функции fetch_atlas_msdl() [2]. Атлас содержит информацию о местоположении различных функциональных областей мозга, что важно для последующего анализа. На основе карт атласа была создана маска NiftiMapsMasker, позволяющая извлекать временные ряды данных фМРТ из заданных мозговых областей, стандартизируя результаты и обеспечивая необходимую нормализацию данных.

Основным компонентом разработки являлась функция extract_features_from_subject, обеспечивающая извлечение характерных признаков из данных фМРТ для каждого испытуемого. Процесс начинается с загрузки временных рядов, после чего строится матрица корреляции, описывающая взаимодействия между различными регионами мозга. Используемая метрика корреляции позволила учитывать нормированные значения и исключить влияние выбросов.

Для получения информации о каждом участнике исследования использовались данные из файла participants.tsv, который насчитывает 155 испытуемых (89 здоровых пациентов и 66 больных), загруженного с помощью библиотеки pandas [6]. Каждый участник был классифицирован как здоровый или пациент на основе пометок из набора данных, что было важно для дальнейшего анализа.

На следующем этапе был осуществлен сбор всех извлеченных признаков и соответствующих меток в массивы X и y. Признаки, полученные из графового анализа, были объединены в одном массиве, что позволяет эффективно использовать их для обучения модели. Данные были сохранены с помощью библиотеки pickle для обеспечения быстрого доступа к собранной информации.

Признаки, собранные на этапе графового анализа, представляют собой количественные характеристики, которые описывают взаимодействия между различными регионами мозга. В процессе извлечения признаков из фМРТ данных для каждого испытуемого было рассчитано несколько ключевых мет-

рик, отражающих структурные и функциональные свойства когнитивных процессов. Полученные признаки включали:

- **Степень центральности** (degree centrality): показывает количество связей, имеющихся у каждого узла (региона мозга), что позволяет выявить наиболее активные области.
- **Промежуточная центральность** (betweenness centrality): измеряет значимость узла в качестве "моста" между другими узлами, что помогает выявлять ключевые регионы, влияющие на передачу информации между различными частями мозга.
- **Собственная центральность** (eigenvector centrality): определяет важность узла в сети, принимая во внимание не только количество соединений, но и значимость соседних узлов.
- **Коэффициент кластеризации** (clustering coefficient): характеризует степень, с которой узлы группируются в плотные кластеры, что может указывать на функциональные ансамбли, работающие совместно.

Эти измерения предоставляют глубокое понимание функционирования мозга и позволяют выявить паттерны, характерные для различных когнитивных состояний.

Разделение данных на обучающую и тестовую выборки: Для разделения данных мы использовали метод, который случайным образом делит весь набор данных на две части: обучающую выборку и тестовую выборку. В нашем случае размер обучающей выборки составил 124 испытуемых, что составляет примерно 80% от общего объема данных, в то время как тестовая выборка включает 31 испытуемого, что составляет около 20%. Такое соотношение позволяет обеспечить достаточное количество данных для обучения модели, сохраняя при этом достаточно данных для проверки ее производительности. Важно отметить, что при разделении данных мы учитывали случайное распределение, чтобы избежать смещения в выборках. Это означает, что каждая запись в наборе данных имела равные шансы попасть как в обучающую, так и в тестовую выборку.

Для предсказательного анализа была выбрана модель Random Forest, обеспечивающая хорошую производительность даже при наличии большого количества признаков. Мы использовали GridSearch чтобы найти оптимальные параметры для RandomForest (например, сколько деревьев использовать и какой порог для классификации лучше). Это помогает сделать модель точнее. Обучили модель на обучающей выборке с использованием кроссвалидации(5 фолдов). Разбили обучающие данные на 5 частей (фолдов), поочередно обучали модель на 4 частях и проверяли на 1, чтобы убедиться, что

модель хорошо работает на новых данных и не переобучается. Для каждого фолда мы перебирали разные пороги классификации, вычисляли метрики (точность, F1-скор, ROC-AUC) и выбирали порог, при котором модель лучше всего предсказывает данные. Финальным этапом работы стало использование обученной модели для проведения предсказаний на тестовых данных, что позволило оценить точность модели и выявить возможные области для улучшения.

Результаты работы

В результате анализа фМРТ, модель продемонстрировала точность 83.12% на тестовой выборке, что подтверждает её способность различать классы участников. Значения метрик указаны на рисунке 1.

```
Final evaluation on test set:
Best threshold: 0.30
              precision    recall  f1-score   support

           0       0.88      0.78      0.82        18
           1       0.73      0.85      0.79        13

    accuracy                           0.81        31
   macro avg       0.80      0.81      0.80        31
weighted avg       0.82      0.81      0.81        31

ROC-AUC: 0.8312
```

Рис. 1: Значения метрик классификации

В ходе работы была выполнена визуализация графов (рисунок 2) и корреляционных матриц.

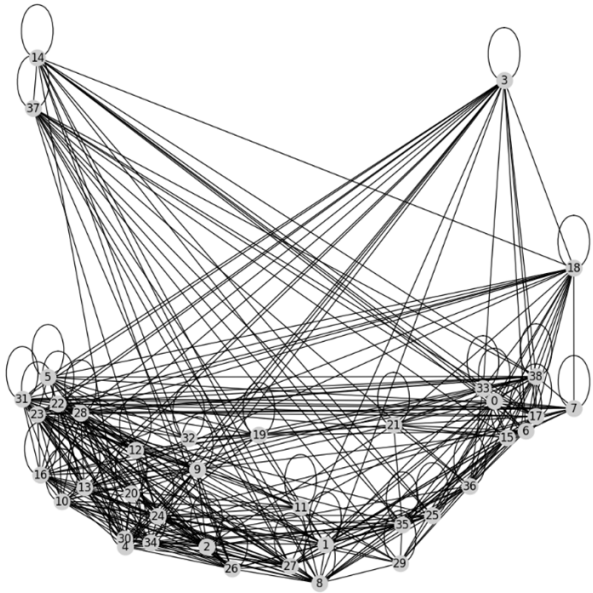


Рис. 2: Визуализация графа, где 1–38 — регионы мозга

Граф строится, чтобы визуализировать и анализировать связи между регионами мозга как сеть, где узлы — это регионы, а ребра — их взаимодействия [3].

Заключение

В ходе проведенного исследования с использованием фМРТ и атласа MSDL удалось достичь результатов в анализе связей между различными регионами мозга. Построение графа мозговых связей, основанного на данных, позволило выявить ключевые взаимодействия и особенности функциональной архитектуры мозга. Полученная модель показала точность 83.12%. Дальнейший анализ с использованием методов машинного обучения нацелен на предсказание разнообразных состояний пациентов и может стать инструментом для улучшения диагностики и лечения [7].

Список литературы

[1] Doyen S. What is tractography? [Электронный ресурс]. — 2022. URL: https://www.o8t.com/blog/tractography

[2] Sughrue M. What are the brain maps? [Электронный ресурс]. — 2022. URL: https://www.o8t.com/blog/brain-maps

[3] Sughrue M. What are brain networks? [Электронный ресурс]. — 2022. URL: https://www.o8t.com/blog/brain-networks

[4] Doyen S. How we build personalized brain maps [Электронный ресурс]. — 2022. URL: https://www.o8t.com/blog/structural-connectivity-atlas

[5] Sughrue M. How Personalized Mapping of Brain Networks aids Neurosurgery [Электронный ресурс]. — 2022. URL: https://www.o8t.com/blog/how-personalized-mapping-of-brain-networks-aids-neurosurgery

[6] OpenNeuro Dataset [Электронный ресурс]. URL: https://openneuro.org/datasets/ds005366/versions/1.2.0

[7] Sughrue M. The trouble with calling brain regions 'eloquent' [Электронный ресурс]. — 2023. URL: https://www.o8t.com/blog/the-trouble-with-calling-brain-regions-eloquent

Построение квадратурных формул наивысшей степени точности обращения преобразования Лапласа

Пийтер Д.С., СПбГУ, Санкт-Петербург piyterds@gmail.com,
Рябов В.М., СПбГУ, Санкт-Петербург victor.ryabov@mail.ru

Аннотация

Преобразование Лапласа широко используется в различных сферах науки и техники. Возникает задача его обращения, то есть задача восстановления функции-оригинала по известной функции-образу. Возможные сферы применения: механика и робототехника, цифровая обработка сигналов, анализ экономических процессов. Общего универсального способа обращения не существует. Существующие подходы к решению проблемы имеют свои достоинства и недостатки. Так аналитический подход во многом основывается на таблицах обращения и малоприменимы к реальным задачам. Численные методы нередко страдают от высокой трудоемкости или недостаточной точности. В данной работе рассматривается обращение преобразования Лапласа с помощью построения квадратурных формул наивысшей степени точности для интеграла Римана-Меллина. В результате работы была успешно создана программа для построения КФНСТ размером до 400 узлов, которая позволяет с различной точностью и быстродействием обращать преобразование Лапласа. Перспективы исследования заключаются в оптимизации работы ПО и его адаптации для более широкого круга пользователей.

Введение

Преобразование Лапласа - известный с 18-го века математический инструмент, который широко используется для решения дифференциальных уравнений и анализа линейных систем [1].

В прикладных задачах появляется необходимость обращения преобразования Лапласа, то есть восстановления функции-оригинала $f(t)$ при известном образе $F(p)$. Общего универсального способа обращения не существует. Существующие подходы к решению проблемы имеют свои достоинства и недостатки.

Стоит отметить, что задача обращения преобразования Лапласа состоит в решении интегрального уравнения первого рода, то есть является некорректной. Это отражается в следующих свойствах:

1. **Особенности функции-оригинала:** В случаях присутствия в функции-оригинале разрывов первого рода данный метод обращения, ввиду нахождения решения, как линейной комбинации образа и его производных, являющимися бесконечно гладкими функциями, в точках разрыва будет возвращать только полусумму левого и правого пределов. Для нахождения точек разрыва и величин скачка оригинала можно воспользоваться методом, описанным в [2].

2. **Теорема о единственности:** Согласно теореме, описанной в [3], если $f(t)$ является кусочно-гладкой и удовлетворяет условиям Больцано-Вейерштрасса, то можно гарантировать единственность решения.

Обзор существующих методов обращения и их особенностей

Существует множество различных методов обращения преобразования Лапласа, как аналитических, так и приближённых.

Аналитические методы, такие как таблицы обращения, теоремы разложения и др. ([4]), хоть и по сути точны, в большинстве своём неприменимы к использованию в практических задачах.

Приближенные методы:

- Приближение с помощью рядов Лагерра - как описано в [5] сходимость метода медленная.

- Дельта-метод обращения преобразования Лапласа высокого порядка точности - коэффициенты быстро растут с увеличением степени точности ([6]).

- ИКФ с равноотстоящими узлами - требует меньше вычислений, но недостаточно точна ([5]).

Поскольку все вычислительные методы обращения преобразования Лапласа либо трудозатратны, либо недостаточно точны, было принято решение использовать квадратурные формулы наивысшей степени точности для приближения интеграла Римана-Меллина. Скорость их сходимости описана в [7]

и составляет

$$O\left(n^{1-s}\left(\frac{3.764}{n^2}\right)^n\right),$$

а устойчивость КФНСТ по отношению к ошибкам в функции $\varphi_s(p) = p^s F(p)$ определяется величиной

$$M_n = \sum_{k=1}^{n} |A_k| = O(n^{1-s} 3.764^n).$$

Соответственно, необходимо будет провести проверку полученного набора коэффициентов.

Математическое основание

Дано уравнение

$$\int_0^\infty e^{-pt} f(t)dt = F(p), \tag{1}$$

где $f(t)$ – искомая функция, $F(p)$ – её образ по Лапласу. Изображение $F(p)$, как известно, регулярно при $\mathrm{Re}\ p > \sigma$, при некотором σ. Не умаляя общности, можем предположить, что $\sigma = 0$ ($F^*(p) = F(p-\sigma)$, $f^*(t) = e^{\sigma t} f(t)$). Также известно, что $F(p) \to 0$ при $\mathrm{Re}\ p \to \infty$. Предположим, что $F(p)$ стремится к нулю, как некоторая степень $1/p$, то есть образ представим в виде $F(p) = \frac{1}{p^s}\varphi_s(p)$, где $s > 0$, а $\varphi_s(\infty)$ существует и конечна.

Обращение преобразования Лапласа производится по формуле Римана-Меллина

$$f(t) = \frac{1}{2\pi i} \int_{c-i\infty}^{c+i\infty} e^{pt} F(p)dp, \quad c > 0, \quad t > 0. \tag{2}$$

Перепишем её для нашей $F(p)$:

$$f(t) = \frac{1}{2\pi i} \int_{c-i\infty}^{c+i\infty} e^{pt} p^{-s} \varphi_s(p)dp = t^{s-1} \frac{1}{2\pi i} \int_{c-i\infty}^{c+i\infty} e^{p} p^{-s} \varphi_s(p/t)dp, \quad c > 0. \tag{3}$$

Для приближенного вычисления интеграла (3) построим квадратурную формулу

$$\frac{1}{2\pi i} \int_{c-i\infty}^{c+i\infty} e^{p} p^{-s} \varphi(p)dp \approx \sum_{k=1}^{n} A_{kn} \varphi\left(p_{kn}\right). \tag{4}$$

Теорема 1 ([5]). Для того, чтобы квадратурная формула (4) была наивысшей степени точности, необходимо и достаточно, чтобы:

1. Формула (4) была интерполяционной. То есть для попарно различных p_{kn} выполнялось

$$\sum_{k=1}^{n} A_{kn} p_{kn}^{-m} = \frac{1}{\Gamma(s+m)}, \quad m = 0, 1, \ldots, n-1.$$

2. Выполнялись условия ортогональности

$$\int_{c-i\infty}^{c+i\infty} e^p p^{-s} \omega_n\left(\frac{1}{p}\right) p^{-m} dp = 0, \quad m = 0, 1, \ldots, n-1, \tag{5}$$

$$\omega_n\left(\frac{1}{p}\right) = \prod_{k=1}^{n} \left(\frac{1}{p} - \frac{1}{p_{kn}}\right). \tag{6}$$

В итоге получаем, что для существования квадратурной формулы наивысшей степени точности для приближения интеграла (3), необходимо и достаточно существования многочленов (6), для которых выполняются условия (5).

Теорема 2 ([8]). Многочлены (6) удовлетворяют уравнению

$$\begin{gathered} B(x)\omega_n''(x) + [A(x) + B'(x)]\,\omega_n'(x) - n(n+s-1)\omega_n(x) = 0, \\ B(x) = x^2, \quad A(x) = (s-2)x - 1. \end{gathered} \tag{7}$$

Многочлены $\omega_n(p)$ с точностью до постоянного сомножителя совпадают с многочленами

$$P_n^s(x) = \sum_{k=0}^{n} (-1)^{n-k} \binom{n}{k} (n+s-1)_k x^k,$$

где $(a)_k$ - символ Похгаммера:

$$(a)_k = \begin{cases} 1, & k = 0; \\ a(a+1)\cdots(a+k-1), & k \geqslant 1, \end{cases}$$

и их корни попарно различны и имеют положительную действительную часть.

Исходя из всего вышеизложенного, можем сделать вывод, что искомая КФНСТ существует, единственна и имеет вид

$$f(t) \approx t^{s-1} \sum_{k=1}^{n} A_{kn} \varphi_s\left(p_{kn}/t\right), \tag{8}$$

где узлы p_{kn} – корни уравнения $P_n^s(1/p_{kn}) = 0$, попарно различны и лежат в правой полуплоскости $\mathrm{Re}\, p > 0$, а коэффициенты A_{kn} вычисляются [9] по формуле:

$$A_{kn} = \frac{(-1)^{n+1} n!(2n+s-2)^2}{n^2 \Gamma(n+s-1) p_{kn}^2 \left(P_{n-1}^s(1/p_{kn})\right)^2}.$$

При увеличении количества узлов программы, использующие стандартные встроенные математические пакеты нахождения корней многочленов, теряют точность и увеличивают время своей работы. Для избежания данной проблемы можно прибегнуть к использованию метода Ньютона.

Как показано в работах [10], узлы КФНСТ лежат в правой половине кольца

$$n + s - 1 \le |p_{kn}| < 2n + s - \frac{2}{3}, \quad k = 1, 2, \ldots, n.$$

Положим

$$z_{kn} = -(2n + s - 2)/p_{kn}, \quad k = 1, 2, \ldots, n.$$

Нормированные корни z_{kn} при $n \to \infty$ стремятся к точкам кривой

$$\gamma = \{z : |\Omega(z)| = 1, \quad \mathrm{Re}\, z < 0\},$$

$$\text{где } \Omega(z) = \exp\left(\sqrt{1 + z^{-2}}\right) / \left[z\left(1 + \sqrt{1 + z^{-2}}\right)\right].$$

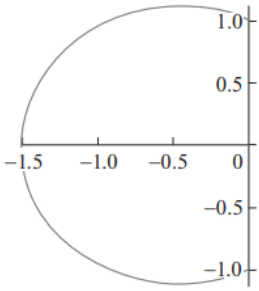


Рис. 1: Кривая $\gamma(z)$

Также было показано, что аргументы корней многочлена $P_n^s(-x)$ распределяются равномерно в интервале $(\pi/2, 3\pi/2)$.

Используя информацию о примерном расположении корней многочлена, воспользуемся методом Ньютона для их поиска.

Программная реализация

Для вычисления узлов КФНСТ и соответствующих коэффициентов, исходя из рассмотренной ранее теории, требуется найти корни многочлена вида

$$P_n^s(x) = \sum_{k=0}^{n} (-1)^{n-k} \binom{n}{k} (n+s-1)_k x^k,$$

где $(a)_k$ - символ Похгаммера.

Реализация вычислений будет происходить с помощью языка программирования Python, поскольку данный язык является пригодным для тонкой настройки параметров и удобным в использовании.

Вычисления требуют достаточно высокой точности, поэтому для хранения чисел и выполнения действий над ними задействуем библиотеку для работы с числами с большим количеством знаков после запятой - ”mpmath”. Также нам понадобится библиотека для реализации метода вычисления количества сочетаний и метода вычисления гамма-функций и бета-функций - ”Scipy”. Для визуализации графиков задействуем библиотеку ”matplotlib.pyplot”. Сохранение результатов реализуем с помощью ”pickle”.

Результаты

В результате работы программы, с кодом которой можно ознакомиться в открытом репозитории [11], были получены наборы узлов и коэффициентов для КНФСТ размером от 2 до 400 узлов (при s=1). Это позволяет обращать преобразование Лапласа с разной точностью. Рассмотрим полученные результаты на конкретных примерах.

Рассмотрим графики обращений уже известных функций. На рисунках 2, 3 представлены графики синуса и функции Бесселя, вычисленные с помощью обращения преобразования Лапласа и вычисленные точно, а также их разность. Отклонение появляется на значениях выше некоторого значения, которое можно увеличить за счёт повышения количества узлов, однако время вычисления также повысится (время работы программы по вычислению 1000 точек обращения преобразования Лапласа в зависимости от количества узлов указано на графике 4).

На рисунках используется цветовое обозначение: оранжевый - точно вычисленная функция, синий - функция полученная в результате обращения, зеленый - их разность.

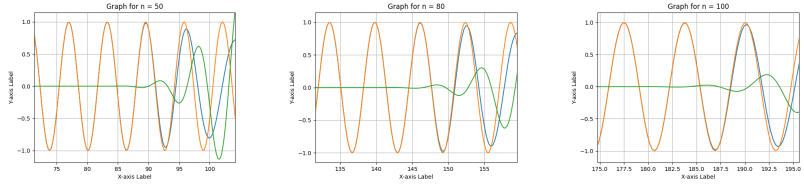


Рис. 2: Синус при n=50, n=80, n=100

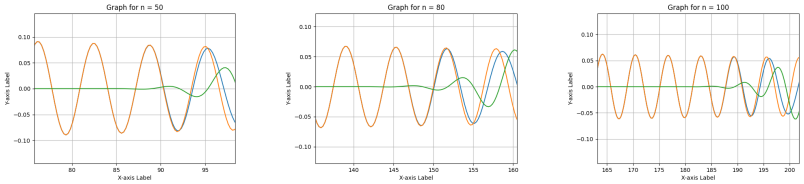


Рис. 3: Функция Бесселя при n=50, n=80, n=100

Выводы

На основе полученных результатов можно сделать следующие выводы:

1. **Результат реализации алгоритма:** требуемые узлы и коэффициенты КФНСТ могут быть достаточно быстро вычислены при использовании метода Ньютона и информации об их примерном местонахождении на комплексной плоскости.

2. **Применимость и универсальность:** КФНСТ способна обращать преобразование Лапласа с разнообразной точностью, что позволяет найти баланс между точностью и быстродействием.

3. **Практическое значение:** преобразование Лапласа имеет широкое распространение, а простой и универсальный инструмент для его численного обращения позволяет специалистам из разных областей науки и техники решать требуемые задачи.

4. **Перспективы:** дальнейшее развитие может быть направлено на оптимизацию работы программы в силу повторяемой природы вычислений. Также возможна реализация на более мощном оборудовании, что позволит увеличить точность за счет грубого увеличения скорости обработки данных.

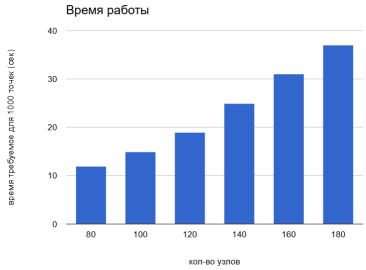


Рис. 4: Время вычисления значений 1000 точек при разном количестве узлов КФНСТ в секундах.

Итогом работы стало создание ПО и подтверждение применимости теоретических данных, описанных в [10], для решения практических задач.

Заключение

В результате данной работы был реализован алгоритм построения КФНСТ для численного обращения преобразования Лапласа. Данный метод является достаточно универсальным и удобным способом обращения, что позволяет использовать его при решении задач, требующих высокой точности.

Созданное в процессе работы ПО может быть использовано в задачах обращения преобразования Лапласа, возникающих у специалистов различных сфер науки и техники.

В дальнейшем созданная программа может быть улучшена. Для повышения точности вычислений и общего быстродействия системы могут быть использованы распределённые вычисления и более оптимальные алгоритмы вычислений.

Список литературы

[1] Erwin Kreyszig, «Advanced Engineering Mathematics», 10th ed. (New York: John Wiley & Sons, 2020). С. 203-253

[2] Лебедева, А. В. «Определение точек разрыва и величины скачка оригинала по его изображению по Лапласу» / А. В. Лебедева, В. М. Рябов // Вест-

ник Санкт-Петербургского университета. Математика. Механика. Астрономия. – 2024. – Т. 11, № 2. – С. 316-323.

[3] Арнольд В. И. «Обыкновенные дифференциальные уравнения» / В. И. Арнольд. — М.: МЦНМО, 2014. — 342 с.

[4] Dwight H. B. «Tables of Laplace Transforms.» — New York: Dover Publications, 1974. — 360 p

[5] Крылов В. И., Скобля Н. С. Методы приближенного преобразования Фурье и обращения преобразования Лапласа: справочная книга. — М. : Мир, 1968. — 224 с.

[6] Рябов В. М. «О точности некоторых методов обращения преобразования Лапласа» // Методы вычислений. Вып. 14. Л. 1985. С. 59–71.

[7] Лебедева А.В., Рябов В.М. «Характеристики сходимости и устойчивости некоторых методов обращения преобразования Лапласа» // Вестник Санкт-Петербургского университета. Математика. Механика. Астрономия. 2024. Т. 11, № 1. С. 115–130

[8] Суетин П.К. «Классические ортогональные многочлены.» М., 1979.

[9] Luke Y. «The special functions and their approximations.» V. 2. New York, 1969.

[10] Bruin M.G., Saff E.B., Varga R.S. «On the zeros of generalised Bessel polynomials.» I, II // Indagat. math. 1981.

[11] Piyter, D. (2025). Construction of quadrature formulas of the highest degree of accuracy for the inversion of the Laplace transform. Zenodo. https://doi.org/10.5281/zenodo.15276516

Применение диффузионной магнитно-резонансной томографии для определения нарушений нейропластичности головного мозга

Прокопьев В.А., СПбГУ, Санкт-Петербург vitaprok896@gmail.com,
Липкович М.М., ИПМаш РАН, Санкт-Петербург lipkovich.mikhail@gmail.com

Аннотация

Разработка более точных и надежных методов для определения регионов коры головного мозга является ключевым шагом в углублении понимания организации мозга. Результаты применения таких методов анализа, как использование теории графов и выполнение диффузионно-тензорной визуализации, и машинного обучения позволяет достичь высокого качества классификации состояния пациентов.

Введение

Поверхность мозга состоит в основном из серого вещества (кортекс). Белое вещество, обеспечивающее взаимодействие частей мозга, в основном находится на глубинных слоях. На основе белого вещества строится карта связей, а сам процесс называется трактографией [1]. Кортекс можно разбить на регионы и получить атлас мозга [2], при этом мозг разбивается на парцели, а сам процесс разбиения называют парцелляцией. Если построить трактограмму и наложить атлас мозга, то можно выяснить, каким образом отделы мозга взаимодействуют между собой [3]. Примеры результатов выполнения процессов представлены на рисунке 1.

Традиционные методы парцелляции обладают ограничениями, существенно снижающими надёжность и интерпретируемость данных. На текущий момент существует несколько эффективных решений. В работе [4] описан следующий алгоритм, использующий диффузионно-тензорную визуализацию (ДТВ): применить методы трактографии к мозгу больного, далее, при помощи алгоритмов машинного обучения, определить, к каким регионам относятся начальные и конечные узлы белого вещества. Существует альтернативный подход, описанный в статье [5] и основанный на теории графов. На построенном при помощи матрицы смежности графе взаимодействия регионов через понятие центральности выделяются важные участки. На рисунках 2 и 3 показаны алгоритмы работ [4] и [5] соответственно.

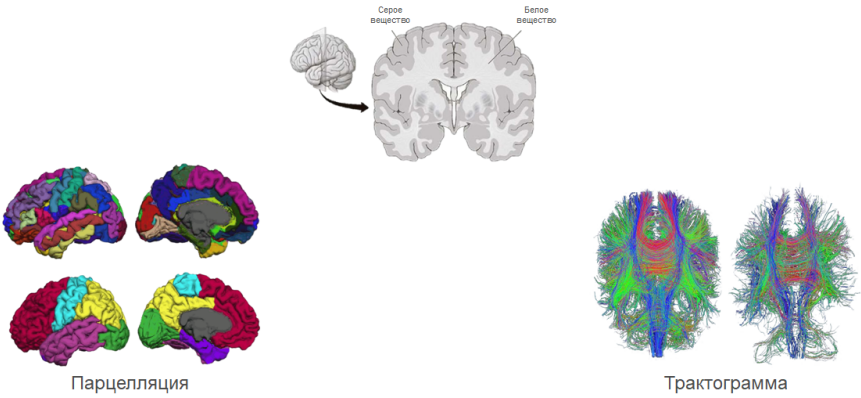


Рис. 1: Пример отображения вещества в головном мозге и выполнения на его основе парцелляции и трактографии

Предлагаемый доклад посвящён применению результатов диффузионной магнитно-резонансной томографии (МРТ) в методах трактографии для определения регионов коры, к которым относятся узлы трактов белого вещества, и классификации состояния пациентов на основе данных МРТ головного мозга. Исследование позволит проверить применяемость существующих методов для разработки более точных и индивидуальных парцелляций, учитывающих как структурную, так и функциональную организацию мозга.

Используемые данные

Для проведения эксперимента был задействован набор данных [6], в котором представлены данные 25 пациентов: 16 здоровых пациентов и 9 пациентов с нарушениями нейропластичности. Были проведены две сессии оценки качества работы мозга: перед началом тренировки и после определённого периода обучения. На основании результатов обучения было принято решение о причислении к конкретной группе пациентов.

Набор данных хранит в себе файл с основными данными в виде 4D-массива, описывающего значения по координатам X, Y, Z в различные периоды мозговой активности; файл со значениями интенсивности градиента (bvals) для каждого направления диффузии; файл с направлениями диффузии (bvecs) для каждого измерения; файл с маской мозга, который определяет область интереса для анализа; файл с метаданными для маски мозга.

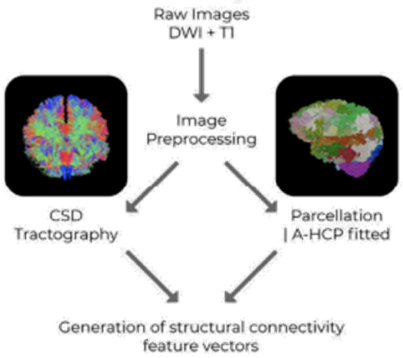


Рис. 2: Алгоритм на основе ДТВ

Реализуемые подходы

Анализ при помощи диффузионно-тензорной визуализации

При помощи bvals и bvecs описывается схема градиентов магнитного поля, которые применялись во время сканирования. Для построения трактографии мозга необходимо создать маску белого вещества, которая будет отображать взаимодействие областей интереса между собой. На основе полученной функции распределения ориентации можно выявить основные направления распространения воды и построить потоки перемещения воды. Для выявления точек, в которых будет проверяться наличие белого вещества и его направление, производится настройка алгоритма диффузионно-тензорной визуализации при помощи схемы градиентов. На основе маски и аффинного преобразования, полученного из данных МРТ, происходит отбор точек. После этого алгоритм выявляет потоки белого вещества.

Для выявления структурной взаимосвязи регионов используется анатомический атлас, предварительно адаптированный под данные. Происходит нанесение потоков белого вещества на применяемый анатомический атлас. Построив матрицу связности, легко визуализировать взаимодействие областей между собой.

Обучение классификации происходит на основе признаков матрицы связности.

Схематическое отображение процесса анализа проведения ДТВ представлено на рисунке 4.

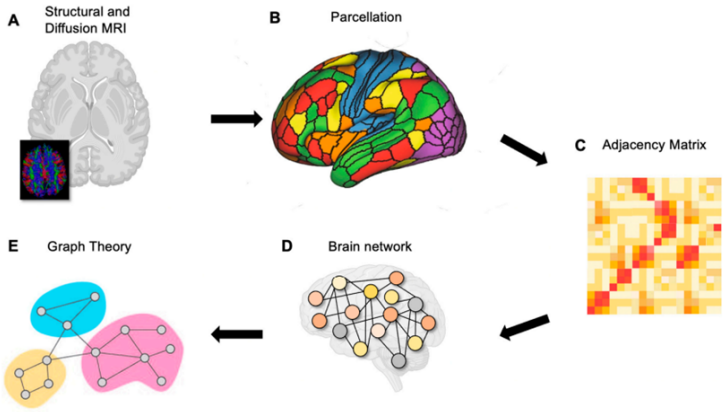


Рис. 3: Алгоритм на основе теории графов

Анализ при помощи теории графов

Для определения связности регионов воспользуемся понятием центральности по посредничеству, мерой, которая оценивает, насколько часто узел лежит на кратчайших путях между всеми парами других узлов в графе. Узел с высокой степенью центральности по посредничеству имеет важное значение для связности всей сети. В случае МРТ головного мозга регион будет активно включён в передвижение белого вещества.

На основе построенной матрицы связности определяется центральность регионов и их связанность между собой. Обучение классификации происходит на основе определяемых параметров теории графов.

Выполнение классификации

В качестве рассматриваемых методов классификации были выбраны такие алгоритмы, как случайный выбор, наивный Байес, случайный лес и метод опорных векторов.

Набор данных был разделён на обучающую и тестовую выборку в соотношении 7:3. Обучение классификации происходит на основе группы пациента.

Для улучшения качества был проведён подбор гиперпараметров модели методом кросс-валидации. Данные разбивались на фолды, содержащие в себе различные поднаборы исходного датасета.

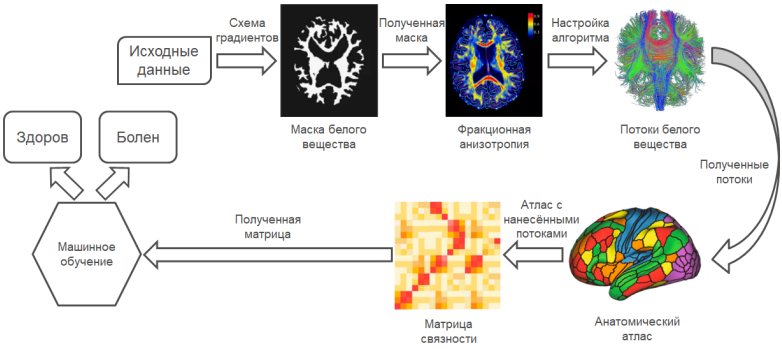


Рис. 4: Реализация подхода ДТВ

Также стоит отметить, что проводится подбор порогового значения вероятности для определения принадлежности к классу «Болен».

Результаты

Для оценивания работы моделей использовалась F1-мера. Высокая F1-мера означает, что модель чаще всего правильно указывает на больного, причём очень редко пропускает тех, кому нужна помощь. Математическое ожидание говорит об общей возможности модели решать поставленную задачу, а дисперсия – о стабильности работы.

Результаты обучения моделей представлены в таблице 1. M(F1) – математическое ожидание по F1, D(F1) – дисперсия по F1.

Название модели	**Вид данных**	**M(F1)**	**D(F1)**
Случайный выбор	-	0.7671	0.0162
Наивный Байес	Граф с нормировкой	0.7378	0.0249
Случайный лес	Граф без нормировки	0.8059	0.0206
Метод опорных векторов	Матрица с нормировкой	0.9008	0.0112

Таблица 1: Показатели моделей

Лучшим результатом стало значение метрики F1 в 0.90 для метода опорных векторов, обученного на признаках нормированной матрицы связности.

Масштабирование

Текущая реализация имеет огромный простор для улучшений. Приведём некоторые возможные улучшения:

1. Классификация

   Текущая реализация рассматривает возможности системы выявить одну конкретную болезнь. Следует рассмотреть возможность выявлять несколько болезней.

2. Обработка данных

   К исходным данным можно применить операцию регистрации, при которой данные из одного пространства измерений переносятся в другое пространство. Это может подстроить процесс анализа под необходимый вариант интерпретации. Также это может повлиять на качество работы алгоритмов определения группы пациентов.

3. Оптимизация ресурсов

   Текущая реализация использует последовательное исследование данных пациентов. Возможно настроить параллельную обработку и тем самым ускорить процесс анализа данных.

Заключение

В докладе приведены основные подходы к анализу МРТ головного мозга. Рассмотренные методы показали свою эффективность в создании точных и надежных парцелляций, учитывающих функциональную значимость мозговых областей. Разработанная система демонстрирует возможность автоматизировать процесс определения регионов коры головного мозга, опираясь на данные о структуре белого вещества и связности между различными областями.

Список литературы

[1] Omniscient Neurotechnology. — What is tractography? https://www.o8t.com/blog/tractography.

[2] Omniscient Neurotechnology. — What are brain maps? https://www.o8t.com/blog/brain-maps.

[3] Omniscient Neurotechnology. — What are brain networks? https://www.o8t.com/blog/brain-networks.

[4] Doyen, S. Connectivity-based parcellation of normal and anatomically distorted human cerebral cortex / S. Doyen, P. Nicholas, A. Poologaindran, L. Crawford, I. M. Young, R. Romero-Garcia, M. E. Sughrue // Omniscient Neurotechnology. — 2021. — С. 1–12.

[5] Tanglay O. Graph Theory Measures and Their Application to Neurosurgical Eloquence / O. Tanglay, N. B. Dadario, E. H N Chong, S. J. Tang, I. M. Young, M.E Sughrue // Cancers (Basel). — 2023. — С. 1–16.

[6] OpenNeuro. — SCA2 Diffusion Tensor Imaging. https://openneuro.org/datasets/ds001378/versions/00003.

Тенденции развития современной инфраструктуры программного обеспечения цифровой экономики

Мирошниченко И.Д., ст. преподаватель СПбГУ i.miroshnichenko@spbu.ru,
Шапилов К.А., студент 2 курса мат.-мех. ф-та st126028@student.spbu.ru

Аннотация

В этой статье отмечаются тенденции формирования и развития инфраструктуры программного обеспечения цифровой экономики с учетом спектра разнообразных стандартов и связи компонентов их интерфейсов. Рассматриваются принципы и направления современных разработок с точки зрения развития технических и программных средств.

Введение

С 2017 года осуществляются процессы цифровизации экономики согласно указам президента РФ по реализации национальных программ, которые успешно были реализованы к 31 декабря 2024 года.

В рамках реализации указов Президента РФ от 07 мая 2018 г. № 204 «О национальных целях и стратегических задачах развития Российской Федерации на период до 2024 года» [1] и от 21 июля 2020 г. № 474 «О национальных целях развития Российской Федерации на период до 2030 года», в том числе с целью решения задачи по обеспечению ускоренного внедрения цифровых технологий в экономике и социальной сфере, Правительством РФ сформирована национальная программа «Цифровая экономика Российской Федерации», утверждённая протоколом заседания президиума Совета при Президенте Российской Федерации по стратегическому развитию и национальным проектам от 04 июня 2019 г. № 7. Эта национальная программа завершена 31 декабря 2024 г., итоги её реализации привели к реализации с 2025 года новой программы, которая является продолжением этих проектов на современном уровне цифровизации.

Чтобы оценить объём отечественного рынка программных продуктов, заметим, что на момент написания статьи официальный сайт Реестра российского программного обеспечения [2] включал 26544 продукта, которые принадлежат 9907 правообладателям.

В Интернет-ресурсах можно найти обзоры тенденций развития инфраструктуры отечественного программного обеспечения (далее ПО) как бизнес-аналитиков [3], так и консалтинговых компаний [4], общих и подробных, относящихся к исследуемых ими областям цифровой экономики.

Цель данной статьи - дать, по возможности, краткий обзор современных тенденций инфраструктуры программного обеспечения и, более конкретно, выделить тенденции инфраструктуры отечественного программного обеспечения с точки зрения применимости в учебном процессе в современных реалиях. В частности, результаты могли бы быть полезны для построения соответствующего курса для магистров или бакалавров, которые должны учитывать особенности современного развития в области больших данных (и их роста), необходимости распределенных вычислений, киберугроз, потребности в высокой доступности и масштабируемости, а также подходов контейнеризации, микросервисных архитектур и облачных технологий.

Основная часть

Прежде всего, нужно отметить, что цифровая экономика базируется не на материальных ресурсах, а на данных, то есть, информации, интеллектуальных продуктах и цифровых платформах. Таким образом, основными отличиями цифровой экономики от традиционной, в первую очередь, считаются следующие:

- глобальность (цифровые услуги могут оказываться независимо от местоположения),
- автоматизация процессов,
- высокая скорость масштабирования и адаптации.

Рассмотрим основные характеристики цифровой экономики:

- ценность данных и их использование как стратегического ресурса (инвестирование в сбор и анализ больших данных, чтобы принимать обоснованные и выгодные решения),
- рост значения цифровых платформ как посредников между бизнесом и потребителем,
- увеличение ценности платформы с ростом числа пользователей (причём, на одной платформе) могут быть объединены миллионы пользователей и поставщиков.

Интернет-технологии и программное обеспечение в цифровой экономике обеспечивают главную инфраструктурную функцию (то есть взаимодействие между пользователями) и вышеуказанные три характеристики. Цифровые

сервисы: (банковские приложения, маркетплейсы, госуслуги) базируются на распределенной IT-инфраструктуре (серверах, базах данных, контейнерных и облачных решениях, средствах доставки контента), главными требованиями к которой являются масштабируемость, отказоустойчивость и гибкость (легкость адаптации к реальным техническим условиям), где современные инструменты позволяют автоматизировать конфигурацию, масштабирование и обновление сервисов практически почти без вмешательства человека.

Инфраструктуру программного обеспечения составляют

- системное программное обеспечение (и, в первую очередь, операционные системы, сетевое программное обеспечение),
- средства виртуализации (гипервизоры, обеспечивающие разделение физических ресурсов между несколькими виртуальными машинами),
- СУБД (например, PostgreSQL, которая обеспечивает надёжность и масштабируемость при хранении и обработке больших объёмов данных).

Цифровые сервисы (банковские приложения маркетплейсы, телемедицинские платформы, госуслуги) функционируют благодаря распределенной ИТ-инфраструктуре (в нее входят серверы, базы данных, контейнерные и облачные решения, системы доставки контента). Например, портал «Госуслуги» взаимодействует с инфраструктурой «ГосТех» для унификации и масштабирования различных ведомств [5].

Программная инфраструктура влияет на скорость цифровой обработки: с внедрением DevOps-практик (методология CI/CD и подхода IaC) в разы быстрее внедряются цифровые продукты, так как минимизируется вмешательство человека в процесс автоматизации, конфигурирование, масштабирование и обновление сервисов [6]. В телемедицине («СберЗдоровье») обеспечивается защита персональных данных, видеосвязь, интеграция с базами рецептов на облачной инфраструктуре, часто с использованием микросервисной архитектуры.

Для хранения и обработки данных используются системы управления базами данных (PostgreSQL), обеспечивающие надёжность и масштабируемость при работе с большими объёмами информации (где используются инструменты мониторинга, которые собирают метрики, предоставляющие данные для анализа производительности и состояния сервисов) [7].

Современная тенденция в развитии виртуализации – контейнеризация (технология, суть которой в том, что приложения запускаются в изолированных средах – контейнерах). Наиболее известный инструмент средств виртуализации Docker, а в распределенных системах для управления множеством контейнеров функционирует оркестратор Kubernetes.

Современные подходы к управлению инфраструктурой требуют внедрения концепции IaS, описывающую всю конфигурацию систем в виде исходного кода, что позволяет автоматизировать развёртывание, тестирование и масштабирование инфраструктуры, а также CI/CD системы (Jenkins или GitLab CI), автоматизирующие процесс интеграции и доставки программных обновлений (обеспечивают непрерывность цифровых сервисов и быстрого внедрения новых функций) [8].

Современная цифровая инфраструктура строится сегодня на принципах базовой модели Открытых систем Open Source, что минимизирует издержки лицензирования, быстрее реагировать на уязвимости и использовать перспективные решения. Сегодня базовая модель открытой системы – часть программной экосистемы, обеспечивающая ускорение цифровой трансформации и технологическую независимость.

Облачные модели предоставления услуг – IaaS, PaaS и SaaS – важные составляющие современной IT-инфраструктуры. Edge и Serverless computing – новые подходы к обработке данных, направленные на повышение производительности и снижение затрат на инфраструктуру.

Edge и serverless computing – новые подходы к обработке больших данных, повышающие производительность и снижение затрат на инфраструктуру. Такой подход переносит вычислительные ресурсы ближе к источникам данных, а это снижает издержки и повышает эффективность обработки Код запускается без того, чтобы управлять серверами, автоматически масштабируются ресурсы в зависимости от загрузки: - технология способна адаптироваться к изменяющимся условиям.

API-first подход и API-экономика – приоритетные подходы в разработке программного обеспечения: API-first – разработка интерфейсов программирования приложений – обеспечивает стандартизированный удобный способ взаимодействия между различными компонентами системы, API-экономика представляет API как стратегический актив, позволяющий расширять сервисы, интегрироваться с партнерами и создавать новые бизнес- модели [9].

DevOpsи DevSecOps – организационно-технологические стандарты, предназначенные для разработки, тестирования и эксплуатации программного обеспечения. Первый стандарт позволяет ускорить процесс выпуска продуктов ввиду тесного взаимодействия разработчиков и операционных команд, а второй стандарт интегрирует аспекты безопасности на всех этапах жизненного цикла разработки, что очень важно в условиях интенсивности киберугроз [10].

Важная тенденция в современном технологическом подходе (упоминавшийся выше Edge computing) – переход к периферийным вычислениям – подход, в котором значительная часть обработки данных переносится ближе к

источникам данных. Это снижает задержки, уменьшает нагрузки на каналы связи и делает критические системы устойчивее к сбоям. Edge-вычисления применяются в устройствах Интернета-вещей (IoT), узлах 5G-сетей или локальных центрах обработки данных, расположенных непосредственно у заказчика.

Отметим тенденцию усиления значения применения интеллектуальных систем в управлении инфраструктурой ввиду усложнения и распределённости IT-сред. В частности, в подходе AIOps, сочетающем анализ больших данных и алгоритмы машинного обучения для мониторинга и поддержки IT-систем (платформы AIOps: Dynatrace, Splunk, Moogsoft), собираются и обрабатываются события, логи и метрики, позволяющие выявлять скрытые зависимости, прогнозировать сбои и автоматически инициировать корректирующие действия до того, как инциденты станут критичными.

В мониторинге надежности современной инфраструктуры также реализуется тенденция проактивного мониторинга – практика, при которой отклонения от норм фиксируется ещё до возникновения серьёзных последствий, проактивные системы непрерывно анализируют телеметрию и реагируют на потенциальные угрозы в реальном времени. Инструменты (Prometheus, Zabbix и Grafana) обеспечивают визуализацию метрик, генерацию оповещений и автоматическое масштабирование или перезапуск компонентов системы. Такой подход не только сокращает время простоя, но и создаёт основу для самовосстанавливающихся архитектур, устойчивых к сбоям и изменяющимся нагрузкам.

Санкции привели к ограничению доступа к западным продуктам и сервисам, начал формироваться внутренний рынок отечественного ПО. Наиболее заметные решения ОС Astra Linux от АО «НПО РусБИТТех» (в военном секторе и госсекторе), платформа виртуализации СКАЛА-Р от ГК «Инфосистемы Джет», отечественные СУБД (PostgreSQL, Postgres Pro), система криптографической защиты от КриптоПро, инструменты от «РЕД СОФТ» и других российских вендоров [11] (в Альфа-Банке, в частности, внедрена специализированная платформа Feature Store, open-source библиотека AUF). Современные разработки инфраструктуры отечественного ПО активно применяют методы машинного обучения.

Заключение

В работе дан краткий обзор понятий, инструментов и подходов, используемых в разработке и функционировании современной инфраструктуры программного обеспечения, что дает представление о современных тенденциях

в этой области. Проведен сравнительный анализ тенденций развития современных ресурсов ПО с учетом их комплексного взаимодействия. Результаты могли бы быть полезны для построения соответствующего курса для магистров или бакалавров, которые должны учитывать особенности современного развития в области больших данных (и их роста), необходимости распределенных вычислений, киберугроз, потребности в высокой доступности и масштабируемости, а также подходов контейнеризации, микросервисных архитектур и облачных технологий, а также выбора конкретного ПО для направлений разработки студенческих проектов.

Список литературы

[1] О стратегии развития информационного общества в Российской Федерации на 2017-2030 годы [Текст]: Указ Президента Российской Федерации от 09.05.2017 г. № 203

[2] Официальный сайт Реестра российского программного обеспечения. URL: https://reestr.digital.gov.ru

[3] Почепский А. Список российского ПО: перечень программного обеспечения отечественного производства. 2022. Электронный ресурс. URL: https://www.cleverence.ru/articles/auto-busines/spisok-rossiyskogo-po-kakoe-programmnoe-obespechenie-otechestvenn #27

[4] Strategy Partners. Обзор российского рынка инфраструктурного ПО и перспективы его развития. Май 2025. Электронный ресурс. URL: https://yandex.ru/search/?text=обзор+тенденций+инфраструктуры+российского+ПО&lr=2&clid=2323970-925&win=688

[5] Минцифры России: направления деятельности. Официальный сайт URL: digital.gov.ru/ru/activity/directions/990 (дата обращения 5.05.2025). Текст электронный.

[6] McKinsey & Company. Technology Trends Outlook 2022 URL: https://mckinsey.com/capabilities/mckinsey-digital/our-insights/the-top-trends-in-tech-2022 (дата обращения: 05.05.2025). Текст электронный.

[7] Prometheus Authors Overview
URL: https://prometeus.io/docs/introduction/overview
(дата обращения: 05.05.2025).
Текст электронный

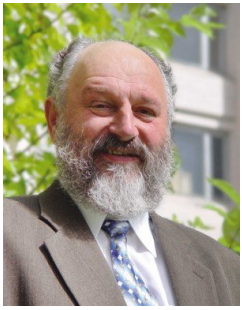
[8] Jenkins. Jenkins User Documentation
URL: https://jenkins.io/doc
(дата обращения 05.05.2025).
Текст электронный

[9] Red Hat Developers. An API-first approach to building Node.js applications
URL: https://developers.redhat.com/articles/2022/10/18/api-first-approach-building-nodejs-applications
(дата обращения 05.05.2025)
Текст электронный

[10] What is DevOps? Meaning, methodology and guide [Электронный ресурс] TechTarget: Sarch IT Operations.
URL: https://www.techtarget.com/searchitoperations/definition/DevOps
(дата обращения 05.05.2025)
Текст электронный.

[11] «Инфосистемы Джет» и ГК «Астра» обеспечат качество и надёжность импортозамещённых ИТ-систем// Астра. 16.05.2022: пресс-центр компании
URL: https://astra.ru/about/press-ctnter/news/infosistemy-dzhet-i-gk-astra-obespechat-kachestvo-i-nadyezhnost-?ysclid=mafvu3l5u715505355
(дата обращения: 07.05.2025)
Текст электронный

Системное программирование и программная инженерия



Терехов Андрей Николаевич
д.ф.-м.н., профессор, заведующий кафедрой системного программирования, президент ООО «Ланит-Терком»

Литвинов Юрий Викторович, к.т.н., старший преподаватель кафедры системного программирования

Луцив Дмитрий Вадимович, к.ф.-м.н., доцент кафедры системного программирования

Сартасов Станислав Юрьевич, к.т.н., Chief Technical Officer, TheDenominator, Inc.

Разработка инфраструктуры для создания специализированных ускорителей на основе Interaction Nets

Кубышкин Е.А., СПбГУ, Санкт-Петербург st098235@student.spbu.ru,
Пономарев Н.А., СПбГУ, Санкт-Петербург n.ponomarev@spbu.ru

Аннотация

Нерегулярный параллелизм — ситуация, когда сложно заранее определить количество независимых подзадач — важная проблема на пути к высокой производительности в таких важных областях, как искусственный интеллект, анализ графов, и многих других. Одним из решений этой проблемы является модель вычислений Interaction Nets, для которой параллелизм подобного рода естественен. Тем не менее на данный момент существуют только программные реализации данной системы. В работе представлен стенд для прототипирования сопроцессоров, основанных на Interaction Nets: от программы на высокоуровневом языке до прошивки ПЛИС.

Введение

Потребность в параллельной обработке данных нарастает. Это связано как с усложнением используемых алгоритмов, так и с ростом объёма обрабатываемых данных. Однако в ряде задач, например, связанных с обработкой графов, искусственным интеллектом и разреженной линейной алгеброй, возникает проблема нерегулярного параллелизма с которой традиционные архитектуры в виду своей специфики справляются неэффективно.

Interaction Nets [1] — модель вычислений, основанная на переписывании графов, для которой естественен нерегулярный параллелизм. Программы в ней представляются в виде неориентированного меченого графа, называемого сетью. Процесс вычисления (редукции) происходит за счёт переписывания подграфов. Важным свойством системы является то, что редукции локальны, что позволяет применять все доступные переписывания одновременно. Кроме того известно, что модель полна по Тьюрингу, а результат вычислений не зависит от порядка редукций.

Важно отметить, что, меняя правила редукции и метки на узлах, можно параметризовать сети и оптимизировать вычисления для какой-то предметной области. Таким образом Interaction Nets можно рассматривать, как

крупно-гранулярную реконфигурируемую архитектуру (CGRA) [2]. Такие архитектуры сочетают гибкость программных моделей с высокой производительностью специализированных аппаратных решений.

На данный момент существуют только программные реализации Interaction Nets [3], поэтому нам хотелось бы изучить можно ли ускорить некоторые алгоритмы в помощью сопроцессора на Interaction Nets.

Цели и задачи

Для исследования данной гипотезы нами был начат проект Lamagraph[1], целью которого является разработка параметризуемого многоядерного вычислителя на основе Interaction Nets.

Для её достижения были выделены следующие этапы.

1. Создание минимальной инфраструктуры для создания специализированных вычислителей на основе Interaction Nets.
2. Разработка eDSL для спецификации Interaction Nets.
3. Добавление поддержки одновременной редукции в ускоритель.
4. Проведение сравнительных экспериментов на задачах разреженной линейной алгебры, анализа графов и искусственного интеллекта.

На данный момент проект находится на первом этапе: создание минимального рабочего окружения для прототипирования вычислителей и работы с ними, который состоит из следующих подшагов.

1.1. Реализация высокоуровневого языка программирования.

1.2. Разработка транслятора из высокоуровневого языка в Interaction Nets.

1.3. Реализация интерпретатора Interaction Nets.

1.4. Разработка генератора прошивки вычислителя для ПЛИС.

Требования к перовому этапу

Первый этап главным образом создаёт фундамент для дальнейшей разработки и экспериментов. Так что фокус смещён в сторону скорости разработки и анализа результатов для возможности оперативно проверять рабочие гипотезы. Исходя из этого, к проекту были выставлены следующие требования.

[1]Репозитории проекта доступны по ссылке: `https://github.com/Lamagraph`

Гомогенность. Использование единого стека технологий позволяет упростить создание и поддержку программно-аппаратного стека, а также цепочку обработки данных.

Целостность. Получение полнофункционального прототипа, содержащего все компоненты, важнее, чем детальная проработка какого-то отдельного компонента. На данном этапе необходимо в первую очередь выстроить взаимодействие между компонентами, а также увидеть, какие сложности возникают не только при реализации отдельных компонент, но и при интеграции между ними. Это позволяет быстрее выявлять и исправлять неудачные архитектурные решения.

Гранулярность. На каждом шаге работы программно-аппаратного стека есть результат исполнения программы. Такой подход позволяет тестировать как каждую отдельную компоненту, так и сразу несколько.

Реализация

В качестве основного языка был выбран Haskell из-за его удобства для работы высокоуровневыми абстракциями, лёгкостью в создании DSL (и eDSL) и наличии HDL, основанного на нём.

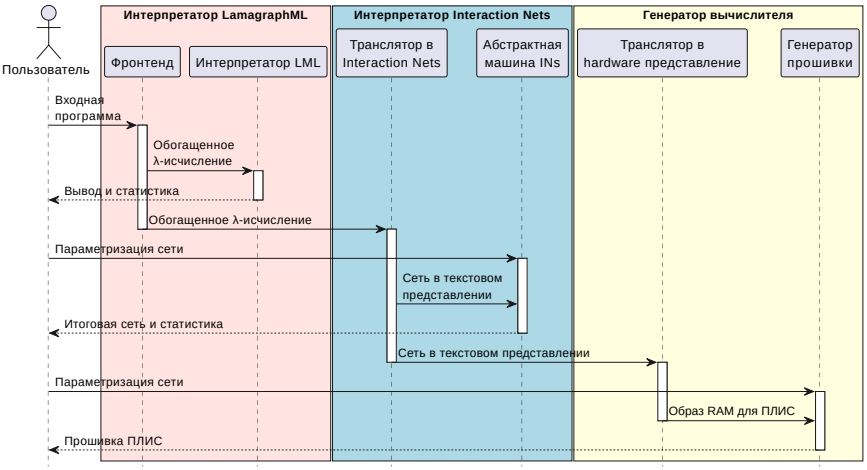


Рис. 1: Диаграмма взаимодействия пользователя с программно-аппаратным стеком Lamagraph.

Взаимодействие с пользователем

Входная программа на LamagraphML сначала транслируется в обогащённое λ-исчисление; далее это представление можно транслировать в сеть в текстовом представлении, для которого реализована абстрактная машина; и из текстового представления можно получить образ памяти для ПЛИС и получить прошивку. На каждом этапе можно исполнить программу и получить статистику и результат исполнения. Весь программно-аппаратный стек представлены на Рис. 1. Было решено разделить проект на три большие части, далее они будут описаны более подробно.

Интерпретатор LamagraphML

Многие реализации Interaction Nets придумывают собственные языки программирования (например, [4] или Bend[2]), которые не привычны программисту, поскольку пытаются текстово описывать граф. Lamagraph решает данную проблему, поскольку в нашем проекте Interaction Nets используется в качестве промежуточного представления, а пользователю предоставляется относительно привычный высокоуровневый ML-подобный язык — LamagraphML. Функциональный язык выбран из-за наличия схем трансляции λ-исчисления в Interaction Nets. После работы фронтенда мы получаем программу в виде обогащенного λ-исчисления, которую потом можно либо интерпретировать, либо транслировать в Interaction Nets.

Интерпретатор Interaction Nets

Стандартным представлением для Interaction Nets является графовое, однако с ним (как и с любым графом) трудно работать в функциональных языках программирования. Тем не менее существует текстовое представление сетей [5], для которого разработана абстрактная машина [6]. Поэтому для реализации было выбрано именно оно.

В процессе реализации транслятора в Interaction Nets была обнаружена проблема: существующие схемы трансляции работают только с чистым λ-исчислением. Попытки расширить схему трансляции предпринимались в [7], однако такой подход требует использование динамического базиса агентов. Но поскольку вычислитель прошивается на ПЛИС, то базис агентов должен быть фиксирован, и подход предложенный в статье нам не подходит. Поэтому трансляция в Interaction Nets конструкций обогащённого λ-исчисления —

[2]Репозиторий проекта: `https://github.com/HigherOrderCO/Bend`

предмет дальнейшей работы.

Генератор вычислителя

Языком программирования ПЛИС был выбран Clash [8] — HDL, основанный на Haskell, который потом транслируется в традиционные HDL. Такое решение продиктовано простотой, в сравнении с традиционными HDL, создания и работы с высокоуровневыми абстракциями. Clash позволяет описывать комбинационные схемы почти как обычный Haskell код, что открывает возможность писать модульные и даже property-based тесты. Более сложные последовательностные схемы же можно симулировать с помощью симулятора Clash и писать для них test bench файлы.

Дизайн вычислителя сделан таким образом, чтобы явно отделить разные этапы выполнения редукции: работа с памятью, применение правила редукции и встраивание локальных изменений в глобальную сеть. Это позволило оставить пространство для дальнейших оптимизаций. Также такое решение помогло вынести реализацию спецификации Interaction Nets почти полностью на уровень программирования на типах, что значительно упрощает отладку и интеграцию с остальными компонентами системы.

Заключение

По итогам работы над первым этапом, можно сказать, что получилось создать полный программно-аппаратный стек разработки вычислителя, основанного на Interaction Nets: от программы на высокоуровневом языке до прошивки ПЛИС. Более подробно, на основе анализа предметной области и общего виденья проекта, были выделены ключевые требования (**гомогенность**, **целостность** и **гранулярность**), и в соответствие с ними реализованы следующие компоненты.

- Интерпретатор языка LamagraphML, поддерживающий невзаимную рекурсию и встроенные алгебраические типы данных `list` и `option`.
- Транслятор из чистого λ-исчилсения в Interaction Nets в стратегии Call-by-Value.
- Полноценный интерпретатор для Interaction Nets на основе абстрактной машины.

- Генератор прошивки одноядерного вычислителя на основе Interaction Nets, поддерживающий параметризацию агентами и правилами редукции.

Также удалось выявить узкое место в виде трансляции обогащённого λ-исчисления в Interaction Nets.

Список литературы

[1] Lafont Yves. Interaction nets // Proceedings of the 17th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. – POPL '90. – New York, USA: Association for Computing Machinery, 1989. – С. 95–108

[2] Liu L. et al. A survey of coarse-grained reconfigurable architecture and design: Taxonomy, challenges, and applications //ACM Computing Surveys (CSUR). – 2019. – Т. 52. – №. 6. – С. 1-39.

[3] Hassan Abubakar, Mackie Ian, Sato Shinya. An Implementation Model for Interaction Nets // Electronic Proceedings in heoretical Computer Science. – 2015. – may. – Vol. 183. – С. 66–80

[4] Sato S. Design and implementation of a low-level language for interaction nets : дис. – University of Sussex, 2015.

[5] Fernández M., Mackie I. A calculus for interaction nets //International Conference on Principles and Practice of Declarative Programming. – Berlin, Heidelberg : Springer Berlin Heidelberg, 1999. – С. 170-187.

[6] Pinto J. S. Sequential and concurrent abstract machines for interaction nets //International Conference on Foundations of Software Science and Computation Structures. – Berlin, Heidelberg : Springer Berlin Heidelberg, 2000. – С. 267-282.

[7] Jiresch E. R. W. Extending interaction nets towards the real world : дис. – Technische Universität Wien, 2012.

[8] Christiaan Baaij, Matthijs Kooijman, Jan Kuper et al. CLaSH: Structural Descriptions of Synchronous Hardware Using Haskell // 2010. – 09. – С. 714–721

Уточнение GNSS-позиционирования при помощи визуально-инерциальной одометрии

Карасева Е.О., СПбГУ, Санкт-Петербург liza.1610@mail.ru

Аннотация

В данной работе предлагается модифицировать визуально-инерциальный SLAM-алгоритм путем добавления в конвейер GNSS-данных. Статья содержит подробный обзор алгоритмов, обоснования и описание работы выбранного алгоритма, а также описание реализации решения.

Введение

Прогресс в робототехнике ускоряет развитие систем автономной навигации для дронов и автомобилей. Такие системы используют данные с различных датчиков для одновременной локализации и картографирования (SLAM). Популярный подход к решению задачи SLAM — визуально-инерциальная одометрия (VIO), основанная на камерах и данных с инерциальных измерительных модулей (IMU), однако она подвержена накоплению ошибок из-за использования относительных измерений.

Добавление GNSS-данных, представляющих абсолютные координаты, позволяет снизить эти ошибки и повысить точность. В работе предлагается интеграция GNSS в модуль Kimera-VIO с последующей оценкой на расширенном датасете EuRoC. Ожидается повышение точности траектории.

Решения, интегрирующие GNSS, VIO и SLAM

Интеграция GNSS в VI-SLAM активно исследуется [8], но доступных решений с открытым исходным кодом немного. Чаще всего это модификации существующих визуально-инерциальных алгоритмов с недостаточной документацией и без оценки точности траекторий, что затрудняет их использование. Кроме того, часть из них работает только с GPS, что снижает надёжность и требует доработок для поддержки данных других навигационных систем. Анализ решений представлен в Таблице 1.

Алгоритм	**Тип**	**Обновления**	**Лицензия**
MINS	GNSS	2024	GPLv3
GNSS-Stereo-Inertial Fusion	GNSS	2024	GPLv3

RTK-Visual-Inertial-Navigation	GNSS	2024	GPLv3
InGVIO	GNSS	2023	GPLv3
IC-GVINS	GNSS	2023	GPLv3
SVO-Pro-GPS	GPS	2023	GPLv3
VINS-Fusion	GPS	2021	GPLv3
GVINS	GNSS	2021	GPLv3

Таблица 1: VI-SLAM алгоритмы с поддержкой GNSS и открытым исходным кодом

Ни одно из рассмотренных решений не подходит для использования в коммерческих проектах, поэтому было принято решение провести внедрение GNSS в один из VI-SLAM-алгоритмов самостоятельно.

Обзор современных реализаций VI-SLAM

В качестве алгоритма для внедрения GNSS-данных было проведено сравнение известных реализаций SLAM с использованием визуально-инерциальной одометрии [2], [3], [4]. Главным критерием отбора являлось наличие открытого исходного кода. Сравнение проводилось по следующим параметрам: тип поддерживаемых камер, метод обработки изображений (прямой или выделения особенностей), способ оценки состояния (вероятностный или на основе оптимизации), наличие механизма коррекции ошибок (замыкание цикла), тип лицензии (необходимая — разрешающая коммерческое использование), а также поддержка проекта — наличие обновлений в течение последних пяти лет.

На основе сравнения VI-SLAM-систем (Таблица 2) можно сделать выводы, что ORB-SLAM3 [9] и Kimera [7] поддерживают все типы камер, VINS-Fusion работает с моно- и стереокамерами, OKVIS ориентирован на стереокамеры, а ROVIO – только на монокулярные. ROVIO существенно отличается благодаря использованию расширенного фильтра Калмана (алгоритма оценки состояния системы на основе нелинейных функции) и прямого метода анализа изображений, что делает его быстрее, но менее точным и более чувствительным к шуму. ORB-SLAM3, VINS-Fusion и Kimera обладают механизмом замыкания циклов для компенсации накопленных ошибок, отсутствующим в OKVIS и ROVIO. Лицензия BSD, подходящая для коммерческого применения, используется в Kimera, OKVIS и ROVIO, тогда как ORB-SLAM3 и VINS-Fusion распространяются под GPLv3. Наибольшую поддержку разработчиков имеют Kimera, ORB-SLAM3 и ROVIO.

Алгоритм	Тип камер	Фронтенд	Бэкенд	Loop closure	Лицензия	Обновления
ORB-SLAM3	Моно, Стерео, RGB-D	Особенности	Оптимизация	Да	GPLv3	2022
VINS-Fusion	Моно, Стерео	Особенности	Оптимизация	Да	GPLv3	2021
OKVIS	Стерео	Особенности	Оптимизация	Нет	BSD 3-Clause	2020
ROVIO	Моно	Прямой метод	Фильтрация	Нет	BSD 3-Clause	2024
Kimera [7]	Моно, Стерео, RGB-D	Особенности	Оптимизация	Да	BSD 2-Clause	2025

Таблица 2: Современные алгоритмы VI-SLAM, данные актуальны на 14.02.25

С учетом указанных преимуществ для интеграции GNSS-данных выбрана библиотека Kimera.

Диаграмма последовательностей Kimera-VIO

На Рисунке 1 представлена диаграмма последовательностей Kimera-VIO. Библиотека реализует конвейерную параллельность: обработка каждого кадра разбита на этапы, а данные передаются через потокобезопасные очереди и буферы асинхронно.

DataProviderModule отслеживает состояние буферов, собирает пришедшие данные в синхронизированные пакеты и отправляет их во FrontendModule. Он распознает ориентиры и передает их и предсказанную на основе IMU-данных позу в BackendModule, который создает из данных факторы и уточняет позу при помощи оптимизации факторграфа. MesherModule строит 3D-сетку на основе данных с фронтенда и бекенда, а VisualizerModule готовит данные для визуализации. Наконец, DisplayModule выводит картинку на экран. По окончании чтения датасета DataProviderInterface сигнализирует о завершении, данные перестают поступать в очереди модулей, Pipeline завершает работу каждого из них и возвращает управление в main(), где фиксирует результаты и завершает программу.

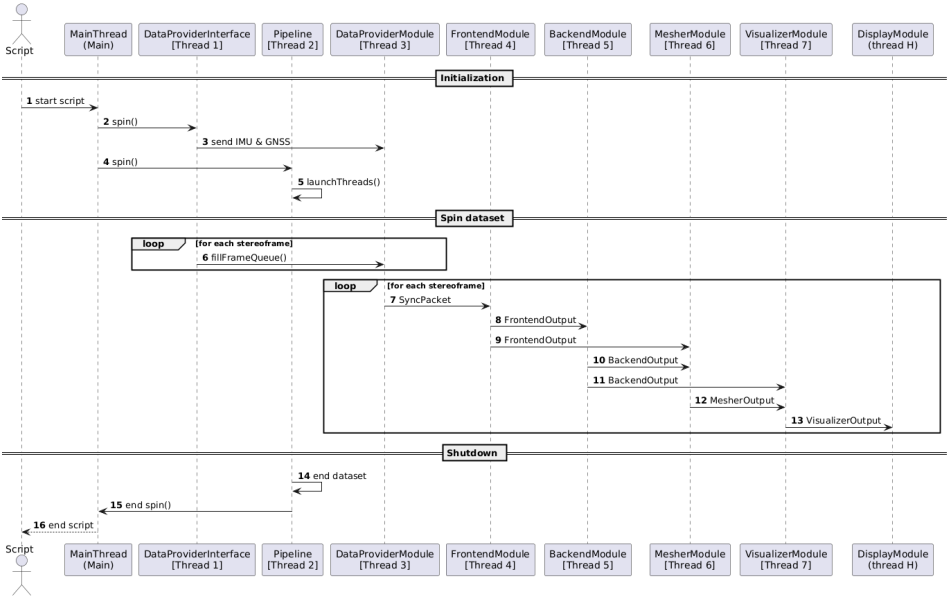


Рис. 1: Диаграмма последовательностей системы Kimera-VIO

Реализация

Основным требованием к решению является наличие поддержки GNSS-данных в алгоритме и повышение точности построенной траектории при запуске алгоритма на датасете EuRoc. Для выполнения задачи необходимо было разобраться с форматом данных, понять устройство конвейера и модифицировать часть модулей Kimera-VIO [6].

Формат и преобразования данных

Запуск Kimera-VIO предполагается на датасете EuRoC, содержащем данные с микро летательного аппарата, оснащённого стереокамерами и IMU. В датасете доступны эталонные измерения от системы Vicon (6-мерный вектор состояния с высокой частотой и точностью) и лазерного трекера Leica MS50 (три координаты с точностью до сотых миллиметра). На основе формата этих данных GNSS-измерения также заданы в виде трёх координат (x, y, z) с временной меткой и хранятся в отдельной папке gnss0.

Так как каждый сенсор имеет свою систему координат, перед использованием данные преобразуются в общую систему (в EuRoC это система IMU), с помощью матрицы трансформации $T_{BS} \in SE(3)$. Для этого координаты GNSS переводятся в однородные, умножаются на матрицу, а результат преобразуется обратно в афинные координаты.

Обработка параметров и чтение датасета

Преобразование выполняется на этапе чтения датасета. Для обработки GNSS-данных был добавлен модуль, отвечающий за чтение и передачу измерений в систему. Матрица TBS, сдвиг по времени и частота измерений хранятся в структуре. Параметры GNSS интегрированы в общую структуру, а сами измерения — в вектор. Передача в следующий модуль идет через callback-функции.

DataProviderModule

По мере чтения изображений разрозненные данные передаются в модуль, который синхронизирует их по времени. Для GNSS callback-функция помещает измерения в потокобезопасный буфер.

Когда буферы содержат достаточно изображений, IMU и GNSS, из них формируется пакет и передается во фронтенд. Сначала выбирается самый старый кадр левой камеры, затем — интерполируются данные IMU и подбирается ближайшее по времени изображение с правой камеры. Если в GNSS-буфере достаточно данных, проводится линейная интерполяция GNSS-координат по ближайшим временным меткам, после чего результат добавляется к пакету.

FrontendModule

Фронтенд проводит трекинг ориентиров, триангуляцию и предсказание позы по IMU. Так как фронтенд не использует глобальную позицию, GNSS-данные не влияют на этот этап обработки, а сразу передаются в Backend.

BackendModule

Бэкенд создает граф факторов [**?**], на каждой итерации добавляя новые факторы, удаляя устаревшие и вызывая оптимизацию. Результат передаётся

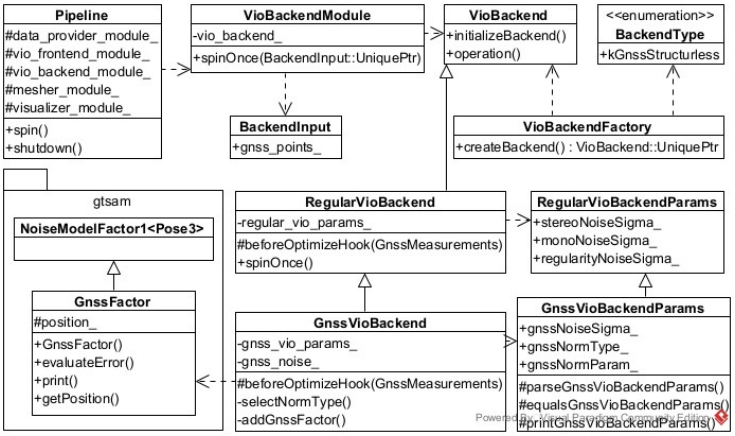


Рис. 2: Архитектура бэкенд модуля системы

модулям визуализации. В рамках работы был добавлен GNSS-фактор, который учитывает только координаты точек. Тип нормы ошибки (L2, Huber или Tukey) выбирается в параметрах в зависимости от ожидаемого уровня шума. После добавления GNSS-факторы учитываются в общей оптимизации графа (Рисунок 2).

Эксперимент

Для оценки точности алгоритма использовался модифицированный датасет EuRoC с симулированными GNSS-данными. Анализ траектории проводился на основе файлов журналов, сгенерированных при запуске. Было проведено более сотни запусков алгоритма и подобраны оптимальные параметры. Для эмуляции шума в GNSS-координатах был использован гауссовский шум, это допустимо [1] при упрощенной модели. В Таблице 3 показана полученная статистика: средние результаты реализации с GNSS превосходят стандартную, и в части случаев оказываются лучше значений на лучших запусках, представленных в оригинальной работе [7].

Заключение

В результате работы был проведен обзор GNSS-VI-SLAM-алгоритмов и VIO-SLAM-систем, потенциально подходящих для внедрения GNSS. После

Location	GNSS				No GNSS	Paper [2]
	Ideal	$\sigma = 0{,}5$м	$\sigma = 1{,}5$м	$\sigma = 3$м		
MH_01	**0.10**	0.27	0.25	0.19	0.16	0.11
MH_02	0.16	0.18	0.29	0.21	0.19	**0.10**
MH_03	**0.16**	0.28	0.19	0.23	0.16	0.16
MH_04	0.34	0.44	0.31	0.37	0.28	**0.24**
MH_05	**0.20**	0.24	0.22	0.23	0.27	0.35

Таблица 3: Сравнение ошибок построенных траекторий (RMSE ATE)

изучения архитектуры библиотеки Kimera была обеспечена загрузка, синхронизация и передача GNSS-данных в конвейер VIO. Также проведены эксперименты на GNSS-данных с разным уровнем шума: точность построенной траектории увеличилась, алгоритм стабилен даже при редких и неточных GNSS-данных. Реализация представлена по ссылке [10].

Список литературы

[1] Analysis and Discussion on the Optimal Noise Model of Global GNSS Long-Term Coordinate Series Considering Hydrological Loading / Yue-fan He, Guigen Nie, Shuguang Wu, Haiyang Li // Remote Sensing. — 2021. — Vol. 13, no. 3. — URL: https://www.mdpi.com/2072-4292/13/3/431 (дата обращения: 10 апреля 2025 г.).

[2] Bala Jibril, Adeshina Steve, Aibinu Abiodun. Advances in Visual Simultaneous Localisation and Mapping Techniques for Autonomous Vehicles: A Review // Sensors. — 2022. — 11. — Vol. 22. — P. 8943 (дата обращения: 20 ноября 2024 г.).

[3] A Comprehensive Survey of Visual SLAM Algorithms / Andréa Macario Barros, Yoann Moline, Frédérick Carrel et al. // Robotics. — 2022. — 02. — Vol. 11 (дата обращения: 20 ноября 2024 г.).

[4] A review of visual SLAM for robotics: evolution, properties, and future applications / Basheer Al-Tawil, Thorsten Hempel, Ahmed Abdelrahman, Ayoub Al-Hamadi // Frontiers in Robotics and AI. — 2024. — Vol. Volume 11 - 2024. — URL:https://www.frontiersin.org/journals/robotics-and-ai/articles/10.3389/frobt.2024.1347985 (дата обращения: 12 марта 2025 г.).

[5] Factor Graphs and GTSAM. — URL: https://gtsam.org/tutorials/intro.html (дата обращения: 12 марта 2025 г.).

[6] Kimera-VIO: Open-Source Visual Inertial Odometry. — URL: https://github.com/MIT-SPARK/Kimera-VIO (дата обращения: 20 ноября 2024 г.).

[7] Kimera: an Open-Source Library for Real-Time Metric-Semantic Localization and Mapping / Antoni Rosinol, Marcus Abate, Yun Chang, Luca Carlone // CoRR. — 2019. — Vol. abs/1910.02490. — arXiv : 1910.02490 (дата обращения: 20 ноября 2024 г.).

[8] Nathan Aaron. Loosely Coupled & Tightly Coupled INS & GNSS [2024 Guide]. — 2024. — URL: https://pointonenav.com/news/loose-vs-tight-coupling-gnss/ (дата обращения: 12 марта 2025 г.).

[9] ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual–Inertial, and Multimap SLAM / Carlos Campos, Richard Elvira, Juan J. Gomez Rodriguez et al. // IEEE Transactions on Robotics. — 2021. — . — Vol. 37, no. 6. — P. 1874–1890. — URL: http://dx.doi.org/10.1109/TRO.2021.3075644 (дата обращения: 20 ноября 2024 г.).

[10] Pull-request with GNSS pipeline. — URL: https://github.com/MIT-SPARK/Kimera-VIO/pull/258 (дата обращения: 20 мая 2025 г.).

Содержание

Организаторы конференции «Мат-мех. Наука 2025» благодарят спонсоров и партнёров конференции за неоценимую ресурсную и организационную помощь в проведении конференции и при подготовке сборника материалов.

В 2025 году нас поддержали:

Санкт-Петербургский государственный университет

Группа компаний «ЯДРО»

Научное издание

Избранные труды весенней научно-практической конференции по вопросам информатики, математики, механики и астрономии «Мат-мех. Наука 2025»

28 апреля – 3 мая 2025 г.
Санкт-Петербург

Компьютерная верстка: Д.В. Луцив

Издательство «ВВМ»
196247, г. Санкт-Петербург,
ул. Бассейная
д. 5, лит. А. кв. 52
E-mail: vvmpub@yandex.ru

Подписано в печать 04.07.2025. Формат 60 × 84 $^{1}/_{16}$.
Бумага офсетная. Гарнитура Times. Печать цифровая.
Усл. печ. л. 10,82. Тираж 100 экз. Заказ № 2517.

Отпечатано в Издательстве ВВМ.
196247, г. Санкт-Петербург, ул. Бассейная д. 5, лит. А. кв. 52.