МГУ им. М. В. Ломоносова
Факультет вычислительной математики и кибернетики
Конференция «Программирование и вычислительная математика»,
посвящённая 100-летию со дня рождения Николая Павловича Трифонова.

Методы глубокого обучения для обнаружения аномалий в данных журналов приложений в задачах надёжности.

Горохов Олег Евгеньевич,

к.ф.-м.н., доцент кафедры ИИТ

Петровский Михаил Игоревич,

д-р ф.-м.н., зав. каферой ИИТ

Машечкин Игорь Валерьевич

2025 г.

Содержание

- Введение.
- Аналитический обзор решений задачи мониторинга надёжности программных комплексов на основе выявления аномалий в потоках данных.
- Построение нейросетевого кластеризующего автокодировщика для мониторинга надёжности автоматизированных систем.
- Программная реализация и экспериментальная оценка предлагаемого решения.
- Заключение.

Содержание

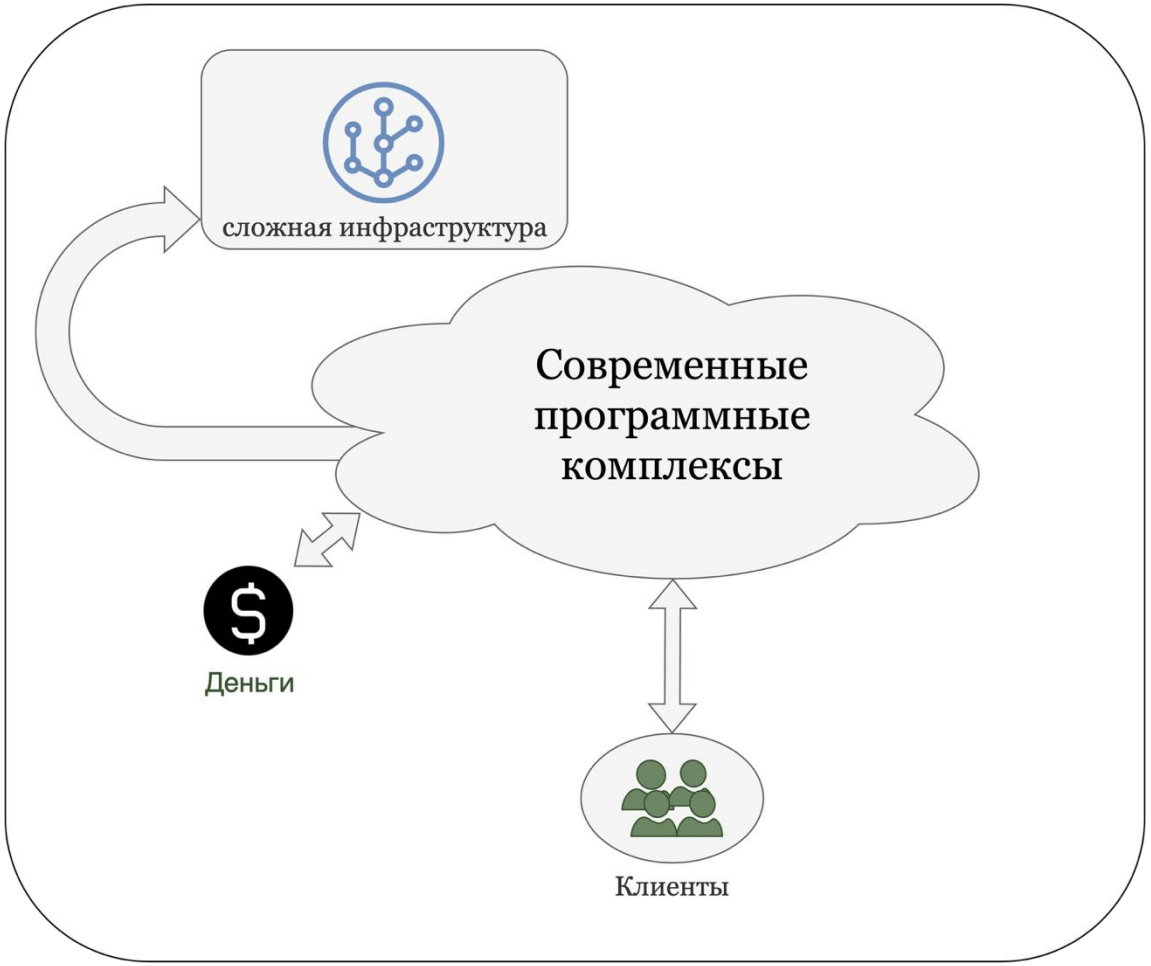
- **Введение.**
- Аналитический обзор решений задачи мониторинга надёжности программных комплексов на основе выявления аномалий в потоках данных.
- Построение нейросетевого кластеризующего автокодировщика для мониторинга надёжности автоматизированных систем.
- Программная реализация и экспериментальная оценка предлагаемого решения.
- Заключение.

Программные комплексы

- ***Программный компонент*** – программа, рассматриваемая как единое целое, выполняющая законченную функцию и применяемая самостоятельно или в составе комплекса[1].
- ***Программный комплекс*** – программа, состоящая из двух или более компонентов и (или) комплексов, выполняющих взаимосвязанные функции, и применяемая самостоятельно или в составе другого комплекса[1].
- Тенденции развития программных комплексов:
  - Расширение области использования;
  - Увеличение объёмов данных;
  - Увеличение числа пользователей.

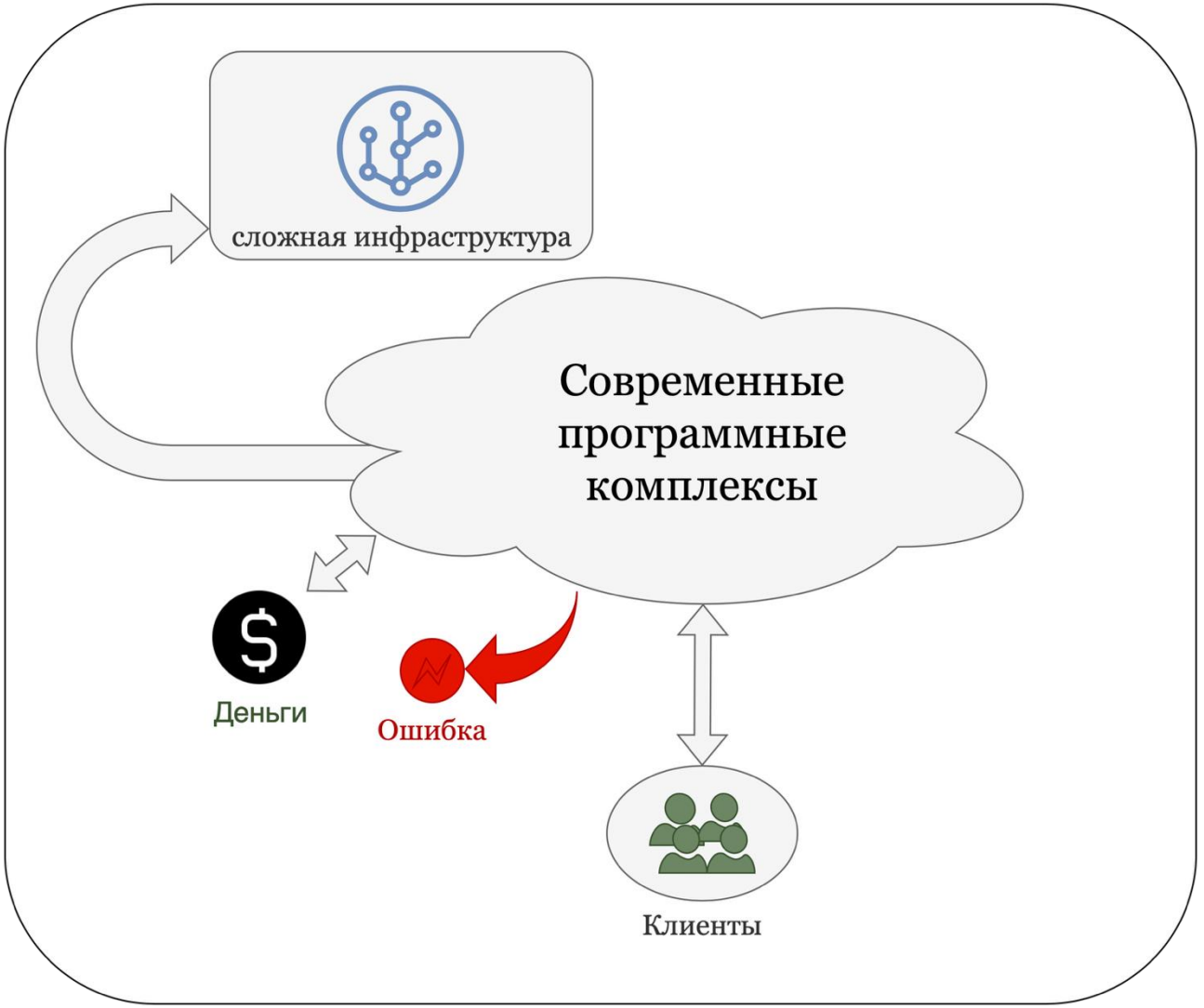
1. ГОСТ 19.101-77. Межгосударственный стандарт. Единая система программной документации. Виды программ и программных документов.

Современные программные комплексы (1)[1, 2]

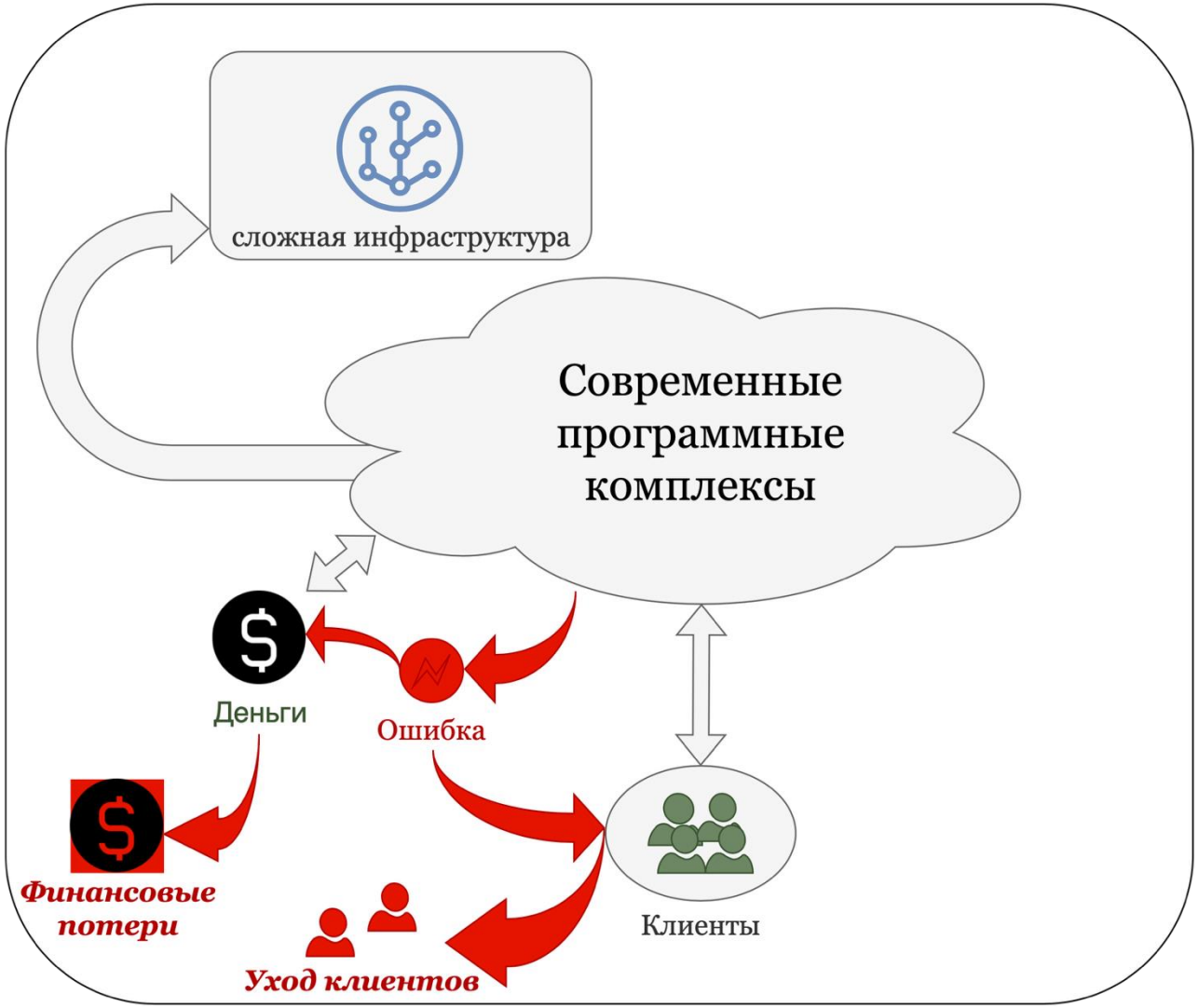


1. Le V. H., Zhang H. Log-based anomaly detection with deep learning: how far are we? //Proceedings of the 44th International Conference on Software Engineering. – 2022. – С. 1356-1367.
2. He S. et al. Identifying impactful service system problems via log analysis //Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. – 2018. – С. 60-70.

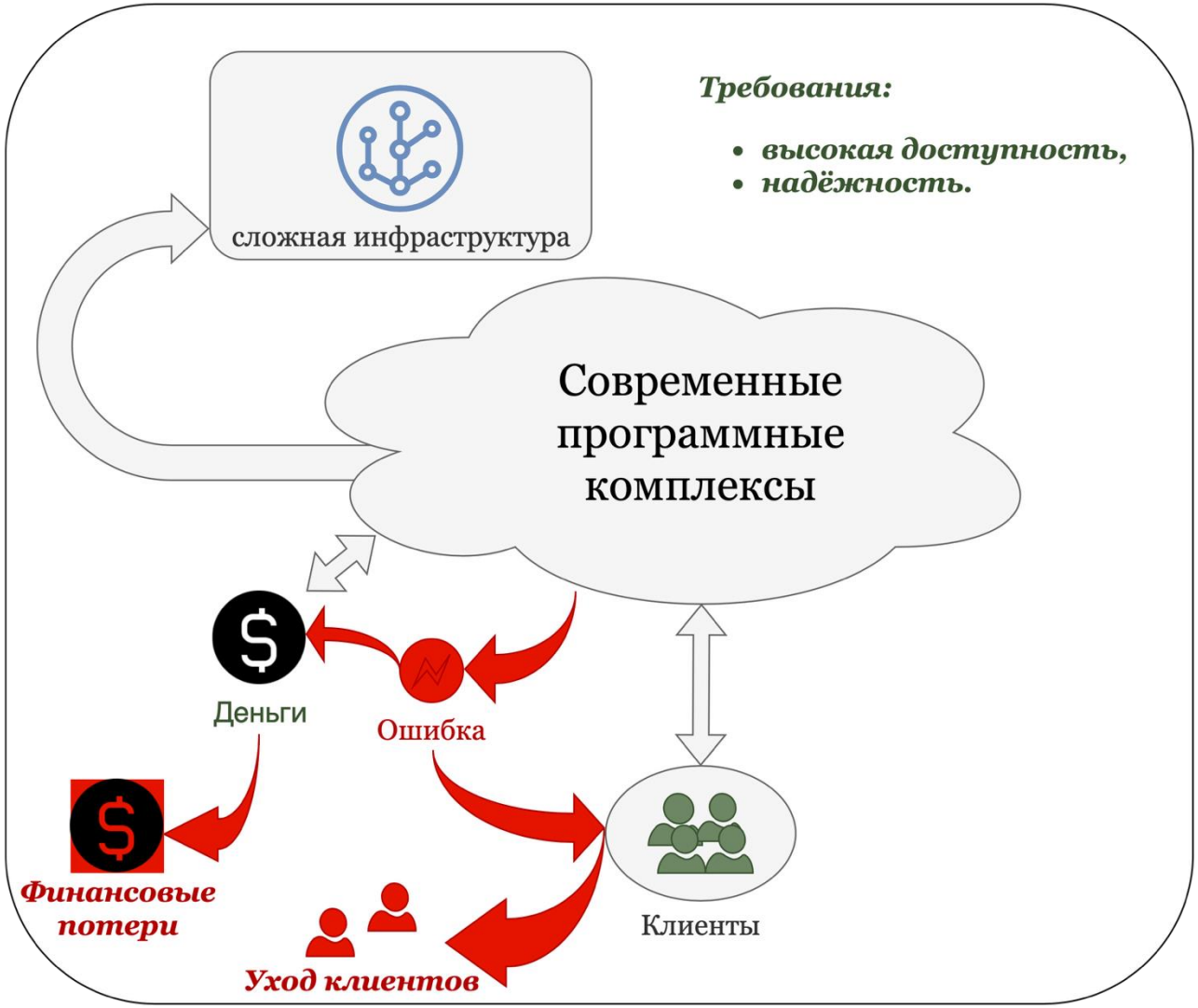
Современные программные комплексы (2)



Современные программные комплексы (3)



Современные программные комплексы (4)

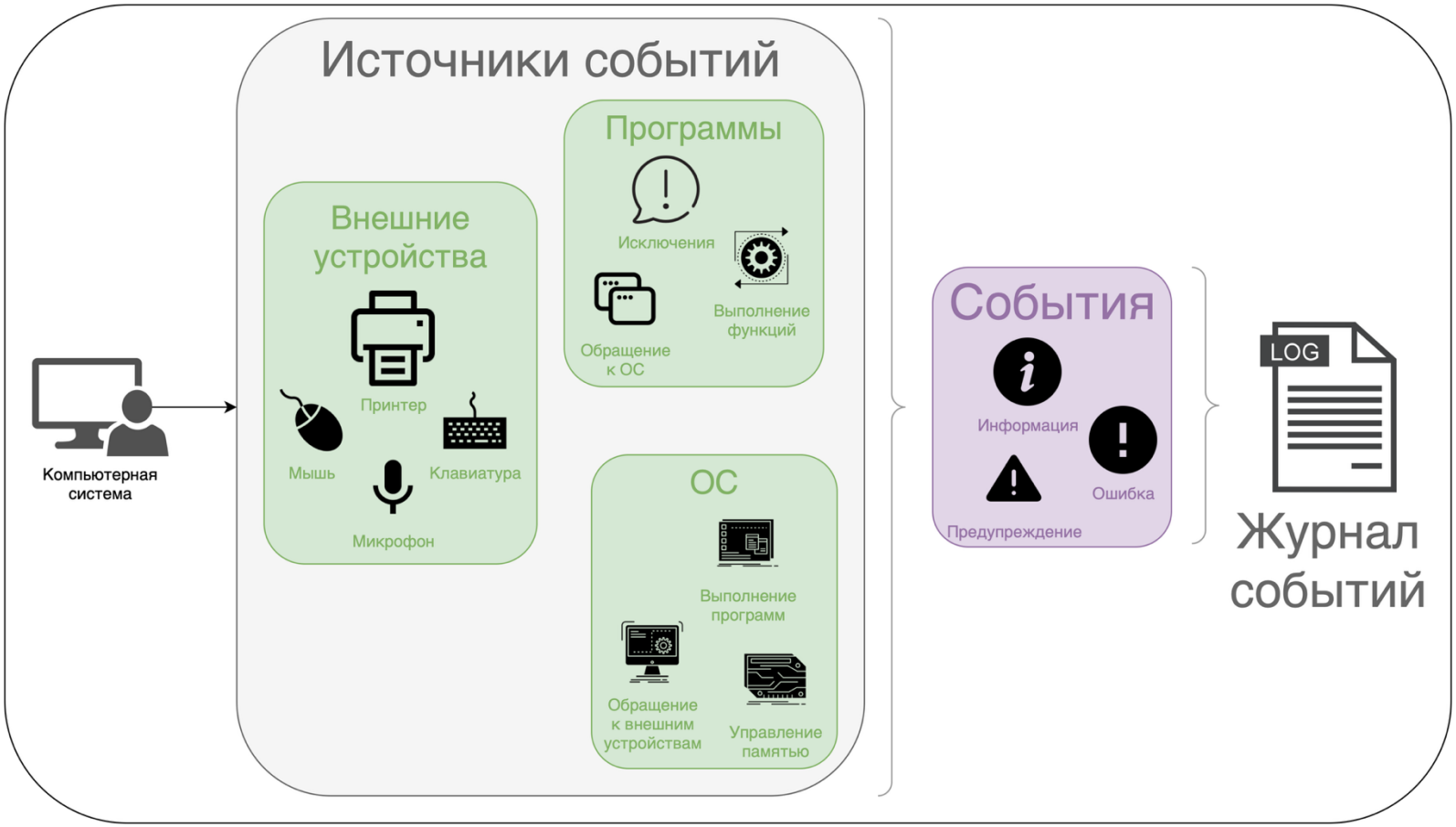


Отказы

- Ключевые ошибки – отказ системы.
- ***Отказ*** – событие, заключающееся в нарушении работоспособного состояния объекта[1].
- ***Предотказное состояние*** – состояние объекта, характеризуемое повышенным риском его отказа[1].
  - Вероятность отказа – вероятность того, что система находится в предотказном состоянии.
- Отказы происходят регулярно в различных программных системах и отраслях.

1. Г. Н. Черкесов, В. А. Нетес, В. А. Шпер, Ю. И. Тарасьев, Г. Ф. Ковалёв, Г. А. Федотова, И. Н. Долгополов, И. Б. Шубинский и А. В. Батурин, «ГОСТ 27.002-2015 Надежность в технике (ССНТ). Термины и определения (с Поправкой),» 1 Январь 2021. [В Интернете]. [Дата обращения: 6 Ноябрь 2022].

Данные о работе комплексов хранятся в журналах событий (логов).



Актуальность

- Данные журналов событий – ***поток структурированных и сложно структурированных сообщений.***
  - Представимы в виде потока текстовых описаний событий с дополнительной служебной информацией (идентификатор устройства, тип события и т.д.).
- Обнаружение предотказного состояния системы сводится к решению задачи ***обнаружения аномалий в потоках данных***.
- Классические подходы машинного обучения
  - не позволяют добиться хороших результатов,
  - привязаны к формату журналов в конкретной системе,
  - не устойчивы к выбросам в обучающей выборке.
- Эффективными на данный момент являются ***устойчивые к выбросам подходы, основанные на архитектуре кодировщиков и методах нечёткой кластеризации.***

Постановка задачи

- Исследование и разработка методов глубокого обучения для обнаружения аномалий в данных журналов приложений для задач надёжности программных комплексов.

Содержание

- Введение.
- **Аналитический обзор решений задачи мониторинга надёжности программных комплексов на основе выявления аномалий в потоках данных.**
- Построение нейросетевого кластеризующего автокодировщика для мониторинга надёжности автоматизированных систем.
- Программная реализация и экспериментальная оценка предлагаемого решения.
- Заключение.

Обзор. Подзадачи

Сбор данных журналов

Сложно структурированные **текстовые описания** происходящих событий

Хранятся **в файлах либо базах данных.**

Обработка

Извлечение шаблонов типовых событий

с применением **регулярных выражений**.

Извлечение признаков

Построение **векторных представлений** блоков событий

Обнаружение аномалий

В последовательности блоков событий,

с применением **одноклассовой классификации**.

Существующие решения. Извлечение групповых признаков блоков событий.

- Классические подходы к векторизации текстовых данных (TF-IDF, мешок слов):
  - не учитывают последовательность событий.
- Подходы, основанные на глубоком обучении (Трансформеры):
  - учитывают последовательность событий,
  - показывают хорошие результаты,
  - обладают высокой вычислительной сложностью.
- ***Автокодировщики:***
  - строят информативное векторное представление для блоков событий,
  - в режиме одноклассовой классификации строят тождественное преобразование для нормальных блоков событий,
  - не устойчивы к выбросам.
- ***Глубокое обучение (свёрточные нейронные сети):***
  - показывают высокие результаты при работе с текстовыми данными,
  - можно использовать для построения признаков в автокодировщике,
  - позволяют получать решения, которые удобно распараллеливать.

Существующие решения. Обнаружение аномалий.

- Одноклассовая классификация.
- Одноклассовый SVM:
  - Модификация классического SVM
  - $\min_{w,\rho,\xi_i} \left[\frac{1}{2}\|w\|^2 + \frac{1}{vl}\sum_{i=1}^{l}\xi_i - \rho\right], \left(w * \Phi(x_i)\right) \geq \rho - \xi_i, \mathrm{i} = 1,..,\mathrm{l}; \xi_i \geq 0,$
  - x_i - feature vector, $v \in (0,1)$ – outliers percent, ρ – hyperplane shift, ξ_i - penalty variables,
  - $f(x) = sign(w * \Phi(x) - \rho);$
  - построение модели для большого объёма данных занимает очень много времени (сложность достигает $O(N^2)$),
  - не позволяет добиться хороших результатов.
- Нечёткая кластеризация:
  - построение нечётких кластеров данных с помощью различных модификаций алгоритма,
  - устойчивы к выбросам,
  - показывают хорошие результаты,
  - построение модели для большого объёма данных занимает очень много времени (сложность достигает $O(N^2)$),
  - нет возможности встраивания алгоритма в подходы глубокого обучения.
  - **Возможно встроить в архитектуру автокодировщика для построения нечёткого кластера полученных кодировщиком признаков.**

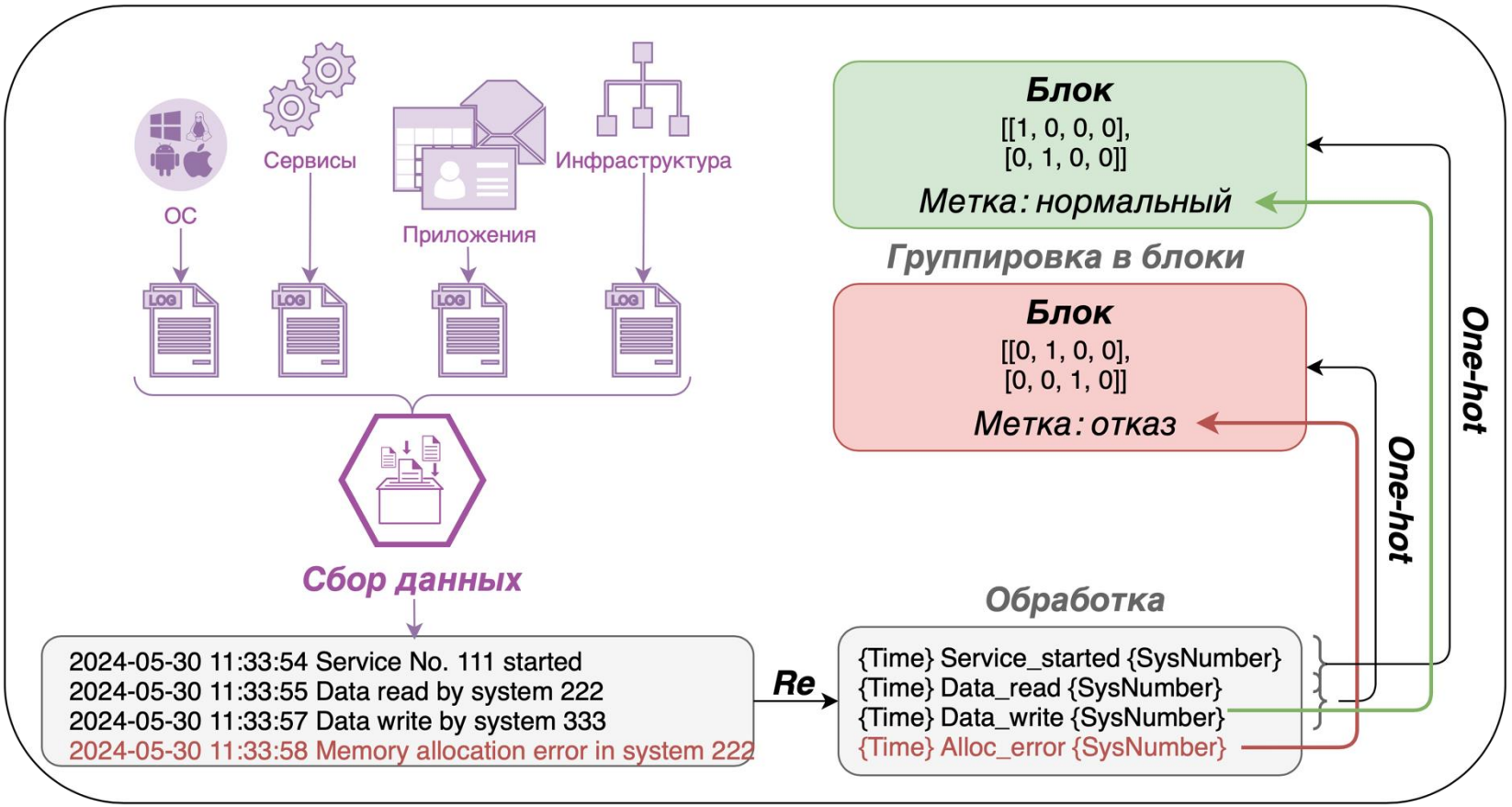
Обзор. Выводы.

- Предлагается построить единое решение, основанное на автокодировщике и самостоятельно реализованном слое, осуществляющем нечёткую кластеризацию.
- Необходимо провести экспериментальное исследование на выбранных наборах данных:
  - Выбрать лучшую конфигурацию решения.
  - Сравнить результаты с существующими подходами.
  - Оценить производительность предложенного решения.
- Необходимо разработать экспериментальный образец программного комплекса для решения задачи обнаружения предотказного состояния программных комплексов.

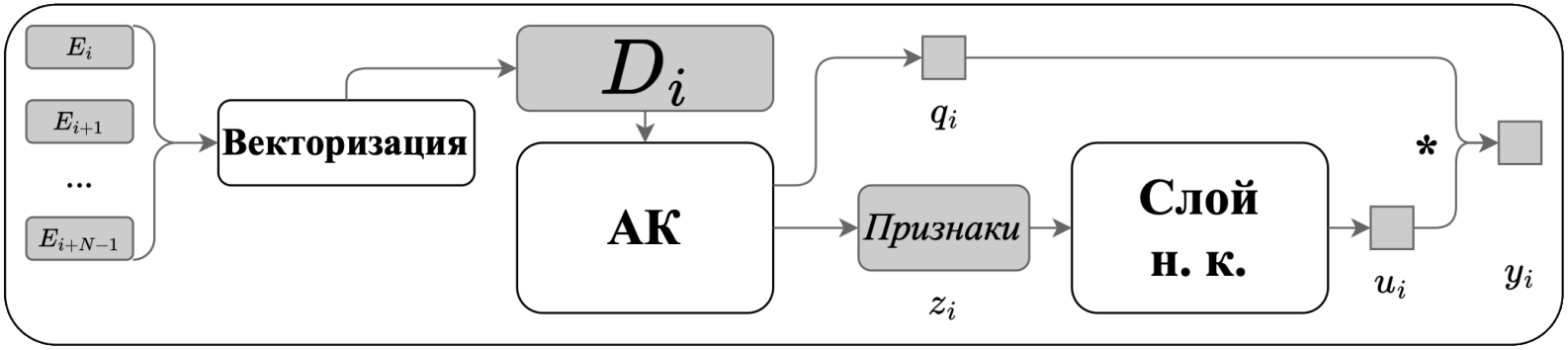
Содержание

- Введение.
- Аналитический обзор решений задачи мониторинга надёжности программных комплексов на основе выявления аномалий в потоках данных.
- **Построение нейросетевого кластеризующего автокодировщика для мониторинга надёжности программных комплексов.**
- Программная реализация и экспериментальная оценка предлагаемого решения.
- Заключение.
- Список литературы.

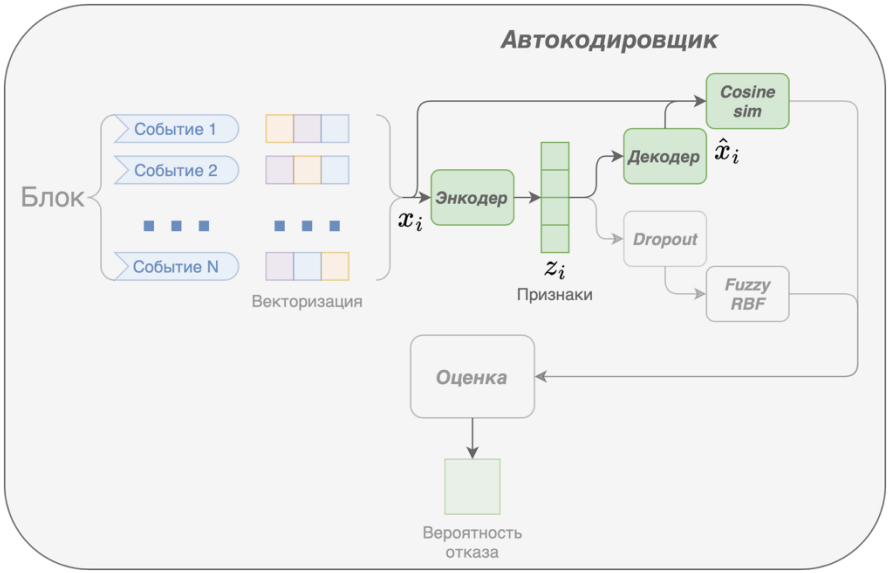
Сбор данных, обработка, векторизация и группировка

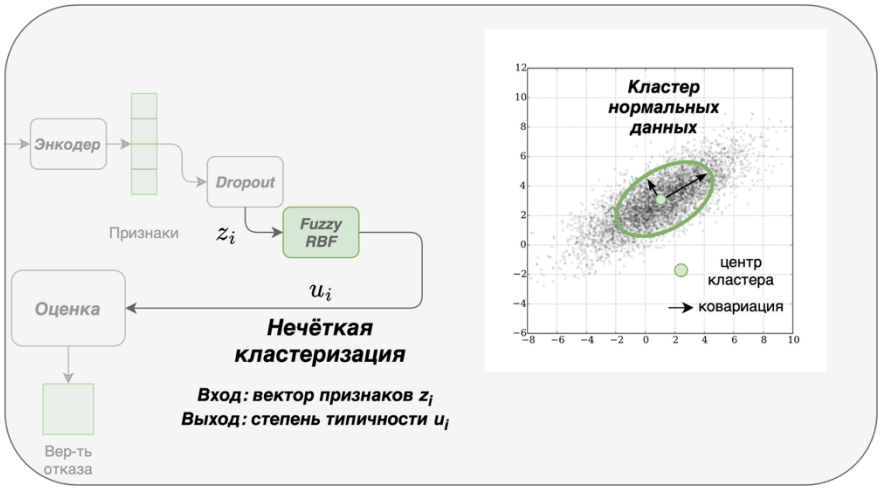


Предложенное решение. Автокодировщик с применением нечёткой кластеризации.



Извлечение признаков, нечёткая кластеризация.





Одноклассовая классификация:

$$\sum_{j=1}^{F} \mathcal{L}_{L2}^{j} + \mathcal{L}_{fuzzy} + \mathcal{L}_{out} \rightarrow \min_{a,u,\mu,C}$$

$\mathcal{L}_{L2}^{j}$ – ошибка L2-регуляизации j-го свёрточного слоя

$\mathcal{L}_{fuzzy}$– ошибка нечёткой кластеризации.

$\mathcal{L}_{out} = \sum_{i=1}(u_i \cos(x_i, \hat{x_i}) - 1)^2$ – ошибка реконструкции автокодировщика.

u_i – степень типичности.

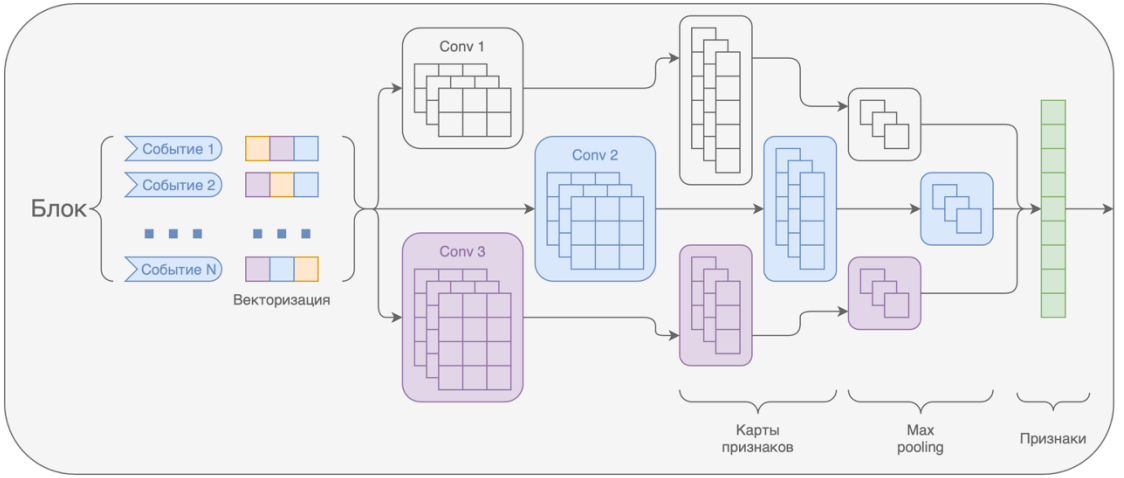
Алгоритм оптимизации – RMSProp.

Исследование предлагаемого решения. Конфигурация.

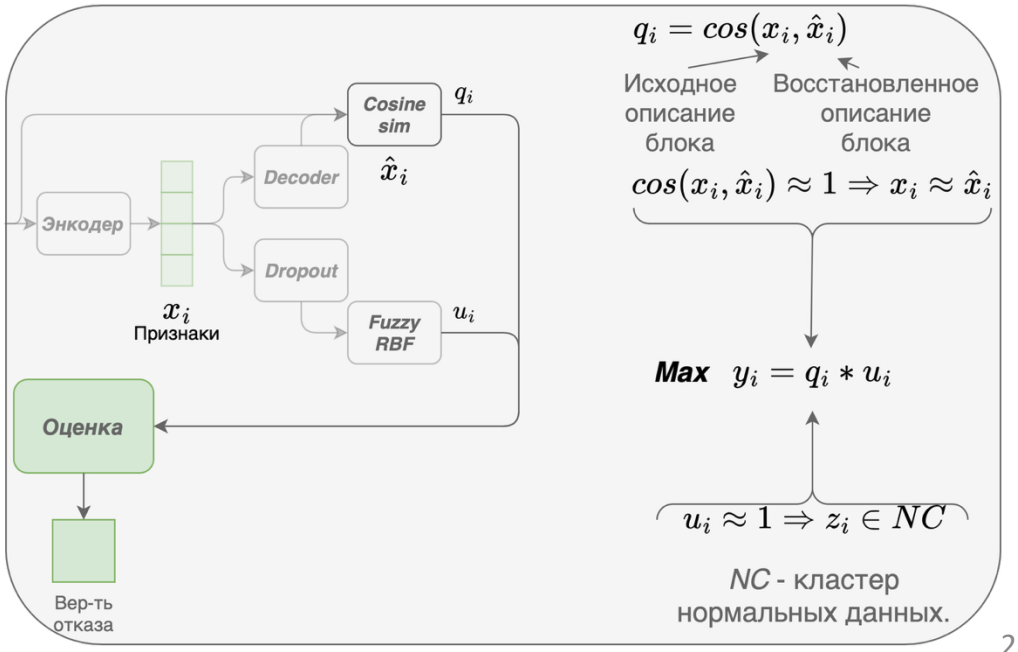
- Вид автокодировщика (симметричный, асимметричный, со свёрткой и без неё).
- Вид алгоритма нечёткой кластеризации:
  - Отсутствие кластеризации
  - Классический подход к нечёткой кластеризации:
    - $\min\limits_{U,a,\mu} \sum_i u_i^m * d_i(a) + \mu * \sum_i (1 - u_i\ )^m$,
    - $u_i = \left[1 + \left(\frac{d_i(a)}{\mu}\right)^{\frac{1}{m-1}}\right]^{-1}$
  - C-Means кластеризация Густавсона Касселя с использованием матрицы ковариаций и расстояния Махалонобиса:
    - $d_j = \|z_i - a\|_C^2 = (z_i - a)^T * C^{-1} * (z_i - a)$,
    - C – матрица ковариаций.
  - Вид кластеризации: однокластерная, многокластерная.
- Функции потерь: неробастные, робастные.
- Поиск лучшей конфигурации – на целевых наборах данных, с вычислением метрик ROC-AUC и PR-AUC.

Параллельный свёрточный автокодировщик.

Слои параллельной свёртки



Оценка аномальности



Предложенное решение. Результаты.

- Автокодировщик с применением нечёткой кластеризацией признаков с применением методов C-Means кластеризации Густавсона-Кесселя.
- Экспериментальное исследование предложенного решения:
  - Определение и исследование модификаций предложенного решения на выбранных наборах данных.
  - Построение программной реализации предложенного решения, оценка производительности и исследование технических средств, оптимизирующих время работы предложенного решения.

Содержание

- Введение.
- Аналитический обзор решений задачи мониторинга надёжности программных комплексов на основе выявления аномалий в потоках данных.
- Построение нейросетевого кластеризующего автокодировщика для мониторинга надёжности автоматизированных систем.
- **Программная реализация и экспериментальная оценка предлагаемого решения.**
- Заключение.
- Список литературы.

Экспериментальное исследование и программная реализация. Цели.

- Оценка пригодности для решения задачи (значения метрик больше 0.85).
- Сравнение предлагаемого решения с существующими подходами.
- Описание программной реализации предложенного решения.
- Исследование времени обучения и применения построенного решения.

Используемые наборы данных[1].

Данные о работе HDFS:

- Система работает на 203 узлах Amazon EC2;
- ~ 11 000 000 событий;
- ~ 16 000 отказов;
- Ручная разметка аномальных блоков событий;
- Примеры отказов: machine down, network disconnection, data read/write error.

Данные о работе BGL (BlueGene/L):

- ~ 5 000 000 событий;
- ~ 21 000 отказов;
- Ручная разметка аномальных событий.
- Примеры отказов: fan module error, memory error.

HPC

- Данные о работе System 20 на кластере HPC;
- 49 узлов, 6152 ядра, 128 Гб на каждый узел;
- ~ 450 тыс. событий;
- ~ 100 тыс. отказов;
- Ручная разметка отказов;
- Примеры отказов: server file system failure, SRM console error.

Экспериментальная оценка.

- Метрика качества: ROC AUC.
- Оценка распределения метрики:
  - разбиение тестовой выборки на подвыборки,
  - оценка медианы, Q1 и Q3 для значений метрик на подвыборках.
- Рассматриваемые алгоритмы: одноклассовый SVM, классический Fuzzy, LogBERT (трансформер).
- Подбор гиперпараметров – на валидационной выборке.

Сравнение с существующими подходами

Набор данных	Подход	ROC-AUC			
		Медиана	*Q1*	*Q3*	*Среднее*
HDFS1	OC-SVM	0.833	0.826	0.837	0.83
	Fuzzy	0.858	0.851	0.862	0.85
	LogBERT	*0.9*	*0.894*	*0.907*	*0.9*
	SymCNNAE	***0.975***	***0.971***	***0.974***	***0.97***
BGL	OC-SVM	0.783	0.779	0.789	0.78
	Fuzzy	0.834	0.821	0.845	0.82
	LogBERT	*0.908*	*0.901*	*0.912*	*0.9*
	SymCNNAE	***0.973***	***0.962***	***0.978***	***0.975***
HPC	OC-SVM	0.953	0.95	0.955	0.95
	Fuzzy	0.947	0.945	0.949	0.95
	LogBERT	*0.962*	*0.96*	*0.968*	*0.96*
	SymCNNAE	***0.968***	***0.964***	***0.969***	***0.968***

Программная реализация. Особенности

- Языки программирования/разметки:
  - Python 3.9 (NumPy, SciPy, Pandas, matplotlib, scikit-learn, Keras Django).
    - Сумарный объем кода: порядка 6500 строк.
  - HTML
    - Суммарный объем кода: порядка 400 строк.
- ОС Ubuntu 20.04.4 с ядром Linux версии 5.4.0-153-generic.
- ЦП AMD EPYC 7532 32.
- 256 Гб оперативной памяти типа DDR4.
- Построение, обучение и применение моделей: NVIDIA RTX A6000 и NVIDIA RTX A100 с 48 Гб оперативной памяти.

Экспериментальная оценка времени обучения (на 1 эпоху)*

L	1xA6000	1xA100	2xA100	4xA100	8xA100
1	13.8	2.27	**2.24**	2.62	3.99
2	27.9	3.44	2.56	**2.46**	3.38
5	68.52	7.59	4.75	**3.8**	4.39
10	137.81	14.59	8.58	6.11	**6.05**
50	682.17	10.48	38.59	23.56	**18.75**
100	1365.4	152	75	42	**27.52**

***Для экспериментов с несколькими GPU используется стратегия зеркалирования.**
Размер блока N=20, размер векторного представления одного блока M=100.
L – число событий (в тысячах).
Время указывается в секундах, усреднённое по 100 независимым запускам.

Экспериментальная оценка. Выводы.

- Предлагаемое решение (SymCNNAE) удовлетворяет критерию пригодности:
  - На всех наборах данных метрики превосходят 0.9
- SymCNNAE показывает лучший результат по сравнению с существующими подходами.
- Разброс результатов (средняя разница Q3-Q1) не превосходит 0.06.
- SymCNNAE показывает достаточно быструю скорость обучения и применения даже на одной видеокарте A6000 (~3.8 часов на 10 эпох обучения):
  - Использование стратегии зеркалирования позволяет ускорить обучение до 4.5 минут на 8 видеокартах A100.
- Время применения алгоритма на 1000 событиях занимает 1.5 секунды (на видеокарте A6000).

Результаты

- Предложен метод мониторинга надёжности программных систем:
  - построение признаков: симметричный свёрточный автокодировщик
    - эффективный способ построения признаков (параллельная свёртка),
    - минимизация потерь при сжатии (автокодировщик);
  - обнаружение предотказного состояния – слой нечёткой кластеризации:
    - оценка степени типичности,
    - устойчивость к выбросам в обучающей выборке.
- Разработана архитектура, реализован экспериментальный образец программного комплекса мониторинга надёжности программных систем:
  - значение целевых метрик превосходит 0.9,

Спасибо за внимание!

Результаты (2)

- Результаты опубликованы в 12 печатных работах, а также докладывались на 12 научных конференциях.
- Предложенное решение представлялось на конференции Lake Baikal Summit 2024:
  - Петровский М.И., Машечкин И.В., ***Горохов О.Е.***, Васильев Ю.А. Machine learning methods to monitor, analyze, and predict computer systems operating behavior.
- 1 статья опубликована в журнале, входящем в ***Q1***, индексируемом в Scopus, WoS и др.:
  - ***Gorokhov O.,*** Petrovskiy M., Mashechkin I., Kazachuk M. Fuzzy CNN Autoencoder for Unsupervised Anomaly Detection in Log Data // Mathematics. Special Issue "Mathematical Modeling, Optimization and Machine Learning, 2nd Edition".

Результаты (3)

- 4 статьи индексируются в WoS, Scopus или RSCI:
  - ***Gorokhov O.,*** Petrovskiy M., Mashechkin I. Convolutional neural networks for unsupervised anomaly detection in text data //International Conference on Intelligent Data Engineering and Automated Learning. – Cham : Springer International Publishing, 2017. – С. 500-507.
  - Kazachuk M., Petrovskiy M., Mashechkin I., ***Gorokhov O.*** Novelty Detection Using Elliptical Fuzzy Clustering in a Reproducing Kernel Hilbert Space //Intelligent Data Engineering and Automated Learning–IDEAL 2018: 19th International Conference, Madrid, Spain, November 21–23, 2018, Proceedings, Part II 19. – Springer International Publishing, 2018. – С. 221-232.
  - Казачук М. А., Петровский М. И., Машечкин И. В., ***Горохов О. Е.*** Методы поиска исключений в потоках сложноструктурированных данных //Вестник Московского университета. Серия 15. Вычислительная математика и кибернетика. – 2019. – №. 3. – С. 17-28.
  - ***Gorokhov O.E.,*** Kazachuk M.A., Mashechkin I.V., Petrovskiy M.I. Parallel Autoencoder for Unsupervised Anomaly Detection in Large Multimodal User Behavior Data //Lobachevskii Journal of Mathematics, 2025. – С. 3647-3661.

Предлагаемое решение. Оценка степени типичности (3).

- $u_j = \left[1 + \left(\frac{d_j(a)}{\mu}\right)^{\frac{1}{m-1}}\right]^{-1}$,
- $d_j = \|z_i - a\|_C^2 = (\Phi(z_i) - a)^T * C^{-1} * (\Phi(z_i) - a)$,
  - расстояние Махалонобиса до центра кластера,
- $\Phi(x)$: $\mathbb{R}^S \to H$
  - ядровая функция преобразования пространства признаков в некоторое новое пространство H большей размерности.
- m – степень нечёткости.

- Обучаемые параметры:
  - $C > 0$ – матрица ковариаций,
  - $\mu > 0$ – радиус кластера, для которого степень типичности равна 0.5,
  - a – центр построенного кластера нормальных данных.

Предлагаемое решение. Регуляризация.

Цели:

- избежание переобучения,
- нормализация признаков.

Используемые подходы:

- L2-регуляризация свёрточных слоёв;
- Использование подходов, основанных на само-нормализующихся сетях[1]:
  - использование функции SELU в свёрточных слоях,
  - AlphaDropout слой перед слоем нечёткой кластеризации.

*текст

Предлагаемое решение. Обучение.

- Решение задачи одноклассовой классификации,
- $$\begin{cases} \min\limits_{w,b,a,C,\mu} \mathcal{L}(w,b,a,C,\mu), \\ \mathcal{L}(w,b,a,C,\mu) = \sum_{i=1}^{F} \mathcal{L}_{L2}^{i} + \mathcal{L}_{fuzzy} + \mathcal{L}_{out}, \\ C > 0, \mu > 0 \end{cases}$$
- $\mathcal{L}_{fuzzy} = \sum_{i=1}^{V} (u_i * d_i(a))$,
- $\mathcal{L}_{out} = \frac{1}{V} \sum_{i=1}^{V} (y_i - 1)^2$,
- решение задачи оптимизации – алгоритмом RMSProp,
- разбиение обучающей выборки на V батчей (фрагментов),
- F – число слоёв в параллельной свёртке.

Содержание

- Введение.
- Аналитический обзор решения поставленной задачи.
- Построение модели обнаружения предотказного состояния системы.
- Экспериментальная оценка предлагаемого решения.
- Программная реализация.
- Заключение.
- Список литературы.

Сравнение с существующими подходами

Набор данных	Подход	ROC AUC			PR AUC		
		Медиана	Q1	Q3	Медиана	Q1	Q3
HDFS1	OCSVC	0.833	0.826	0.837	0.828	0.821	0.832
	Fuzzy	0.858	0.851	0.862	0.853	0.847	0.856
	LogBERT	*0.9*	*0.894*	*0.907*	*0.901*	*0.898*	*0.911*
	FuzzyCNN	***0.973***	***0.971***	***0.974***	***0.97***	***0.969***	***0.972***
BGL	OCSVC	0.783	0.779	0.789	0.778	0.772	0.783
	Fuzzy	0.834	0.821	0.845	0.827	0.82	0.83
	LogBERT	*0.908*	*0.901*	*0.912*	*0.898*	*0.893*	*0.901*
	FuzzyCNN	***0.939***	***0.934***	***0.94***	***0.921***	***0.916***	***0.926***

Исследование устойчивости к выбросам

Набор данных	Процент выбросов	ROC AUC			PR AUC		
		Медиана	Q1	Q3	Медиана	Q1	Q3
HDFS1	20%	0.959	0.96	0.963	0.95	0.953	0.963
	50%	0.937	0.93	0.94	0.944	0.94	0.949
BGL	20%	0.846	0.84	0.85	0.872	0.871	0.874
	50%	0.805	0.8	0.81	0.845	0.843	0.848

Содержание

- Введение.
- Аналитический обзор решения поставленной задачи.
- Построение модели обнаружения предотказного состояния системы.
- Экспериментальная оценка предлагаемого решения.
- Программная реализация.
- Заключение.
- Список литературы.

*текст

Примеры обнаружения предотказного состояния (1)

- Наличие критической ошибки:
  - RAS KERNEL INFO critical input interrupt warning for torus y wire
  - RAS KERNEL INFO critical input interrupt warning for torus z wire
  - RAS KERNEL INFO critical input interrupt warning for torus y wire
  - RAS KERNEL INFO critical input interrupt warning for torus z wire
  - ***RAS KERNEL FATAL external input interrupt torus sender x retransmission error was corrected***
  - ***Вероятность отказа: 0.9824644.***
- Блоки, где система обнаружила ошибки и смогла их исправить, но произошёл отказ:
  - RAS KERNEL INFO ciod generated core files for program
  - RAS KERNEL INFO instruction cache parity error corrected
  - RAS KERNEL INFO torus receiver x input pipe errors dcr detected and corrected
  - RAS KERNEL INFO torus receiver z input pipe errors dcr detected and corrected
  - RAS KERNEL INFO torus receiver y input pipe errors dcr detected and corrected
  - ***Вероятность отказа: 0.98334394.***

Примеры обнаружения предотказного состояния (2)

- Блок, не содержащий отказ:
  - RAS KERNEL INFO microseconds spent in the rbs signal handler…
  - RAS KERNEL INFO CE sym at mask
  - RAS KERNEL INFO total interrupts ***critical input interrupts***…
  - RAS KERNEL INFO microseconds spent in the rbs signal handler…
  - RAS KERNEL INFO total interrupts ***critical input interrupts***…
  - ***Вероятность отказа: 0.2430820999999996.***

Содержание

- Введение.
- Аналитический обзор решения поставленной задачи.
- Построение модели обнаружения предотказного состояния системы.
- Экспериментальная оценка предлагаемого решения.
- Программная реализация.
- Заключение.
- Список литературы.

*текст