DOI 10.37882/2223-2966.2024.06.33

МОДЕЛЬ СИСТЕМЫ ОБНАРУЖЕНИЯ ДЕФЕКТОВ ПРОГРАММНОЙ СРЕДЫ НА ОСНОВЕ ГЛУБОКОГО ОБУЧЕНИЯ С НАИБОЛЕЕ ПОДХОДЯЩИМИ ГИПЕРПАРАМЕТРАМИ

MODEL OF A SOFTWARE ENVIRONMENT DEFECT DETECTION SYSTEM BASED ON DEEP LEARNING WITH THE MOST SUITABLE HYPERPARAMETERS

D. Potapov
S. Kornienko

Summary. This article discusses methods for detecting errors in software code, lists existing tools for searching for defects in the software environment, focuses on a way to find and predict the appearance of software code defects using machine learning, lists well-known scientific works on finding errors and anomalies in the software environment, considers a typical scheme of artificial neural network operation in training A hybrid model of a software environment defect detection system based on deep learning with the most appropriate hyperparameters is proposed, which is based on machine learning technology with hyperparameters calculated using the hierarchy analysis method.

Keywords: machine learning, software environment defects, hyperparameters, hybrid model, hierarchy analysis method.

Потапов Дмитрий Артёмович
Аспирант, Петербургский государственный университет путей сообщения Императора Александра I;
главный специалист. Дирекция информационной безопасности ПАО Банк «Санкт-Петербург»
Корниенко Светлана Владимировна
кандидат технических наук, доцент, Петербургский государственный университет путей сообщения Императора Александра I
89025610510@mail.ru

Аннотация. В данной статье рассмотрены методы обнаружения ошибок в программном коде, перечислены существующие инструменты для поиска дефектов в программной среде, акцент сделан на способ нахождения и прогнозирования появления дефектов программного кода с использованием машинного обучения, перечислены известные научные работы по поиску ошибок и аномалий в программной среде, рассмотрена типовая схема работы искусственной нейронной сети при обучении, предложена гибридная модель системы обнаружения дефектов программной среды на основе глубокого обучения с наиболее подходящими гиперпараметрами, которая основана на технологии машинного обучения с гиперпараметрами, вычисленными по методу анализа иерархий.

Ключевые слова: машинное обучение, дефекты программной среды, гиперпараметры, гибридная модель, метод анализа иерархий.

Развитие в информационной сфере повлекло за собой появление современных технических устройств, облегчающих жизнь человечеству. Современные смарт технологии, имплантируемые медицинские устройства, автомобили с автопилотом и множество других изобретений ежедневно используют программную среду, которая требует наличие и совершенствование программного обеспечения. Существует множество языков программирования и средств для помощи в написании программного кода, а также его корректировки в целях адекватного использования. Несмотря на существующее многообразие методов и способов прогнозирования появления и выявления существующих ошибок в программном коде, исключить полностью вероятность появления дефекта в программной среде нет. Автоматизация процесса поиска ошибок программного кода экономит время на написание программы и повышает ее качество.

Ошибки (баги) могут возникать из-за различных причин, таких как ошибки в коде программы, некорректное взаимодействие компонентов, проблемы с аппаратным оборудованием или чрезвычайные происшествия. Преодолеть эти проблемы, можно за счет методов, используемых в сфере искусственного интеллекта.

Наиболее важное место в борьбе с появлением ошибок в программном коде занимает использование машинного обучения, которое может быть использовано для автоматического анализа и обнаружения ошибок и аномалий, что в свою очередь улучшает качество продукта и ускоряет процесс разработки с возможностью ранней локализации проблем на ранних стадиях разработки [8, 9]. Исходными данными для работы искусственной нейронной сети являются исходный код программы, логи событий, стеки вызовов и другие метрики.

Тема поиска ошибок в программной среде на основе машинного обучения в очень популярна, многие ученые трудились над развитием методов прогнозирования дефектов в программном обеспечении, использующие свои уникальные алгоритмы.

Существует несколько вариантов направлений машинного обучения, которые могут быть использованы

для обнаружения дефектов в программном обеспечении: классификация, кластеризация, извлечение признаков, обучение с учителем и без учителя, анализ временных рядов. Основными способами обнаружения дефектов программного обеспечения являются: тестирование программного обеспечения, использование инструментов для статического и динамического анализа кода, Peer code review, использование методов формальной верификации, мониторинг и отладка программы на этапе ее работы и сбор обратной связи от пользователей. В настоящее время существует ряд специальных инструментов, которые позволяют автоматизировать процесс тестирования и выявления дефектов, например: Pylint, SonarQube, ESLint, Checkstyle, Coverity.

Среди научных работ по применению машинного обучения для выявления ошибок в программном коде следует отметить зарубежные работы некоторых авторов в которых рассмотрены:

— работа алгоритмов классификации для поиска ошибок кода, таких как нейронные сети, наивный Байес, опорные вектора (SVM), линейная регрессия и K-ближайший сосед подробно описаны в [5];
— процесс используемый в SVM для классификации дефектов, в котором особое внимание уделяется предварительной обработке входных данных, с учетом применения байесовской классификации в сочетании с особенностью выбора признаков для эффективного прогнозирования описан в научном труде [4].

Также ранее были описаны гибридные подходы для оптимального исследования и кластеризации реальных наборов данных, так в [7] разработана модель, использующая метод оптимизации муравьиной колонии для прогнозирования появления ошибок программного обеспечения. В другом исследовании [6] была предложена гибридная модель машины опорных векторов в сочетании с алгоритмом поиска летучих мышей. В [10] представлена гибридная модель с использованием нейронной сети на основе метода пакетирования и генетического алгоритма для прогнозирования ошибки программного обеспечения для повышения производительности.

В данной работе используется термин «гиперпараметры» в нейронных сетях, под которыми понимаются параметры, которые не обучаются моделью, а задаются заранее и влияют на процесс обучения и структуру нейронной сети.

Предлагаемая гибридная модель представляет собой совокупность алгоритма определения наиболее лучших гиперпараметров на основе метода анализа иерархий и машинного обучения.

Для определения гиперпараметров нейронной сети подходят различные методы экспертных оценок: инициализация весов, ручная настройка, оптимизация параметров, адаптивные методы обучения такие как методы обратного распространения ошибки или методы оптимизации градиентного спуска.

Экспертные методы могут быть очень полезны при обучении нейронных сетей, особенно если у нас есть ограниченное количество данных или требуется решить сложную задачу, требующую дополнительные знания и опыт.

Существуют многокритериальные методы принятия управленческих решений такие как: метод простого аддитивного взвешивания, метод идеальной точки, методы исключения и выбора, отражающего реальность, метод анализа иерархий, предложенный Т. Саати [3]. Данный метод универсальный и может применяться для решения различных задач, учитывая многие критерии. Появление ошибок у данного метода минимальна за счет применения индекса согласованности между мнениями экспертов и весовыми коэффициентами каждого эксперта. Поэтому для выбора наиболее подходящих гиперпараметров нейронной сети предлагается применить метод анализа иерархий.

В результате проведенного анализа источников по функционированию искусственных нейронных сетей [1] в качестве критериев, по которым необходимо выбрать наиболее приоритетные гиперпараметры, рекомендуется использовать следующие:

1. Архитектура нейронной сети: выбор количества слоев, нейронов и их типов (например, плотные слои, сверточные слои, рекуррентные слои) влияет на способность сети к обучению и предсказанию.
2. Функции активации: выбор подходящей функции активации для каждого слоя в сети может улучшить ее способность к обучению и предсказанию.
3. Параметры обучения: скорость обучения, количество эпох, размер мини-батча и другие параметры обучения могут влиять на скорость сходимости и качество модели.
4. Регуляризация: использование методов регуляризации, таких как dropout, L1/L2 регуляризация, может помочь в предотвращении переобучения и улучшении обобщающей способности сети.
5. Выбор оптимизатора для обучения сети (например, Adam, SGD, RMSprop) также влияет на скорость сходимости и качество модели.

Для успешной реализации необходимо привлекать в качестве экспертов наиболее подготовленных специалистов, имеющих опыт в сфере проектирования и тестирования нейронных сетей, при наличии пяти выбранных основных критериев количество экспертов должно быть не менее 10 [2].

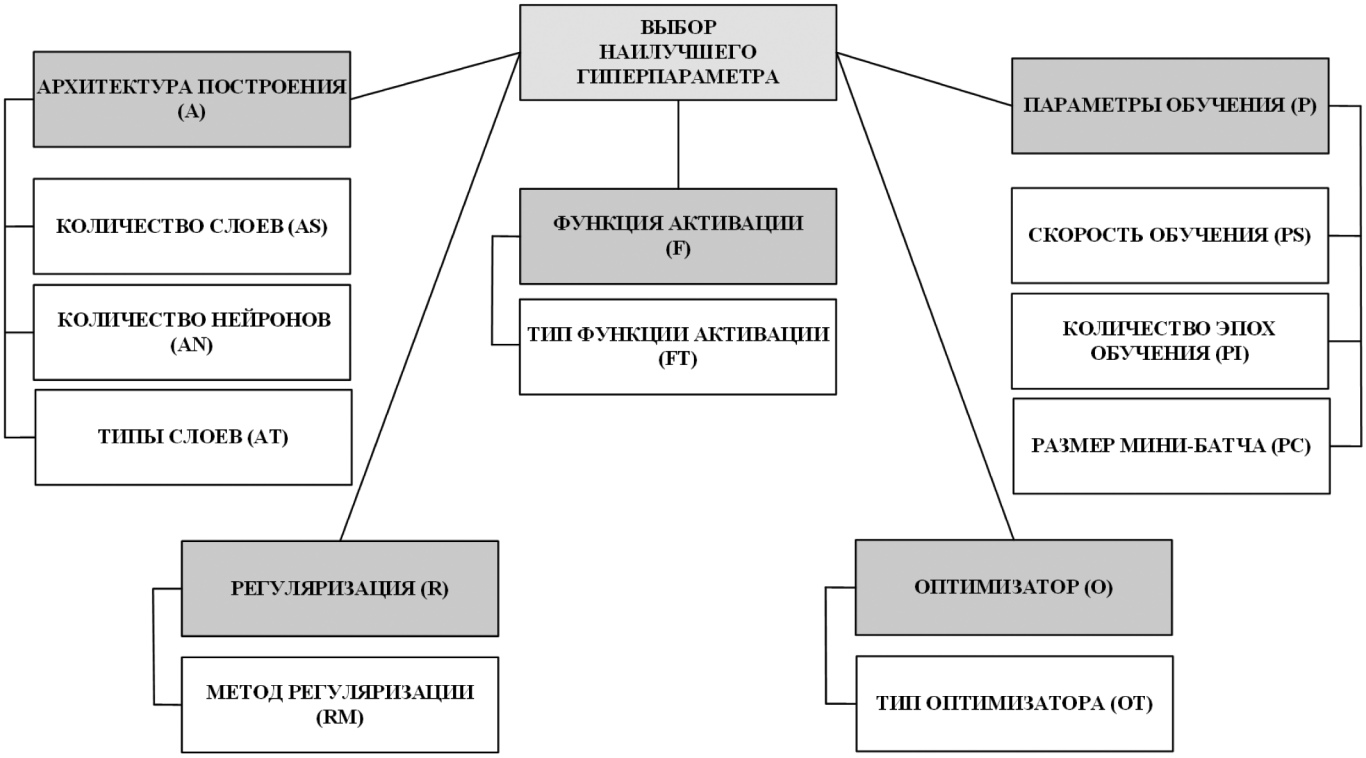

Рис. 1. Критерии для выбора наиболее лучшего гиперпараметра при работе нейронной сети

Основной задачей метода является нахождение собственного вектора с определением наибольшего значения. Определение вектора производят методом попарного сравнения критериев (рисунок 1).

Цель в данном случае — выбор наиболее подходящего гиперпараметра.

Критериями 1 уровня являются архитектура построения нейросети (*A*), функция активации (*F*), параметры обучения (*P*), регуляризация (*R*), оптимизатор (*O*). Критериями 2 уровня являются количество слоев (*AS*), количество нейронов (*AN*), типы слоев (*AT*), тип функции активации (*FT*), скорость обучения (*PS*), количество эпох обучения (*PI*), размер мини-батча (*PC*), метод регуляризации (*RM*), тип оптимизатора (*OT*).

На 1 этапе производится декомпозиция проблемы принятия решений с выделением главных целей, подцелей и различных целевых функций. На 2 этапе определяется значимость критериев первого уровня. Для этого составляем квадратную матрицу парных сравнений *M* размера $k \times k$, в нашем случае $k = 5$ (критериев первого уровня 5) [2].

$$M = \begin{pmatrix} m_{11} & m_{12} & m_{13} & m_{14} & m_{15} \\ m_{21} & m_{22} & m_{23} & m_{24} & m_{25} \\ m_{31} & m_{32} & m_{33} & m_{34} & m_{35} \\ m_{41} & m_{42} & m_{43} & m_{44} & m_{45} \\ m_{51} & m_{52} & m_{53} & m_{54} & m_{55} \end{pmatrix}$$

Элементы матрицы mij являются соотношениями весов W_{ij} (значимость критерия от 0 до 9) к соответствующим элементам в матрице (от 1 до 5).

$$m_{ij} = \frac{W_i}{W_j}$$

Если веса заранее неизвестны, то попарные сравнения элементов производятся с использованием суждений, численно оцениваемых по фундаментальной шкале предпочтений. Далее вычисляем локальный вектор приоритетов, который будет определять приоритет того или иного критерия при выборе гиперпараметров. Далее для каждой строки матрицы определяем геометрическое среднее элементов.

$$g_i = \sqrt[k]{\prod_{j=1}^{k} m_{ij}},$$

где k — это количество рассматриваемых элементов.

Далее вычисляются компоненты нормализованного вектора приоритетов.

$$y_i^{\text{н}} = \frac{g_i}{\sum_{i=1}^{k} g_i}$$

После этого определяется однородность суждений через индекс согласованности и сравнивается со значением случайной согласованности по фундаментальной шкале для заданного значения (значение не должно быть более 10 % иначе необходимо проводить заново присваивание весов критериям экспертами).

Глобальные приоритеты для каждой альтернативы a_i (элемент множества гиперпараметров) находятся по формуле, указанной ниже [2].

$$v(a_i) = \sum_{i=1}^{n} y_i^{H'} \times y_i^{H''}$$

где $y_i^{H'}$ — это компоненты нормализованного вектора приоритетов первого уровня иерархии,

$y_i^{H''}$— это компоненты нормализованного вектора приоритетов второго уровня иерархии.

Каждый раз после принятия решения о необходимости настройки гиперпараметров происходит оценивание по указанным критериям.

На 3 этапе переходим к оценке наилучшего гиперпараметра. Если значения критериев строго определены и составлен локальный вектор приоритетов, то матрица парных сравнений для этого гиперпараметра не составляется, а приоритетность определяется компонентами нормализованного вектора для каждого гиперпараметра.

Если по какому-либо критерию значения слабо формализуемые, то оценка по этому критерию осуществляется составлением матрицы парных сравнений с получением локального вектора приоритетов.

Глобальный вектор приоритетов позволяет выбрать наиболее предпочтительное решение по выбору значений гиперпараметров для эффективного функционирования нейронной сети.

Модель системы обнаружения дефектов программной среды на основе глубокого обучения с наиболее подходящими гиперпараметрами (Модель СОДПС) представлена на рисунке 2.

Данная модель состоит из модуля поиска дефектов, модуля обобщения результатов поиска, модуля сравнения, модуля определения наилучших гиперпараметров и модуля фиксации. Главными функциями данных модулей являются:

- модуль поиска дефектов осуществляет заданный режим работы за счет гиперпараметров и помечает подозрительные участки подаваемого на него объекта в виде программного кода;
- модуль обобщения результатов поиска подготавливает формализованный результат работы нейронной сети, подходящий для работы функции потерь;
- модуль сравнения осуществляет сравнения результата работы нейронной сети с правильным ответом и определяет корректность настройки гиперпараметров нейронной сети (если ответ близок с правильным ответом, то он передается в модуль фиксации, а при большом расхождении происходит регулировка гиперпараметров);
- модуль определения наилучших гиперпараметров предназначен для осуществления регулировки работы нейронной сети с целью повышения точности ее работы, регулировка осуществляется на основе произведенных вычислений по методу анализа иерархий;
- модуль фиксации осуществляет запись полученного результата работы нейронной сети, а также его разницу от правильного ответа и готовит данные по состоянию гиперпараметров нейронной сети для формализованной справки с рекомендациями к обработанному программному коду.

Следует подчеркнуть, что реализация вычислений в модуле определения наилучших гиперпараметров может осуществляться с помощью программы с заранее заданными экспертами конфигурациями.

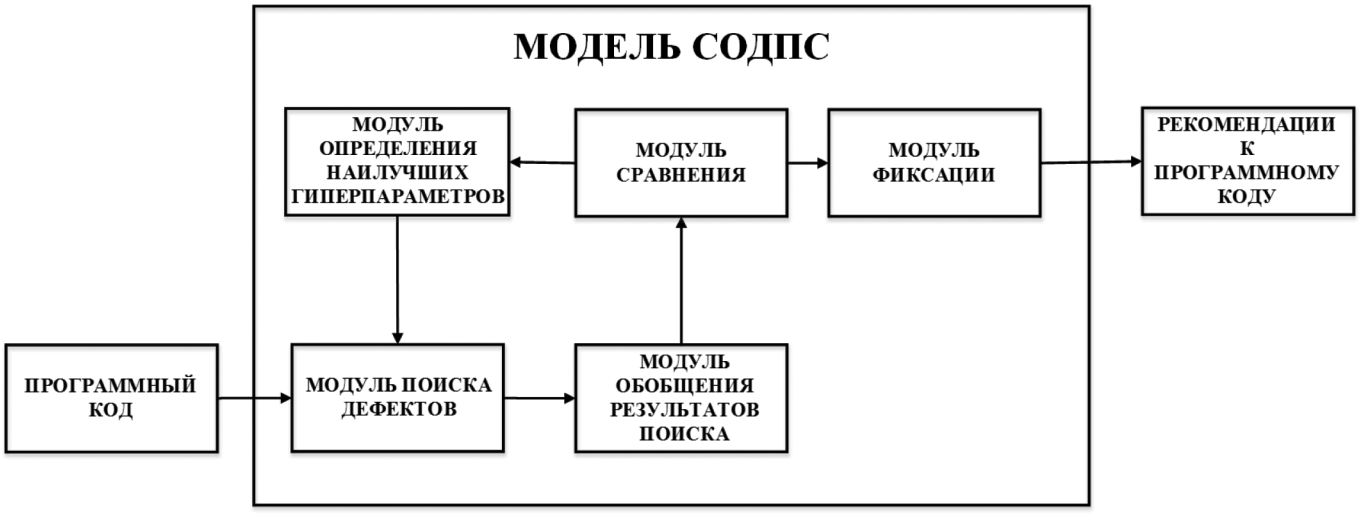

Рис. 2. Модель СОДПС

Разработанная модель СОДПС является концептуальной и может дополняться необходимыми модулями и блоками. Одним из перспективных направления по совершенствованию данной модели СОДПС является дополнение ее модулем по подготовке рекомендаций к обработанному программному коду, результатом работы которого будут являться возможные варианты исправления ошибок и варианты замены участков кода, определенными как аномальные.

За последние годы появилось много программных продуктов с возможностью раннего прогнозирования появления дефектов программного обеспечения и связанных с ними проблем, что говорит об актуальности исследования. В этой статье представлена гибридная модель по прогнозированию дефектов программного обеспечения с использованием технологий, применяемых в глубоких нейронных сетях в сочетании с методом анализа иерархии для поиска наилучших гиперпараметров. Глубокое машинное обучение и метод анализа иерархий могут использоваться вместе для оптимизации архитектуры нейронных сетей, настройки гиперпараметров или обучения моделей. Таким образом, комбинированная связка глубокого машинного обучения с методом анализа иерархий может помочь улучшить качество и эффективность обучения моделей и соответственно более качественно находить, и прогнозировать появление дефектов в программной среде. Дальнейшем исследованием для разработанной модели будет проведение эксперимента с целью сравнения эффективности функционирования нейронной сети.

ЛИТЕРАТУРА

1. Галушкин, А.И. Нейронные сети: основы теории / А.И. Галушкин. М.: Горячая линия-Телеком, 2017. 496 с.
2. Павлов, И.П. Алгоритм ранжирования событий безопасности по приоритетности в автоматизированной системе в защищенном исполнении на основе метода анализа иерархий / И.П. Павлов // Электронный сетевой политематический журнал «Научные труды КубГТУ». 2021. № 4. С. 67–78.
3. Саати Т. Принятие решений: метод анализа иерархий / Т. Саати; пер. с англ. Р.Г. Вачнадзе. М.: Радио и связь, 1993. 314 [1] с.
4. Gray, D., Bowes, D., Davey, N., Sun, Y., Christianson, B., Using the support vector machine as a classification method for software defect prediction with static code metrics. Engineering Applications of Neural Networks, pp. 223–234. Springer, Berlin, 2009.
5. Perreault, L., Berardinelli S., Izurieta C. and Sheppard J., Using Classifiers for Software Defect Detection. 26th International Conference on Software Engineering and Data Engineering (SEDE), 2017.
6. Rasneet K.C. and Iqbal S., Latest Research and Development on Software Testing Techniques and Tools, International Journal of Current Engineering and Technology, 4(4), 2014.
7. Azar, D., Vybihal, J.: An ant colony optimization algorithm to improve software quality prediction models: case of class stability.Inf. Softw. Technol. 53(4), 388–393,2011.
8. Puranika S., Deshpandea P., Chandrasekaran K., A Novel Machine Learning Approach for Bug Prediction, 6th International Conference on Advances In Computing & Communications, ICACC, 6–8 September 2016.
9. Menzies T., Turhan B., Bener A., Gay G., Cukic B., and JiangY., Implications of ceiling effects in defect predictors. In Proceedings of the 4th international workshop on Predictor models in software engineering, pp. 47–54. ACM, 2008.
10. Wahono, R.S., Herman, N.S., Ahmad, S.: Neural network parameter optimization based on genetic algorithm for software defect prediction. Adv. Sci. Lett. 20, 1951–1955 (2014).

Журнал «Современная наука: актуальные проблемы теории и практики»