МАШИННЕ НАВЧАННЯ ДЛЯ ВІЗУАЛІЗАЦІЇ МОДЕЛЕЙ АРХІТЕКТУР ПРОГРАМ НА МОБІЛЬНИХ ПРИСТРОЯХ / MACHINE LEARNING FOR THE SOFTWARE ARCHITECTURE MODELS VISUALISATION ON MOBILE DEVICES

Франків О.О. / Frankiv O.O.
Національний університет «Києво-Могилянська академія» / National University of Kyiv-Mohyla Academy
04070, м. Київ, вул. Г. Сковороди 2, каф. інформатики, тел. (044) 425 60 64
E-mail: o.frankiv@ukma.edu.ua, тел. (044) 425 60 64

In the context of creating the new tools for automated quality assurance it is important to enable convenient ways of tool integration into the development process while providing high quality interaction with the analysis results. In this work author extends the existing system for the software architecture visualization in the form of graph by proposing the novel specialized approach to graph placement in space. The solution improves the performance of the existing force-directed placement algorithm without negative aesthetic impact by providing the specialized heuristics. The machine learning techniques are used to reveal the implicit aspects of placements of the architecture models of the real software and provide the heuristics. Based on the performance of the new approach depending on graph size, we implemented the combined solution.

Розвиток технологій в галузі розробки програмного забезпечення створює постійну потребу в дослідженні нових способів перевірки якості продуктів. В той час як виявлення відносно простих вад не потребує значних затрат і може бути автоматизовано, аналіз архітектури програми є нетривіальним. З огляду на високу ціну такого аналізу у формі аудиту та високі ризики, пов’язані з несвоєчасним виявленням проблем на такого роду, пошук можливості прискорити та спростити процес перевірки залишається актуальним.

Раніше було запропоновано програмний комплекс, що дозволяє автоматично аналізувати архітектуру програм мовою Swift, будувати її візуальну модель у формі зваженого орієнтованого графа та відображати за допомогою спеціального застосунку-переглядача в просторі. Пізніше, у роботі [1] було запропоновано вдосконалення візуальної моделі, що збільшило її інформативність завдяки включенню частин класів, структур та переліків (а саме полів, методів, індексаторів, тощо) в загальну модель в явному вигляді. Це, з іншого боку, значно збільшило її розмір, що створило потребу в пошуку можливостей оптимізації алгоритму розміщення графу в просторі.

Окрім того серйозним недоліком системи є те, що аналіз та розміщення графа у просторі відбувається на комп’ютері, а перегляд — на мобільному пристрої. Це негативно впливає на гнучкість взаємодії користувача з візуальним представленням. Для того, щоб нівелювати цю проблему, важливо було запропонувати рішення, що буде працювати і на мобільних пристроях.

Проблема розміщення графів у просторі не є новою, однак вона має певні додаткові виклики, коли йдеться про візуалізацію архітектури програми. Рівень сприйняття певного розміщення графа залежить від відповідності цього розміщення певним критеріям естетичності, як-от відсутність накладання вузлів чи відсутність перетинів ребер. Найбільш високих показників естетичності дозволяють досягнути алгоритми, що базуються на фізичних моделях. Саме тому для візуалізації було обрано силовий алгоритм. В контексті цього алгоритму граф розглядають як фізичну систему, в якій вузли моделюють позитивними точковими зарядами, а ребра — пружинами. Точкові заряди відштовхуються за законом Кулона, а пружини, що з’єднують заряди, створюють протидію за законом Гука. Вважається, що розміщення графа досягає естетичності, коли воно відповідає розміщенню такої системи в стані спокою. Додатковою перевагою алгоритму є те, що він дозволяє легко працювати зі зваженими графами. В такому разі ваги ребер моделюються як жорсткості пружин, що моделюють ребра.

Серед недоліків силового алгоритму варто підкреслити нестабільність та високу часову складність.

Нестабільність полягає у невизначеності кількості кроків, які знадобляться для того, щоб система перейшла у стан спокою. Очевидно, що ця кількість сильно залежить від початкового положення графа. Для її зменшення зазвичай використовують випадкову евристику, тобто всі вузли графа розміщують в просторі випадковим чином, що працює краще, ніж якщо на початку роботи алгоритму всі вузли були б в одній точці. Існують підходи, що дозволяють побудувати більш інформовану евристику, однак вони часто не є достатньо універсальними.

Висока часова складність алгоритму ґрунтується на тому, що на кожній ітерації загального алгоритму для досягнення спокою системи потрібно обійти весь граф і перерахувати сили, що діють на кожен з елементів. Існують оптимізації, що дозволяють за певних умов перейти до обходу дерева замість графа, однак це має негативний вплив на естетичність отриманого результату.

Завдяки доступності спеціалізованого програмного та апаратного забезпечення для роботи з машинним навчанням, зокрема нейронними мережами, було розглянуто можливість використати існуючі рішення для розміщення графів. Хоча підходи, описані в [2], [3] та [4] дозволяють отримати естетичні представлення для певних задач, в нашому випадку їхнє використання обмеженим з точки зору універсальності, зокрема з огляду на потребу в тонкому налаштуванні чи окремих реалізаціях залежно від розмірів та особливостей будови графів.

Задля створення універсального підходу, що водночас дозволив би забезпечити такий же високий рівень естетичності, як і силовий алгоритм, автори розглянули можливість застосувати графові згорткові мережі [5] для неявної побудови спеціалізованої евристики для силового алгоритму. Використання шарів GCNConv також забезпечило повну сумісність з CoreML та дозволило застосування даного підходу на iOS пристроях. Наявність невипадкової евристики забезпечує стабільність алгоритму. Окрім того так можна отримати деяке покращення продуктивності за умови, що використання евристики зменшить кількість необхідних ітерацій I для досягнення стану спокою системи на стільки, що обчислення мережі потребуватиме менше часу, ніж потенційне обчислення ΔI ітерацій.

Для тренування власної мережі було автоматизовано збір даних: збір реальних проектів для конкретної технології зі сховища GitHub та побудову їхніх візуальних моделей. Результат побудови за допомогою класичного силового алгоритму взято за еталонний.

З огляду на описані в [5] результати роботи з шарами GCNConv з метою виявлення нюансів роботи з нашими даними автори підготували три споріднені мережі, що значно не відрізнялись. Перший шар мережі приймає на вхід індекс вузлів, ребер та їхніх ваг. Оскільки вузли наших графів не мають чітко визначених властивостей, для забезпечення сумісності властивістю вузла було визначено його степінь. Останній шар повертає набір тривимірних векторів, що позначають передбачені координати в просторі для кожного вузла. Розроблені мережі мали від 0 до 2 прихованих шарів. Відповідно модель дворівнева мережа L2 мала 0 прихованих шарів, а трирівнева L3 та чотирирівнева L4 мали по 1 і 2 приховані шари відповідно.

Для порівняння ефективності спеціалізованих евристик було проведено порівняння за кількістю ітерацій та часу, необхідного для розміщення графа.

Позитивним результатом стало забезпечення константної кількості ітерацій зовнішнього циклу алгоритму для всіх евристик.

Відповідно до отриманих даних також вдалося з’ясувати, що евристика обчислена за допомогою мережі L2 справді суттєво зменшує кількість ітерацій до 40% для невеликих графів з малою кількістю вузлів та ребер (до 50), L3 аналогічно працює для графів з кількостями вузлів та ребер від 50 до 300, L4 — для більших графів. В контексті затрат часу варто зауважити, що для графів з кількістю вузлів та ребер до 100 накладні витрати на обчислення евристики більші, ніж вона дозволяє зекономити,

зменшуючи кількість необхідних ітерацій. Відповідно, в такому разі більш ефективною є випадкова евристика, якщо стабільність алгоритму не є принциповою.

Спираючись на отримані результати було створено новий модуль для розміщення графів в системі візуалізації, що динамічно підбирає більш ефективну евристику залежно від розмірів графа. Цей модуль також сумісний з CoreML, а тому дозволив перенести обчислення на мобільну платформу для розширення можливостей взаємодії користувача з моделлю під час перегляду.

Важливо підкреслити, що процес створення таких мереж повністю автоматизовано, що дозволяє легко за потреби адаптувати рішення для інших технологій.

Пошук найбільш ефективних мереж для створення евристик для графів більших розмірів, а також можливості їх вдосконалення є предметом майбутніх досліджень.

Використані джерела:

1. Frankiv O., Glybovets M. Automated Visualization of Program Architecture Components for the Swift Language // Cybernetics and Systems Analysis. – 2024. – DOI: 10.1007/s10559-024-00737-9.
2. Grover A., Leskovec J. node2vec: Scalable Feature Learning for Networks // Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. – 2016. – P. 855–864. – DOI: 10.1145/2939672.2939754 .
3. Rahman M. K., Sujon M. H., Azad A. Force2Vec: Parallel Force-Directed Graph Embedding // 2020 IEEE International Conference on Data Mining (ICDM). – Sorrento, Italy, 2020. – P. 442–451. – DOI: 10.1109/ICDM50108.2020.00053.
4. Wang Y., Jin Z., Wang Q., Cui W., Ma T., Qu H. DeepDrawing: A Deep Learning Approach to Graph Drawing // IEEE Transactions on Visualization and Computer Graphics. – 2020. – Vol. 26, No. 1. – P. 676–686. – DOI: 10.1109/TVCG.2019.2934798.
5. Kipf T. N., Welling M. Semi-supervised classification with graph convolutional networks // arXiv preprint. – 2017. – arXiv:1609.02907. – Published as a conference paper at ICLR 2017. – DOI: 10.48550/arXiv.160 .

МЕТОДИ РЕГУЛЯРИЗАЦІЇ МОДЕЛЕЙ МАШИННОГО НАВЧАННЯ / METHODS OF REGULARIZATION O F MACHINE LEARNING MODELSч

Гордуз О.С. / Horduz O.S.

Національний університет «Києво-Могилянська академія» / National University of “Kyiv-Mohyla Academy”

54003, Миколаївська обл., м.Миколаїв, вул.Чкалова, 110А, кв.147. Тел. +380959000281

Email: alex.horduz@gmail.com

The given work is devoted to the problem of overfitting in machine learning models. Such models learn the training data rather than its patterns, and fail to generalize on the out-of-distribution data. While increasing the dataset size helps with the problem, in practice it’s hard or even impossible to gather enough labeled data. The weight decay method is described, which includes $L1$, $L2$ and $ElasticNet$ regularization, along with their advantages and disadvantages. Dropout is another regularization method, which is applied to deep neural networks. It works by randomly turning off some neurons during training, allowing, at the inference time, for the ensemble effect produced by a single model. Additionally, the work highlights other regularization methods, such as tuning hyperparameters for decision tree models, using smoothed labels, or adding noise to features.

Важливою проблемою, яка з’являється при роботі з моделями машинного навчання, є перенавчання (overfit). Воно відбувається, коли модель вивчає навчальну вибірку, а не шаблони та закономірності. Такі моделі погано показують себе на out-of-distribution даних (тих, що не було у тренувальній вибірці), в той час як точність їх на тренувальних даних є високою. Одним