Министерство науки и высшего образования
Российской Федерации

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Тверской государственный университет»

СТУДЕНЧЕСКАЯ КОНФЕРЕНЦИЯ ФАКУЛЬТЕТА ПМиК. СБОРНИК ТРУДОВ

Тверь
17–28 апреля 2023 г.

Под редакцией Б. Н. Карлова

Тверь 2023

УДК 004 + 303 + 330 + 519
ББК 22.18 + 65.05я43
С88

Тверской государственный университет
Факультет прикладной математики и кибернетики

С88 Студенческая конференция факультета ПМиК. Сборник трудов. Тверь, 17–28 апреля 2023 г. / под ред. Б. Н. Карлова. — Тверь : ТвГУ, 2023. — 120 с.

Сборник содержит научные работы, представленные на студенческой конференции, проводимой факультетом прикладной математики и кибернетики Тверского государственного университета. Сборник предназначен для научных работников, аспирантов и студентов старших курсов.

УДК 004 + 303 + 330 + 519
ББК 22.18 + 65.05я43

Содержание

УДК 330.322 + 519.832

Теоретико-игровая модель формирования портфеля многопериодных инвестиций по критерию минимального риска

Баяндина П. В.

Тверской государственный университет

Аннотация. В статье рассматривается задача формирования портфеля многопериодных инвестиционных проектов в условиях неопределнности процентных ставок по заимствованиям и инвестициям. Показана возможнсть моделирования данной задачи антагонистической игрой между инвестором и рынком капитала. Конкретизированы компоненты данной игры, метод ее решения и связь решения игры с решением исходной задачи. Применение модели демонстрируется на тестовом примере.

Введение

Одной из проблем в задачах анализа многопериодных инвестиционных проектов является неопределенность прогнозных процентных ставок по дополняющим инвестициям и по заимствованиям в процессе реализациии проектов. Известно [2], что в таких условиях целесообразным является диверсификация инвестиционных вложений. Вопросы диверсификации при стохастической неопределенности в настоящее время проработаны недостаточно, поэтому рассматриваемая в работе задача является актуальной.

Целью работы является повышение обоснованности формирования портфелей многопериодных инвестиционных проектов, с учетом интервальных оценок процентных ставок по инвестированию и заимствованию в плановые периоды реализации проектов.

Достижение цели основано на обосновании возможности представления задачи формирования портфеля многопериодных инвестиций моделью бесконечной антагонистической игры [3]. Анализ данной модели проводится путем аппроксимации бесконечной антагонистической игры ее конечным аналогом. Решение конечной

антагонистической игры основано на ее сведении к паре двойственных задач линейного программирования. Связь решения последних с решением игры и с решением исходной задачи обеспечивает формирования оптимального инвестиционного портфеля.

1. Постановка задачи

Для формулировки задачи ввдем следующие обозначения [2]

- $I = \{1, 2, \ldots, N\}$ — множество многопериодных инвестиционных проектов;
- T — горизонт планирования с периодами реализации $t = 0, 1, \ldots, T$;
- $h_t \in [\underline{h_t}, \overline{h_t}]$ — диапазоны ставок по дополняющим инвестициям в моменты времени $t = 1, 2, \ldots, T$;
- $s_t \in [\underline{s_t},\ \overline{s_t}]$ — диапазоны ставок по дополняющему заимствованию в моменты времени $t = 1, 2, \ldots, T$;
- $z_t(i)$ — платеж, порождаемый инвестиционным проектом $i \in I$ в момент времени t;
- M_t — базовый платеж в момент времени t;
- Y — заданный, одинаковый для всех проектов уровень изъятий (дохода) с определенной структурой $f = (f_0, f_1, \ldots,\ f_T)$ изъятий, при которой изымаемый доход в момент времени t равен $f_t Y$;
- G — лимит заимствования (при неограниченном рынке капитала $G = \infty$).

В работе рассматривается несовершенный рынок капитала.

Пусть пара $\lambda_t = (h_t, s_t)$ определяет выбор процентных ставок рынком капитала на период времени $t = \overline{1, T}$. Тогда вектор $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_T)$ определяет возможную реализацию процентных ставок рынком капитала на горизонте планирования T. Через Q будем обозначать множество всех возможных реализаций процентных ставок. Через $C_T(i, \lambda)$ обозначим остаточную стоимость по

реализации проекта i в условии использования процентных ставок $\lambda \in Q$.

Требуется найти оптимальную (в смысле минимума возможных потерь в остаточной стоимости) диверсификацию (распределение) $x^* \in X$ инвестиционного капитала на множестве анализируемых инвестиционных проектов, где

$$X = \Big\{ x \in \mathbb{R}^N : x_i \geqslant 0,\ i = \overline{1,N},\ \sum_{i=1}^{N} x_i = 1 \Big\},$$

компоненты x_i вектора $x = (x_1, x_2, \ldots, x_N) \in X$ определяют доли инвестиционных вложений в проект $i \in I$.

Рассматриваемая задача может трактоваться как конфликт между инвестором и рынком капитала, который может моделироваться бесконченой антагонистической игрой [3] $\Gamma = (I,\ Q,\ H)$, где:

- I — конечное множество чистых стратегий первого игрока (инвестора);
- Q — бесконченое множество чистых стратегий второго игрока (рынка капитала);
- $H(i, \lambda)$ — функция выигрыша первого игрока (инвестора), измеряемая потерями в остаточной стоимости при выборе инвестором проекта i в случае реализации варианта процентных ставок λ.

Известно [3], что решение игры включает оптимальную смешанную стратегию первого игрока (инвестора)

$$\alpha^* = (\alpha_1^*,\ \alpha_2^*, \ldots, \alpha_N^*),$$

где α_i^* — вероятность выбора инвестором стратегии i.

Нетрудно показать, что данная оптимальная стратегия имеет содержательную экономическую интерпретацию, а именно, вероятность α_i^* применения стратегии i определяет оптимальную долю инвестиционных вложений в проект $i \in I$. Таким образом, решение игры Γ обеспечивает решение задачи формирование портфеля: $x^* = \alpha^*$.

Согласно сформулированной задаче для формирования портфеля многопериодных инвестиций необходимо конкретизировать следующие положения:

1) функцию выигрыша H;
2) метод решения БАИ, включающий аппроксимацию диапазонов процентных ставок конечными наборами процентных ставок по инвестированию и заимствованию;
3) способ вычисления остаточной стоимости на стратегиях рынка;
4) метод формирования многопериодного инвестиционного портфеля;
5) комплексный алгоритм формирования портфеля многопериодных инвестиционных проектов и численную аттестацию предлагаемой модели.

2. Модель формирования портфеля

Осуществим переход к конечному множеству вариантов чистых стратегий рынка капитала путем ε аппроксимаций $\varepsilon > 0$ возможных диапазонов процентных ставок. Каждый диапазон аппроксимируется n_t и m_t точками возможных значений процентных ставок, соответственно для ставок по инвестированию и по заимствованию:

$$n_t = \frac{\left|\underline{h_t} - \overline{h_t}\right|}{\varepsilon} + 1, \quad t = \overline{1,T};$$
$$m_t = \frac{\left|\underline{s_t} - \overline{s_t}\right|}{\varepsilon} + 1, \quad t = \overline{1,T}.$$

Тогда множество Q всех возможных вариантов процентных ставок по дополняющему инвестированию и множество K всех возможных вариантов процентных ставок по дополняющему заимствованию, представляются конечными множествами $\overline{Q}$ и $\overline{K}$, а множество чистых стратегий рынка капитала — конечным множеством $\overline{Q} \times \overline{K}$.

Мощность данного множества определяется соотношением

$$L = \prod_{t=1}^{T} n_t \times m_t = \left|\overline{Q} \times \overline{K}\right|.$$

Обозначим через $C_t(i)$ остаточную стоимость по i-му проекту на момент времени $t = 0, \ldots, T$.

Для конкретной стратегии рынка капитала $\lambda \in Q$ алгоритм вычисления остаточной стоимости $C_t(i)$ определяется следующими соотношениями [2].

При $t = 0$:

$$C_0 = M_0 - f_0 Y + z_0.$$

При $t \geqslant 1$:

$$C_{t-1}(i) > 0 \Rightarrow C_t = M_t - f_t Y + z_t + (1 + h_t)\, C_{t-1}(\lambda),\ t = 2, \ldots, T;$$
$$C_{t-1}(i) < 0 \Rightarrow C_t = M_t - f_t Y + z_t + (1 + s_t)\, C_{t-1}(\lambda),\ t = 2, \ldots, T.$$

Если в определенный момент времени планового периода возникает ситуация, при которой необходимо осуществить дополняющее заимствование, превышающее лимит G, то такой проект неосуществим.

Результаты вычисления остаточной стоимости для каждого $i \in I$ и каждого $\lambda \in \overline{Q} \times \overline{K}$ удобно представить матрицей

$$C = \begin{pmatrix} C_T(1,1) & C_T(1,2) & \ldots & C_T(1,L) \\ \ldots & \ldots & \ldots & \ldots \\ C_T(N,1) & C_T(N,2) & \ldots & C_T(N,L) \end{pmatrix}.$$

В данной матрице каждый элемент $C_T(i,l)$ определяет остаточную стоимость i-го проекта в условии использования варианта l процентных ставок, $l = 1, \ldots, L$.

В качестве функции выигрыша $H(i,l)$ будем использовать меру сожаления (риска) от потери остаточной стоимости в условиях реализации i-го проекта, при принятии рынком процентных ставок l:

$$H(i,l) = \max_{j \in I} C_T(j,l) - C_T(i,l).$$

Для конечной антагонистической игры функцию выигрыша удобно представить матрицей

$$H = \begin{pmatrix} h_{11} & h_{12} & \ldots & h_{1l} \\ h_{21} & h_{22} & \ldots & h_{2l} \\ \ldots & \ldots & \ldots & \ldots \\ h_{N1} & h_{N2} & \ldots & h_{Nl} \end{pmatrix},$$

где

$$h_{ij} = H(i,l) = \max_{j \in I} C_T(j,l) - C_T(i,l).$$

В качестве метода решения игры используем метод ее сведения к паре двойственных задач линейного программирования [1].

Прямая задача линейного программирвоания (ПЗЛП):

$$\begin{cases} F(u) = \sum\limits_{j=1}^{L} u_j \to \max\limits_{u}, \\ \sum\limits_{j=1}^{L} h_{ij} u_j \leqslant 1, \ i = \overline{1,N}, \\ u_j \geqslant 0, \quad j = \overline{1,L}, \end{cases}$$

где $u = (u_1, u_2, \ldots, u_L)$ — вектор переменных прямой задачи линейного программирования.

Если $u^* = (u_1^*, u_2^*, \ldots, u_L^*)$ — решение ПЗЛП, то оптимальная стратегия рынка капитала представляется вектором

$$\beta^* = (\beta_1^*, \ \beta_2^*, \ldots, \beta_L^*), \quad \beta_j^* = \frac{u_j^*}{\sum\limits_{l=1}^{L} u_l^*}, \quad j = \overline{1,L}.$$

Двойственная задача линейного программирования (ДЗЛП):

$$\begin{cases} \Psi(\gamma) = \sum\limits_{i=1}^{N} \gamma_i \to \min\limits_{\gamma}, \\ \sum\limits_{i=1}^{N} h_{ij} \gamma_i \geqslant \ 1, \ j = \overline{1,L}, \\ \gamma_N \geqslant 0, \ i = \overline{1,N}, \end{cases}$$

где $\gamma = (\gamma_1, \ \gamma_2, \ \ldots, \gamma_N)$ — вектор переменных двойственной задачи линейного программирования, если решение ДЗЛП, $\gamma^* = (\gamma_1^*, \gamma_2^*, \ldots, \gamma_N^*)$, то оптимальная стратегия инвестора определяется вектором

$$\alpha^* = (\alpha_1^*, \alpha_2^*, \ldots, \alpha_N^*), \quad \alpha_i^* = \frac{\gamma_i^*}{\sum\limits_{l=1}^{N} \gamma_l^*}, \quad i = \overline{1,L}.$$

Проект 1	$z_{t,1}$	-500	-400	800	400
Проект 2	$z_{t,2}$	-300	-800	1200	200
Проект 3	$z_{t,3}$	-900	800	360	-10
Базовые платежи	M_t	600	100	-200	800
Структура изъятий	f_t	1	1,1	1,2	1,3
Проценты по заимствованию	$[\underline{s_t}, \overline{s_t}]$		[0,11; 0,13]	[0,09; 0,11]	[0,10; 0,13]
Проценты по инвестированию	$[\underline{h_t}, \overline{h_t}]$		[0,04; 0,06]	[0,06; 0,08]	[0,05; 0,09]

Таблица 1.

При этом искомая диверсификация инвестиционного капитала равна $x^* = \alpha^*$.

Заметим, что в смешанном расширении игры выигрышем инвестора является математическое ожидание функции выигрыша (средний риск потери):

$$\overline{H} = \sum_{i=1}^{N} \sum_{j=1}^{L} \alpha_i^* \beta_j^* h_{ij}.$$

3. Применение модели

Рассмотрим задачу диверсификации инвестиционных вложений на множестве проектов $I = \{1, 2, 3\}$, исходные данные по которым представлены в табл. 1. Плановый горизонт $T = 3$.

Результаты применения рассмотренного комплексного алгоритма приводят к оптимальной стратегии инвестора $\alpha^* = (0{,}101; 0{,}81; 0{,}089)$ и, следовательно, к оптимальной диверсификации инвестиционных вложений $x^* = (0{,}101; 0{,}81; 0{,}089)$.

Средний риск данной диверсификации инвестиционного портфеля составляет $\overline{H} = 9{,}127$.

Заключение

Рассмотренная в статье модель формирования портфеля многопериодных инвестиционных проектов позволяет провести оптимальную диверсификацию инвестиционных вложений в условиях полной неопределенности процентных ставок по дополняющим инвестированиям и заимствованиям, задаваемых возможными диапазонами их изменения в плановый период реализации проекта.

Перспективным направлением развития рассмотренного подхода является исследование возможностей моделирования конфликта между инвестором и рынком капитала бескоалиционной игрой.

Список литературы

[1] Болотникова, О. В. Линейное программирование: симплекс-метод и двойственность : учеб. пособие / О. В. Болотникова, Д. В. Тарасов, Р. В. Тарасов. — Пенза : Изд-во ПГУ, 2015. — 84 с.

[2] Крушвиц, Л. Инвестиционные расчеты : учебник для вузов. — СПб. : Питер, 2001. — 414 с.

[3] Петросян, Л. А. Теория игр : учеб. пособие для ун-тов / Л. А. Петросян, Н. А. Зенкевич, Е. А. Семина. — М. : Высш. шк., Книжный дом «Университет», 1998. — 304 с.

Библиографическая ссылка

Баяндина, П. В. Теоретико-игровая модель формирования портфеля многопериодных инвестиций по критерию минимального риска // Студенческая конференция факультета ПМиК. Сборник трудов. — Тверь : ТвГУ, 2023. — С. 4–11.

Сведения об авторах

Баяндина Полина Викторовна
Студентка магистратуры
Направление «Прикладная информатика»

УДК 303.722.4

Применение методов кластеризации и оценка качества их результатов

Журавлёва В. А.

Тверской государственный университет

Аннотация. Статья посвящена применению методов кластеризации и оценке качества полученных результатов. Приведены основные понятия и виды. Предложены индексы, с помощью которых анализируется качество кластеризации. Алгоритм получения кластеров реализован в статистическом пакете R. Работа алгоритма демонстрируется на примере.

Введение

Кластеризация часто выступает первым шагом при анализе данных. Выделение групп похожих объектов помогает понять структуру данных и использовать свой подход к обработке каждой из них. Применение методов кластерного анализа позволяет сократить объем хранимых данных; выявить новизну, то есть обнаружить нетипичные объекты, которые не удается причислить ни к одному из кластеров; упростить дальнейшую обработку. На сегодняшний момент число методов разбиения групп объектов на кластеры довольно велико — несколько десятков алгоритмов и еще больше их модификаций.

Результаты, полученные различными методами кластерного анализа, могут значительно отличаться друг от друга. В большинстве задач возникает проблема выбора оптимального числа кластеров, соответствующего природе изучаемых объектов. Поэтому одним из актуальных вопросов кластерного анализа является оценка качества полученных результатов и поиск разбиения, которое наиболее соответствует структуре исследуемых данных.

1. Кластерный анализ. Основные понятия, виды. Методы кластеризации

Кластерный анализ или кластеризация — это задача группировки набора объектов таким образом, чтобы объекты в одной и той же группе, называемой кластером, были более похожи друг на друга, чем объекты в других группах — кластерах.

Кластеризация может быть достигнута с помощью различных алгоритмов, которые существенно различаются в своем понимании того, что представляет собой кластер и как их эффективно находить. Популярные понятия кластеров включают группы с небольшими расстояниями между членами кластера, плотные области пространства данных, интервалы или конкретные статистические распределения.

Иерархическая кластеризация — это общее семейство алгоритмов кластеризации, которые создают вложенные кластеры путем их последовательного слияния или разделения. Эта иерархия кластеров представлена в виде дерева (или дендрограммы). Корень дерева — это уникальный кластер, который собирает все образцы, а листья дерева — это кластеры только с одним образцом.

Существует два типа методов иерархической кластеризации:

- агломеративная кластеризация (снизу вверх);
- разделяющая кластеризация (сверху вниз).

Агломеративная кластеризация является наиболее распространенным типом иерархической кластеризации, используемой для группировки объектов в кластеры на основе их сходства. Она также известна как AGNES (Agglomerative Nesting, агломеративная вложенность). Алгоритм начинается с обработки каждого объекта как одноэлементного кластера. Затем пары кластеров последовательно объединяются, пока все кластеры не будут объединены в один большой кластер, содержащий все объекты. Результатом является древовидное представление объектов, называемое дендрограммой.

Разделяющая иерархическая кластеризация, также известная как DIANA (Divisive Analysis, разделяющий анализ), является обратной по отношению к агломеративной кластеризации. Она начинается с включения всех объектов в один большой кластер. На каждом шаге итерации наиболее разнородный кластер делится на два. Процесс

повторяется до тех пор, пока все объекты не окажутся в своем собственном кластере.

Разделяющая кластеризация хороша для идентификации больших кластеров, в то время как агломеративная кластеризация хороша для идентификации небольших кластеров.

Меры связи между кластерами (методы иерархической кластеризации).

- Метод одиночной связи (Single Linkage)— наиболее понятный метод, который часто называют методом «ближайшего соседа» (Nearest Neighbor). Алгоритм начинается с поиска двух наиболее близких объектов, пара которых образует первичный кластер. Каждый последующий объект присоединяется к тому кластеру, к одному из объектов которого он ближе. Метрика задаётся следующим образом:

$$D_{ij} = \min_{x \in C_i, y \in C_j} d(x,\ y),$$

где расстояние D_{ij} между двумя кластерами C_i и C_j — это минимальное расстояние между двумя точками x и y, где $x \in C_i$, $y \in C_j$.

- Метод полной связи (Complete Linkage) часто называют методом «дальнего соседа» (Furthest Neighbor). Правило объединения этого метода подразумевает, что новый объект присоединяется к тому кластеру, самый далекий элемент которого находится ближе к новому объекту, чем самые далекие элементы других кластеров. Это правило является противоположным предыдущему и более жестким. Поэтому здесь наблюдается тенденция к выделению большего числа компактных кластеров, состоящих из наиболее похожих элементов. Метрика задаётся следующим образом:

$$D_{ij} = \max_{x \in C_i, y \in C_j} d(x,\ y),$$

где расстояние D_{ij} между двумя кластерами C_i и C_j — это максимальное расстояние между двумя точками x и y, где $x \in C_i$, $y \in C_j$.

- Метод средней связи (Average Linkage) или межгрупповой связи (Between Groups Linkage) занимает промежуточное положение относительно крайностей методов одиночной и полной связи. На каждом шаге вычисляется среднее арифметическое расстояние между каждым объектом из одного кластера и каждым объектом из другого кластера. Объект присоединяется к данному кластеру, если это среднее расстояние меньше, чем среднее расстояние до любого другого кластера. По своему принципу этот метод должен давать более точные результаты классификации, чем остальные методы. То, что объединение кластеров в методе средней связи происходит при расстоянии большем, чем в методе одиночной связи, но меньшем, чем в методе полной связи, и объясняет промежуточное положение этого метода. Этот метод имеет тенденцию формировать кластеры с одинаковой дисперсией и, в частности, с небольшой дисперсией. Метрика задаётся следующим образом:

$$D_{ij} = \sum_{x \in C_i, y \in C_j} \frac{d(x,\ y)}{n_i \times n_j},$$

где расстояние D_{ij} между двумя кластерами C_i и C_j — это среднее значение расстояний между парой точек x и y, где $x \in C_i$, $y \in C_j$, n_i и n_j — соответственно количество элементов в кластерах C_i и C_j.

- Центроидный метод (Centroid clustering). В обоих кластерах рассчитываются средние значения переменных относящихся к ним наблюдений. Затем расстояние между двумя кластерами рассчитывается как дистанция между двумя осредненными наблюдениями. Этот метод более надежен, чем другие, с точки зрения изолированных точек. Метрика задаётся следующим образом:

$$D_{ij} = \|\overline{x}_i - \overline{x}_j\|^2,$$

где расстояние D_{ij} между двумя кластерами C_i и C_j равно квадрату евклидова расстояния между центрами тяжести двух кластеров, то есть между средними векторами двух кластеров, $\overline{x_i}$и $\overline{x_j}$соответственно.

- Метод Уорда (Ward's Method). В отличие от других методов кластерного анализа, для оценки расстояний между кластерами здесь используются методы дисперсионного анализа. Сначала в обоих кластерах для всех имеющихся наблюдений производится расчет средних значений отдельных переменных. Затем вычисляются квадраты евклидовых расстояний от отдельных наблюдений каждого кластера до этого кластерного среднего значения. Эти дистанции суммируются. На каждом шаге алгоритма объединяются такие два кластера, которые приводят к минимальному увеличению дисперсии. Этот метод применяется для задач с близко расположенными кластерами [1].

2. Оценка качества кластеризации

Основная задача кластеризации обычно формулируется так: разделить объекты на группы таким образом, чтобы сходство между объектами одной группы было велико, а сходство между объектами разных групп — мало. Исходя из этого, понятие качества кластеризации состоит из следующих пунктов:

Компактность: элементы кластера должны быть как можно ближе друг к другу. Это свойство можно выразить через расстояния между элементами в кластере, плотностью внутри кластера или же объемом, занимаемым кластером в многомерном пространстве.

Отделимость: расстояние между различными кластерами должно быть как можно больше.

Концентрация: элементы кластера должны быть сконцентрированы вокруг центра кластера. Этот пункт используется гораздо реже, чем первые два, потому что далеко не во всех алгоритмах кластеризации используется понятие центра кластера.

Оценка качества кластеризации осуществляется с использованием специализированных модулей статистического пакета R. Для каждого из выбранных методов необходимо определить оптимальное количество кластеров, для этого применяются тридцать признаков из пакета R. После чего автором выполняется сравнительный анализ, оценка качества кластеризации и определяется наилучший из предложенных методов.

Примеры некоторых индексов оценки качества кластеризации:

- Индекс Данна (Dunn Index; Dunn, 1974). Индекс Данна имеет множество вариаций, оригинальная версия выглядит следующим образом:

$$D(C) = \frac{\min\limits_{c_k \in C}\left\{\min\limits_{c_l \in C \setminus c_k}\left\{\delta\left(c_k, c_l\right)\right\}\right\}}{\max\limits_{c_k \in C}\left\{\Delta\left(c_k\right)\right\}},$$

где:

  - δ — межкластерное расстояние (оценка разделения):

$$\delta\left(c_k,\ c_l\right) = \min_{x_i \in c_k, x_j \in c_l} \left\|x_i - x_j\right\|;$$

  - $\Delta\left(c_k\right)$ — диаметр кластера (оценка сплоченности),
  - $\Delta\left(c_k\right) = \max\limits_{x_i, x_j \in c_k} \left\|x_i - x_j\right\|$.

- Индекс SDbw (SDbw Index; Halkidi, Vazirgiannis, 2001). Он основан на вычислении Евклидовой нормы:

$$\|x\| = \left(x^T x\right)^{1/2}$$

и стандартных отклонений:

$$\sigma(X) = \frac{1}{|x|} \sum_{x_i \in X} \left(x_i - \overline{x}\right)^2,$$

$$stdev(C) = \frac{1}{K} \sqrt{\sum_{c_k \in C} \left\|\sigma\left(_k\right)\right\|}.$$

Сам индекс определяется формулой

$$SDbw(C) = \frac{1}{K} \sum_{c_k \in C} \frac{\left\|\sigma\left(c_k\right)\right\|}{\|\sigma(X)\|} + \\ + \frac{1}{K(K-1)} \sum_{c_k \in C} \sum_{c_l \in C \setminus c_k} \frac{den(c_k, c_l)}{\max\left(den\left(c_k\right), den\left(c_l\right)\right)},$$

здесь

$$den\,(c_k) = \sum_{x_i \in c_k} f(x_i, \overline{c_k}),$$

$$den\,(c_k,\ c_l) = \sum_{x_i \in c_k \cup c_l} f\left(x_i, \frac{\overline{c}_k + \overline{c}_l}{2}\right),$$

$$f\,(x_i, c_k) = \begin{cases} 0, & \text{если } \|x_i - \overline{c}_k\| > stdev(C), \\ 1 & \text{в ином случае.} \end{cases}$$

Индекс должен снижаться с улучшением кластеризации.

- Силуэт (Silhouette Index; Rousseeuw, 1987). Значение силуэта показывает, насколько объект похож на свой кластер по сравнению с другими кластерами.

  Оценка для всей кластерной структуры:

$$Sil(C) = \frac{1}{N} \sum_{c_k \in C} \sum_{x_i \in c_k} \frac{b\,(x_i, c_k) - a\,(x_i, c_k)}{\max\{a\,(x_i, c_k)\,, b\,(x_i, c_k)\}},$$

  где

  - $a\,(x_i, c_k) = \dfrac{1}{|c_k|} \sum\limits_{x_j \in c_k} \|x_i - x_j\|$ — среднее расстояние от $x_i \in c_k$ до других объектов из кластера c_k (компактность),
  - $b\,(x_i, c_k) = \min\limits_{c_l \in C \setminus c_k} \left\{ \dfrac{1}{|c_l|} \sum\limits_{x_j \in c_l} \|x_i - x_j\| \right\}$ — среднее расстояние от $x_i \in c_k$ до объектов из другого кластера $c_l : k \neq l$ (отделимость).

  Можно заметить, что $-1 \leqslant Sil(C) \leqslant 1$. Чем ближе данная оценка к 1, тем лучше.

- Индекс Калинского – Харабаша (Calinski – Harabasz Index; Calinski, Harabasz, 1974):

$$CH(C) = \frac{N - K}{K - 1} \times \frac{\sum\limits_{c_k \in C} |c_k| \times \left\|\overline{c}_k - \overline{X}\right\|}{\sum\limits_{c_k \in C} \sum\limits_{x_i \in c_k} \|x_i - \overline{c}_k\|}.$$

Компактность основана на расстоянии от точек кластера до их центроидов, а разделимость — на расстоянии от центроидов кластеров до глобального центроида. Индекс должен возрастать.

- Индекс C (C Index; Hubert, Levin, 1976). Индекс C представляет собой нормализованную оценку компактности:

$$CI(C) = \frac{S(C) - S_{\min}(C)}{S_{\max}(C) - S_{\min}(C)},$$

где

  - $S(C) = \sum_{c_k \in C} \sum_{x_i, x_j \in c_k} \|x_i - x_j\|$,
  - $S_{\min}(C)$ (соответственно $S_{\max}(C)$) — это сумма

$$\frac{|c_k| \times (|c_k| - 1)}{2}$$

    минимальных (максимальных) расстояний между парами всех объектов во всем наборе данных.

- Индекс Дэвиса – Болдуина (Davies – Bouldin Index; Davies, Bouldin, 1979) Это, возможно, одна из самых используемых мер оценки качества кластеризации.

  Она вычисляет компактность как расстояние от объектов кластера до их центроидов, а отделимость — как расстояние между центроидами:

$$DB(C) = \frac{1}{K} \sum_{c_k \in C} \max_{c_l \in C \setminus c_k} \left\{ \frac{S(c_k) + S(c_l)}{\|\overline{c_k} - \overline{c_l}\|} \right\}.$$

  C-индекс и индекс Дэвиса – Болдуина должны минимизироваться для роста кластеризации.

- Индекс Дуды (Duda Index; Duda, Hart, 1973):

$$duda = \frac{J_e(2)}{J_e(1)} = \frac{W_k + W_l}{W_m},$$

где

- $J_e(2)$ — сумма квадратов ошибок внутри кластера, когда данные разделены на два кластера;
- $J_e(1)$ выдает квадраты ошибок, когда присутствует только один кластер;
- W_k, W_l, W_m определяются как W_q:

$$W_q = \sum_{k=1}^{q} \sum_{i \in x_k} (x_i - c_k)(x_i - c_k)^T;$$

- W_q представляет собой матрицу внутригрупповой дисперсии для данных, сгруппированных в q кластеров.

Обозначим через n_m количество наблюдений в кластере c_m. Оптимальным числом кластеров является наименьшее q, такое что

$$duda = 1 - \frac{2}{\pi p} - z\sqrt{\frac{2\left(1 - \left(\frac{s}{\pi^2 p}\right)\right)}{n_m p}},$$

где p — количество переменных в наборе данных, z — стандартный нормальный балл [2].

3. Результаты

В рассматриваемом примере в роли кластеризуемого множества выступают регионы Российской Федерации, а набором данных являются такие переменные как оборот розничной, оптовой торговли, объем платных услуг населению и др.

На рис. 1–5 представлены результаты работы программы.

Анализируя, полученную информацию, можно сделать вывод, что разбиение регионов РФ на 4 кластера наилучшее, так как все предложенные методы кластеризации показывают наибольшее количество критериев качества для этого значения.

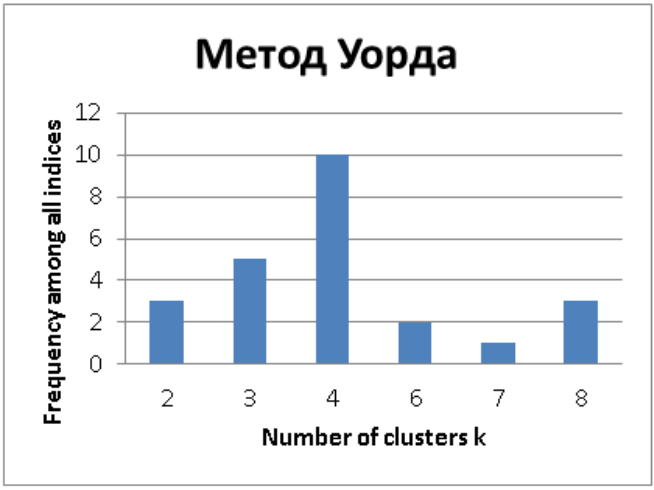

Рис. 1. Метод Уорда.

Рис. 2. Центроидный метод.

Рис. 3. Метод средней связи.

Рис. 4. Метод полной связи.

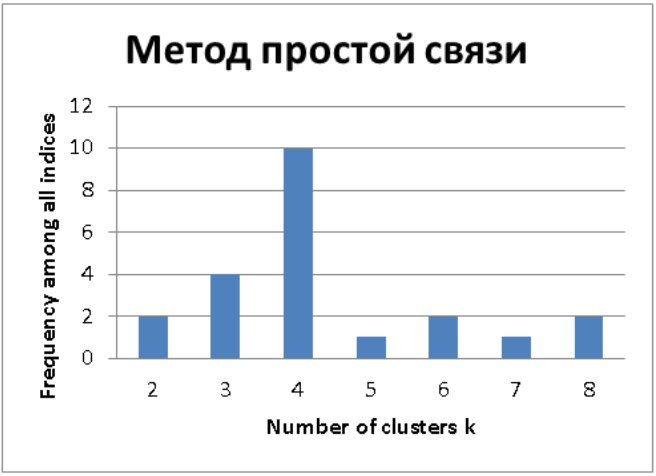

Рис. 5. Метод простой связи.

Заключение

В статье рассмотрены основные аспекты кластерного анализа, описана методология оценки качества кластеризации, позволяющая определить оптимальное количество кластеров. Создана программная реализация в статистическом пакете R, работа которой демонстрируется на примере классификации регионов Российской Федерации.

Список литературы

[1] Шитиков, В. К. Классификация, регрессия и другие алгоритмы Data Mining с использованием R / В. К. Шитиков, С. Э. Мастицкий. — Тольятти : Creative Commons, 2017. — 351 с.

[2] Das, A. K. Application of clustering techniques in defining level of service criteria of urban streets : Master of Technology in Transportation Engineering Thesis. — Rourkela : 2013. — 64 p.

Библиографическая ссылка

Журавлёва, В. А. Применение методов кластеризации и оценка качества их результатов // Студенческая конференция факультета ПМиК. Сборник трудов. — Тверь : ТвГУ, 2023. — С. 12–24.

Сведения об авторах

Журавлёва Виктория Анатольевна
Студентка магистратуры
Направление «Прикладная информатика»

УДК 004.415 + 004.78 + 519.237.5

Система прогнозирования уровня образования с использованием методов интеллектуального анализа данных

Исанбаев Д. Б.

Тверской государственный университет

Аннотация. В данной статье дано описание разработанной системы прогнозирования уровня образования с использованием методов интеллектуального анализа данных и представлен анализ ее работы на реальных данных. Проведен сравнительный анализ работы выбранных алгоритмов и дана оценка эффективности их применения к реальным данным.

Введение

В настоящее время образование является ключевым фактором успешного становления человека в обществе, а качество полученного им обучения напрямую влияет на его дальнейшую профессиональную карьеру. В процессе модернизации и развития систем обучения, для повышения их эффективности, особое внимание уделяется внедрению инновационных технологий. Именно в таком контексте разработана данная система прогнозирования уровня образования с использованием методов интеллектуального анализа данных. Основная задача системы заключается в прогнозировании возможности получения обучающимся высшего образования (уровня образования), с учетом имеющихся знаний и навыков.

1. Постановка задачи

Учитывая описание каждого из учеников, необходимо выявить наиболее значимые параметры для получения целевых значений, используя методы интеллектуального анализа данных. Путем выделения таких параметров сокращается количество необходимой информация для получения оптимального решения.

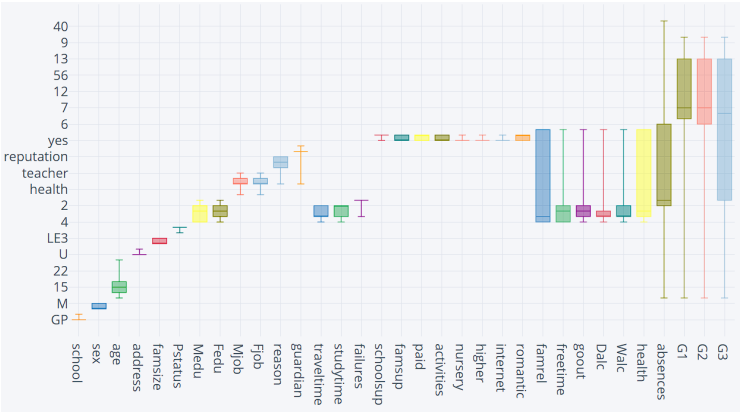

Рис. 1. Атрибуты тестовой выборки.

После анализа данных, на основе самых коррелирующих параметров, обучить систему выявлять, какую из ступеней образования может получить тот или иной обучающийся.

Таким образом, достигается поставленная цель — прогноз возможности получения студентом высшего образования, а также выявления уровня такого образования.

2. Анализ и первичная обработка исходных данных

Для тестовой выборки были использованы данные об успеваемости учащихся. Параметры включают в себя оценки учащихся, демографические и социальные характеристики, а также показатели, связанные со школой. Выборка исходных данных осуществлялась с использованием сети Интернет на основе размещенных в открытом доступе отчетов и результатов анкетирования. Рассматриваются параметры по 395 ученикам, каждый из которых имеет по 33 характеристики. Целевыми атрибутами являются «G1», «G2», «G3», которые характеризуют получение высших степеней образования (рис. 1).

Для поиска наиболее важных параметров, влияющих на получения образования, по результатам исследования было решено исклю-

	MAE	RMSE
Linear Regression	3.485115	4.432597
ElasticNet	3.608051	4.573274
Random Forest	3.680629	4.619445
Extra Trees	3.813434	4.775211
SVM	3.549266	4.581466
Gradient Boosted	3.572048	4.500625
Baseline	3.787879	4.825228

Рис. 2. Результаты полной выборки.

	MAE	RMSE		MAE	RMSE		MAE	RMSE
Linear Regression	3.415824	4.373226	**Linear Regression**	3.478875	4.362786	**Linear Regression**	3.507428	4.408219
ElasticNet	3.608051	4.573274	**ElasticNet**	3.633929	4.594937	**ElasticNet**	3.617368	4.583982
Random Forest	3.510955	4.388521	**Random Forest**	3.624504	4.495170	**Random Forest**	3.601296	4.402734
Extra Trees	3.767879	4.573706	**Extra Trees**	3.672138	4.577411	**Extra Trees**	3.637876	4.448406
SVM	3.544303	4.513978	**SVM**	3.472808	4.391310	**SVM**	3.476464	4.423282
Gradient Boosted	3.623946	4.525077	**Gradient Boosted**	3.569086	4.404425	**Gradient Boosted**	3.613708	4.454123
Baseline	3.787879	4.825228	**Baseline**	3.787879	4.825228	**Baseline**	3.787879	4.825228

Рис. 3. Результаты выборки по каждому кластеру.

чить из рассмотрения параметры «G1», «G2», которые являются периодическими ступенями и сильно коррелируют с итоговой оценкой «G3». Сложнее предсказать «G3» без «G2» и «G1», но такое предсказание гораздо полезнее, потому что мы хотим найти другие факторы, влияющие на оценку. Произведем оценку нескольких моделей обучений на полной выборке параметров (рис. 2).

Для сравнительного анализа также производим тест системы на другом количестве параметров для определения наиболее эффективного способа прогнозирования. Параметры кластеризуем по следующим признакам (рис. 3):

- обстоятельства, обусловленные родственными связями;
- обстоятельства, связанные с личными интересами;

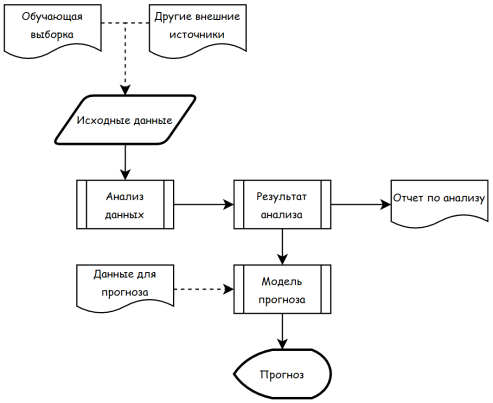

Рис. 4. Диаграмма структуры приложения.

- временные и другие количественные характеристики.

На основании полученных результатов можем говорить о том, что самыми важными являются признаки, связанные с семьей (особенно с матерью). По ним и будем строить целевой прогноз.

3. Архитектура системы и ее реализация

Для эффективной разработки системы была спроектирована архитектуры приложения в формате UML-диаграммы (рис. 4).

Архитектура данного приложения может быть разбита на три основных модуля.

1) Модуль сбора данных.
   Этот модуль отвечает за сбор данных из различных источников, таких как онлайн курсы, тесты, исследования и т. д. С данными можно работать только в том случае, если они структурированы и соответствуют определенному формату. Поэтому модуль сбора данных также должен включать процессы очистки и преобразования данных для последующего использования в аналитике.
2) Модуль анализа данных.
   Этот модуль использует различные методы интеллектуального

анализа данных (например, машинное обучение) и статистического моделирования для объяснения паттернов и трендов в данных. Кроме того, этот модуль должен позволять пользователям интерактивно взаимодействовать с данными и быстро получать результаты.

3) Модуль предсказания.
Этот модуль использует полученные знания и инсайты, которые предоставляет модуль анализа данных, для прогнозирования будущих тенденций развития образования. Он также может помочь пользователям в принятии решений, например, о выборе определенного курса или направления обучения.

4. Алгоритм прогнозирования

Для обучения алгоритма были использованы следующие методы [2].

1) Линейная регрессия:

$$\gamma = \beta_0 x + \beta_1 + \varepsilon.$$

2) Метод эластичной сети:

$$\min_{\widehat{\beta}_j} \sum_{i=1}^{n} \left(y_i - \widehat{y}_i\right)^2 + \lambda_1 \sum_{j=1}^{k} |\widehat{\beta}_j| + \lambda_2 \sum_{j=1}^{k} \widehat{\beta}_j^2.$$

3) Метод случайного леса:

$$a(x) = \frac{1}{N} \sum_{i=1}^{N} b_i(x).$$

4) Метод градиентного бустинга:

$$\min_{\eta} \min_{h} \frac{1}{N} \Big(\sum_{r=1}^{t-1} a_T g_T \left(x_n\right) + \eta h \left(x_n\right), y_n \Big).$$

	MAE	RMSE
Linear Regression	3.439686	4.395349
ElasticNet	3.429911	4.478035
Random Forest	3.723816	4.763088
Extra Trees	3.917518	4.983893
SVM	3.367254	4.374602
Gradient Boosted	3.542394	4.445384
Baseline	3.575758	4.712974

Рис. 5. Результаты тренировочной выборки.

Обучение производим на полной выборке, содержащей 30 параметров об учениках. Затем производим тестовую проверку работоспособности алгоритма. Для проверки на адекватность запустим программу на остаточных данных тренировочной выборки и проверим правильность прогнозирования (рис. 5).

Для более расширенного анализа работоспособности системы был проведен опрос среди людей в возрасте 18–27 лет по целевым признакам, которые были отобраны при кластеризации данных. Перед запуском алгоритма прогнозирования на новых данных, было произведено его обучение на полной тестовой выборке для улучшения результата.

Проведенная оптимизация исходных данных, в результате проведенного опроса, позволила значительно повысить эффективность системы: показатели эффективности системы выросли с 83% до 91%.

Это подтверждает наличие у системы потенциала к развитию и корректность ее работы.

Заключение

С помощью современных систем прогнозирования, использующих машинное обучение, можно значительно повысить точность прогнозирования в различных сферах жизни. Применение разработанной системы прогнозирования может быть полезно высшим учебным заведениям при отборе потенциальных студентов, что еще раз подтверждает актуальность работы и важность исследований в данной сфере.

Список литературы

[1] Documenting Software Architectures: Views and Beyond / P. Clements, F. Bachmann, L. Bass [et al.]. — 2nd ed. — Boston : Addison-Wesley Professional, 2010. — 592 p.

[2] Michie, D. Machine Learning: Neural and Statistical Classification / D. Michie, D. J. Spiegelhalter, C. C. Taylor — New Delhi : Overseas Press, 2009. — 290 p.

Библиографическая ссылка

Исанбаев, Д. Б. Система прогнозирования уровня образования с использованием методов интеллектуального анализа данных // Студенческая конференция факультета ПМиК. Сборник трудов. — Тверь : ТвГУ, 2023. — С. 25–31.

Сведения об авторах

Исанбаев Даниил Богданович
Студент магистратуры
Направление «Фундаментальная информатика и информационные технологии»

УДК 004.85

Построение и анализ модели на основе сверточных нейронных сетей для распознавания наличия маски на лице человека

Каленский И. С.

Тверской государственный университет

Аннотация. Статья представляет сравнительный анализ популярных архитектур сверточных нейронных сетей, таких как VGG16, VGG19, ResNet50, ResNet101, ResNet152, DenseNet121, DenseNet169 и DenseNet201, в контексте эффективного распознавания маски на лицах людей. Целью исследования является построение и обучение оптимальной модели для данной задачи, которая может быть применена в общественной безопасности, контроле доступа и медицинских приложениях. Результаты обзора помогут выбрать наиболее подходящую архитектуру для данной задачи.

Введение

Распознавание наличия маски на лице человека стало актуальной задачей в свете пандемии COVID-19 и необходимости соблюдения мер безопасности. Сверточные нейронные сети (CNN) представляют собой мощный инструмент для анализа изображений и могут быть применены для эффективного распознавания наличия маски на лицах людей [3].

Целью работы является построение и обучение оптимальной модели для распознавания наличия маски на лице человека, что может быть полезно в различных сферах, включая общественную безопасность, контроль доступа и медицинские приложениях. В работе рассматриваются такие популярные архитектуры сверточных нейронных сетей, как VGG16, VGG19, ResNet50, ResNet101, ResNet152, DenseNet121, DenseNet169 и DenseNet201.

1. Создание учебной выборки для классификации изображений

Изначально было собрано по 100 изображений на каждую категорию: «mask_correct» (маска надета правильно), «mask_missing» (маска отсутствует) и «mask_not_correct» (маска надета неправильно). Это гарантировало равномерное представление каждого класса и предотвращало несбалансированные данные.

Для повышения точности результатов было принято решение размножить данные с помощью добавления различных фильтров и аугментации изображений, таких как размытие, повышение резкости, grayscale, увеличение и уменьшение контрастности. В результате общий размер датасета составил 6000 изображений размером 526×561 пикселей. Далее данные были размечены, присваивая каждому изображению метку, соответствующую одному из трех классов. Для обучения модели выборка была разделена на три части: 70% данных использовались для обучения, 10% — для оценки и 20% — для тестирования модели. Описанный процесс создания обучающей выборки предоставляет основу для последующего обучения модели распознавания маски на лице.

2. Обучение моделей сверточных нейронных сетей на учебной выборке

Было выбрано несколько популярных архитектур сверточных нейронных сетей, таких как VGG16, VGG19, ResNet50, ResNet101, ResNet152, DenseNet121, DenseNet169 и DenseNet201. Каждая из этих моделей имеет свою структуру и глубину, что может влиять на их способность обучаться и точность предсказаний [1, 2, 4].

Для каждой выбранной архитектуры модели создается экземпляр сверточной нейронной сети с соответствующей конфигурацией слоев. Веса моделей инициализируются предварительно обученными весами, если они доступны. Обучение может занимать значительное время, особенно при использовании глубоких моделей с большим количеством параметров. Длительность обучения может зависеть от размера учебной выборки, сложности задачи, доступных вычислительных ресурсов и выбранных параметров обучения, таких как скорость обучения и размер мини-пакетов. В связи с переносом

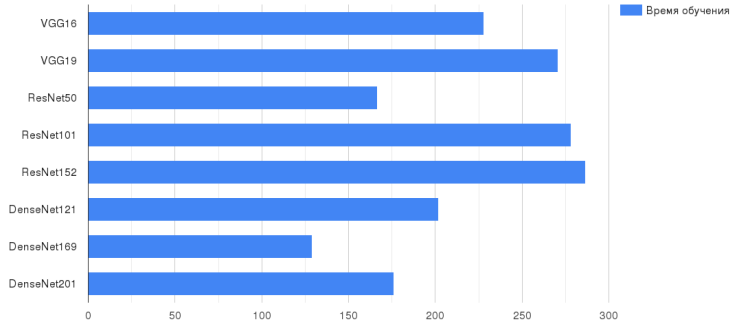

Рис. 1. Время обучения каждой модели на GPU в секундах.

вычислений на GPU, обучение значительно ускорилось и стало занимать не более пяти минут для каждой из рассматриваемых моделей, что можно увидеть на рис. 1. Результаты показали, что это в 15 раз быстрее, чем на CPU.

Во время обучения моделей важно следить за процессом обучения, чтобы оценить их производительность. Можно отслеживать метрики, такие как точность (accuracy) и функцию потерь (loss), на обучающей (рис. 2, 3) и валидационной (рис. 4, 5) выборках. Это позволяет оптимизировать параметры обучения и принимать решения о прекращении обучения, если модель достигает наилучших результатов.

Потери для моделей семейства DenseNet не отображаются, так как очень малы, по сравнению с остальными моделями:

- DenseNet121: 0,00000000105352353330090 4;
- DenseNet169: 0,00000316258160637517;
- DenseNet201: 0,0000086686072268847027.

На основе предоставленных данных, все модели демонстрируют высокую точность (accuracy) 1,0 на обучающих данных. Однако для выбора лучшей модели следует рассмотреть также значения validation accuracy и validation loss.

Сравнение моделей на основе validation accuracy:

- ResNet50 показывает наивысшую validation accuracy 0,9967,

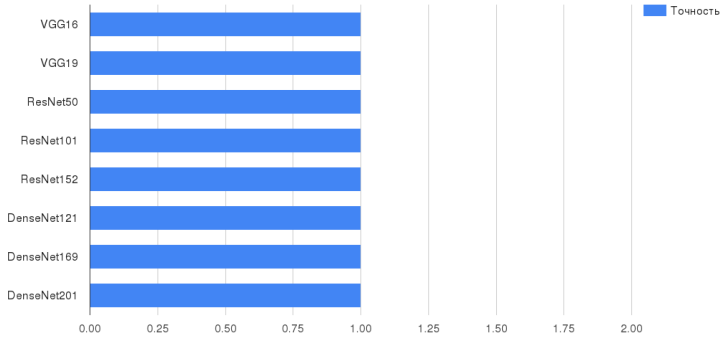

Рис. 2. Точность на обучающей выборке.

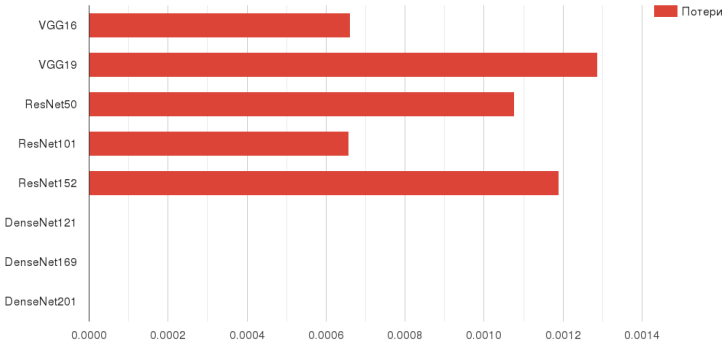

Рис. 3. Потери на обучающей выборке.

что означает, что она достигает наилучшей точности на валидационных данных из всех представленных моделей;

- затем следует DenseNet201 с validation accuracy 0,9917;
- остальные модели имеют незначительные различия в validation accuracy, но все они также достигают высокой точности на валидационных данных.

Сравнение моделей на основе validation loss:

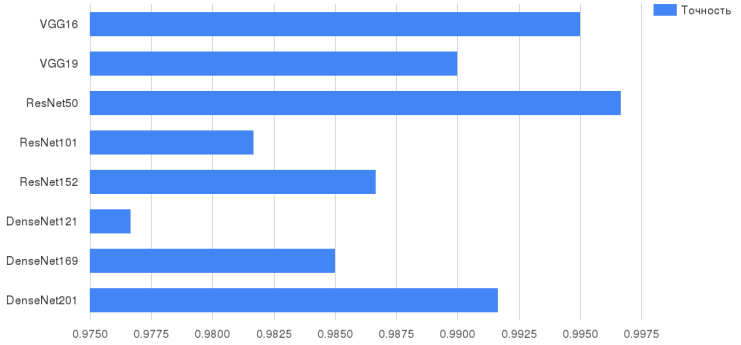

Рис. 4. Точность на валидационной выборке.

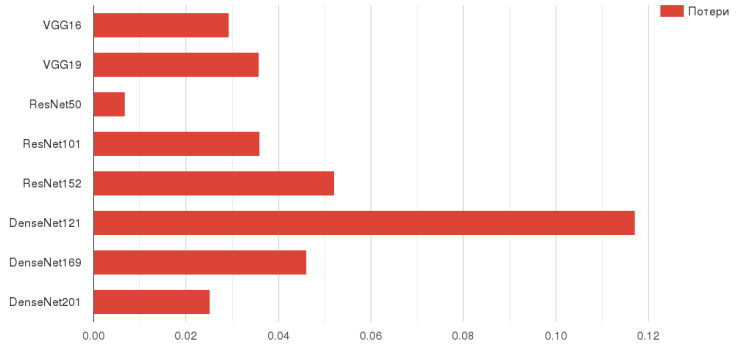

Рис. 5. Потери на валидационной выборке.

- ResNet50 имеет наименьший validation loss 0,0067, что говорит о его способности к наилучшей минимизации потерь на валидационных данных;
- затем следует DenseNet201 с validation loss 0,0252;
- наибольшее значение validation loss 0,1171 достигается для модели DenseNet121.

В целом, на основе предоставленных данных, на данном этапе ResNet50 проявляет себя как лучшая модель с самой высокой

validation accuracy и наименьшим validation loss.

3. Тестирование и анализ обученных моделей на тестовой выборке

Для тестирования моделей создается отдельный набор данных, который не использовался в процессе обучения и валидации. Тестовая выборка должна содержать разнообразные изображения с лицами людей в правильно надетых масках, неправильно надетых масках и без масок. Как было упомянуто выше, это 20% от нашего датасета (1200 изображений). Модели, полученные в процессе обучения, загружаются для применения на тестовой выборке. При этом сохраняется архитектура и веса моделей, которые были получены в результате обучения. Для каждой модели производится применение на тестовой выборке путем подачи изображений лиц на вход модели. Модели генерируют предсказания о наличии маски на лице человека для каждого изображения.

После получения предсказаний моделей, их результаты сравниваются с правильными метками из тестовой выборки. Метрики производительности, такие как полнота (recall), точность предсказания класса (precision) и F1-мера (рис. 6), точность и потери (рис. 7, 8), вычисляются для каждой модели.

Анализируя все полученные данные, можно сделать следующие выводы.

- **Точность на обучающей выборке.** Все модели показывают 100% точность на обучающем наборе данных. Это говорит о том, что все модели правильно классифицируют изображения с лицами людей с маской.
- **Потери на обучающей выборке.** Все модели имеют низкие значения функции потерь на обучающем наборе данных, что указывает на хорошую способность моделей обучаться и соответствовать данным. Наименьшее значение имеет модель DensNet121 ($1{,}0535 \cdot 10^{-9}$).
- **Точность на валидационной выборке.** ResNet50 показывает самую высокую точность на проверочном наборе данных (0,9967), что означает, что эта модель хорошо обобщает и обнаруживает маски на лицах в новых изображениях.

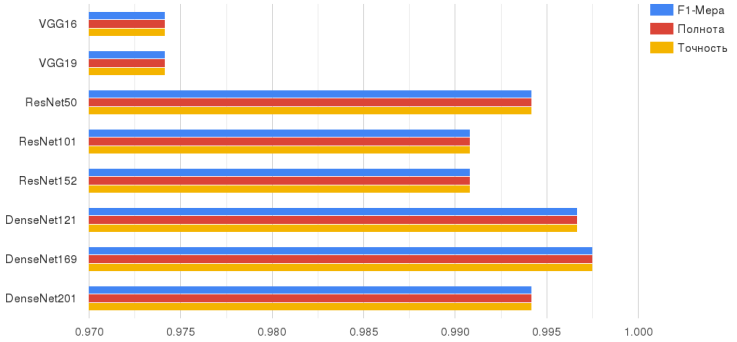

Рис. 6. Метрики производительности моделей (F1-мера, полнота, точность).

- **Потери на валидационной выборке.** ResNet50 имеет наименьшие потери на проверочном наборе данных (0,0067), что свидетельствует о хорошей способности модели обобщаться и обнаруживать маски на лицах в новых изображениях.
- **Точность на тестовой выборке.** DenseNet169 показывает самую высокую точность на тестовом наборе данных (0,9975), что говорит о способности эффективно классифицировать лица с масками.
- **Потери на тестовой выборке.** DenseNet121 и DenseNet169 имеют наименьшие потери на тестовом наборе данных (0,0131 и 0,0173 соответственно), что подтверждает их способность обобщаться и эффективно обнаруживать маски на лицах в реальных условиях.
- **F1-мера, полнота, точность (F1-score, recall и precision).** Все модели показывают высокие значения этих метрик, что говорит о их способности эффективно обнаруживать маски на лицах.
- **Время обучения:**
    - VGG16 и VGG19 требуют больше времени для обучения, превышая 200 секунд;

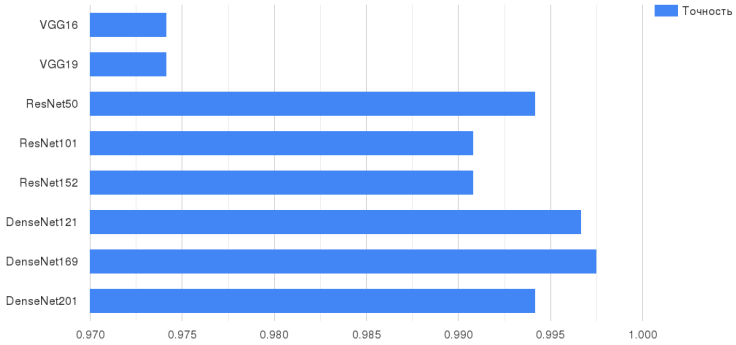

Рис. 7. Точность на тестовой выборке.

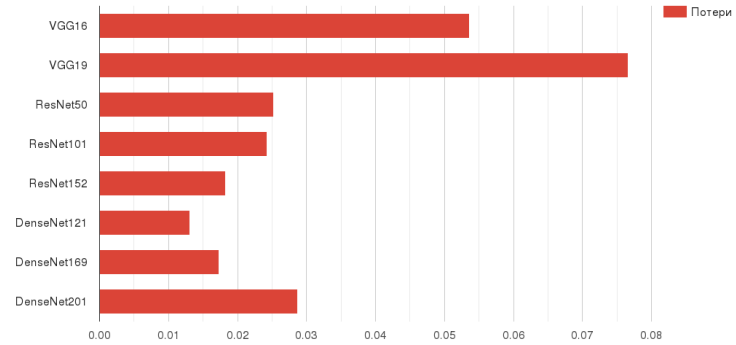

Рис. 8. Потери на тестовой выборке.

- ResNet50 имеет относительно низкое время обучения, составляющее около 167 секунд;
- ResNet101 и ResNet152 требуют значительно больше времени для обучения, превышая 270 секунд;
- DenseNet121 имеет промежуточное время обучения около 202 секунд;
- DenseNet169 и DenseNet201 имеют самое низкое время обучения с примерно 129 и 176 секундами соответственно.

Рис. 9. Результат работы модели ResNet50.

Исходя из анализа, можно сделать вывод, что модели ResNet50, DenseNet121 и DenseNet169 демонстрируют хорошие результаты на всех основных метриках.

Результат работы модели для распознавания наличия маски на лице человека продемонстрирован на рис. 9.

Для каждого изображения выводится следующая информация.

- Исходное изображение. Слева отображается название изображения. Снизу подписывается предсказанный класс и истинный класс, который указывается в скобках.

- Для демонстрации работы алгоритма также выводится Grad-CAM Heatmap. Основная идея Grad-CAM: мы хотим использовать информацию, полученную через нашу свёрточную сеть, чтобы понять какие части входного изображения были важны для принятия решения о классификации. Grad-CAM очень популярный метод для создания heatmap для конкретных классов на основе входных изображений.
- Диаграмма вероятностей принадлежности классу. Демонстрирует с какой вероятность и к какому классу модель соотнесла изображение, где 0, 1, 2 — это «mask_correct», «mask_missing», «mask_not_correct» соответственно.

Заключение

Все модели демонстрировали высокую точность на обучающей выборке и способность правильно классифицировать изображения. ResNet50 продемонстрировала наивысшую точность и наименьшие потери на провероной выборке, что свидетельствует о ее хорошей способности обобщаться и обнаруживать маски на лицах в новых изображениях. DenseNet121 и DenseNet169 показали наивысшую точность и наименьшие потери на тестовой выборке, что подтверждает их способность эффективно обнаруживать маски на лицах в реальных условиях. Все модели также показали высокие значения метрик F1-меры, полноты и точности, что свидетельствует о их эффективности в обнаружении масок на лицах. Время обучения различных моделей варьируется, но модели DenseNet169 и DenseNet201 имеют наименьшее время обучения.

Однако стоит отметить, что лучшая модель среди рассмотренных зависит от конкретной задачи, набора данных и ресурсов, доступных для обучения и применения модели. Можно сделать вывод о том, что не существует однозначного ответа на вопрос о лучшей модели, и выбор должен быть основан на конкретных требованиях и ограничениях. В данной задаче лучшими моделями оказались ResNet50, DenseNet121 и DenseNet169.

В целом, результаты исследования указывают на эффективность и применимость свёрточных нейронных сетей для задачи распознавания наличия маски на лице человека. Это открывает перспективы для применения подобных моделей в различных областях, включая

безопасность, контроль доступа, и ограничение распространения инфекционных заболеваний, где точное и автоматизированное распознавание масок на лицах играет важную роль.

Список литературы

[1] Deep Residual Learning for Image Recognition / K. He, X. Zhang, S. Ren, J. Sun. — 2015. — URL: https://arxiv.org/abs/1512.03385.

[2] Densely Connected Convolutional Networks / G. Huang, Z. Liu, L. van der Maaten, K. Q. Weinberger. — 2016. — URL: https://arxiv.org/abs/1608.06993.

[3] Krizhevsky, A. ImageNet Classification with Deep Convolutional Neural Networks / A. Krizhevsky, I. Sutskever, E. G. Hinton // Communications of the ACM. — 2017. — Vol. 60, iss. 6. — P. 84–90.

[4] Simonyan, K. Very Deep Convolutional Networks for Large-Scale Image Recognition / K. Simonyan, A. Zisserman. — 2015. — URL: https://arxiv.org/abs/1409.1556.

Библиографическая ссылка

Каленский, И. С. Построение и анализ модели на основе сверточных нейронных сетей для распознавания наличия маски на лице человека // Студенческая конференция факультета ПМиК. Сборник трудов. — Тверь : ТвГУ, 2023. — С. 32–42.

Сведения об авторах

Каленский Иван Сергеевич
Студент магистратуры
Направление «Фундаментальная информатика и информационные технологии»

УДК 004.855.5 + 519.254

Интеллектуальный анализ данных в прогнозировании временных рядов

Ножкин М. А.

Тверской государственный университет

Аннотация. Статья посвящена сравнению двух методов прогнозирования временных рядов — с помощью рекуррентной нейронной сети и с помощью модели прогнозирования, основанной на концепции градиентного бустинга — для решения задачи предсказания продаж товаров на следующий месяц. Была реализована программа с графическим интерфейсом, что позволяет выбрать метод прогнозирования и на основе загруженных данных провести обучение, а затем предсказание с выводом среднеквадратичной ошибки во время обучения и предсказания, а также выводом самого предсказания с возможностью записи его в файл.

Введение

Прогнозирование спроса на современном мировом рынке информационных технологий — важный процесс, который минимизирует риски, затраты и позволяет составить точный план работы любой организации. Ее программные продукты становятся актуальны и прибыльны среди потребителей. Это также укрепляет положение организации на рынке.

В сегодняшнем нестабильном экономическом климате многие предприятия недооценивают или чаще переоценивают свои прогнозы продаж и теряют деньги.

Отсутствие аналитической и прогностической информации и проверенных прогностических знаний привело к тому, что многие компании столкнулись с неудачами в продуктах, что негативно сказалось на экономике растущей компании и общей репутации компании, приводя к убыткам. Если существует интерес к подобной проблеме, значит, требуется углубленное изучение методов прогнозирования, эффективности и методов реализации для недопущения подобных ошибок в будущем [5].

Существующие методы субъективного прогнозирования для розничной торговли недостаточно информативны и не поддаются автоматизации. Эта проблема усугубляется по мере того, как все больше и больше разработчиков программного обеспечения создают все новые программы, которые одновременно удовлетворяют многочисленные потребности компаний и отдельных пользователей программного обеспечения, что увеличивает конкуренцию и сложность разработки [4].

Чтобы облегчить процесс прогнозирования, многие компании и их аналитики обращаются к нейронным сетям и моделям прогнозирования. Нейронные сети могут прогнозировать количество продаж того или иного продукта на основе данных о продажах за определенный период времени. А модели прогнозирования, опираясь на данные о продажах за предыдущие периоды времени, могут часто дать более развернутую информацию про сами данные и как те или иные признаки могут зависеть друг от друга. Разработчики любят использовать данные технологии, потому что они дешевы в создании и эксплуатации.

1. Рекуррентная нейронная сеть для прогнозирования

Для реализации прогноза продаж была выбрана рекуррентная нейронная сеть с обратным распространением. Связи между элементами в такой нейросети образуют направленную последовательность. Благодаря этому появляется возможность обрабатывать серии событий во времени или последовательные пространственные цепочки [3]. Схема работы нейронной сети представлена на рис. 1.

Здесь $x_1, x_2, x_3, \ldots, x_n$ представляют входные данные из датасета, $y_1, y_2, y_3, \ldots, y_M$ представляют предсказанные значения продаж, а $\overline{h}_0, \overline{h}_1, \overline{h}_2, \overline{h}_3, \ldots, \overline{h}_t$ содержат информацию для предыдущих входных данных.

Сами данные перед подачей их нейронной сети должны пройти предобработку. Так как самих файлов с данными несколько, сначала необходимо загрузить их в различные переменные, а после соединить их в отдельную таблицу и преобразовать в нужный для нейросети вид. После слития данных в одну таблицу необходимо произвести

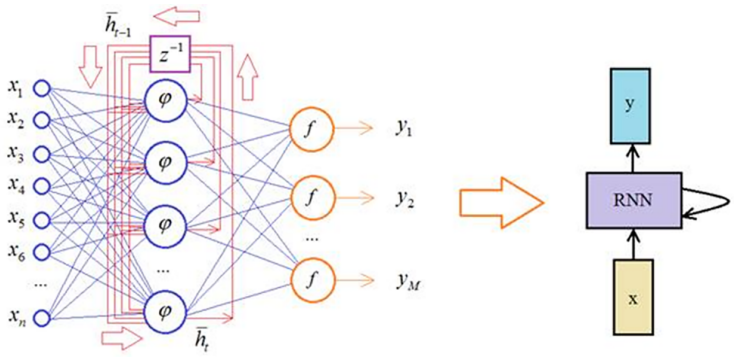

Рис. 1. Схема работы рекуррентной нейронной сети.

удаление признаков, которые будут лишними для прогнозирования, поскольку не содержат никаких важных данных. Как только ненужные признаки были удалены, далее нужно сложить данные о продажах в каждом месяце по каждому товару в каждом магазине, чтобы уже эта таблица была конечной точкой для «скармливания» ее нейросети (рис. 2).

Строится модель — последовательная нейросеть с одним LSTM-слоем (долгая краткосрочная память для того, чтобы запоминать предыдущие месяцы с продажами) с количеством «ячеек памяти», равным 64 нейрона. На вход подается вектор 33×1 с признаками. Сами признаки и являются этими самыми продажами за месяц. На выходе получается один плотный слой с одним нейроном, который и будет ответом на поставленную задачу. Также при сборе модели нейросети указывается метрика исчисления ошибки — Mean Squared Error (MSE), так как решается задача регрессии. Во время обучения можно также видеть, какая величина у среднеквадратичной ошибки, то есть насколько сильно ошибается нейросеть (рис. 3).

После того, как нейросеть обучилась, выводится сам прогноз. Он записывается в отдельную таблицу. Содержащую три признака: магазин, товар в этом магазине и количество продаж данного товара (рис. 4).

	(item_cnt_day, 1.0)	(item_cnt_day, 2.0)	(item_cnt_day, 3.0)	(item_cnt_day, 4.0)	(item_cnt_day, 5.0)	(item_cnt_day, 6.0)	(item_cnt_day, 7.0)	(item_cnt_day, 8.0)	(item_cnt_day, 9.0)	(item_cnt_day, 10.0)
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...
214195	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
214196	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
214197	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0
214198	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
214199	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

...	(item_cnt_day, 24.0)	(item_cnt_day, 25.0)	(item_cnt_day, 26.0)	(item_cnt_day, 27.0)	(item_cnt_day, 28.0)	(item_cnt_day, 29.0)	(item_cnt_day, 30.0)	(item_cnt_day, 31.0)	(item_cnt_day, 32.0)	(item_cnt_day, 33.0)
...	2.0	0.0	0.0	0.0	1.0	1.0	1.0	3.0	1.0	0.0
...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	0.0	0.0	0.0	0.0	3.0	2.0	0.0	1.0	3.0	0.0
...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...
...	2.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Рис. 2. Итоговая таблица для подачи нейросети.

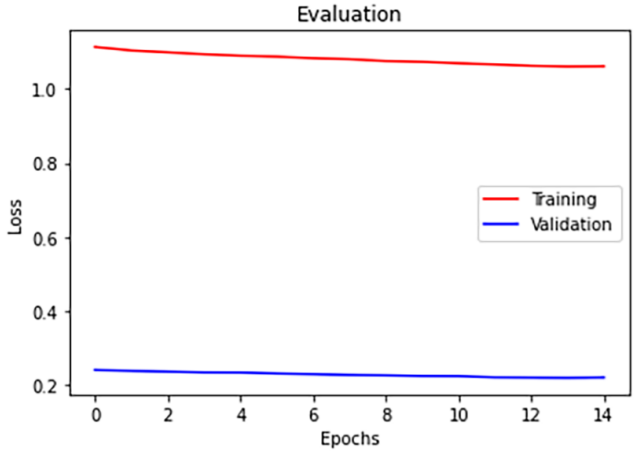

Рис. 3. График потерь во время обучения нейросети.

shop_id	item_id	item_cnt_month
5.0	5037.0	0.22958387
5.0	5320.0	1.9857826
5.0	5233.0	0.22879909
5.0	5232.0	0.08641611
5.0	5268.0	1.9857826
5.0	5039.0	0.07596864
5.0	5041.0	0.15906471
5.0	5046.0	0.032641903
5.0	5319.0	0.3433079
5.0	5003.0	1.9857826
5.0	4806.0	0.49624687
5.0	4843.0	0.12207484
5.0	4607.0	-0.0039809644
5.0	4869.0	0.11732496
5.0	4870.0	0.34831458
5.0	4872.0	0.21379988

Рис. 4. Фрагмент таблицы с предсказанными продажами.

2. Модель прогнозирования

В противопоставление результату предсказания нейронной сетью в этой работе также необходимо реализовать модель прогнозирования, которая также на основе данных о продажах за предыдущий отрезок времени способна выдать необходимый ответ [1]. Для начала необходимо посмотреть на графики сезонности, тренда и остатков для суммарного количества продаж по всем магазинам, чтобы определить соответствующие признаки для построения прогноза (рис. 5).

Видно, что суммарно от года к году продажи товаров снижались. При этом на графике хорошо видна зависимость количества продаж товаров от месяца. Далее нужно посмотреть на график автокорреляции временного ряда. На рис. 6, 7 можно увидеть данные графики. На графиках видно, что на продажу следующего месяца напрямую влияют продажи за предыдущие месяцы.

Поэтому в качестве признаков, по которым будет предсказано

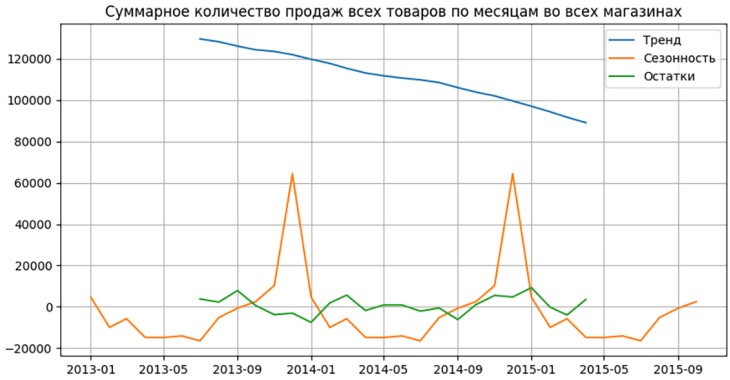

Рис. 5. Суммарное количество продаж товаров по месяцам во всех магазинах.

количество продаж, будут использованы:

1) магазин — продажи в каждом магазине могут отличаться в зависимости от его расположения, а также маркетинговой политики;
2) товар — продажи каждого товара могут отличаться, так как некоторые товары могут пользоваться большим спросом, нежели другие;
3) категорию товара — продажи товаров схожих категорий могут иметь схожие показатели. Этот признак может помочь предсказывать сумму продаж для новых товаров, которые модель не встречала ранее.
4) месяц продажи — как мы могли наблюдать ранее, на графике тренда – сезонности – остатков наблюдается зависимость продаж от месяца.

Данные о продажах за последние три месяца могут помочь более точно предсказать количество продаж в текущем месяце. Для предсказания продаж была использована модель градиентного бустинга, техники построения модель предсказания в форме ансамбля слабых предсказывающих моделей, обычно деревьев решений [2]. Чтобы

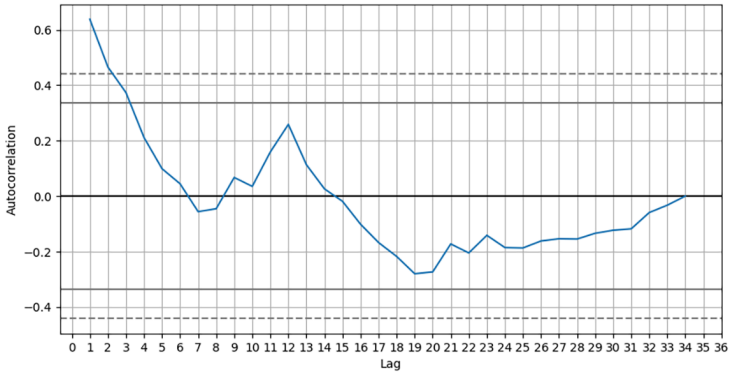

Рис. 6. График автокорреляции лагов.

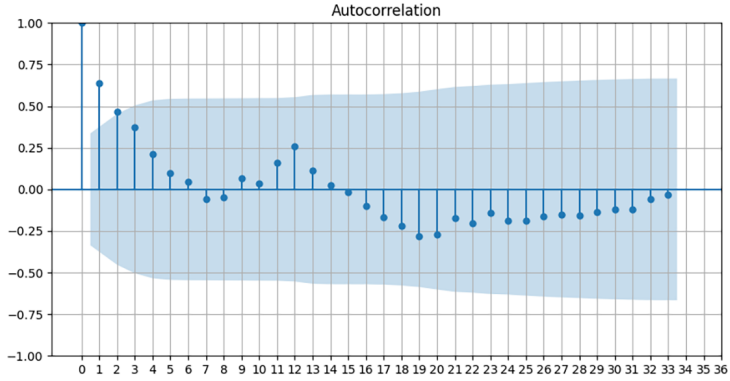

Рис. 7. График автокорреляции зависимостей продаж.

понять, влияют ли данные о продажах за последние три месяца на количество продаж для месяца, что будет предсказан, были рассмотрены два варианта предсказания: с дополнительными данными и без дополнительных данных.

После обучения модели выведем график зависимостей признаков, что могут повлиять на продажи (рис. 8, 9).

Теперь добавим новые признаки и вновь выведем график зави-

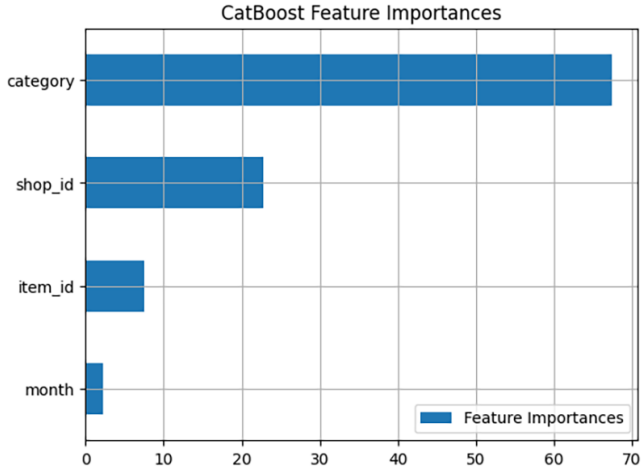

Рис. 8. Зависимость продаж от признаков при использовании алгоритма градиентного бустинга.

Рис. 9. Истинные и предсказанные значения после отработки алгоритма градиентного бустинга.

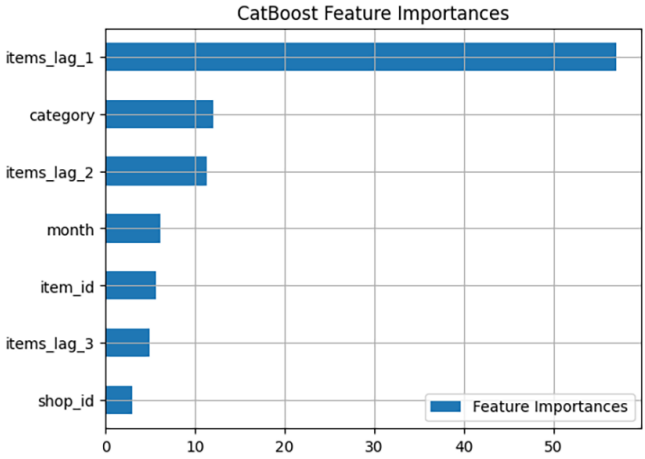

Рис. 10. График зависимостей продажи с дополнительными признаками после отработки алгоритма градиентного бустинга.

симостей и график предсказанных значений (рис. 10, 11).

Также была сравнена ошибка обучения алгоритма градиентного бустинга без признаков и с признаками:

GradientBoosting, RMSE
без дополнительных признаков: 8,971969732834388
GradientBoosting, RMSE
с дополнительными признаками: 8,434113155281425

Как видно, данные о продажах за последние три месяца помогли улучшить качество алгоритма и уменьшить среднеквадратичную ошибку. А наиболее важным признаком алгоритм градиентного бустинга посчитал продажу за предыдущий месяц.

3. Графический интерфейс

Поскольку готовые решения для предсказания продаж были созданы как отдельные приложения, было принято решения разработать свой собственный графический интерфейс и подключить его

Рис. 11. Истинные и предсказанные значения после отработки алгоритма градиентного бустинга с дополнительными признаками.

к библиотеке с открытым исходным кодом Streamlit, что позволяет загружать свои приложения и пользоваться ими как в браузере, так и локально на пользовательском ПК [6]. Сам интерфейс представлен на рис. 12.

Работает приложение следующим образом: сначала выбирается сама модель предсказания (градиентный бустинг либо нейронная сеть), после в само приложение загружаются файлы, содержащие информацию о продажах. После загрузки файлов с помощью «ползунков» настраиваются параметры прогнозирования для модели: у нейросети это количество нейронов скрытого слоя, размер батча — сколько информации будет подаваться за раз для обучения — а также количество эпох обучения, у градиентного бустинга — количество деревьев выбора, а также максимальную глубину данных деревьев.

Чтобы модель либо нейросеть выдали пользователю прогноз, необходимо сначала провести обучение. После самого обучения уже можно будет вывести сам прогноз. При выборе градиентного бустинга выводится среднеквадратичная ошибка, а также графики

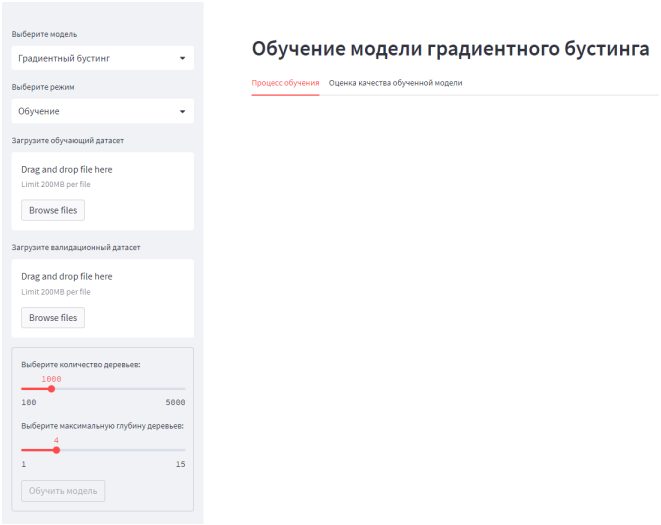

Рис. 12. Графический интерфейс для программы предсказания.

зависимости признаков и точности предсказанных значений. При выборе нейронной сети выводится график потерь при обучении и среднеквадратическая ошибка. Само же предсказание выводится отдельной таблицей, которую можно скачать отдельным файлом. На рис. 13 показано предсказание модели, полученное после работы нейросети.

Заключение

В ходе сравнения, рассматриваемого в данной статье, было выявлено, что нейросеть выдает большую ошибку при обучении (4,621097 против 3,820307 у градиентного бустинга), но меньшую ошибку при самом предсказании (0,888221 против 8,336551 у градиентного бустинга), что делает его более привлекательным методом для прогнозирования продаж. Но нельзя также недооценивать сами методы предсказания с помощью временных рядов. Во многих случаях с ними нужно дольше разбираться, но они могут дать больше наглядности для исследования скрытых закономерностей в данных.

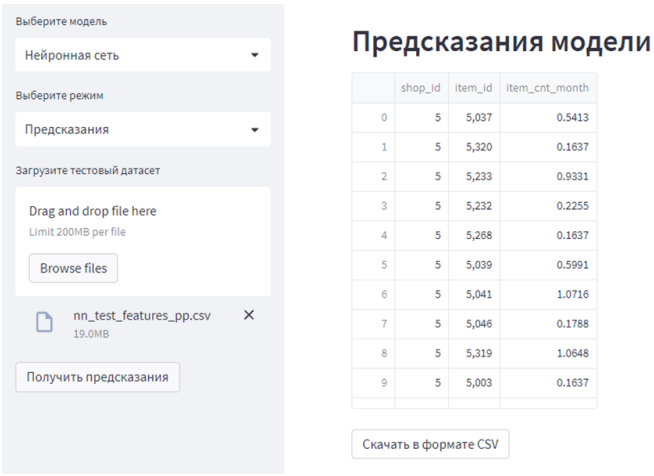

Рис. 13. Предсказание, полученное после работы нейронной сети.

Список литературы

[1] Анализ временных рядов // StatSoft : [сайт]. — URL: http://statsoft.ru/home/textbook/modules/sttimser.html (дата обращения: 15.05.2023). — Загл. с титул. экрана.

[2] Градиентный бустинг — просто о сложном // Neurohive : [сайт]. — URL: https://neurohive.io/ru/osnovy-data-science/gradientyj-busting/ (дата обращения: 15.05.2023). — Загл. с титул. экрана.

[3] Ростовцев, В. С. Искусственные нейронные сети : учебник. — СПб. : Лань, 2019. — 216 с.

[4] Elkan, C. Predictive Analytics and Data Mining. — San Diego : University of California, 2013. — 164 p.

[5] Larose, D. T. Data Mining and Predictive Analytics / D. T. Larose, C. D. Larose. — 2nd ed. — Hoboken, NJ : John Wiley & Sons, 2015. — 827 p.

[6] Streamlit documentation : [сайт]. — URL: https://docs.streamlit.io/. — Загл. с титул. экрана.

Библиографическая ссылка

Ножкин, М. А. Интеллектуальный анализ данных в прогнозировании временных рядов // Студенческая конференция факультета ПМиК. Сборник трудов. — Тверь : ТвГУ, 2023. — С. 43–55.

Сведения об авторах

Ножкин Михаил Александрович
Студент магистратуры
Направление «Фундаментальная информатика и информационные технологии»

УДК 519.876.3

Построение и оптимизация сетевых графиков комплексов работ в условиях неопределенности

Руденок М. А.

Тверской государственный университет

Аннотация. Рассматривается задачи оптимизации сетевых графиков комплекса работ в условиях неопределенности, которая имеет несколько вариантов оптимизации. В исследуемой задаче входными данными являются структурная таблица, определены сроки выполнения и стоимость проекта. Рассмотрены следующие методы оптимизации: выполнение комплекса работ в заданный срок и «время – затраты». Приведен пример, иллюстрирующий работоспособность приведенных методов оптимизации.

Введение

В современных экономических условиях при быстром развитии технологий и возрастающей конкуренции заставляет компании максимально эффективно использовать все ресурсы, начиная от материальных и заканчивая трудовыми. Однако в условиях неопределенности, когда имеются ряд неизвестных факторов сложно точно установить время выполнения комплекса работ. Для начала, нужно избавиться или минимизировать влияние неопределенности на время выполнения. После можно прибегать к построению и оптимизации дабы повысить эффективность.

Цель работы состоит в разработке и обосновании метода оптимизации сетевого графика. Решив поставленную задачу, мы получим оптимальное время выполнения и стоимость комплекса работ.

Оптимизация сетевого графика позволяет оценить риски и возможности, связанные с выполнением проекта, определить наиболее эффективный план действий, сократить время и затраты на его реализацию.

По указанным причинам тема работы: «Построение и оптимизация сетевых графиков комплекса работ в условиях неопределенности» является чрезвычайно актуальной.

1. Методы оптимизации сетевых графиков в условиях неопределенности

Оптимизация сетевого графика состоит в определении оптимальных резервов времени работ и нахождении оптимального критического пути, то есть состава проводимых работ и выполненных событий.

Оптимизация сетевого графика предполагает уменьшение общей длительности выполнения комплекса работ до минимальной величины, либо до величины, соответствующей заданному сроку. Расчет сетевого графика заключается в нахождении оптимального критического пути и определение резервов времени для работ на этом пути.

Модель проекта включает в себя следующие исходные данные:

- структурная таблица работ (то есть перечень работ с номерами $1, \ldots, n$);
- t_0 — сроки выполнения всех работ (проекта);
- C — стоимость проекта.

В таблице каждая работа имеет номер n и связана с $a(i,j)$ — минимальным временем выполнения и $b(i,j)$ — максимальным временем выполнения.

Оптимизация сетевого графика происходит после нахождения критического пути $t_{\text{кр}}$ и резервов времени выполнения работ. Методы оптимизации сетевого графика показаны на рис. 1.

Оптимизацию сетевого графика можно условно разделить на частную и комплексную. К частной оптимизации можно отнести минимизацию времени выполнения комплекса работ при заданной его стоимости и минимизацию стоимости комплекса работ при заданном времени выполнения проекта. К комплексной оптимизации относится нахождение оптимального соотношения величин стоимости и сроков выполнения проекта в зависимости от заданных целей, ставящихся при его реализации.

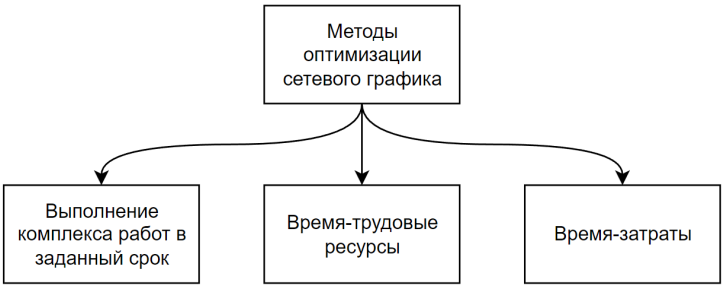

Рис. 1. Методы оптимизации сетевого графика.

1.1. Оптимизация по методу «Выполнение комплекса работ в заданный срок»

Оптимизация сетевой модели по этому методу предполагает уменьшение общей длительности выполнения комплекса работ до минимальной величины, либо до величины, соответствующей заданному сроку.

Время $t_{\text{кр}}$ соответствует времени первоначально рассчитанного критического пути в сетевом графике.

Если первоначально рассчитанное критическое время выполнения работ $t_{\text{кр}}$ может превышать требуемое время выполнения комплекса работ t_0 ($t_{\text{кр}} > t_0$), то возникает отрицательный резерв времени. Однако осмысленное вложение y единиц средств в работу под номером n, как правило, приводит к выполнению ее за время $t_{i1} < t_i$. Тогда возникает задача: найти такое количество дополнительных средств, выделяемых на каждую, или некоторые отдельные, работы с номерами n $(y_1, y_2, \ldots, y_n)$, чтобы общий объем дополнительных средств на работы при ограничениях $t_1+t_2+\cdots+t_n \leqslant t_0$ был минимальным. Формально указанное требование можно записать в виде выражения (1):

$$y_1 + y_2 + \cdots + y_n \Big|_{t_1+t_2+\cdots+t_n} \Rightarrow \min\,. \tag{1}$$

При изменении сетевого графика по данному критерию необходимо постоянно проверять остальные пути графика, так как в связи с сокращением данного критического пути, могут измениться резервы

времени на другие. Это может привести к возникновению новых критических работ, и тогда оптимизация может быть бесполезной, а в отдельных случаях, приводить к увеличению потребного времени.

1.2. Метод оптимизации «Время – трудовые ресурсы»

Используя данный метод, можно более точно распределить трудовые ресурсы по периодам времени. Для каждой специальности отдельно распределяется численность исполнителей для всех календарных периодов и ведется расчет их потребности

$$\sum_{k=1}^{e} Ч_k, \tag{2}$$

где

- e — количество работ, потребляющих данный ресурс и попадающих в данный период времени;
- k — номер специальности.

Сетевой график при данном методе оптимизации строится по шкале времени. Работы отображаются сплошными стрелками в масштабе времени их свершения по ранним или поздним срокам свершения $(t^{p.н}_{(i-j)}; t^{p.о}_{(i-j)}; t^{п.н}_{(i-j)}; t^{п.о}_{(i-j)})$, где резервы времени работ первого $(r'_{i=j})$ и второго $(r''_{i=j})$ вида, отображаются волнистыми линиями

Далее находится суммарная потребность в трудовых ресурсах

$$\sum_{k=1}^{e} Ч_k, \tag{3}$$

По каждой специальности $(1, \ldots, k)$ по периодам времени отдельно суммируются количество исполнителей в графах по ранним или поздним срокам выполнения проекта.

Оптимизацию по критерию «время – трудовые ресурсы» следует применять при соблюдении определенных условий: специалисты, выполняющие параллельные работы, должны быть взаимозаменяемые; группы специалистов должны быть подчинены одному руководителю. В случае несоблюдения условий организация работ и сокращение

сроков выполнения могут потребоваться дополнительные ресурсы (работников и средств) на работы критического пути.

Оптимизация по критерию «трудовые ресурсы» ведет к увеличению продолжительности выполнения отдельных работ в пределах резервов времени. Увеличение продолжительности работ до некоторых пределов уменьшает затраты на их выполнение. Возникает возможность при найденном критическом пути $t_{\text{кр}}$ использовать резервы времени некритических работ $(r''_{i=j})$ и $(r'_{i=j})$ и получить сетевой график с минимальными денежными затратами на весь комплекс работ.

Так же возможно за счет увеличения затрат на работы критического пути сократить время выполнения этого пути, тем самым сократить сроки выполнения всего комплекса работ.

При оптимизации сетевого графика учитывается не только сроки выполнения работ и численность исполнителей, но и стоимость работ. При определении стоимости работ учитывается все проводимые работы.

1.3. Метод оптимизации «Время – затраты»

Метод оптимизации «время – затраты» заключается в установлении зависимости между производительностью $t_{(i-j)}$ и стоимостью $C_{(i-j)}$ работ с целью их оптимизации.

Используя метод «Время – затраты», предполагают, что сокращение продолжительности работы пропорционально возрастанию ее стоимости. Главная цель оптимизации по данному методу — сократить продолжительность критического пути $t_{\text{кр}}$.

Каждая работа с индексами (i, j) характеризуется некоторой продолжительностью $t_{(i,j)}$, которая лежит в пределах

$$a_{(i,j)} \leqslant t_{(i,j)} \leqslant b_{(i,j)}, \tag{4}$$

где

- $a_{(i,j)}$ — минимально возможная (экстренная) продолжительность выполнения комплекса работы с индексами (i, j);
- $b_{(i,j)}$ — максимальная продолжительность выполнения работ (i, j).

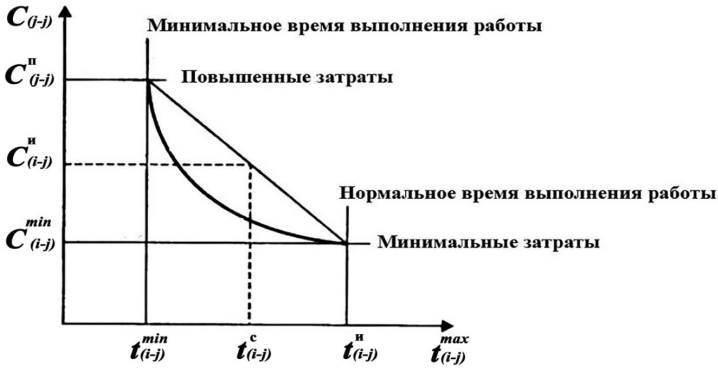

Рис. 2. Оптимизация сетевого графика по методу «время – затраты».

Для каждой работы задается минимальная величина затрат $C^{\min}_{(i,j)}$ и минимальное время выполнения $t^{\min}_{(i,j)}$ работы.

Определяется пара оценок, которые позволяют оценить максимальное сокращение затрат.

Рассчитывается вторая пара оценок, которая позволяет максимально сократить время выполнения комплекса работ.

Для нахождения величины затраты $C^{И}_{(i,j)}$ на графике проводят аппроксимирующую кривую, соединяющую найденные пары оценок (рис. 2).

По полученной кривой оценивают размеры увеличения расходов при сокращении сроков выполнения работ или увеличение времени выполнения работ при уменьшении затрат.

Величина затрат, необходимая для выполнения работ в сокращенное время, находится по формуле:

$$C^{И}_{(i,j)} = \frac{(C^{\text{п}}_{(i,j)} - C^{\min}_{(i,j)})(t^{o}_{(i,j)} - t^{c}_{(i,j)})}{t^{o}_{(i,j)} - t^{\max}_{(i,j)}}, \tag{5}$$

где

- $C^{И}_{(i,j)}$ — величина затрат, необходимая для выполнения (i,j) работ в сокращенное время $t^{c}_{(i,j)}$;

A	B	C	D
1	2	9	17
2	3	8	13
...	...	...	...
28	29	4	13

Таблица 1. Исходные данные.

- $C^{\text{п}}_{(i,j)}$ — повышенные размеры денежных затрат для выполнения (i,j) работ;
- $C^{\min}_{(i,j)}$ — минимально возможная величина затрат для выполнения (i,j) работ;
- $t^{o}_{(i,j)}$ — ожидаемое время выполнения, при котором может быть достигнуто $C^{\min}_{(i,j)}$ для (i,j) работы;
- $t^{\max}_{(i,j)}$ — максимальное время выполнения (i,j) работы;
- $t^{c}_{(i,j)}$ — сокращенное время выполнения работ для (i,j) работы.

2. Математическая модель описания построения и оптимизации сетевых графиков

Рассмотрим задачу оптимизации сетевого графа на примере проекта с 29 работами (табл. 1), где

- A — начало работы;
- B — конец работы;
- C — ранний срок выполнения работы;
- D — поздний срок выполнения работы.

Для автоматизированной обработки разработана компьютерная программа в среде MATLAB.

Пользователь перед началом моделирования формирует исходные данные и составляет таблицу для программной обработки. Данные представляются в виде отсортированного, в порядке возрастания списка работ, csv-файла.

С использованием программы проводятся расчеты ожидаемого времени выполнения каждой работы. Считается, что величина искомого времени распределена в пределах интервала от a до b по равномерному закону:

$$f_X(x) = \begin{cases} \dfrac{1}{b-a}, & \text{если } x \in [a,b], \\ 0, & \text{если } x \notin [a,b]. \end{cases} \tag{6}$$

Находят критический путь в сетевом графе. Для нахождения критического пути достаточно перебрать все пути и выбрать тот, или те из них, которые имеют наибольшую суммарную продолжительность выполнения работ.

Далее убеждаются, что комплекс работ будет выполнен в срок. Для этого находят оценку вероятности закончить проект в срок:

$$P(t_{\text{кр}} \leqslant T) = \frac{1}{2} + \frac{1}{2}\Phi\left(\frac{T - \bar{t}_{\text{кр}}}{\sigma_{\text{кр}}}\right), \tag{7}$$

где:

- $t_{\text{кр}}$ — длина (время) критического пути;
- Φ — значения интеграла вероятности Лапласа;
- T — максимальный срок выполнения проекта;
- $\bar{t}_{\text{кр}}$ — средняя длина (время) критического пути;
- $\sigma_{\text{кр}}$ — среднеквадратичное отклонение длины критического пути.

Если в задаче T неизвестно, то его находят по формуле (8):

$$T = \bar{t}_{\text{кр}} + z_\beta \cdot \sigma^2_{\text{кр}}, \tag{8}$$

где z_β — нормальное отклонение случайной величины, определяемое с помощью функции Лапласа.

Квадрат σ определяется по формуле (9):

$$\sigma^2(i,j) = \left[\frac{t_n(i,j) - t_o(i,j)}{6}\right]^2, \tag{9}$$

где:

- $t_n(i,j)$ — пессимистическая оценка;
- $t_o(i,j)$ — оптимистичная оценка.

Если $P(t_{\text{кр}} \leqslant T) \leqslant 0{,}3$, то считают, что с высокой вероятностью, работа не будет закончена в срок и нужно прибегать к перераспределению ресурсов, пересмотру состава работ, модернизации или отмене проекта.

Если $0{,}3 < P(t_{\text{кр}} \leqslant T) < 0{,}8$, то считают, что успех закончить работы в срок вероятен.

Если $P(t_{\text{кр}} \leqslant T) \geqslant 0{,}8$, то считают, что работа будет закончена в срок с высокой вероятностью.

Следующим этапом работы является оптимизация графа по одному из описанных выше методов. В качестве примера, выполним оптимизацию по методу «время – трудовые ресурсы».

На рис. 3 приведен результат работы компьютерной программы, реализующей алгоритм оптимизации по методу «время – трудовые затраты». В центре изображен сетевой граф с критическим путем. Правее от него статистические оценки и ожидаемое время выполнение проекта. Слева входные данные, которые пользователь сам составляет в документе формата .csv и указывает данное название в поле «Введите название документа». Справа в таблице отображены выходные данные, где указано оптимальное время выполнения каждой работы.

Основными результатами оптимизации являются:

- ожидаемое время выполнения комплекса работ (155 дней при изначально установленных 160 дней);
- ожидаемая стоимость (334,3 тысяч рублей при изначально выделенных 350 тысячах рублей).

Заключение

В настоящей работе:

- проанализированы основные методы оптимизации сетевых графиков в условиях неопределенности;

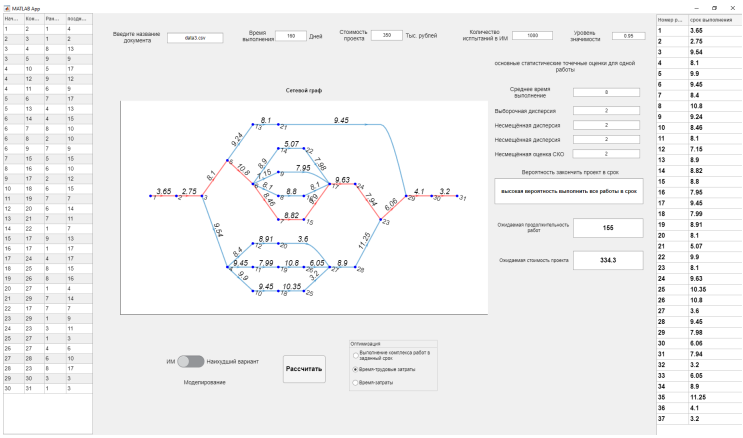

Рис. 3. Результат оптимизации по методу «время – трудовые ресурсы».

- исследована математическая модель построения и оптимизации сетевых графиков и на ее основе разработана компьютерная программа оптимизации сетевого графика в среде MATLAB;
- с использованием программы проведены расчеты ожидаемого времени выполнения и выполнена оптимизация сетевого графика, в результате которой получены оценки ожидаемого времени выполнения комплекса работ и стоимости проекта.

Это позволяет значительно ускорить выполнение проекта, сократить расходы и достичь более высоких результатов в условиях быстро развивающегося рынка.

Разработанную программу можно применять в различных областях деятельности людей: строительство объектов, освоение нового производства продукции в промышленных/торговых компаниях, реструктуризации производства.

Список литературы

[1] Зенкин, А. А. Методы и задачи сетевого планирования : учебное пособие. — М. : КНОРУС, 2021. — 206 с.

[2] Исследование операций в экономике : учебник / Н. Ш. Кремер, Б. А. Путко, И. М. Тришин, М. Н. Фридман. — М. : Издательство Юрайт, 2017. — 438 с.

[3] Капулин, Д. В. Автоматизация планирования мелкосерийного производства сетевыми методами / Д. В. Капулин, М. В. Винниченко, Д. И. Винниченко // Прикладная информатика. — 2016. — Т. 11, № 6 (66). — С. 6–18.

[4] Катаргин, Н. В. Оптимизация сетевого графика выполнения комплекса работ // Управленческие науки. — 2012. — № 1. — С. 87–93.

Библиографическая ссылка

Руденок, М. А. Построение и оптимизация сетевых графиков комплексов работ в условиях неопределенности // Студенческая конференция факультета ПМиК. Сборник трудов. — Тверь : ТвГУ, 2023. — С. 56–66.

Сведения об авторах

Руденок Михаил Александрович
Студент магистратуры
Направление «Прикладная информатика»

УДК 519.1

Реализация эффективных алгоритмов поиска оптимальных потоков в сетях

Савельев С. А.

Тверской государственный университет

Аннотация. В данной статье предоставлен сравнительный анализ алгоритмов решения задачи о максимальном потоке на различных видах сетей. Реализованы модификации алгоритма Диница. Модификация заключается в различных способах поиска пути в сети. Приведены графики, сравнивающие эффективность алгоритмов.

Введение

Задача о максимальном потоке является классической и имеет множество применений. Значимость задачи о максимальном потоке все время возрастает вместе со строительством трубопроводов, новых дорог, роста пользователей всемирной сети и любых других сетей. В повседневной жизни мы часто сталкиваемся с различными видами сетей: телефонными, автомобильными, железнодорожными, производственными, компьютерными и другими сетями. В каждой из этих сетей, и во многих других, перед нами стоит задача переместить некоторое количество любого рода «вещества» из одного пункта назначения в другой, и сделать это максимально быстро, насколько это возможно. За последние десятилетия теория графов превратилась в один из наиболее бурно развивающихся разделов математики. Это вызвано запросами стремительно расширяющейся области приложений. В теоретико-графовых терминах формулируется большое число задач, связанных с дискретными объектами. Рассмотрим примеры некоторых практических задач.

1) «Транспортные» задачи, в которых вершинами графа являются пункты, а ребрами — дороги (автомобильные, железные и др.) или другие транспортные (например, авиационные) маршруты. Другой пример — сети снабжения (энергоснабжения, газоснабжения, снабжения товарами и т. д.), в которых вершинами

являются пункты производства и потребления, а ребрами — возможные маршруты перемещения (линии электропередач, газопроводы, дороги и т. д.). Соответствующий класс задач оптимизации потоков грузов, размещения пунктов производства и потребления и т. д., иногда называется задачами обеспечения или задачами о размещении. Их подклассом являются задачи о грузоперевозках.

2) «Технологические задачи», в которых вершины отражают производственные элементы (заводы, цеха, станки и т. д.), а дуги — потоки сырья, материалов и продукции между ними, заключаются в определении оптимальной загрузки производственных элементов и обеспечивающих эту загрузку потоков.

3) Обменные схемы, являющиеся моделями таких явлений как бартер, взаимозачеты и т. д. Вершины графа при этом описывают участников обменной схемы (цепочки), а дуги — потоки материальных и финансовых ресурсов между ними. Задача заключается в определении цепочки обменов, оптимальной с точки зрения, например, организатора обмена и согласованной с интересами участников цепочки и существующими ограничениями.

4) Управление проектами. С точки зрения теории графов проект — совокупность операций и зависимостей между ними. Хрестоматийным примером является проект строительства некоторого объекта. Совокупность моделей и методов, использующих язык и результаты теории графов и ориентированных на решение задач управления проектами, получила название календарно-сетевого планирования и управления (КСПУ).

1. Необходимые понятия

Сеть представляет собой граф, который в свою очередь состоит из вершин и ребер, и в котором отмечены «источник» — начальная вершина и «сток» — конечная вершина сети. По ребрам сети, как уже было сказано, может перемещаться некоторое «вещество», называемое потоком. Сети могут иметь ограничения. Примерами таких ограничений могут служить: для транспортных — максимальная пропускная способность автомобильной дороги или максимальное

сечение трубопровода, который соединяют различные заводы. Для производственных — максимальное количество продукции, которую может перевезти одно транспортное средство.

Определение 1. *Сеть $N = (V, E, s, t, c)$ — это связный ориентированный граф без петель $G = (V, E)$, у которого*

1) *имеется одна вершина s, в которую не входят дуги (источник);*
2) *имеется одна вершина t, из которой не выходят дуги (сток);*
3) *каждой дуге $e = (u, v)$ из E сопоставлена ее пропускная способность $c(e) \geqslant 0$ (см. [3]).*

Определение 2. *Поток f в сети N — это функция $f\colon E \to \mathbb{R}$ такая, что*

1) *$0 \leqslant f(e) \leqslant c(e)$ для каждой дуги e из E;*
2) *Для каждой вершины v из $V \setminus \{s, t\}$*

$$\sum\{f(e) \mid e \text{ входит в } v\} = \sum\{f(e) \mid e \text{ выходит из } v\}.$$

Здесь $\mathbb{R}$ — множество действительных чисел. Величиной потока f называется значение $\mathrm{val}(f) = \sum f(s, u)$. Поток с наибольшим значением величины называется максимальным.

Определение 3. *Пусть N — сеть, f — поток в ней. Увеличивающий путь (увеличивающая цепь) из s в v относительно f — это любой путь P из s в v в неориентированном графе, полученном из G игнорированием направлений дуг, в котором:*

1) *для любой дуги (u, v), проходимой в P в прямом направлении (прямой дуги), $f(u, v) \leqslant c(u, v)$;*
2) *для любой дуги (v, u), проходимой в P в обратном направлении (обратной дуги), $f(u, v) > 0$.*

Увеличивающий путь из s в t будем просто называть увеличивающем путем.

2. Виды сетей

Ациклический граф. Ациклический граф — граф без циклов.

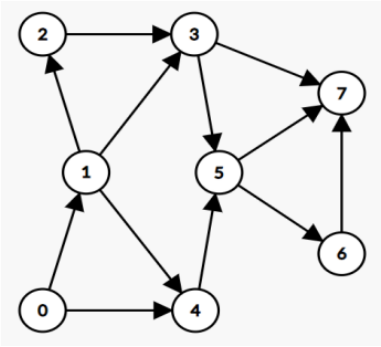

Рис. 1. Ациклический граф.

Граф «решетка». Регулярная решетка — это плоский граф, все грани которого являются равными правильными многоугольниками. Существует три типа таких решеток, в которых грани являются квадратами, правильными треугольниками или правильными шестиугольниками. В дальнейшем в работе рассматриваются только решетки с квадратными гранями, причем ребра ориентируются таким образом, чтобы не было циклов (рис. 2).

3. Алгоритмы поиска пути в графе

Поиск в глубину. Стратегия поиска в глубину, как и следует из названия, состоит в том, чтобы идти «вглубь» графа, насколько это возможно. Алгоритм поиска описывается рекурсивно: перебираем все исходящие из рассматриваемой вершины ребра. Если ребро ведет в вершину, которая не была рассмотрена ранее, то запускаем алгоритм от этой не рассмотренной вершины, а после возвращаемся и продолжаем перебирать ребра. Возврат происходит в том случае, если в рассматриваемой вершине не осталось ребер, которые ведут в не рассмотренную вершину.

Поиск в ширину. Алгоритм поиска в древовидной структуре дан-

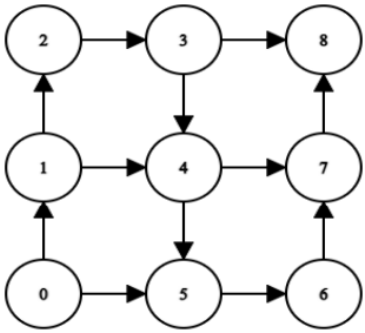

Рис. 2. Граф «решетка».

ных узла, удовлетворяющего заданному свойству. Он начинается с корня дерева и исследует все узлы на текущей глубине, прежде чем перейти к узлам на следующем уровне глубины.

Алгоритм Дейкстры. Алгоритм на графах, изобретенный нидерландским ученым Эдсгером Дейкстрой в 1959 году. Находит кратчайшие пути от одной из вершин графа до всех остальных. Алгоритм работает только для графов без ребер отрицательного веса. В качестве модификации на каждом шаге алгоритма выбиралось ребро с максимальной пропускной способность.

4. Алгоритм поиска максимального потока

Алгоритм Диница — полиномиальный алгоритм для нахождения максимального потока в транспортной сети, предложенный в 1970 году советским математиком Ефимом Диницем. Алгоритм представляет собой несколько фаз. На каждой фазе сначала строится остаточная сеть, затем по отношению к ней строится слоистая сеть (обходом в ширину), а в ней ищется произвольный блокирующий поток. Найденный блокирующий поток прибавляется к текущему потоку, и на этом очередная итерация заканчивается.

Описание алгоритма.

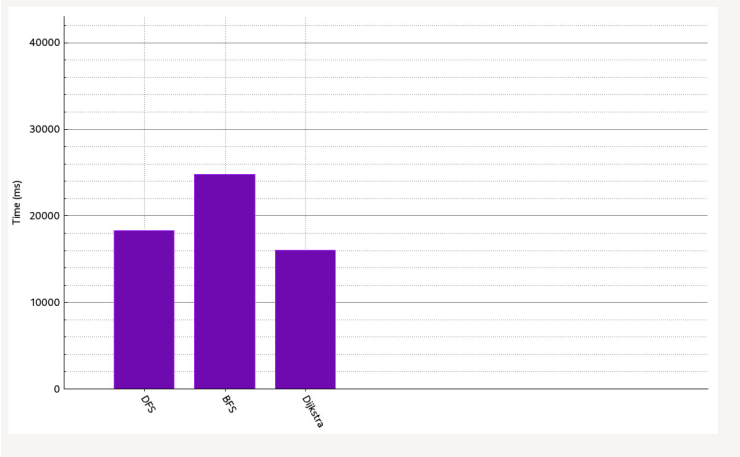

Рис. 3. Результат работы на ациклических сетях.

1) Обнуляем все потоки сети G.
2) Построим слоистую сеть GL из остаточной сети Gf. Если в ней нет пути из s в t остановиться и вывести f.
3) Найдем блокирующий поток в GL.
4) Дополним поток найденным блокирующим потоком f и перейдем к шагу 2.

Далее описаны результаты экспериментов с различными сетями. Эксперименты проводились на 100 графах размерностью 500 вершин. На рис. 3 представлены сравнительные графики работы алгоритма Диница с: 1) поиском в глубину; 2) поиском в ширину; 3) алгоритмом Дейкстры на ациклических сетях.

На рис. 4 представлены сравнительные графики на графах «решетка».

Заключение

В этой статье представлены результаты работы алгоритма Диница с различными реализациями поиска блокирующего потока в

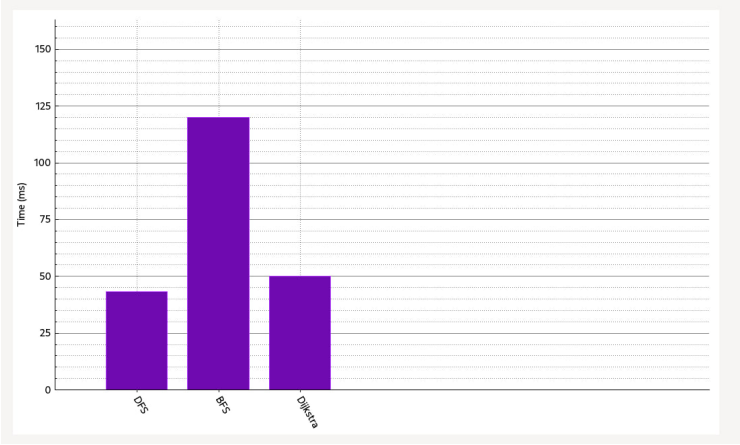

Рис. 4. Результат работы на сетях «Решетка».

сетях на ациклических графах и графах «решетках». Исходя из полученных результатов очевидно, что вид сети и способ поиска блокирующего пути в ней напрямую влияет на быстродействие алгоритма. При тестах на графах «решетках» можно видеть, что при увеличении «решетки» алгоритм Диница с поиском в глубину, срабатывает быстрее остальных, так как пути от истока в сток становятся длиннее, и поиск в глубину быстрее находит увеличивающий путь. Однако на ациклических графах, где длина путей от источника в сток не такая большая, оказывается, что быстрее искать путь с помощью алгоритмы Дейкстры, где на каждом шагу выбирается вершина для обхода, к которой ведет ребро с наибольшей пропускной способностью.

Список литературы

[1] Ахо, А. Структуры данных и алгоритмы / А. Ахо, Дж. Ульман, Дж. Хопкрофт. — М. : Вильямс, 2015. — 384 с.

[2] Дехтярь, М. И. Алгоритмические задачи на графах. — Тверь : Тверской государственный университет, 2014. — 118 с.

[3] Касьянов, В. Н. Графы в программировании: обработка, визуализация и применение / В. Н. Касьянов, В. А. Евстигнеев. — СПб. : БХВ-Петербург, 2003. — 1104 с.

[4] Подбельский, В. В. Практикум по программированию на языке Си. — М. : Финансы и статистика, 2004. — 600 с.

Библиографическая ссылка

Савельев, С. А. Реализация эффективных алгоритмов поиска оптимальных потоков в сетях // Студенческая конференция факультета ПМиК. Сборник трудов. — Тверь : ТвГУ, 2023. — С. 67–74.

Сведения об авторах

Савельев Сергей Андреевич
Студент магистратуры
Направление «Прикладная информатика»

УДК 004.93 + 004.932

Метод автоматизированной обработки сканированных анкетных документов

Сирукова И. С.

Тверской государственный университет

Аннотация. В данной статье рассматривается процесс решения задачи автоматизированной обработки сканированных анкетных документов. В ней описываются методы генерирования эталонных и искаженных тестовых данных, обнаружения особых точек на тестовых данных. Также рассматривается приведение искаженных анкет к эталонному виду и получение данных из них.

Введение

В настоящее время мы имеем тенденцию к увеличению необходимости в методах получения данных с бумажных носителей. Ведь данные, сохраненные на электронных носителях не подвергаются старению, не нуждаются в физическом хранении, легко редактируются, копируются и распространяются, а также, благодаря различным облачным сервисам, могут быть доступны для просмотра и изменения на различных устройствах и в любое время.

Сейчас, несмотря на существование возможности проводить анкетирование сразу в электронном формате в режиме удаленного доступа, все еще достаточно часто при проведении социологических, психологических или других исследований при опросе респондентов, а также на экзаменах, используют бумажные анкеты, в которые заносятся ответы. Для получения данных из этих анкет существует 2 общих способа оцифровки: ручной, когда оператор вводит данные вручную и сохраняет их в базе, и автоматизированный, когда документ сканируется с дальнейшей его обработкой при помощи различных программных средств. Если первый способ достаточно затратный по времени и труду, а также подвержен человеческому фактору, то второй способ не обладает перечисленными недостатками и позволяет свести к минимуму участие в процессе обработки оператора, что и уменьшает вероятность появления ошибок.

Одним из первых этапов оцифровки документа является его сканирование (фотографирование), после происходит обработка изображения. Однако, на практике часто получается так, что оцифровка происходит неровно, анкета искажается поворотами и перспективой. В данной статье будет описан метод автоматизированной обработки сканированных (сфотографированных) анкетных документов, в котором учтены эти моменты. Пункты, рассматриваемые при работе над методом:

1) генерация эталонной анкеты;
2) генерация тестовых данных;
3) обнаружение меток на изображениях;
4) приведение сканированных анкет к эталонному виду.

1. Генерация эталонной анкеты

Эталонная анкета (рис. 1) в дальнейшем будет использоваться как для искусственной генерации тестовых данных, так и для сопоставления с ней искаженных изображений с целью нахождения точек соответствия, которыми являются квадратные маркеры по краям листа. Они позволяют определять примерные границы анкеты на изображении, а также ее ориентацию.

Исходные координаты и размеры элементов относительно листа задаются в качестве входных данных для метода генерации.

Сгенерировать анкету можно с различным dpi (dot per inch). Таким образом, происходит моделирование различных разрешающих возможностей различных устройств ввода (камеры, сканера и т. д.). Подразумевается, что чем больше разрешающая способность устройства, тем точнее электронное отображение данных.

При генерации эталонной анкеты происходит пересчет исходных координат и размеров, заданных в программе, из миллиметров в пиксели по следующей формуле:

$$pixels = \frac{mm}{25{,}4} \cdot dpi, \tag{1}$$

где mm — исходные данные, dpi — заданное разрешение, в котором «считывается» анкета, а константа 25,4 — количество пикселей в миллиметре.

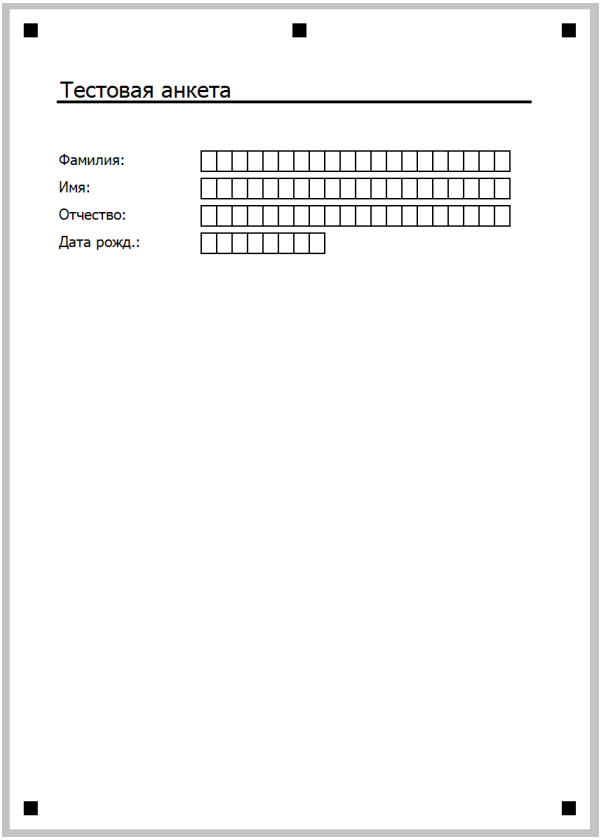

Рис. 1. Сгенерированная эталонная анкета.

На анкете имеются поля для ввода информации, из которых можно будет вырезать сегменты с рукописными буквами для дальнейшей их обработки.

2. Генерация тестовых данных

Для генерации тестовых данных был реализован кастомный рендер с усеченным функционалом, необходимым для генерации

$$P_r = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & r \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

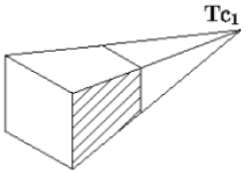

Рис. 2. Одноточечная (центральная) перспектива.

искаженных относительно эталонных изображений при помощи масштабирования, переноса, поворотов вокруг осей абсцисс, ординат, аппликат, а также перспективных преобразований. Для моделирования работы считывающего устройства будем использовать одноточечное перспективное (центральное) проектирование, где r — это обратная заданному фокусу величина, отложенная по оси аппликат (рис. 2).

Таким образом моделируется различное положение анкеты относительно камеры.

Опишем последовательность действий при работе рендера [1].

1) Изначально будущая анкета представляет из себя экземпляр модели листа и задается двумя треугольниками, состоящими из четырех вершин.
2) Далее координаты этих вершин переводятся в пространство модели таким образом, чтобы середина листа совпадала с центром системы координат.
3) Теперь можно исказить объект при помощи изменения масштаба (матрица S_{xyz}), поворотов вокруг осей (матрицы R_x, R_y, R_z) и сдвигов (матрица T_{xyz}):

$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) & 0 \\ 0 & -\sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{2}$$

$$R_y = \begin{bmatrix} \cos(\psi) & 0 & -\sin(\psi) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(\psi) & 0 & \cos(\psi) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3}$$

$$R_z = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 & 0 \\ -\sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{4}$$

$$T_{xyz} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ dx & dy & dz & 1 \end{bmatrix}, \tag{5}$$

$$S_{xyz} = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ dx & dy & dz & 1 \end{bmatrix}. \tag{6}$$

4) Так как нам интересен лист целиком, то сдвигаем модель в глобальном пространстве с таким расчетом, чтобы весь объект находился перед камерой.

5) Применяем сначала перспективное преобразование к объекту, а затем ортографическое с $z = 0$, тем самым получая 2D отображение нашего объекта на экранной плоскости.

6) Текстурируем объект. Текстурирование происходит для каждого треугольника объекта при помощи линейной интерполяции. Каждый треугольник делится точкой D на два треугольника, для точек которых вычисляются соответствующие точки текстуры (u, v). В качестве текстуры выступает эталонное изображение.

3. Обнаружение меток на изображении

Одним из подходов к сопоставлению изображений компьютером является обнаружение так называемых особых точек на нем.

Особой точкой изображения называется такая точка, окрестность которой $o(m)$ отличима от окрестности любой другой точки изображения $o(n)$ в некоторой другой окрестности особой точки $o_2(m)$. Такие точки должны удовлетворять приципам отличимости, инвариантности, стабильности, уникальности и интерпретируемости.

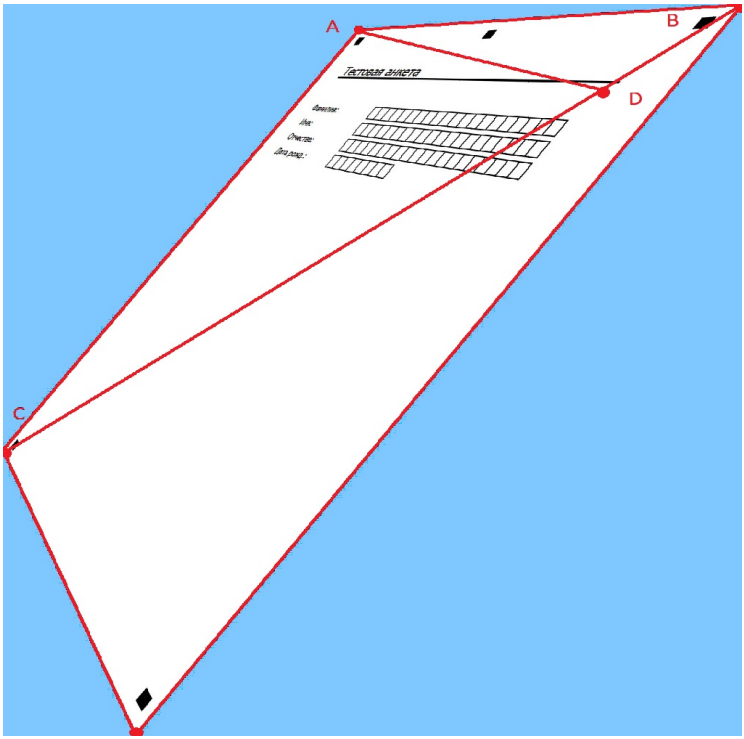

Рис. 3. Результат работы рендера.

Процесс сравнения изображений разделяется на три этапа. Первый этап — нахождение множества особых точек с помощью методов, называемых детекторами. Данные методы обеспечивают инвариантность нахождения одних и тех же точек относительно преобразований изображения. Однако, недостаточно использования лишь детектора, так как результатом его работы является множество координат особых точек, которые на каждом изображении различны. Для этого на втором этапе происходит построение дескрипторов. Дескриптор — это описание точки, уникально идентифицирующее ее среди множества всех точек. Дескриптор должен обеспечивать инвариантность нахождения соответствий между точками относительно

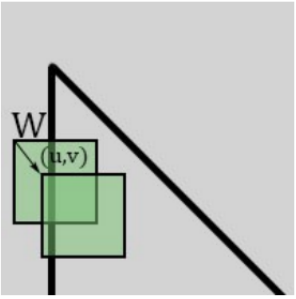

Рис. 4. Используемое окно.

преобразований изображения. Некоторые методы выполняют сразу обе задачи, детектирование особых точек и построение дескрипторов. И третий этап заключается в сравнении дескрипторов и поиске точек, совпадающих на обоих изображениях.

В ходе изучения различных детектеров особых точек был выбран угловой детектор Харриса в связи с особенностями входных данных (ярко выраженные углы на изображениях), поступающих на обработку разрабатываемого ПО, достаточно хорошей скорости по сравнению с другими рассмотренными детекторами, а также его инвариантности к поворотам [2].

Метод Харриса основан на детекторе Моравеца и является его улучшением, так как для него характерна анизотропия по всем направлениям. Харрис и Стефенс предложили рассмотреть производные по множеству направлений.

Рассмотрим окно W (обычно его размер выбирается равным 5×5 пикселей, но может быть другим в зависимости от размера изображения) с центром в точке (x, y), а также сдвиг этого окна на (u, v) (изображено на рисунке 4). Тогда взвешенная сумма квадрата разностей между окном W и окном W, сдвинутым на (u, v) (то есть изменение окрестности точки (x, y) при сдвиге на (u, v)) равна

$$E(u, v) = \sum_{(u,v) \in W} w(x, y)(I(x + u, y + v) - I(x, y))^2, \qquad (7)$$

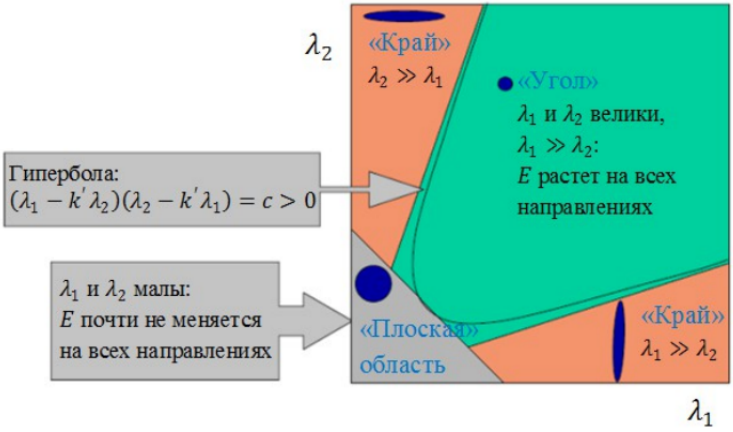

Рис. 5. Расположение собственных значений.

$$E(u,v) \approx \sum_{(u,v)\in W} w(x,y)(I_x(x,y)u - I_y(x,y)v)^2, \tag{8}$$

где $w(x,y)$ — весовая функция (обычно используется функция Гаусса или бинарное окно), M — автокорреляционная матрица:

$$M = \sum_{(u,v)\in W} w(u,v) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}. \tag{9}$$

Угол характеризуется большими изменениями функции $E(x,y)$ по множеству всех направлений (x,y), что эквивалентно большим по модулю собственным значениям матрицы M. Расположение собственных значений приведено на рис. 5.

Так как считать собственные значения напрямую затратно с точки зрения ресурсов, Харрис и Стефен предложили меру отклика:

$$R = \det M - k(\operatorname{tr} M)^2 > k, \tag{10}$$

где k — эмпирическая константа, $k \in [0{,}04; 0{,}06]$. Таким образом, значение R неотрицательно для угловых особых точек. Затем производится фильтрация точек по R (то есть те точки, у которых

значение R меньше определенного порога, исключаются из рассмотрения). Далее находятся локальные максимумы функции отклика по окрестности заданного радиуса и выбираются в качестве особых точек.

4. Приведение анкет к эталонному виду

В предыдущем пункте были найдены особые точки и точки соответствия эталонного и обрабатываемого изображений. Теперь, необходимо найти гомографию между этими изображениями.

Гомография — это преобразование, которое отображает точки одного изображения в точки соответствия другого изображения. Чтобы рассчитать гомографию между двумя изображениями, нужно знать как минимум 4 точки соответствия между двумя изображениями.

Для каждой пары соответствующих точек на двух изображениях требуется отыскать гомографию H:

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = H \times \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \times \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}. \qquad (11)$$

Функционал нахождения гомографий предоставляет библиотека с открытым исходным кодом OpenCV, специализирующаяся на алгоритмах компьютерного зрения и обработке изображений. Используем findHomography для нахождения гомографии и warpPerspective, чтобы развернуть обрабатываемое изображение в плоскость эталонной анкеты.

Так как для эталонной анкеты нам известны координаты полей ввода информации, то теперь мы можем вырезать пиксели внутри ячеек из приведенной к эталонной анкеты и сохранить полученные сегменты для последующей обработки с целью распознавания символов.

Заключение

В процессе работы сопутствующих исследований были изучены аффинные и перспективные преобразования, способы нахождения опорных точек на изображениях, а также функционал и возможности библиотеки OpenCV.

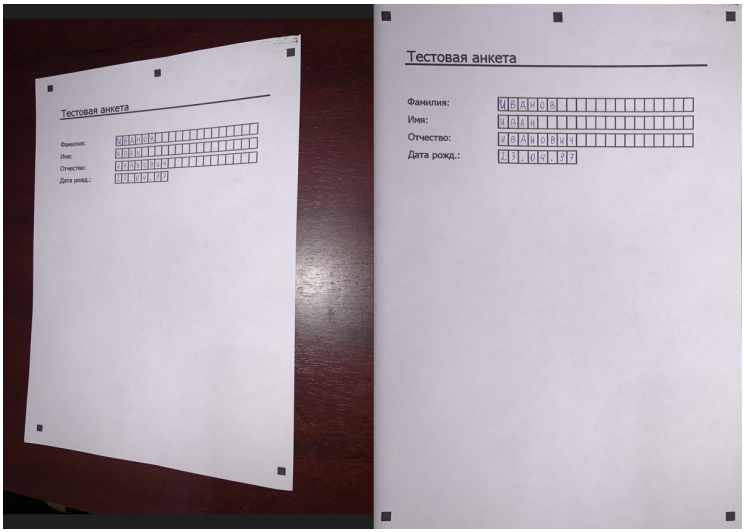

Рис. 6. Пример обратного искажения обрабатываемого изображения.

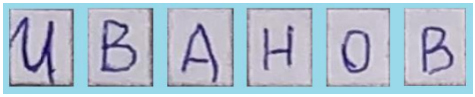

Рис. 7. Пример полученных промежуточных данных из анкеты.

Итоговым результатом работы стала реализация ПО, позволяющего генерировать тестовые данные с заданным разрешением, предоставляющего функционал для искажения данных аффинными и перспективными преобразованиями, поиска опорных точек, выравнивания искаженных изображений, а также получения из этих изображений данных для дальнейшей обработки.

Список литературы

[1] Gambetta, G. Computer Graphics from Scratch : a Programmer's Introduction to 3D Rendering. — San Francisco, CA : No Starch Press, 2021. — 248 p.

[2] Harris, C. A Combined Corner and Edge Detector / C. Harris, M. Stephens // Proc. Fourth Alvey Vision Conference (Manchester, UK, September, 1988). — [S. l.] : Alvey Vision Club, 1988. — P. 147–151.

Библиографическая ссылка

Сирукова, И. С. Метод автоматизированной обработки сканированных анкетных документов // Студенческая конференция факультета ПМиК. Сборник трудов. — Тверь : ТвГУ, 2023. — С. 75–85.

Сведения об авторах

Сирукова Ирина Сергеевна
Студент магистратуры
Направление «Фундаментальная информатика и информационные технологии»

УДК 004.85

Алгоритмы жанровой классификации текста

Табалов А. А.

Тверской государственный университет

Аннотация. Данная статья представляет собой исследование, посвященное классификации текста с использованием семи различных моделей: наивный байесовский классификатор, метод опорных векторов, метод случайного леса, логистическая регрессия, метод k-ближайших соседей, рекуррентная нейронная сеть и сверточная нейронная сеть. В работе произведено сравнение этих моделей на основе точности классификации, времени обучения и других параметров. Результаты эксперимента позволяют определить наиболее эффективную модель для многоклассовой классификации текста.

Введение

Автоматическая классификация текста является важной задачей в области обработки естественного языка. С ее помощью можно эффективно обрабатывать большие объемы текстовых данных, проводить анализ и выявлять закономерности. В последние годы интерес к этой области стал расти в связи с развитием машинного обучения и искусственного интеллекта.

Классификация текста — одна из самых распространенных задач в автоматической обработке текста. Это процесс, в ходе которого каждый текстовый документ относится к одной из нескольких заранее определенных категорий. Классификация текста широко используется в различных областях, таких как медицина, право, финансы и др.

Цель данной статьи заключается в исследовании эффективности различных моделей для классификации текста: наивный байесовский классификатор, метод опорных векторов, метод случайного леса, логистическая регрессия, метод k-ближайших соседей, рекуррентная нейронная сеть и сверточная нейронная сеть. Будут рассмотрены различные метрики для оценки качества работы каждой модели,

включая точность классификации, скорость обучения и другие параметры.

Данная работа актуальна, так как обработка текстовых данных является неотъемлемой частью современного мира, и важно иметь эффективные инструменты для классификации больших объемов текстовой информации.

1. Постановка задачи

Математическая постановка задачи жанровой классификации текста заключается в определении функции, которая отображает пространство всех возможных текстов в пространство жанров.

Пусть есть множество текстовых документов $D = \{d_1, d_2, \ldots, d_n\}$ и множество жанровых категорий $C = \{c_1, c_2, \ldots, c_k\}$. Каждый текст d_i может быть представлен в виде вектора $x \in \mathbb{R}^m$, где m — количество признаков, используемых для описания текста. Каждая жанровая категория c_j может быть представлена вектором $y_j \in \{0,1\}^k$, где k — количество жанровых категорий. Этот вектор содержит 1 в j-м элементе и 0 во всех остальных элементах. Тогда задача жанровой классификации текста заключается в поиске функции $f\colon \mathbb{R}^m \to \{0,1\}^k$, которая отображает каждый текст x_i в вектор жанровых категорий y_j.

Функция f может быть определена различными методами, включая правила и эвристики, методы машинного обучения, а также комбинации этих методов. Важно, чтобы функция была обучена на достаточно большом и разнообразном наборе текстов для обеспечения точности и универсальности ее работы.

2. Этапы классификации текста

Первый этап жанровой классификации текста — это стандартизация и нормализация данных. Это необходимо, чтобы обеспечить единообразие текстовых данных и убрать все лишнее, что может влиять на точность классификации. На этом этапе мы можем провести следующие операции.

- Удаление пунктуации и специальных символов. Это позволяет убрать все символы, которые не являются буквами или

цифрами.

- Приведение текста к нижнему регистру. Это помогает избежать проблем, связанных с различным написанием слов с заглавной буквы или в нижнем регистре.

- Удаление стоп-слов. Стоп-слова — это слова, которые не несут никакой информации и могут быть исключены из текста. Примерами стоп-слов являются артикли, союзы и предлоги.

Второй этап жанровой классификации текста — это векторизация данных. Векторизация текста — это процесс преобразования текстовых данных в числовые векторы. Для этого часто используется мера TF-IDF. TF-IDF — это мера, которая учитывает важность слова в документе. Слова, которые часто встречаются в документе, но редко встречаются в других документах, имеют более высокий вес, чем слова, которые встречаются часто во всех документах. Другие меры векторизации текста, такие как мультимножество слов (Bag of Words) или представление слов в виде векторов (Word2Vec), также могут быть использованы для жанровой классификации.

После векторизации текстовых данных следующим этапом является определение модели для жанровой классификации. Существует множество моделей машинного обучения, которые можно использовать для классификации текстов. Некоторые из наиболее популярных моделей включают в себя:

- наивный байесовский классификатор (Naive Bayes Classifier);

- метод опорных векторов (Support Vector Machine, SVM);

- логистическая регрессия (Logistics Regression);

- метод ближайших соседей (K-Nearest Neighbors, KNN);

- случайный лес (Random Forest);

- нейронные сети (Neural Networks).

Выбор конкретной модели для жанровой классификации зависит от многих факторов, таких как размер и структура обучающей

выборки, количество классов, требования к скорости и точности работы модели и др.

Например, алгоритмы классификации, такие как наивный байесовский классификатор, случайный лес и метод опорных векторов, обладают высокой точностью классификации и относительно просты в реализации. Однако эти методы не учитывают контекст и последовательность слов в тексте, что может приводить к снижению точности классификации в некоторых случаях. В отличие от этого, нейронные сети, такие как рекуррентные и сверточные сети, могут учитывать контекст и последовательность слов, но требуют больше вычислительных ресурсов и экспертных знаний для их оптимальной настройки.

Таким образом, жанровая классификация текста является многопроцессным подходом, который включает в себя предобработку текста, векторизацию текстовых данных и определение модели для классификации. При правильном подходе и настройке параметров этот подход может дать высокую точность классификации.

3. Краткий обзор моделей

Опишем кратко некоторые методы классификации.

1) Модель наивного байесовского классификатора (Naive Bayes Classifier). Новый документ относится к классу

$$\arg\max \left(P(Q_k) \prod P(x_i \mid Q_k) \right),$$

где

- $P(Q_k) = \dfrac{\text{число документов класса } Q_k}{\text{общее количество документов}}$ — вероятность класса Q_k;
- $P\left(x_i \mid Q_k\right) = \dfrac{\alpha + N_{ik}}{\alpha M + N_k}$ — вероятность вхождения слова x_i в документ класса Q_k;
- N_k — количество слов, входящих во все документы класса Q_k;
- M — количество слов из обучающей выборки;

- N_{ik} — количество вхождений слова x_i во все документы класса Q_k;
- α — параметр для сглаживания.

2) K-ближайших соседей.

- Зафиксирован параметр k — число соседей.
- Вычисляется расстояние от нового документа до каждого документа обучающей выборки.
- Выбираются k документов с наименьшим расстоянием.
- Класс нового документа — класс, который встречается чаще всего среди k ближайших соседей.

3) Модель на основе метода опорных векторов (SVM).

- Главная цель SVM как классификатора — найти уравнение разделяющей гиперплоскости

$$w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_m x_m = 0$$

в пространстве $\mathbb{R}^m$.

- Общий вид преобразования объекта x в метку класса Y:

$$F(x) = \operatorname{sign}\left(w^T x \ - b\right),$$

где b — вектор смещения.

4) Логистическая регрессия. Обозначим через $P(y = 1 \mid x)$ вероятность того, что классифицируемый объект принадлежит классу 1 (то есть «положительному»), при условии значения признаков x. Пусть $w_0, w_1, \ldots, w_n$ — это веса модели, которые определяют важность каждого признака в процессе классификации, а $x_1, x_2, \ldots, x_n$ — значения признаков объекта, которые используются для определения его класса. Тогда вероятность вычисляется по формуле

$$P(y = 1 \mid x) = \frac{1}{1 + \exp\left(-w_0 - \sum\limits_{i=1}^{n} w_i x_i\right)}.$$

5) Случайный лес (Random Forest). Основная идея заключается в построении большого количества деревьев решений на основе случайно выбранных признаков и случайной выборки обучающих данных. Каждое дерево классифицирует объект на основе значения его признаков и «проголосовавшие» деревья определяют итоговый класс объекта.

6) Сверточная нейронная сеть для текстовой классификации (Convolutional Neural Network for Text Classification) — это модель, которая использует сверточные слои для обработки текста. Она состоит из нескольких слоев: входной слой, сверточный слой, операция подвыборки (max-pooling) и выходной слой. Сверточный слой находит признаки в тексте, а операция подвыборки (max-pooling) собирает эти признаки и передает их на выходной слой для классификации текста.

7) Рекуррентная нейронная сеть на основе LSTM (Long Short-Term Memory) — это модель, которая использует рекуррентные слои для обработки текста. Она состоит из нескольких слоев: входной слой, рекуррентный слой и выходной слой. Рекуррентный слой позволяет модели учитывать предыдущие контексты при обработке текущего входного слова.

4. Результаты

В исследовании использовалось два набора данных. Первый из них — набор новостей с сайта Lenta.ru, разбитый на 5 классов, общий объем данных составляет 25 тысяч новостных статей, на каждый класс приходится 5 тысяч размеченных примеров. Второй набор данных — набор книг, разбитый на 5 классов, но в связи с трудностью получения необходимого количества примеров для каждого класса набор данных не сбалансирован. Далее будет видно, как это повлияло на результат классификации. Общий объем данных для второго набора составляет порядка 10 тысяч экземпляров.

Введем обозначение для моделей с целью экономия места для таблиц.

1) Наивный байесовский классификатор — NBC.
2) K-ближайших соседей — KNN.

	NBC	KNN	SVM	LR	RF	RNN	CNN
Новости	92%	90%	94%	94%	92%	95%	97%
Книги	73%	67%	78%	78%	73%	80%	55%

Таблица 1. Точность моделей.

	NBC	KNN	SVM	LR	RF	RNN	CNN
Новости	2,5 с	3,5 с	3 с	4 с	36 с	27 с	27 с
Книги	50 с	52 с	53 с	54 с	150 с	35 с	35 с

Таблица 2. Время обучения.

	NBC	KNN	SVM	LR	RF	RNN	CNN
Новости	0,7 с	32,5 с	0,5 с	0,5 с	0,8 с	0,1 с	0,1 с
Книги	5 с	610 с	5 с	5 с	5 с	0,5 с	0,5 с

Таблица 3. Время предсказания.

3) Метод опорных векторов — SVM.
4) Логистическая регрессия — LR.
5) Случайный лес — RF.
6) Рекуррентная нейросеть — RNN.
7) Сверточная нейросеть — CNN.

В табл. 1, 2 и 3 приведены результаты вычислительных экспериментов.

Заключение

В результате исследования можно отметить, что все модели показывают достаточно высокий результат классификации, если данные были представлены в большом объеме, хорошо сбалансированы и правильно подобраны параметры для моделей. Важно отметить, что представленные модели могут быть доработаны, усовершенствованы и объединены в одно целое. Однако важно понимать, что жанровая классификация текста — это достаточно сложная задача, которая

может требовать большого количества ресурсов и времени для ее решения. Кроме того, результаты классификации могут зависеть от многих факторов, таких как особенности текста, способ представления данных, выбор алгоритмов и параметров модели. Поэтому при решении задач жанровой классификации текста важно проводить тщательный анализ данных, выбирать подходящие алгоритмы и параметры модели, а также оценивать качество результатов классификации на разных метриках.

Список литературы

[1] Уоссермен, Ф. Нейрокомпьютерная техника: Теория и практика. — М.: Мир, 1992. — 240 с.

[2] Hastie, T. The Elements of Statistical Learning. Data Mining, Inference, and Prediction / T. Hastie, R. Tibshirani, J. Friedman. New York : Springer New York, 2009. — 745 p.

Библиографическая ссылка

Табалов, А. А. Алгоритмы жанровой классификации текста // Студенческая конференция факультета ПМиК. Сборник трудов. — Тверь : ТвГУ, 2023. — С. 86–93.

Сведения об авторах

Табалов Артем Алексеевич
Студент магистратуры
Направление «Фундаментальная информатика и информационные технологии»

УДК 519.688

Маршрутизация в среде с циркулярными препятствиями

Титов А. Д.

Тверской государственный университет

Аннотация. Задача поиска кратчайшего пути появилась достаточно давно и, несмотря на множество подходов, алгоритмов и методов, существующих для ее решения, эффективность и возможность использования того или иного метода во многом зависит от постановки задачи и области применения. Задача поиска кратчайшего пути очень часто встречается на практике во многих научных направлениях, при решении проблем бизнеса и повседневной жизни, она нисколько не теряет своей актуальности и в настоящее время. В данной статье представлен алгоритм для построения по карте циркулярных препятствий графа и поиска по нему кратчайшего пути, который учитывает пересечения препятствий и переменный радиус исполнителя.

Введение

Темой работы является разработка метода маршрутизации в средах с циркулярными препятствиями, то есть задача поиска кратчайшего пути в таких средах. Данная математическая модель может быть получена, если рассматривать задачу поиска оптимального маршрута передвижения на плоскости с препятствиями для человека, автотранспортного средства или иного исполнителя. Препятствия представляют собой круги различных радиусов и могут пересекаться между собой. В дальнейшем данную плоскость будем называть картой циркулярных препятствий. В компьютерной графике можно прийти к такой модели, если использовать аппроксимацию точных границ объектов сложной формы с помощью сферы и применить операцию проекции на плоскость. Такой способ представления позволяет получить оптимальный маршрут, плавно обходящий препятствия, которые имеют сложную геометрию.

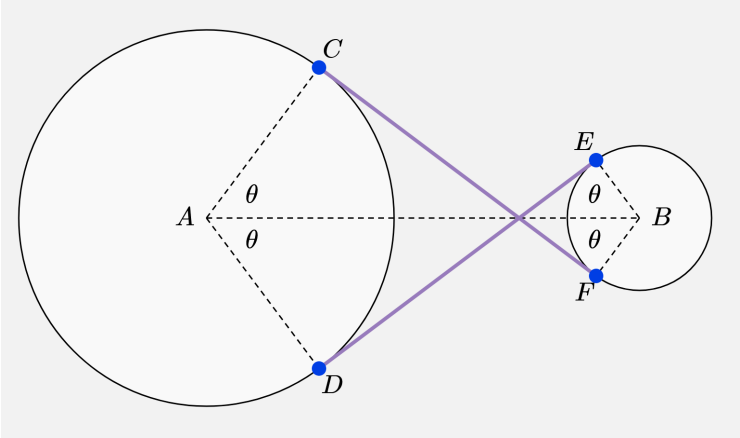

Рис. 1. График к «задаче ремня».

1. Построение графа

Входными данными для работы алгоритма является карта циркулярных препятствий, которая представляет собой множество кругов, заданных координатами центра и радиусом, а так же начальной и конечной точками для построения оптимального пути. Отдельно может быть задан радиус исполнителя, если он отличен от нуля.

Любой оптимальный путь сквозь лес циркулярных препятствий будет состоять из перемежающихся отрезков касательных и их дуг. Они будут являться ребрами графа, а их конечные точки будут вершинами. Отрезки касательных будем называть ребрами перехода, а дуги — огибающими ребрами.

Ребра перехода можно разделить на внешние и внутренние. Построение внутренних ребер перехода между двумя кругами приводит к математической «задаче ремня», которая требует определения длины скрещенного ремня, соединяющего два круговых шкива разного радиуса.

Для кругов с центрами в точках A и B с радиусами R_a и R_b соответственно, расположенных на расстоянии d, как показано на

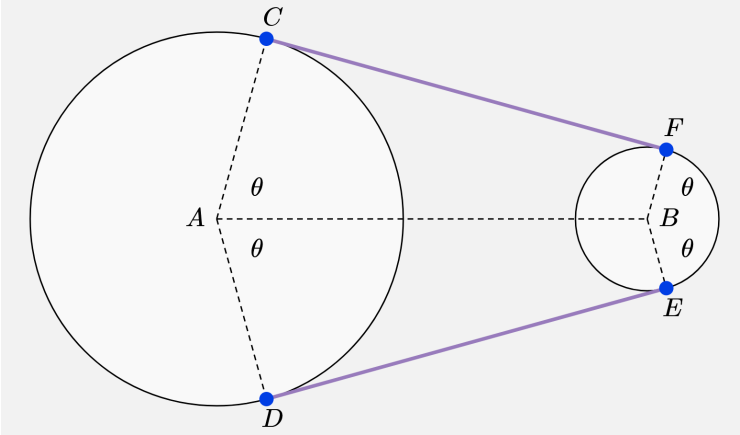

Рис. 2. График к «проблеме со шкивом».

рис. 1, угол θ может быть найден по следующей формуле:

$$\theta = \arccos \frac{R_a + R_b}{d}.$$

Теперь, зная угол θ, можно легко найти координаты точек C, D, E и F и построить отрезки внутренних касательных между двумя циркулярными препятствиями.

Построение внешних ребер перехода между двумя циркулярными препятствиями приводит к «проблеме со шкивом». Проблема шкива аналогична проблеме ремня, однако ремень не перекрещивается.

Как показано на рис. 2, для кругов с центрами A и B с радиусами R_a и R_b соответственно, расположенных на расстоянии d, угол θ находится по следующей формуле:

$$\theta = \arccos \frac{R_a - R_b}{d}.$$

Чтобы получить концы отрезков внешних касательных C, D, E и F, угол θ следует отложить на противоположных сторонах кругов. Некоторые из полученных ребер могут оказаться заблокированными другими препятствиями и их следует отбросить. Для того чтобы

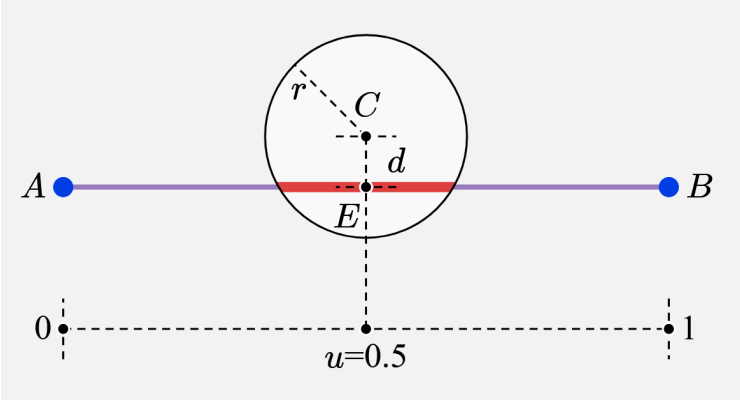

Рис. 3. Иллюстрация к методу определения пересечений препятствия и ребра.

определить, заблокировано ли ребро конкретным препятствием, нужно найти расстояние до его центра и сравнить с радиусом.

На рис. 3 представлена геометрическая интерпретация задачи. Во-первых, найдем параметр u, отражающий позицию точки E — проекции центра препятствия C на прямую, содержащую отрезок:

$$u = \frac{(C - A)(B - A)}{(B - A)(B - A)}.$$

Координаты точки E можно найти по формуле:

$$E = A + clamp(u, 0, 1) \cdot (B - A).$$

Далее найдем расстояние от нее до точки C и сравним с радиусом препятствия r. Если расстояние больше или равно радиусу, то данное препятствие не блокирует ребро. Проверку следует произвести для всех препятствий, если ни одно их них не блокирует ребро перехода, то его следует добавить в граф.

Когда все ребра перехода сгенерированы, можно создать огибающие ребра, соединив друг с другом попарно дугами все вершины, лежащие на одном циркулярном препятствии.

Начальная и конечная точки маршрута рассматриваются как препятствия с радиусом ноль и имеют только два ребра перехода к остальным, так как внешние и внутренние ребра перехода совпадут.

Так по карте циркулярных препятствий построен неориентированный граф.

2. Обработка пересечений препятствий

Для корректной обработки пересечений препятствий необходимо добавить этап предварительной обработки карты до генерации графа. Пусть имеется два круга с центрами в точках A и B с радиусами R_a и R_b соответственно, а расстояние между их центрами равно d. Тогда на предварительном этапе возможны следующие варианты их взаимодействия.

1) Если $d > R_a + R_b$, то препятствия не пересекаются. В этом случае, они не влияют на огибающие ребра друг друга и все касательные существуют.
2) Если $d < |R_a - R_b|$, тогда одно из препятствий полностью содержит другое. Следовательно меньшее оказывается поглощено препятствием с большим радиусом и не добавит в граф ни одной вершины или ребра. Такие препятствия можно сразу исключить из рассмотрения.
3) Если $d \leqslant R_a + R_b$, то два циркулярных препятствия пересекаются, то есть имеют две точки пересечения. В частном случае, когда они лишь касаются друг друга, эти точки равны.

При пересечении препятствий внутренние касательные между ними не будет, а огибающие ребра обоих кругов могут оказаться заблокированы. Чтобы найти точки касания между двумя циркулярными препятствиями, рассмотрим геометрическую задачу изображенную на рис. 4.

Пусть имеются два круга с центрами в точках A и B с радиусами R_a и R_b соответственно, d — расстояние между их центрами. Рассмотрим прямую, на которой лежит отрезок ED, соединяющий точки пересечения кругов. Она перпендикулярна прямой AC, так как AC является высотой равнобедренного треугольника AED, и

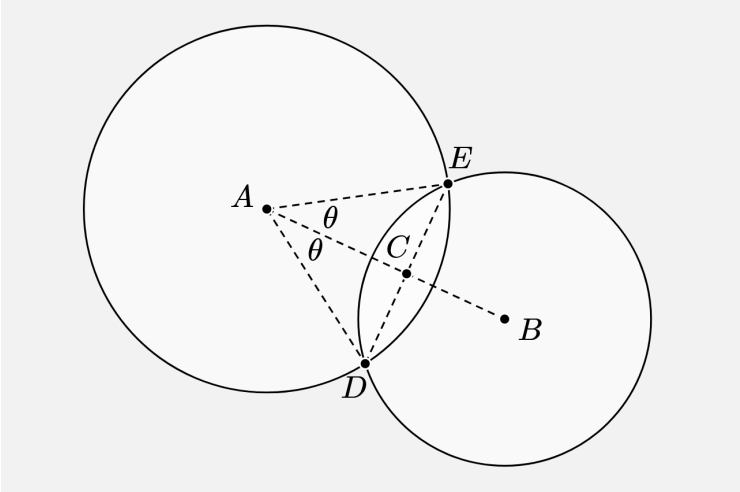

Рис. 4. График к задаче поиска пересечений между препятствиями.

пересекает ее в точке C. Тогда длину a отрезка AC найдем по следующей формуле:

$$a = \frac{R_a^2 - R_b^2 + d^2}{2d}.$$

Теперь найдем угол θ:

$$\theta = \arccos \frac{a}{R_a}.$$

Если угол θ равен 0, то препятствия касаются в точке C. Иначе, существует две точки касания E и D, координаты которых легко вычислить.

Чтобы проверить, что ребро не блокируется, достаточно убедиться, что на дуге не лежит ни одна из точек пересечения с другими препятствиями.

3. Поиск кратчайшего пути

Поиск кратчайшего пути из начальной точки в конечную в полученном по карте графе будем производить с помощью алгоритма A*.

Так как заведомо нет никакой априорной информации о расположении препятствий на карте, требуется универсальный алгоритм и A* подходит для решения данной задачи как нельзя лучше. Алгоритм A* является полным, как и поиск в ширину, а значит он всегда находит решение, если такое существует. Алгоритм будет являться оптимальным, если эвристическая функция $h(x)$, оценивающая расстояние до конечной точки, допустима, то есть не переоценивает минимальную стоимость достижения цели. В качестве функции $h(x)$ используем евклидово расстояние.

4. Переменный радиус исполнителя

Пусть исполнитель имеет радиус R, тогда его движение в среде с циркулярными препятствиями эквивалентно движению точки с единственным изменением: радиус всех препятствий увеличивается на R. Данное свойство следует из того, что плоскость является линейным пространством,а радиусы препятствий и исполнителя будут простым случаем применения суммы Минковского.

Суммой Минковского двух подмножеств A и B линейного пространства V (или произвольной группы) называется множество C, состоящее из сумм всевозможных векторов из A и B. На рис. 5 представлена графическая интерпретация суммы Минковского:

$$C = \{c \mid c = a + b, a \in A, b \in B\}.$$

5. Улучшения и оптимизации

Алгоритм A* в ходе работы может оставить некоторую часть графа вовсе нерассмотренной. Например, это может произойти, если часть графа оказывается не достижима из начальной вершины. Можно отказаться от полной генерации графа и динамически создавать его по мере необходимости.

Для начальной и конечной точек сразу генерируем все ребра перехода к остальным препятствиям и применяем алгоритм A*. Получив очередную вершину, проверяем, принадлежит ли она препятствию, для которого еще не генерировались ребра перехода. Если это так,

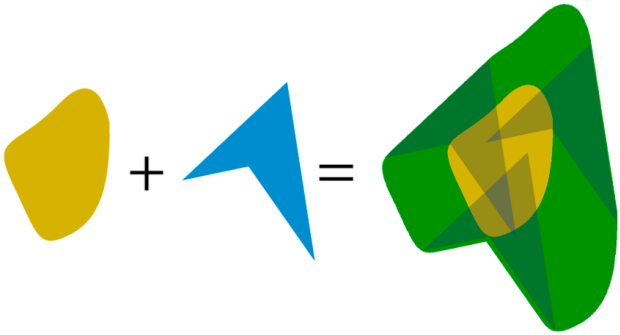

Рис. 5. Графическая интерпретация суммы Минковского.

то нужно их сгенерировать и добавить, а также создать огибающие ребра.

Такой подход позволяет значительно ускорить работу алгоритма и оптимизировать потребление памяти. Сравнение скорости работы полной и динамической генерации, а так же количества хранимых вершин и ребер для карт различного размера приведены на рис. 6, 7 и 8.

Еще одно улучшение — это отбрасывание заостренных огибающих ребер. Идея данного метода заключается в отбрасывании заведомо не оптимальных огибающих ребер, которые не могут быть частью оптимального пути.

Заметим, что алгоритм A*, рассматривая очередную вершину, обрабатывает каждое неориентированное ребро, как два ориентированных. Тогда сразу будем строить ориентированный граф по следующим правилам.

1) Каждая вершина P разбивается на две с различными ориентациями по часовой и против часовой стрелки.
2) Неориентированные ребра перехода и огибающие становятся ориентированными, при этом:
    а) огибающие ребра и внешние касательные сохраняют на-

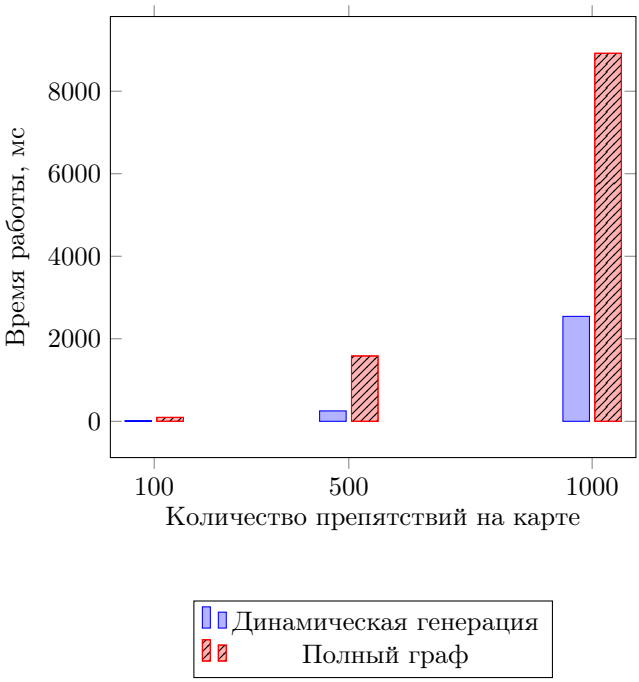

Рис. 6. Сравнение времени работы алгоритма при динамической и полной генерации графа.

правление обхода вершин, то есть соединяют только вершины с одинаковым направлением;

б) внутренние касательные меняют направление обхода на противоположное.

3) Начальная и конечная вершины остаются единственными в своем роде и не имеют направления. Направления обхода появляется у непосредственных потомков начальной вершины, исходя из того, по какому ребру они были достигнуты, в соответствии с пунктом 2.

На рис. 9, 10 и 11 приведены данные тестов скорости работы

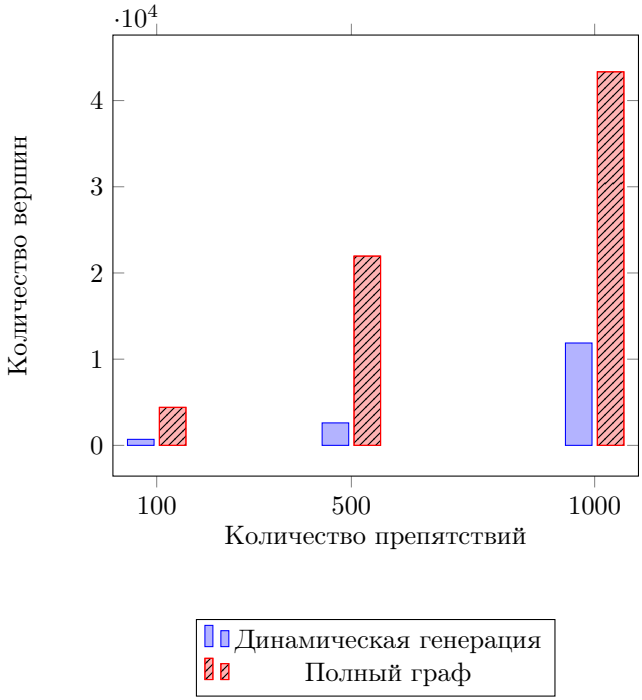

Рис. 7. Сравнение количества вершин при динамической и полной генерации графа.

и число сгенерированных вершин и ребер без применения данного улучшения и с ним.

Рассмотренное выше преобразование графа замедляет работу алгоритма и увеличивает объем памяти необходимый для его хранения.

Заключение

В процессе работы был разработан алгоритм генерации графа по карте циркулярных препятствий. Изучены методы поиска кратчайшего пути в графах с применением различных алгоритмов. Изучен

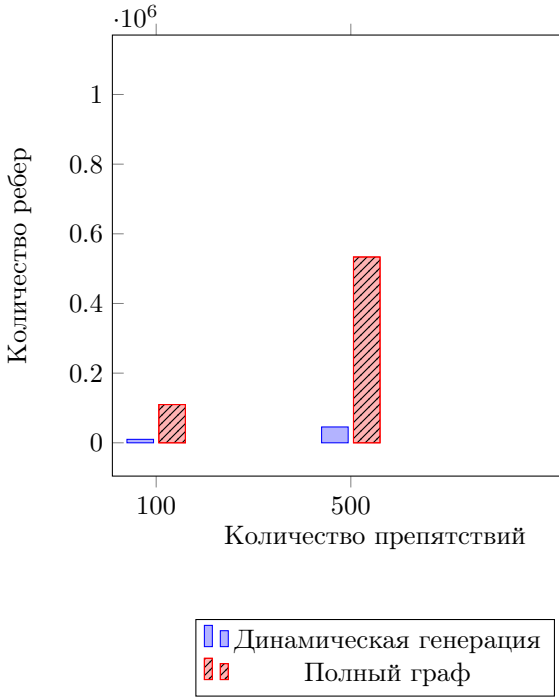

Рис. 8. Сравнение количества ребер при динамической и полной генерации графа.

алгоритм A* и применен для поиска кратчайшего пути. Исследованы возможности улучшения и оптимизации полученного алгоритма, а так же учтена возможность пересечения циркулярных препятствий и переменным радиус исполнителя.

Алгоритм реализован на языке программирования C# в виде отдельного программного модуля, который может быть повторно использован в других программных продуктах.

Разработан редактор циркулярных препятствий позволяющий создавать, сохранять в формате xml и редактировать карты. Данный редактор реализует описанный в статье алгоритм поиска кратчайшего пути и его эффективные улучшения.

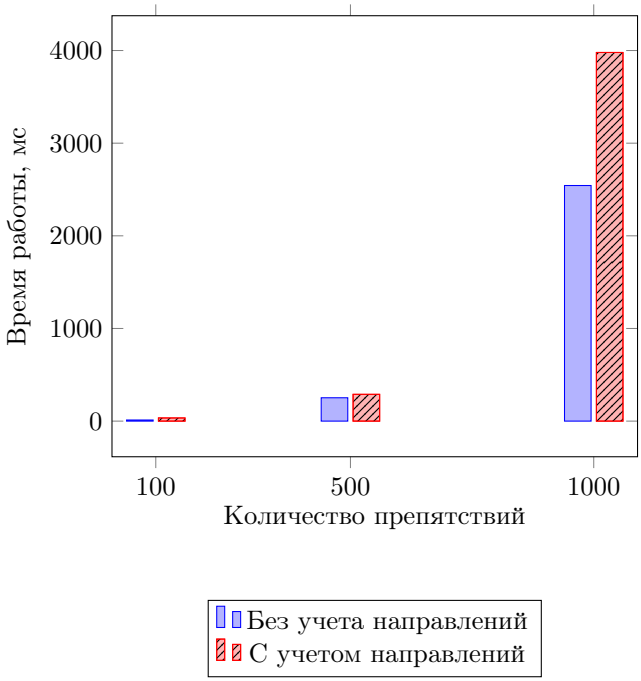

Рис. 9. Сравнение времени работы алгоритма без учета направления и с учетом.

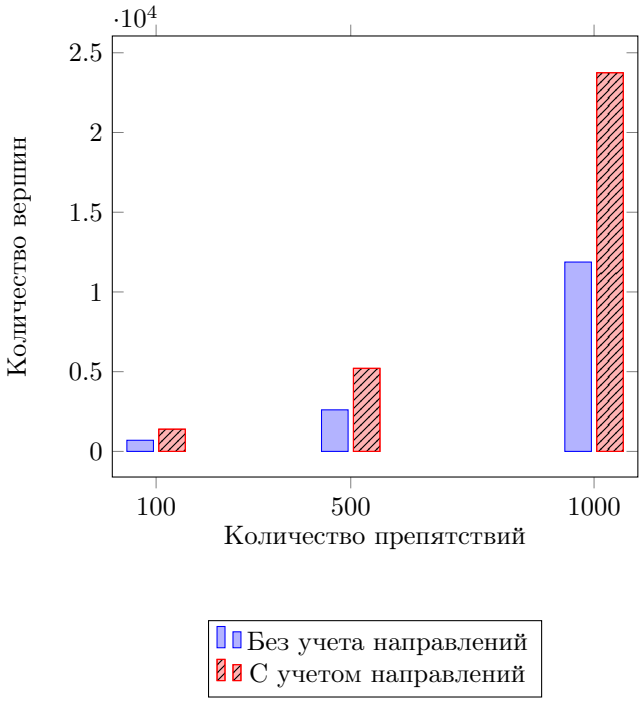

Рис. 10. Сравнение количества вершин без учета направления и с учетом.

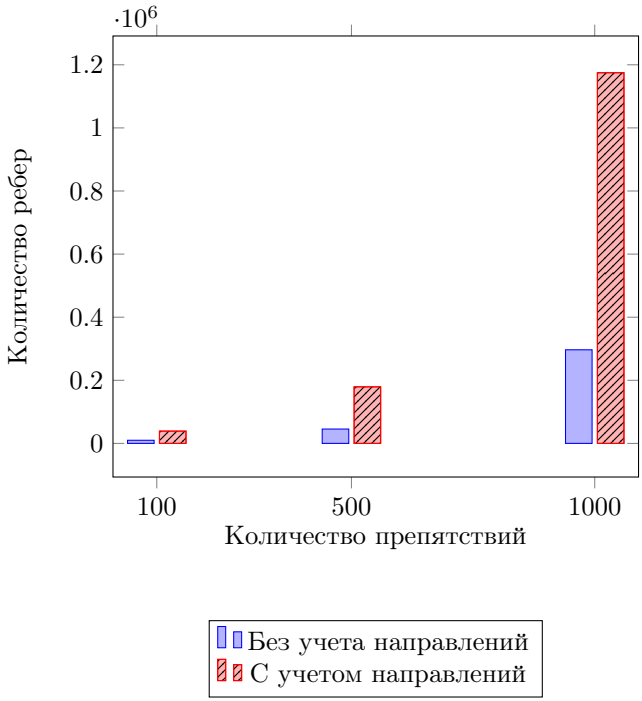

Рис. 11. Сравнение количества ребер без учета направления и с учетом.

Список литературы

[1] Circular Obstacle Pathfinding : [сайт]. — 2017. — URL: https://redblobgames.github.io/circular-obstacle-pathfinding/ (дата обращения: 01.05.2023).

[2] DeLoura, M. A. Game Programming Gems 2. — Needham Heights, MA : Charles River Media, 2001. — 575 p.

Библиографическая ссылка

Титов, А. Д. Маршрутизация в среде с циркулярными препятствиями // Студенческая конференция факультета ПМиК. Сборник трудов. — Тверь : ТвГУ, 2023. — С. 94–108.

Сведения об авторах

Титов Артём Дмитриевич
Студент магистратуры
Направление «Фундаментальная информатика и информационные технологии»

УДК 004.85

Повышение разрешения изображения с помощью сверточных нейронных сетей

Чумаков М. С.

Тверской государственный университет

Аннотация. В данной статье представлен обзор сверточной нейронной сети для повышения разрешения изображений, известной как Fast Super-Resolution Convolutional Neural Network (FSRCNN). Предлагаемая нейронная сеть обладает уникальной архитектурой, которая позволяет увеличить разрешение изображений с высокой скоростью и с отличным качеством результата. Для оценки производительности FSRCNN производится сравнение с классическим методом билинейной интерполяции. Результаты экспериментов показывают, что FSRCNN достигает более высокого качества восстановленных изображений по сравнению с билинейной интерполяцией. Особенно заметно улучшение в деталях изображений и сохранение текстур.

Введение

Обработка изображений несколько последних десятилетий является активно развивающейся научной областью. Одной из активно решаемых задач в данной области является повышение разрешения изображения. Это объясняется тем, что изображения с высоким разрешением имеют лучшие характеристики для восприятия человеком и позволяют проводить более качественный последующий анализ. Изначально считалось, что главным способом получения изображений с высоким разрешением является использование технически совершенного оборудования существующего на данный момент времени. Однако при таком подходе возникают сложности, связанные со стоимостью оборудования и его техническими ограничениями. Для их преодоления был предложен подход, который состоит в получении изображения высокого разрешения из одного или нескольких изображений с низким разрешением. Благодаря расширению сфер применения в последние несколько лет систем компьютерного зрения

задача повышения разрешения изображений на основе вычислений все более актуальна. Для решения данной задачи перспективным является использование методов, основанных на глубоком машинном обучении, которые в последние годы доказали практическую значимость при решении различных научных задач.

1. Сверточные нейронные сети для увеличения разрешения изображения

Сверточные нейронные сети (CNN) существуют уже несколько десятилетий, а глубокие CNN продемонстрировали взрывную популярность, частично благодаря своему успеху в классификации изображений. Они также были успешно применены в других областях компьютерного зрения, таких как обнаружение объектов, распознавание лиц и обнаружение пешеходов. Несколько факторов имеют центральное значение в этом прогрессе: эффективная реализация обучения на современных мощных графических процессорах, предложение выпрямленного линейного блока (ReLU), который значительно ускоряет конвергенцию, сохраняя при этом хорошее качество, и легкий доступ к большому количеству данных (например, ImageNet) для обучения больших моделей [2].

В задаче повышения разрешения одиночного изображения сеть получает на вход изображение с низким разрешением и выдает на выходе версию этого изображения с более высоким разрешением. В целом у этой задачи нет единственно правильного решения — существует много возможных изображений с высоким разрешением, которые могут соответствовать одному и тому же изображению с низким разрешением. Однако, обучаясь на наборах данных соответствующих изображений с высоким и низким разрешением, нейронная сеть может получить априорные данные о визуальном мире, которые позволяют ей предсказать правдоподобный вывод с высоким разрешением для любого ввода с низким разрешением.

Подобно семантической сегментации, мы можем использовать полностью сверточную сеть для построения модели сверхвысокого разрешения для одного изображения. Входными данными для модели является изображение с низким разрешением формы $3 \times H \times W$, изображение проходит через серию слоев свертки и слоев повышения дискретизации и в конечном итоге выводит выходные данные с

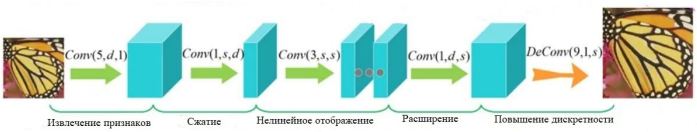

Рис. 1. Архитектура FSRCNN.

высоким разрешением формы $3 \times fH \times fW$, где f — коэффициент повышения разрешения. Во время обучения выходные данные сети сравниваются с реальными выходными данными с использованием L2, L1 или какой-либо другой функции потерь.

Одна из подходящих моделей для решения задачи повышения разрешения изображения с помощью сверточных нейронных сетей — это FSRCNN.

2. FSRCNN

Fast Super-Resolution Convolutional Neural Networks (FSRCNN) — это улучшение модели SRCNN. FSRCNN имеет гораздо более высокое качество изображения и более короткое время выполнения [1].

FSRCNN можно разложить на пять частей — извлечение признаков, сжатие, отображение, расширение и деконволюция. А также между частями используется функция активации PReLU. На рис. 1 представлена архитектура модели [1].

3. Реализация и оценка модели FSRCNN

При настройке искусственных нейронных сетей важно подобрать качественный материал для обучения и тестирования. Для того, чтобы алгоритм хорошо произвел обработку, изображения должны быть одного формата, качества и размера.

Одним из простых способов получения изображения с низким разрешением является ухудшение изображения с высоким разрешением. Это часто делается с помощью размытия или добавления шума.

Существуют свободные ресурсы с базами изображений, например, ImageNet, 91-image, General100, Set5 и Set14.

Затем необходимо определить архитектуру FSRCNN, включая количество сверточных слоев, размеры фильтров, число фильтров в каждом слое и другие параметры. Архитектура FSRCNN может быть настроена на основе предварительных исследований и экспериментов для достижения лучших результатов.

Таким образом была выбрана следующая архитектура FSRCNN.

- Этап извлечения признаков(feature extraction) состоит из одного сверточного слоя с следующими параметрами: один входной канала, пятьдесят шесть выходных каналов, размер ядра равен пяти, паддинг равен двум.

- Этап сжатия состоит из одного сверточного слоя с следующими параметрами: пятьдесят шесть входных каналов, двенадцать выходных каналов, размер ядра равен единице.

- Этап отображения состоит из четырех слоев свертки, каждый из которых с следующими параметрами: двенадцать входных каналов, двенадцать выходных каналов, размер ядра равен трем, паддинг равен единице.

- Этап расширения состоит из одного сверточного слоя с следующими параметрами: двенадцать входных каналов, пятьдесят шесть выходных каналов, размер ядра равен единице.

- Этап деконволюции состоит из одного слоя деконволюции (можно рассматривать как обратную операцию свертки) с следующими параметрами: пятьдесят шесть входных каналов, один выходной канал, размер ядра равен девяти, паддинг равен четырем, шаг равен коэффициенту повышения разрешения изображения, выходной паддинг равен коэффициенту повышения разрешения изображения минус единица.

Входное изображение имеет цветовое пространство YCbCr вместо RGB, а нейронная сеть использует оптимизатор Adam(Adaptive Moment Estimation) и функцию потерь MSE (Mean Squared Error).

Оптимизатор Adam (Adaptive Moment Estimation) является одним из наиболее популярных и эффективных методов оптимизации

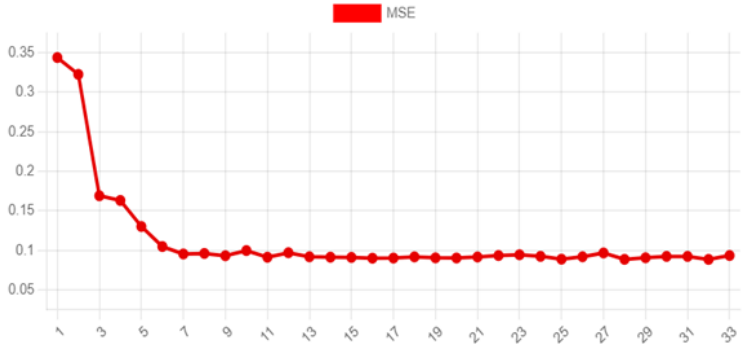

Рис. 2. Потери при обучении FSRCNN.

в области глубокого обучения. Он комбинирует преимущества алгоритмов градиентного спуска и стохастического градиентного спуска, обеспечивая быструю и стабильную сходимость при обучении нейронных сетей.

MSE (Mean Squared Error) — это одна из наиболее распространенных функций потерь (loss function), используемых в задачах регрессии для измерения разницы между предсказанными и фактическими значениями целевой переменной.

MSE рассчитывается как среднее арифметическое квадратов отклонений предсказанных значений от фактических значений, где отклонение — это разница между предсказанным значением и соответствующим фактическим значением:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2, \tag{1}$$

где y — исходное изображение с высоким разрешением, а $\hat{y}$ — полученное в результате работы изображение с высоким разрешением. На рис.2 представлен график потерь при обучении FSRCNN по эпохам.

Для оценки результатов используется два критерия: PSNR и SSIM.

Peak Signal-to-Noise Ratio (PSNR) — одна из метрик, широко используемых для оценки качества восстановленных изображений в

задачах обработки изображений с использованием нейронных сетей. PSNR измеряет отношение между максимально возможной интенсивностью сигнала и среднеквадратичной ошибкой между исходным и восстановленным изображениями.

В контексте нейронных сетей, PSNR может быть использован для сравнения восстановленного изображения с оригинальным высококачественным изображением. Чем выше значение PSNR, тем меньше различие между восстановленным и исходным изображением, и, следовательно, считается, что качество восстановления выше.

PSNR вычисляется по формуле

$$PSNR = 10 \log_{10} \frac{MAX_I^2}{MSE}, \tag{2}$$

где MAX_I — это максимальное значение, принимаемое пикселем изображения, в случае цветового пространства YCbCr равен единице.

Structural Similarity Index (SSIM) является метрикой, широко используемой в задачах обработки изображений с использованием нейронных сетей для оценки сходства между восстановленным и исходным изображениями. Она измеряет структурное подобие между двумя изображениями, учитывая их яркость, контрастность и структуру.

В контексте нейронных сетей, SSIM часто используется для сравнения качества восстановленного изображения с исходным изображением. Высокое значение SSIM указывает на более высокое структурное сходство и, следовательно, на более высокое визуальное качество восстановления.

SSIM вычисляется по формуле

$$SSIM = l(x, y) \cdot c(x, y) \cdot s(x, y), \tag{3}$$

где

- x и y — два изображения, для которых вычисляется SSIM;
- $l(x, y)$ — яркостное сходство (luminance similarity);
- $c(x, y)$ — контрастное сходство (contrast similarity);
- $s(x, y)$ — структурное сходство (structure similarity).

Набор данных	Коэффициент повышения разрешения	Бикубическая интерполяция	FSRCNN
Set5	2	33,66	37,8364
Set14	2	30,23	33,9429
Set5	3	30,39	33,9512
Set14	3	27,54	30,7836

Таблица 1. Оценка метода бикубической интерполяции и FSRCNN по критерию PSNR.

Набор данных	Коэффициент повышения разрешения	Бикубическая интерполяция	SSIM
Set5	2	0,9299	0,9672
Set14	2	0,8687	0,9134
Set5	3	0,9199	0,9237
Set14	3	0,7736	0,8481

Таблица 2. Оценка метода бикубической интерполяции и FSRCNN по критерию SSIM.

Формулы для вычисления каждой компоненты выглядят следующим образом:

$$l(x,y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}, \tag{4}$$

где μ_x и μ_y — средние значения пикселей в изображениях x и y соответственно, C_1 — небольшая константа для стабилизации деления.

$$c(x,y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}, \tag{5}$$

где σ_x и σ_y — дисперсии пикселей в изображениях x и y соответственно, C_2 — небольшая константа для стабилизации деления.

$$s(x,y) = \frac{\sigma_{xy} + C_3}{\sigma_x + \sigma_y + C_3}, \tag{6}$$

Рис. 3. Оригинальное изображение с высоким разрешением.

Рис. 4. Полученное изображение с высоким разрешением.

где σ_{xy} — ковариация пикселей в изображениях x и y, C_3 — небольшая константа для стабилизации деления.

В табл. 1 приведены оценки метода бикубической интерполяции и FSRCNN по критерию PSNR на разных наборах данных и с разными коэффициентами повышения разрешения.

В табл. 2 приведены оценки метода бикубической интерполяции и FSRCNN по критерию SSIM на разных наборах данных и с разными коэффициентами повышения разрешения.

Из таблиц можно видеть, что бикубическая интерполяция проигрывает FSRCNN по всем критериям. Несколько примеров работы модели при коэффициенте повышения разрешения равным трем. Примеры показаны ни рис. 3–6.

Рис. 5. Оригинальное изображение с высоким разрешением.

Рис. 6. Полученное изображение с высоким разрешением.

Заключение

Проведенные исследования показывают, что сверточные нейронные сети, особенно модель FSRCNN, значительно превосходят метод бикубической интерполяции в качестве и точности восстановления изображений.

Модель FSRCNN позволяет получить более четкие и детализированные изображения, благодаря своей способности изучать более сложные иерархические структуры изображений и использовать информацию о контексте для эффективного повышения разрешения. Сравнительные эксперименты показали, что FSRCNN превосходит метод бикубической интерполяции как в визуальном качестве, так и в показателях качества изображения, таких как пиковое отношение сигнал/шум (PSNR) и структурное сходство (SSIM).

Эти результаты подтверждают значительный потенциал сверточных нейронных сетей в области повышения разрешения изображений. Однако следует отметить, что использование модели FSRCNN требует вычислительных ресурсов и времени для обучения и применения. Поэтому при выборе метода повышения разрешения необходимо учитывать и конкретные требования к скорости обработки изображений.

В целом, разработка и исследование сверточных нейронных сетей для повышения разрешения изображений представляет собой важную область в компьютерном зрении и обработке изображений. Модель FSRCNN показала отличные результаты и может быть применена в различных задачах, таких как восстановление изображений низкого разрешения, увеличение детализации и улучшение качества визуализации.

Список литературы

[1] Dong, C. Accelerating the Super-Resolution Convolutional Neural Network / C. Dong, C. C. Loy, X. Tang // Computer Vision — ECCV 2016. Lecture Notes in Computer Science, vol 9906. — Cham : Springer, 2016. — P. 391–407.

[2] Image Super-Resolution Using Deep Convolutional Networks / C. Dong, C. C. Loy, K. He, X. Tang. — 2015. — URL: https://arxiv.org/abs/1501.00092.

Библиографическая ссылка

Чумаков, М. С. Повышение разрешения изображения с помощью сверточных нейронных сетей // Студенческая конференция факультета ПМиК. Сборник трудов. — Тверь : ТвГУ, 2023. — С. 109–119.

Сведения об авторах

Чумаков Максим Сергеевич
Студент магистратуры
Направление «Фундаментальная информатика и информационные технологии»

Научное издание

Студенческая конференция факультета ПМиК. Сборник трудов

Тверь
17–28 апреля 2023 г.

Под редакцией Б. Н. Карлова

Подписано в печать 26.05.2023. Формат 60x84 1/16*.
Усл. печ. л. 6,98. Тираж 100. Заказ № 142.
Издательство Тверского государственного университета.
Адрес: 170100, г. Тверь, Студенческий пер. 12, корпус Б.
Тел. (4822) 35-60-63.