ISSN 1561-2430 (Print)
ISSN 2524-2415 (Online)

ИНФОРМАТИКА

INFORMATICS

УДК 004.89
https://doi.org/10.29235/1561-2430-2025-61-2-159-174

Поступила в редакцию 27.01.2025
Received 27.01.2025

В. И. Бегунков, М. Я. Ковалев

Объединенный институт проблем информатики Национальной академии наук Беларуси, Минск, Республика Беларусь

КЛАССИФИКАЦИЯ ЗАЙМА С ИСПОЛЬЗОВАНИЕМ ГЛУБОКОЙ НЕЙРОННОЙ СЕТИ ПРЯМОГО РАСПРОСТРАНЕНИЯ

Аннотация. Разработана и проанализирована модель глубокой нейронной сети прямого распространения для решения задачи классификации финансового займа. С помощью этой модели на основе исторических данных по выданным ранее займам вычисляются значения следующих традиционных для машинного обучения метрик, которые определяют качество прогнозирования: стоимостная функция, истинность, точность, полнота и мера F_1. Для получения большей точности прогнозирования использованы оптимизационные методы мини-пакетного градиентного спуска, градиентного спуска с импульсом, адаптивной оценки момента, а также метод исключения на нулевом уровне. Определена улучшенная структура предложенной нейронной сети, проанализировано воздействие использования так называемой инициализации *He* на итоговый результат, а также целесообразность применения конкретных алгоритмов оптимизации. Исследование показало, что использование глубокой нейронной сети прямого распространения целесообразно при разработке классификаторов займов.

Ключевые слова: классификация займа, скоринг, глубокая нейронная сеть, машинное обучение

Для цитирования. Бегунков, В. И. Классификация займа с использованием глубокой нейронной сети прямого распространения / В. И. Бегунков, М. Я. Ковалев // Весці Нацыянальнай акадэміі навук Беларусі. Серыя фізіка-матэматычных навук. – 2025. – Т. 61, № 2. – С. 159–174. https://doi.org/10.29235/1561-2430-2025-61-2-159-174

Uladzimir I. Behunkou, Mikhail Y. Kovalyov

United Institute of Informatics Problems of the National Academy of Sciences of Belarus, Minsk, Republic of Belarus

LOAN CLASSIFICATION USING A DEEP FEED-FORWARD NEURAL NETWORK

Abstract. A deep feed-forward neural network model is developed and analyzed in this article to solve the financial loan classification problem. Using this model, based on historical data on previously issued loans, the values of the following traditional machine learning metrics that determine the quality of forecasting are calculated: cost function, truth, accuracy, completeness and F_1 measure. In order to obtain greater forecasting accuracy, optimization methods of mini-batch gradient descent, gradient descent with momentum, adaptive momentum estimation, and zero-level elimination method were used. An improved structure of the proposed neural network was determined, the impact of the so-called *He* initialization on the final result was analyzed, as well as the efficiency of using specific optimization algorithms. The study showed that the use of deep feed-forward neural network is reasonable in developing loan classifiers.

Keywords: loan classification, scoring, deep neural network, machine learning

For citation. Behunkou U. I., Kovalyov M. Y. Loan classification using a deep feed-forward neural network. *Vestsi Natsyyanal'nai akademii navuk Belarusi. Seryya fizika-matematychnykh navuk = Proceedings of the National Academy of Sciences of Belarus. Physics and Mathematics series*, 2025, vol. 61, no. 2, pp. 159–174 (in Russian). https://doi.org/10.29235/1561-2430-2025-61-2-159-174

Введение. В предыдущих работах авторов данной статьи [1, 2] отмечались важность и актуальность поиска решения стоящей перед финансовыми институтами задачи классификации займа, которая представляется как бинарная с делением заемщиков на хороших (без дефолта)

и плохих (дефолт). Частью решения данной задачи является определение подходящего классификатора займов. В [3] в список основных индивидуальных классификаторов оценки заемщиков, кроме рассмотренного прежде [1] классификатора на основе логистической регрессии (LR), также входит классификатор на основе искусственной нейронной сети – ANN. В свою очередь ANN может представлять собой как нейронную сеть прямого распространения, так и глубокую нейронную сеть прямого распространения. Учитывая, что классификатор на базе нейронной сети прямого распространения уже был ранее [2] исследован в рамках решения задачи классификации займа, то целесообразно провести анализ классификатора на основе глубокой нейронной сети прямого распространения

Несмотря на то что в литературе описывается использование методов искусственного интеллекта для решения задачи классификации займов, детальное изучение свойств этих методов представлено недостаточно.

Целью данной работы является исследование возможности эффективного использования глубокой нейронной сети прямого распространения для решения задачи классификации займа и сравнение результатов со значениями, полученными при использовании нейронной сети прямого распространения и логистической регрессии.

Описание данных. С целью решения задачи все используемые данные можно разделить на 3 группы: входные данные, настраиваемые параметры рассматриваемых методов и выходные данные.

Входные данные. Для определения значений параметров и проведения исследований с рассматриваемыми методами используются исторические данные по выданным на платформе для кредитования LendingClub займам[1], состоящие из 2 260 668 строк (кредиты, выданные за период с апреля 2016 г. по сентябрь 2018 г.). Набор входных показателей и принцип преобразования входных данных аналогичны тем, которые были описаны ранее при рассмотрении логистической регрессии [1] и нейронной сети прямого распространения [2]. В итоге окончательный набор входных данных состоит из $m = 1\,221\,731$ позиций и $n = 73$ входных показателей.

Считается, что значения этих показателей были известны до принятия решения о выдаче соответствующего займа. Обозначим значение показателя j в займе i из исходного набора данных через элементы $x_j^{(i)}$ матрицы X размером m на n, где $i = 1, \ldots, m$, $j = 1, \ldots, n$. Обозначим через x_j столбец матрицы X, а через $x^{(i)}$ – строку матрицы X, которая содержит значения независимых показателей в отдельной позиции (займе) i набора данных.

Еще в качестве исходных данных используются целевые значения $y^{(i)}$ (итоговый результат по займам, где $i = 1, \ldots, m$), которые определены в поле *loan_status* исходного набора данных и могут быть также представлены в виде вектора Y. Показатель $y^{(i)}$ принимает 2 значения.

1. Возвратный займ (со значением *Fully Paid*). Данные займы были погашены. Соответствует значению $y^{(i)} = 1$.

2. Невозвратный займ (*Charged Off* или *Default*). Кредиты, по которым был объявлен дефолт или погашение займа, просрочены более чем на 180 дней. Соответствует значению $y^{(i)} = 0$.

Займы со значениями *Current*, *In Grace period*, *Late* (16–30 *days*) и *Late* (31–120 *days*) исключаются из анализа, так как однозначно нельзя понять, были такие кредиты возвратными или невозвратными.

Параметры используемого алгоритма. В исследуемом алгоритме используется набор настраиваемых параметров:

1) $w^{(l)}$, $l = 1, \ldots, L$, – матрица весов нейронной сети, где вектор-строка $w_k^{(l)}$, $k = 1, \ldots, K^{(l)}$, в свою очередь содержит коэффициенты (числа) $w_{kh}^{(l)}$, $h = 1, \ldots, H^{(l-1)}$, L – количество уровней сети, $K^{(l)}$ – количество нейронов в уровне l, а $H^{(l-1)}$ – количество нейронов в уровне l–1, при этом $H^{(0)} = n$;

2) $b^{(l)}$ – вектор-столбцы, состоящие из значений $b_k^{(l)}$ коэффициентов взвешенного набора сигналов нейрона k, $k = 1, \ldots, K^{(l)}$;

3) функции активации $a_k^{(l)}(x^{(i)})$ нейронов, $k = 1, \ldots, K^{(l)}$.

[1] Loan data // Lending Club. URL: https://www.kaggle.com/datasets/wordsforthewise/lending-club.

Выходные данные. Выходными данными изучаемой бинарной задачи классификации (т. е. определения кредита как возвратного или потенциально невозвратного) являются величины $\hat{y}^{(i)} \in \{0,1\}$, где 1 соответствует возвратному, а 0 – потенциально невозвратному займу i, $i \in \in \{1, \ldots, m\}$.

Постановка задачи при использовании глубокой нейронной сети прямого распространения. Глубокая нейронная сеть прямого распространения (рисунок) состоит из множества уровней: входного, нескольких скрытных и выходного уровня. Необходимо отметить, что в соответствии с существующим подходом об обозначениях, набор входных показателей $x_1,..,x_n$ считается нулевым уровнем нейронной сети и не учитывается при подсчете количества уровней. Матрицы $w^{(l)}$ и вектора $b^{(l)}$ контролируют функциональное преобразование из уровня $l-1$ в уровень l, где $l = 1, \ldots, L$. Так как решаемая задача относится к бинарной классификации, то выходной уровень состоит из одного нейрона, который рассчитывает значение функции активации $a_k^{(L)}(x^{(i)})$ на последнем уровне. В свою очередь скрытые уровни состоят из множества нейронов. Описание функционирования нейронной сети представлено в [4, p. 269–270].

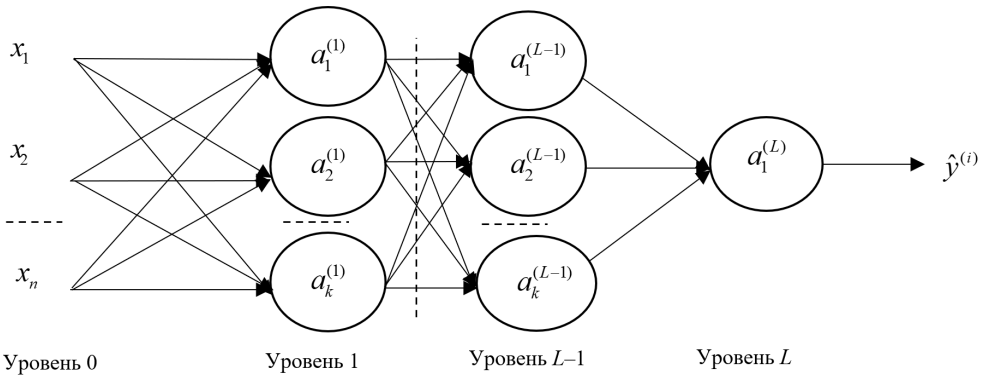
Как отмечено в том же исследовании, для нейронов скрытого уровня целесообразно в качестве функции активации применять функцию ReLu, $f(z)$. При этом в нейроне выходного уровня функция активации определяется на основе сигмоидной функции σ, которая используется при применении логистической регрессии в связи с тем, что диапазон определяемых значений 0 или 1 является целесообразным при решении задачи бинарной классификации [4, p. 12–127].

Обучение глубокой нейронной сети схоже с подходом, описанным ранее для нейронной сети прямого распространения, с отличием в том, что глубокая нейронная сеть содержит множество скрытых уровней. Таким образом, обучение глубокой нейронной сети можно представить в следующей последовательности.

1. Задание начальных значений весов $w_{kh}^{(l)}$ нейронов и величин $b_k^{(l)}$. В отличие от $b_k^{(l)}$, которые можно принять равными 0, веса $w_{kh}^{(l)}$ не могут быть изначально равны 0, так как в этом случае значения функций активации нейронов на одном уровне равны для любого i. Таким образом, нейронная сеть становится неэффективной, так как скрытый уровень осуществляет расчет одной и той же функции активации вне зависимости от их количества в скрытом уровне. Во избежание такой ситуации значения весов $w_{kh}^{(l)}$ задаются как малые величины произвольным образом [5] на основе стандартного нормального распределения со средним значением, равным 0, и со стандартным отклонением, равным 1, умноженных на 10^{-2}.

2. Реализация прямого распространения данных в нейронной сети для расчета $a_1^{(L)}(x^{(i)})$. По аналогии с расчетами для отдельного нейрона расчеты, выполняемые на первом уровне нейронной сети (рисунок), можно представить в векторном виде, где T означает транспонирование:

$$z_k^{(1)}(x^{(i)}) = w_k^{(1)} x^{(i)T} + b_k^{(1)}; \quad a_k^{(1)}(x^{(i)}) = f\left(z_k^{(1)}(x^{(i)})\right). \tag{1}$$



Глубокая нейронная сеть прямого распространения

Deep feed-forward neural network

Данные формулы можно представить и в матричном виде. Если использовать матрицу $w^{(1)}$ и вектор-столбец $b^{(1)}$, то для первого уровня расчет векторов $z^{(1)}(x^{(i)})$ и $a^{(1)}(x^{(i)})$ выглядит следующим образом:

$$z^{(1)}(x^{(i)}) = w^{(1)}x^{(i)T} + b^{(1)}; \quad a^{(1)}(x^{(i)}) = f\left(z^{(1)}(x^{(i)})\right). \tag{2}$$

На уровне l определение векторов $z^{(l)}(x^{(i)})$ и $a^{(l)}(x^{(i)})$ осуществляется на основе формул

$$z^{(l)}(x^{(i)}) = w^{(l)}a^{(l-1)}(x^{(i)}) + b^{(l)}; \quad a^{(l)}(x^{(i)}) = f\left(z^{(l)}(x^{(i)})\right). \tag{3}$$

Аналогичным образом на выходном уровне, содержащем один нейрон, двухшаговый расчет выполняется с помощью формул

$$z_1^{(L)}(x^{(i)}) = w_1^{(L)}a^{(L-1)}(x^{(i)}) + b_1^{(L)}; \quad a_1^{(L)}(x^{(i)}) = \sigma\left(z_1^{(L)}(x^{(i)})\right). \tag{4}$$

В связи с тем, что выходной уровень содержит один нейрон, представление в матричном виде данных формул нецелесообразно. Таким образом, определена функция активации нейронной сети для одного займа. Аналогично выполняется расчет функций активации $a_1^{(L)}(x^{(i)})$ для всех i элементов набора для обучения, где $i \in \{1, \ldots, m\}$.

3. Расчет функции потерь, подлежащей минимизации, от несоответствия значений, рассчитанных функцией активации $a_1^{(L)}(x^{(i)})$, значениям $y^{(i)}$. Данный расчет осуществляется с использованием стоимостной функции нейронной сети по следующей формуле [6, p. 205–207, 370–371]:

$$J(w,b) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\ln\left(a_1^{(L)}(x^{(i)})\right) + (1-y^{(i)})\ln\left(1-a_1^{(L)}(x^{(i)})\right)\right]. \tag{5}$$

4. Реализация алгоритма обратного распространения ошибки для вычисления градиента с целью минимизации стоимостной функции $J(w,b)$. Суть алгоритма состоит в расчете дельты (ошибки) при активации каждого нейрона в каждом уровне сети. Для минимизации ошибки используется производная от функции $J(w,b)$, так как она определяет, каким образом изменить входные параметры для требуемого изменения соответствующей функции [7, p. 82–86]. В данном случае рассчитывается степень изменения параметров $w_{kh}^{(l)}$ и $b_k^{(l)}$ для получения минимального значения стоимостной функции $J(w,b)$. Так как в общем виде $J(w,b)$ является функцией переменных двух типов, то фактически необходимо определить частные производные двух типов $\frac{\partial J}{\partial w}$ и $\frac{\partial J}{\partial b}$. Стоит отметить, что функция $J(w,b)$ является сложной, поскольку напрямую зависит от функции активации α, в свою очередь зависящей от линейной функции z, которая, согласно формуле (1), уже напрямую зависит от w и b. Поэтому обозначенные выше частные производные могут быть найдены с помощью цепного правила [7, p. 205]:

$$\begin{aligned}
\frac{\partial J}{\partial z} &= \frac{\partial J}{\partial a}\frac{\partial a}{\partial z};\\
\frac{\partial J}{\partial w} &= \frac{\partial J}{\partial z}\frac{\partial z}{\partial w};\\
\frac{\partial J}{\partial b} &= \frac{\partial J}{\partial z}\frac{\partial z}{\partial b}.
\end{aligned} \tag{6}$$

Таким образом, в обратном порядке, начиная с последнего до первого уровня, рассчитываются частные производные функции $J(w,b)$ по a, z, w и b для каждого нейрона в каждом уровне. Так как выходной уровень нейронной сети прямого распространения аналогичен выходному уровню глубокой нейронной сети, то на основе определенных в [2] формул $\frac{\partial J}{\partial w}$ и $\frac{\partial J}{\partial b}$ для выходного уровня L исследуемой глубокой сети определяются следующим образом:

$$\frac{\partial J}{\partial w_1^{(L)(i)}} = \left(a_1^{(L)}(x^{(i)}) - y^{(i)}\right) a_1^{(L-1)}(x^{(i)})^T, \tag{7}$$

$$\frac{\partial J}{\partial b_1^{(L)(i)}} = a_1^{(L)}(x^{(i)}) - y^{(i)}. \tag{8}$$

Также уровень L–1 глубокой нейронной сети прямого распространения соответствует первому (единственному скрытому) уровню нейронной сети прямого распространения, что позволяет найти значения $\frac{\partial J}{\partial w_1^{(L-1)(i)}}$ и $\frac{\partial J}{\partial b_1^{(L-1)(i)}}$ на основе формул (18), (19), (20) [2] через $\frac{\partial J}{\partial z^{(L-1)}(x^{(i)})}$:

$$\frac{\partial J}{\partial z^{(L-1)}(x^{(i)})} = \left[\left(a_1^{(L)}(x^{(i)}) - y^{(i)}\right) w_1^{(L)T}\right] * \left[\left(a^{(L-1)}(x^{(i)})\right)'\right]. \tag{9}$$

Здесь $C * D$ тоже означает поэлементное произведение вектор-столбцов C и D. Также функция активации ReLu, представленная как $\left(a^{(L-1)}(x^{(i)})\right)'$, имеет следующие значения:

$$\left(a^{(L-1)}(x^{(i)})\right)' = \begin{cases} 0 \text{ для } z_k^{(L-1)}(x^{(i)}) < 0, \\ 1 \text{ для } z_k^{(L-1)}(x^{(i)}) \geq 0. \end{cases} \tag{10}$$

В свою очередь

$$\frac{\partial J}{\partial w^{(L-1)(i)}} = \frac{\partial J}{\partial z^{(L-1)}(x^{(i)})} a^{(L-2)}(x^{(i)})^T, \tag{11}$$

$$\frac{\partial J}{\partial b^{(L-1)(i)}} = \frac{\partial J}{\partial z^{(L-1)}(x^{(i)})}. \tag{12}$$

При рассмотрении нейронной сети прямого распространения в формуле (14) [2] отмечено, что $\left(a_1^{(L)}(x^{(i)}) - y^{(i)}\right)$ для выходного уровня равняется $\frac{\partial J}{\partial z^{(L)}(x^{(i)})}$. Таким образом, на скрытом уровне l глубокой сети прямого распространения значение $\frac{\partial J}{\partial z^{(l)}(x^{(i)})}$, с использованием формулы (9), находится следующим образом:

$$\frac{\partial J}{\partial z^{(l)}(x^{(i)})} = \left[w^{(l+1)T} \frac{\partial J}{\partial z^{(l+1)}(x^{(i)})}\right] * \left[\left(a^{(l)}(x^{(i)})\right)'\right]. \tag{13}$$

По аналогии для скрытого уровня l значения $\frac{\partial J}{\partial w^{(l)(i)}}$ и $\frac{\partial J}{\partial b^{(l)(i)}}$ определяются как

$$\frac{\partial J}{\partial w^{(l)(i)}} = \frac{\partial J}{\partial z^{(l)}(x^{(i)})} a^{(l-1)}(x^{(i)})^T, \tag{14}$$

$$\frac{\partial J}{\partial b^{(l)(i)}} = \frac{\partial J}{\partial z^{(l)}(x^{(i)})}. \tag{15}$$

При этом для первого уровня (l = 1) значение $a^{(l-1)}(x^{(i)})$ равняется $x^{(i)}$. Так как размерности $a^{(l-1)}(x^{(i)})$ и $x^{(i)}$ отличаются, то при нахождении $\frac{\partial J}{\partial z^{(1)}(x^{(i)})}$, $\frac{\partial J}{\partial w^{(1)(i)}}$ и $\frac{\partial J}{\partial b^{(1)(i)}}$ формула отличается только для $\frac{\partial J}{\partial w^{(1)(i)}}$, которая рассчитывается следующим образом:

$$\frac{\partial J}{\partial w^{(1)(i)}} = \frac{\partial J}{\partial z^{(1)}(x^{(i)})} x^{(i)}. \tag{16}$$

Значения $\frac{\partial J}{\partial z^{(1)}(x^{(i)})}$ и $\frac{\partial J}{\partial b^{(1)(i)}}$ определяются на основе формул (13), 1(5). Также, как отмечалось при исследовании сети прямого распространения, для нулевого уровня производные не рассчитываются, так как входные показатели x_j были заданы и принимаются неизменными.

В результате найдены значения частных производных, $\frac{\partial J}{\partial w_1^{(L)(i)}}$, $\frac{\partial J}{\partial b_1^{(L)(i)}}$, ..., $\frac{\partial J}{\partial w^{(1)(i)}}$ и $\frac{\partial J}{\partial b^{(1)(i)}}$ для одного займа. Таким же образом осуществляется расчет значений данных частных производных для всех i займов, $i \in \{1, \ldots, m\}$. В конце рассчитываются средние значения $\frac{\partial J}{\partial w_1^{(L)}}$, $\frac{\partial J}{\partial b_1^{(L)}}$, ..., $\frac{\partial J}{\partial w^{(1)}}$ и $\frac{\partial J}{\partial b^{(1)}}$ по множеству i.

5. Использование метода градиентного спуска для нахождения лучшего значения параметров. Все последующие параметры нейронной сети обновляются одновременно с использованием следующих формул на основе данного метода:

$$\begin{aligned} w^{(1)} &:= w^{(1)} - \alpha \frac{\partial J}{\partial w^{(1)}}; \\ b^{(1)} &:= b^{(1)} - \alpha \frac{\partial J}{\partial b^{(1)}}; \\ &\ldots\ldots\ldots\ldots\ldots\ldots \\ w_1^{(L)} &:= w_1^{(L)} - \alpha \frac{\partial J}{\partial w_1^{(L)}}; \\ b_1^{(L)} &:= b_1^{(L)} - \alpha \frac{\partial J}{\partial b_1^{(L)}}. \end{aligned} \tag{17}$$

Здесь параметр α определяет размер шага градиентного спуска.

6. Обучение нейронной сети на тренировочном наборе данных путем многократного (10 000) повторения шагов 2–5. При обучении на каждой из итераций значение стоимостной функции должно быть меньше, чем на предыдущей. В результате определяются лучшие значения $w_{kh}^{(l)}$ и $b_k^{(l)}$, а также минимальное значение стоимостной функции $J(w,b)$.

После завершения обучения нейронной сети необходимо рассчитать ее точность при прогнозировании. Для этого с помощью лучших величин $w_{kh}^{(l)}$, $b_k^{(l)}$ и метода прямого распространения нейронной сети на тестовых данных рассчитываются значения $a_1^{(L)}(x^{(i)})$ и определяются $\hat{y}^{(i)} \in \{0,1\}$ для всех займов с учетом симметричности логистической функции относительно значения 0,51, которое было определено для нейронной сети прямого распространения [2]:

$$\begin{aligned} a^{(2)}(x^{(i)}) \geq 0,51 &\rightarrow \hat{y} = 1; \\ a^{(2)}(x^{(i)}) < 0,51 &\rightarrow \hat{y} = 0. \end{aligned}$$

После этого необходимо оценить эффективность исследуемой глубокой нейронной сети. Для этого нужно, используя $\hat{y}^{(i)}$ и $y^{(i)}$, рассчитать 4 основные метрики аналогично подходу, задействованному при использовании логистической регрессии и нейронной сети прямого распространения [1]: коэффициенты эффективности *Accuracy (A)*, *Precision (P)*, *Recall (R)* и меру F_1.

Классификация займов при разной структуре глубокой нейронной сети. Учитывая имеющиеся вычислительные мощности, исследование проводится в пределах 10 скрытых уровней нейронной сети. В обширном эксперименте с простой нейронной сетью были получены лучшие значения: α = 0,87, количество нейронов на каждом скрытом уровне составило 93 и гра-

ничное значение достигло 0,51 [2]. Мы предполагаем, что эти же значения должны быть подходящими для рассматриваемой задачи. При этом последовательно анализируются результаты обучения нейронной сети, состоящего из 10 000 итераций градиентного спуска с количеством скрытых уровней от 2 до 10, и определяется лучшая глубина данной сети. Результаты, соответствующие максимальному значению A на тестовых данных, представлены в табл. 1 и 2.

Таблица 1. **Результаты эксперимента по определению лучшей структуры глубокой нейронной сети**

Table 1. **Experiment results on determining the best structure of a deep neural network**

Результаты 10 000 итераций при α = 0,87	Значение
Лучшее количество нейронов в скрытом слое	93
Лучший α	0,87
Лучшее количество скрытых уровней	2
Длительность обучения алгоритма (одной итерации), с	5,62
Величина стоимостной функции	0,442 15
Accuracy training, %	80,46
Accuracy testing, %	80,35

Таблица 2. **Ключевые метрики при лучшей структуре глубокой нейронной сети**

Table 2. **Key metrics when using the best structure of a deep neural network**

Класс	*Precision*	*Recall*	Мера F_1
Невозвратные займы, %	58,61	10,47	17,77
Возвратные займы, %	81,17	98,12	88,84
Средневзвешенное, %	76,60	80,35	74,44

В итоге проведенного эксперимента лучшей (имеющей максимальное значение A на тестовых данных) является глубокая нейронная сеть с двумя скрытыми уровнями по 93 нейрона в каждом. При этом значение A составило 80,35 %, что выше значения, полученного при исследовании нейронной сети прямого распространения (с одним скрытым уровнем) [2].

Альтернативная инициализация весов глубокой нейронной сети. Как было отмечено ранее [2], начальные значения весов $w_{kh}^{(l)}$ нейронов должны быть отличными от нуля для нейронной сети прямого распространения. Данное требование распространяется и на глубокую нейронную сеть. Однако при обучении очень глубокой сети может возникнуть другое осложнение: проблема исчезающего либо взрывного градиента [8]. В случае исчезающего градиента при обучении очень глубокой нейронной сети производные на последующем уровне могут экспоненциально уменьшаться при $w_{kh}^{(l)} < 1,$ что приводит к быстрому изменению весов на начальных уровнях, но к замедлению обучения на конечных уровнях, а значит, к несущественным изменениям конечной функции активации сети или к полной остановке обучения. При взрывном градиенте, когда $w_{kh}^{(l)} > 1,$ производные могут увеличиваться экспоненциально, что приведет к очень быстрому изменению конечной функции активации и усложнит процесс обучения такой сети.

Примененный ранее подход к инициализации весов (на основе стандартного нормального распределения со средним значением, равным 0, стандартным отклонением, равным 1, умноженным на 10^{-2}) решает данную проблему. Однако так как при решении данной задачи применяется функция активации *Relu* для скрытых уровней, то на практике для данной функции активации часто применяется инициализация *He*, т. е. для отдельного нейрона задается случайное значение веса $w_{kh}^{(l)}$ на базе стандартного гауссовского распределения со средним значением, равным 0, и стандартным отклонением, равным $\sqrt{\frac{2}{H^{(l-1)}}}$ [9]. В итоге $w_{kh}^{(l)}$ имеют значения незначительно выше или ниже 1, а значение $z_k^{(l)}$ является более стабильным, что также должно предотвращать возникновение проблемы исчезающего или взрывного градиента.

Для оценки влияния применения *He* инициализации на результаты обучения глубокой нейронной сети нужно провести эксперимент, аналогично подходу выше, и сравнить результаты по сравнению с инициализацией весов $w_{kh}^{(l)}$ произвольным образом на основе стандартного нормального распределения (табл. 3, 4).

Таблица 3. **Результаты при использовании *He* инициализации весов**

Table 3. **Results when using *He* initialization of weights**

Результаты 10 000 итераций при α = 0,87	Значение
Лучшее количество нейронов в скрытом слое (определенное ранее)	93
Лучший α (определенный ранее)	0,87
Лучшее количество скрытых уровней (определенное ранее)	2
Длительность обучения алгоритма (одной итерации), с	5,18
Величина стоимостной функции	0,441 11
Accuracy training, %	80,50
Accuracy testing, %	80,20

Таблица 4. **Ключевые метрики при использовании *He* инициализации весов**

Table 4. **Key metrics when using *He* initialization of weights**

Класс	*Precision*	*Recall*	Мера F_1
Невозвратные займы, %	56,23	10,60	17,84
Возвратные займы, %	81,16	97,90	88,74
Средневзвешенное, %	76,11	80,21	74,37

В результате проведенного исследования лучшей (имеющей максимальное значение *A* на тестовых данных) является глубокая сеть с двумя скрытыми уровнями по 93 нейрона в каждом. При этом использование *He* инициализации не привело к улучшению исследуемого параметра *A*. Это можно объяснить тем, что значения весов $w_{kh}^{(l)}$ нейронов при начальном подходе к инициализации в абсолютном значении были меньше, чем веса при *He* инициализации из-за уменьшения первых в 10^{-2} раз. Поэтому в дальнейшем инициализация весов нейронов осуществляется, как и ранее, на основе стандартного нормального распределения со средним значением, равным 0, стандартным отклонением, равным 1, умноженными на 10^{-2}.

Использование мини-пакетного градиентного спуска для обучения глубокой нейронной сети. Во всех предыдущих экспериментах проводилось обучение глубокой нейронной сети с использованием стандартного (пакетного) градиентного спуска, который предполагает обработку всего тренировочного набора данных, состоящего из 855 211 позиций, для выполнения одного шага градиентного спуска. В свою очередь для большого набора данных можно использовать более быстрый мини-пакетный алгоритм градиентного спуска [10], который позволит алгоритму градиентного спуска начать делать шаги к лучшему значению до того, как закончится обработка всего набора тренировочных данных. Реализация алгоритма осуществляется следующим образом: весь набор тренировочных данных разделяется для *X* и *Y*, как пример, на равные мини-партии по 1024 позиции, т. е. 855211/1024 = 835 плюс дополнительная партия, которая содержит остаток, содержащий 171 позицию, что в итоге составляет количество мини-партий $t = 836$. Модифицированный набор тренировочных данных можно представить в виде

$$\begin{aligned} X_{n,m} &= [X^{\{1\}}, X^{\{2\}}, \ ..., X^{\{t\}}]; \\ Y_{1,m} &= [Y^{\{1\}}, Y^{\{2\}}, \ ..., Y^{\{t\}}], \end{aligned} \tag{18}$$

где $X^{\{1\}}$ – матрица, размером *n* на 1024, а $Y^{\{1\}}$ – вектор с 1024 позициями.

Далее обучение глубокой нейронной сети осуществляется с использованием градиентного спуска аналогично шагам, обозначенным выше, с той лишь разницей, что в рамках одной итерации используется не весь набор тренировочных данных, а только позиции из партии. Таким образом, по результатам обработки всего набора тренировочных данных выполняется 836 шагов алгоритма градиентного спуска вместо одного шага, что существенно ускоряет процесс обучения данной сети. При этом однократное использование всех мини-партий (в данном случае 836) для обучения сети принято называть эпохой.

В свою очередь возникает вопрос о лучшем размере мини-партии. Для начала рассматриваются 2 граничных значения. В первом случае размер мини-партии равняется всему набору тренировочных данных. Тогда фактически обучение нейронной сети проходит так же, как и при использовании стандартного градиентного спуска ранее. При втором крайнем значении размер партии составит 1 (т. е. каждая позиция является партией), и в таком случае алгоритм градиентного спуска принято называть стохастическим. Ни один из граничных вариантов не является лучшим: как отмечалось ранее, при стандартном градиентном спуске для реализации одного шага алгоритма требуется обработать весь набор данных, а при стохастическом градиенте, несмотря на прогресс в обучении сети при обработке каждой позиции, теряется ускорение обучения сети из-за невозможности использования векторизации (в связи с необходимостью обработки каждой позиции). Соответственно, лучшее значение мини-партии должно находиться между двумя граничными величинами, которое должно приводить к более быстрому обучению нейронной сети. На практике при большом наборе тренировочных данных (более 2000) подходящими размерами мини-партий являются 64, 128, 256 и 512, что связывают с размерностью размещения и использования памяти на вычислительном устройстве [7, p. 276]. Таким образом, принято считать, что реализации алгоритма мини-пакетного градиентного спуска осуществляются быстрее, если размер партии соответствует величине 2 в степени от 6 и выше.

Для определения лучшего размера партии в рамках текущей задачи рассматриваются следующие размеры мини-партий, соответствующие величине 2 в степени от 6 по 19: 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16 384, 32 768, 65 536, 131 072, 262 144 и 524 288. Однако в ходе предварительного анализа выяснилось, что при реализации мини-пакетного алгоритма градиентного спуска для отдельных размеров партий стоимостная функция перестает сходиться и начинает ухудшаться. Это означает, что текущее значение $\alpha = 0{,}87$ при использовании мини-пакетного подхода перестает быть лучшим и требует нахождения нового лучшего значения. Поэтому одновременно с поиском лучшего значения мини-партии необходимо определить лучшее значение α с помощью двухшагового подхода, который был рассмотрен при исследовании нейронной сети прямого распространения с использованием 10 000 эпох (табл. 5, 6).

Таблица 5. **Итоги исследования при использовании мини-пакетного градиентного спуска**

Table 5. **Research results when using mini-batch gradient descent**

Результаты 10 000 эпох при $\alpha = 0{,}87$	Значение
Лучшее количество нейронов в скрытом слое (определенное ранее)	93
Лучшее количество скрытых уровней (определенное ранее)	2
Лучший размер мини-партии	32 768
Лучший α	0,007 20
Длительность обучения алгоритма (одной эпохи), с	17,07
Величина стоимостной функции	0,415 33
Accuracy training, %	80,45
Accuracy testing, %	80,34

Таблица 6. **Ключевые метрики при использовании мини-пакетного градиентного спуска**

Table 6. **Key metrics when using mini-batch gradient descent**

Класс	*Precision*	*Recall*	Мера F_1
Невозвратные займы, %	56,45	13,22	21,42
Возвратные займы, %	81,53	97,41	88,77
Средневзвешенное, %	76,45	80,34	75,11

По итогам исследования видно, что лучшим размером мини-партии является 32 768, а значением $\alpha = 0{,}007\,20$. Также значение величины A на тестовых данных было 80,34 %, что ниже значения 80,35 %, полученного ранее. При этом стоит отметить, что значение меры F_1 увеличилось и составило 75,11 %. Так как целью оптимизации решаемой задачи является поиск максимального значения величины A, то в дальнейших исследованиях мини-пакетный градиентный спуск не используется. Однако в случае поиска максимального значения меры F_1 применение мини-пакетного градиентного спуска видится обоснованным. Таким образом, применение мини-пакетного градиентного спуска может быть целесообразным при оптимизации отдельных метрик.

Использование альтернативных методов оптимизации. Существуют альтернативные методы оптимизации обучения нейронной сети, которые, как считается, являются более быстрыми, чем алгоритм градиентного спуска. Данные алгоритмы основаны на использовании экспоненциально взвешенных скользящих средних [11], определение которых можно представить с помощью следующей формулы:

$$V_t = \beta V_{t-1} + (1-\beta)\theta_t. \tag{19}$$

Таким образом, если имеется временной ряд параметра θ, то его экспоненциально взвешенное скользящее среднее значение в момент времени t зависит от коэффициента взвешенного уменьшения $\beta \in [0,1]$, экспоненциального взвешенного скользящего среднего значения в предыдущий момент времени V_{t-1} и значения параметра в данный момент θ_t при $V_0 = 0$. Использование данного подхода позволяет определить тренд изменения параметра θ несмотря на существенные изменения значений данного параметра в отдельные моменты времени.

Рассмотрим следующие альтернативные алгоритмы обучения нейронной сети, которые используют принцип экспоненциального взвешенного скользящего среднего.

1. Градиентный спуск с импульсом [5]. В отличие от обычного градиентного спуска, при котором направление обновления параметров нейронной сети имеет существенную дисперсию и соответственно замедляет нахождение лучшего значения, в случае использования данного алгоритма рассчитывается экспоненциальное взвешенное скользящее среднее градиента и далее применяется для обновления параметров нейронной сети. В результате дисперсия направления обновления параметров значительно уменьшается, что существенно ускоряет нахождение лучшего значения. Если обратиться к этапам обучения глубокой нейронной сети, указанным выше, то после нахождения значений $\frac{\partial J}{\partial w_1^{(L)}}$, $\frac{\partial J}{\partial b_1^{(L)}}$, ..., $\frac{\partial J}{\partial w^{(1)}}$ и $\frac{\partial J}{\partial b^{(1)}}$ необходимо рассчитать экспоненциальные взвешенные скользящие средние значения при $l = 1, \ldots, L$:

$$\begin{aligned} V_{\partial w^{(l)}} &:= \beta V_{\partial w^{(l)}} + (1-\beta)\frac{\partial J}{\partial w^{(l)}}; \\ V_{\partial b^{(l)}} &:= \beta V_{\partial b^{(l)}} + (1-\beta)\frac{\partial J}{\partial b^{(l)}}. \end{aligned} \tag{20}$$

В связи с этим формулы для обновления параметров нейронной сети видоизменяются:

$$\begin{aligned} w^{(l)} &:= w^{(l)} - \alpha V_{\partial w^{(l)}}; \\ b^{(l)} &:= b^{(l)} - \alpha V_{\partial b^{(l)}}. \end{aligned} \tag{21}$$

В результате внесенных изменений процесс обучения нейронной сети должен выполняться быстрее по сравнению с использованием стандартного градиента. Также стоит отметить, что

изначально $V_{\partial w^{(l)}}$ и $V_{\partial b^{(l)}}$ равны нулю. Таким образом, значения $V_{\partial w^{(l)}}$ и $V_{\partial b^{(l)}}$ на каждой итерации градиентного спуска рассчитываются на основе значений предыдущей операции и величины градиента на текущей итерации. В свою очередь общепринятым значением β принято считать величину 0,9. Несмотря на то, что данный алгоритм в основном лучше стандартного градиентного спуска, требуется оценить целесообразность его использования для решения текущей задачи, а также установить лучшее для этой задачи значение β в диапазоне от 0 до 1 с шагом 0,01. В табл. 7 и 8 представлены результаты проведенного эксперимента.

Таблица 7. **Итоги исследования при использовании градиентного спуска с импульсом**

Table 7. **Research results when using gradient descent with momentum**

Результаты 10 000 итераций при α = 0,87	Значение
Лучшее количество нейронов в скрытом слое (определенное ранее)	93
Лучшее количество скрытых уровней (определенное ранее)	2
Лучший коэффициент взвешенного уменьшения β	0,16
Длительность обучения алгоритма (одной итерации), с	11,68
Величина стоимостной функции	0,441 73
Accuracy training, %	80,49
Accuracy testing, %	80,36

Таблица 8. **Ключевые метрики при использовании градиентного спуска с импульсом**

Table 8. **Key metrics when using gradient descent with momentum**

Класс	*Precision*	*Recall*	Мера F_1
Невозвратные займы, %	58,17	11,01	18,51
Возвратные займы, %	81,24	97,99	88,83
Средневзвешенное, %	76,56	80, 36	74,58

2. Адаптивная оценка момента (Adam) состоит из комбинации рассмотренного ранее алгоритма градиентного спуска с импульсом и алгоритма RMSprop [12]. Алгоритм RMSprop, по сути, применяется с той же целью, что и алгоритм градиентного спуска с импульсом: уменьшение дисперсии направления при обновлении параметров градиентного спуска. Однако в случае RMSprop для расчета экспоненциальных взвешенных скользящих средних значений используются следующие формулы при $l = 1, \ldots, L$:

$$S_{\partial w^{(l)}} := \beta S_{\partial w^{(l)}} + (1-\beta)\left(\frac{\partial J}{\partial w^{(l)}}\right)^2;$$
$$S_{\partial b^{(l)}} := \beta S_{\partial b^{(l)}} + (1-\beta)\left(\frac{\partial J}{\partial b^{(l)}}\right)^2, \tag{22}$$

$$w^{(l)} := w^{(l)} - \alpha \frac{\dfrac{\partial J}{\partial w^{(l)}}}{\sqrt{S_{\partial w^{(l)}}}};$$
$$b^{(l)} := b^{(l)} - \alpha \frac{\dfrac{\partial J}{\partial b^{(l)}}}{\sqrt{S_{\partial b^{(l)}}}}. \tag{23}$$

В свою очередь алгоритм Adam считается очень эффективным при решении различных задач с использованием разнообразных архитектурных решений глубокого машинного обучения. Как отмечалось выше, он объединяет алгоритмы обучения с импульсом и RMSprop. Для реализации алгоритма требуется задать начальные значения: $V_{\partial w^{(l)}} = 0$, $V_{\partial b^{(l)}} = 0$, $S_{\partial w^{(l)}} = 0$ и $S_{\partial b^{(l)}} = 0$. Далее осуществляются следующие вычисления:

$$
\begin{aligned}
V_{\partial w^{(l)}} &:= \beta_1 V_{\partial w^{(l)}} + (1-\beta_1)\frac{\partial J}{\partial w^{(l)}};\\
V_{\partial b^{(l)}} &:= \beta_1 V_{\partial b^{(l)}} + (1-\beta_1)\frac{\partial J}{\partial b^{(l)}};\\
S_{\partial w^{(l)}} &:= \beta_2 S_{\partial w^{(l)}} + (1-\beta_2)\left(\frac{\partial J}{\partial w^{(l)}}\right)^2;\\
S_{\partial b^{(l)}} &:= \beta_2 S_{\partial b^{(l)}} + (1-\beta_2)\left(\frac{\partial J}{\partial b^{(l)}}\right)^2;\\
V^{\text{corrected}}_{\partial w^{(l)}} &:= \frac{V_{\partial w^{(l)}}}{1-(\beta_1)^t};\\
V^{\text{corrected}}_{\partial b^{(l)}} &:= \frac{V_{\partial b^{(l)}}}{1-(\beta_1)^t};\\
S^{\text{corrected}}_{\partial w^{(l)}} &:= \frac{S_{\partial w^{(l)}}}{1-(\beta_2)^t};\\
S^{\text{corrected}}_{\partial b^{(l)}} &:= \frac{S_{\partial b^{(l)}}}{1-(\beta_2)^t};\\
w^{(l)} &:= w^{(l)} - \alpha\frac{V^{\text{corrected}}_{\partial w^{(l)}}}{\sqrt{S^{\text{corrected}}_{\partial w^{(l)}}}+\varepsilon};\\
b^{(l)} &:= b^{(l)} - \alpha\frac{V^{\text{corrected}}_{\partial b^{(l)}}}{\sqrt{S^{\text{corrected}}_{\partial b^{(l)}}}+\varepsilon}.
\end{aligned}
\tag{24}
$$

При этом t является счетчиком шагов, выполненных алгоритмом Adam, и имеет начальное значение, равное 0; $l = 1, \ldots, L$; α – коэффициент скорости обучения; β_1 и β_2 – параметры, контролирующие расчет экспоненциальных взвешенных скользящих средних значений; ε – очень малое значение для избегания деления на 0. В соответствии с рекомендацией [12] при использовании алгоритма Adam хорошими значениями по умолчанию являются $\alpha = 0{,}001$; $\beta_1 = 0{,}9$; $\beta_2 = 0{,}999$; $\varepsilon = 10^{-8}$. На практике при использовании алгоритма Adam наиболее часто анализируется влияние α на результат обучения нейронной сети. В связи с тем, что алгоритм Adam существенно отличается от стандартного градиентного спуска, определенная ранее величина $\alpha = 0{,}87$ уже может не являться лучшей и целесообразно дополнительно ее исследовать. Поэтому был проведен анализ применения алгоритма Adam при решении текущей задачи и определено лучшее значение α с помощью двухшагового подхода по аналогии с подходом, использованным при исследовании нейронной сети прямого распространения (табл. 9, 10).

Таблица 9. **Итоги исследования при использовании алгоритма Adam**

Table 9. **Research results when using algorithm Adam**

Результаты 10 000 итераций	Значение
Лучшее количество нейронов в скрытом слое (определенное ранее)	93
Лучшее количество скрытых уровней (определенное ранее)	2
Лучший α	0,0001
Параметр β_1	0,9
Параметр β_2	0,999
Параметр ε	1e-8
Длительность обучения алгоритма (одной итерации), с	11,94

Окончание табл. 9

Результаты 10 000 итераций	Значение
Величина стоимостной функции	0,443 65
Accuracy training, %	80,45
Accuracy testing, %	80,28

Таблица 10. **Ключевые метрики при использовании алгоритма Adam**

Table 10. **Key metrics when using algorithm Adam**

Класс	*Precision*	*Recall*	Мера F_1
Невозвратные займы, %	55,75	13,17	21,31
Возвратные займы, %	81,51	97,34	88,73
Средневзвешенное, %	76,29	80,28	75,06

Из полученных результатов видно, что использование градиентного спуска с импульсом привело к увеличению (улучшению) значения величины A на тестовых данных до 80,36 %, что выше значения, полученного ранее. При этом лучший коэффициент взвешенного уменьшения β равен 0,16. В свою очередь применение алгоритма Adam не привело к улучшению выбранного параметра A. Однако стоит отметить, что при применении алгоритма Adam использовались значения по умолчанию для параметров $\beta_1 = 0{,}9$; $\beta_2 = 0{,}999$; $\varepsilon = 10^{-8}$. Дальнейшее изучение каждого из этих параметров потенциально может привести к лучшему результату.

Использование альтернативного разделения данных на тренировочный и тестовый наборы. В предыдущих экспериментах весь набор исходных данных был разделен на 2 множества: тренировочные (70 %) и тестовые (30 %) данные. Такой подход к разделению часто использовался на практике на ранних этапах развития машинного обучения, когда исходные наборы данных находились в пределах 10 000. В настоящее время с ростом объема исходных данных часто прибегают к изменению данной пропорции в сторону увеличения доли тренировочного набора данных. В связи с этим целесообразно провести анализ влияния изменения пропорции тренировочного набора данных на точность прогнозирования глубокой нейронной сети в рамках текущей задачи классификации займа. Для этого доля тренировочного набора данных при обучении нейронной сети изменяется с 70 до 99 % с шагом 1 %. В свою очередь доля тестового набора данных уменьшается с 30 до 1 % с тем же шагом. Полученные результаты проведенного эксперимента отражены в табл. 11 и 12.

Таблица 11. **Итоги исследования при использовании альтернативного разделения данных**

Table 11. **Research results when using alternative data separation**

Результаты 10 000 итераций при α = 0,87	Значение
Лучшее количество нейронов в скрытом слое (определенное ранее)	93
Лучшее количество скрытых уровней (определенное ранее)	2
Лучший коэффициент взвешенного уменьшения β	0,16
Доля лучшего тренировочного набора данных, %	94
Длительность обучения алгоритма (одной итерации), с	7,38
Величина стоимостной функции	0,442 56
Accuracy training, %	80,40
Accuracy testing, %	80,50

Таблица 12. **Ключевые метрики при использовании альтернативного разделения данных**

Table 12. **Key metrics when using alternative data separation**

Класс	*Precision*	*Recall*	Мера F_1
Невозвратные займы, %	0,542c86	0,174 87	0,264 53
Возвратные займы, %	0,823 08	0,963 06	0,887 58
Средневзвешенное, %	0,766 88	0,804 98	0,762 62

В данном случае изменение доли тренировочного набора с 70 до 94 % привело к улучшению точности прогнозирования выраженной коэффициентом эффективности A глубокой нейронной сети в рамках рассматриваемой задачи до 80,50 %, что является наилучшим полученным результатом. При этом отметим, что изменение доли тренировочного набора по-разному влияет на остальные метрики P, R и F_1. Соответственно для каждой метрики может быть своя лучшая доля тренировочного набора данных. Таким образом, как видно из результатов проведенных исследований, целесообразно проводить анализ поиска лучшей пропорции разбиения тренировочного и тестового наборов данных.

Использование метода исключения для задачи классификации займа. Как было обнаружено в [2], увеличение количества входных показателей с 54 до 73 улучшило метрики нейронной сети прямого распространения. Дальнейший анализ на избыточность количества входных показателей с использованием метода главных компонент не привел к улучшению коэффициента эффективности A на тестовых данных. Однако остается возможность для получения лучших результатов обучения нейронной сети при влиянии на вес каждого из 73 входных показателей при обучении глубокой нейронной сети прямого распространения. Для устранения чрезмерной аллокации веса для одних входных показателей по сравнению другими при обучении глубокой нейронной сети целесообразно использовать метод исключения [13] – на каждой итерации обучения сети с вероятностью p значение $x_j^{(i)}$ приравнивается 0, т. е. не учитывается (исключается) из обучения глубокой нейронной сети. Следовательно, сеть не сможет полагаться (присваивать больший вес) на небольшое количество входных показателей, так как они могут быть исключены на отдельной итерации обучения, и вынуждена более равномерно распределять веса между входными показателями. В рамках данного исследования анализируются значения вероятности p для $x_j^{(i)}$ на отдельной итерации обучения от 1 по 99 %. Однако стоит отметить, что при использовании тестовых данных исключение не применяется.

Согласно результатам, лучшее значение вероятности p равнялось 2 %. Таким образом, исключение 2 % значений $x_j^{(i)}$ при обучении глубокой нейронной сети привели к результатам, представленным в табл. 13, 14.

Таблица 13. **Результаты исследования при использовании метода исключения**

Table 13. **Results of the study using the dropout method**

Результаты 10 000 итераций при α = 0,87	Значение
Лучшее количество нейронов в скрытом слое (определенное ранее)	93
Лучшее количество скрытых уровней (определенное ранее)	2
Лучший коэффициент взвешенного уменьшения β	0,16
Доля лучшего тренировочного набора данных, %	94
Лучшая вероятность исключения значений входных параметров p, %	2
Длительность обучения алгоритма (одной итерации), с	15,16
Величина стоимостной функции	0,442 56
Accuracy training, %	80,40
Accuracy testing, %	80,65

Таблица 14. **Ключевые метрики при использовании метода исключения**

Table 14. **Key metrics when using the dropout method**

Класс	*Precision*	*Recall*	Мера F_1
Невозвратные займы, %	59,79	10,74	18,21
Возвратные займы, %	81,43	98,19	89,03
Средневзвешенное, %	77,09	80,65	74,82

Как следует из полученных результатов, применение метода исключения привело к увеличению значения коэффициента эффективности A = 80,65 %, что больше значения, полученного ра-

нее – 80,50 %. Таким образом, использование данного метода целесообразно, но ситуация может измениться, если в качестве оптимизации выбирается другой параметр.

Заключение. В данной работе рассмотрено использование глубокой нейронной сети прямого распространения для решения задачи классификации займа. В результате определена лучшая структура изучаемой нейронной сети. Однако установлено, что альтернативное применение инициализации *He* не привело к улучшению выбранной метрики модели. Также показано, что использование мини-пакетного градиентного спуска не привело к улучшению значения величины *A*. В свою очередь при задействовании альтернативных методов оптимизации глубокой нейронной сети выявлено, что применение градиентного спуска с импульсом привело к улучшению модели. Вместе с тем обнаружено, что осуществление поиска альтернативного разбиения данных на тренировочный и тестовый наборы является целесообразным, так как это приводит к улучшению модели на основе глубокой нейронной сети по выбранной метрике. Также определено, что использование метода исключения на нулевом уровне глубокой нейронной сети приводит к улучшению коэффициента эффективности *A*.

Итоговая точность прогнозирования при использовании глубокой сети прямого распространения оказалась выше полученной при применении нейронной сети прямого распространения и логистической регрессии, рассмотренных в [1, 2].

Список использованных источников

1. Бегунков, В. И. Классификация займов с использованием логистической регрессии / В. И. Бегунков, М. Я. Ковалев // Информатика. – 2023. − Т. 20, № 1. – С. 55–74. https://doi.org/10.37661/1816-0301-2023-20-1-55-74

2. Бегунков, В. И. Классификация займа с использованием нейронной сети прямого распространения / В. И. Бегунков // Информатика. – 2024. − Т. 21, № 1. – С. 55–74. https://doi.org/10.37661/1816-0301-2024-21-1-83-104

3. Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research / S. Lessmann, B. Baesens, H.-V. Seow, L. C. Thomas // European Journal of Operational Research. – 2015. – Vol. 247, № 1. – P. 124–136. https://doi.org/10.1016/j.ejor.2015.05.030

4. Shalev-Shwartz, S. Understanding Machine Learning: From Theory to Algorithms / S. Shalev-Shwartz, S. Ben-David. – Cambridge University Press, 2014. – 397 p. https://doi.org/10.1017/CBO9781107298019

5. Rumelhart, D. Learning representations by back-propagating errors / D. Rumelhart, G. Hinton, R. Williams // Nature. – 1986. – Vol. 323. – P. 533–536. https://doi.org/10.1038/323533a0

6. Geron, A. Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow / A. Geron. – 2nd ed. – O'Reilly Media, 2019. – 483 p.

7. Goodfellow, I. Deep Learning / I. Goodfellow, Y. Bengio, A. Courville. – MIT Press, 2016. – 800 p.

8. Glorot, X. Understanding the difficulty of training deep feedforward neural networks / X. Glorot, Y. Bengio // Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS), 2010, Chia Laguna Resort, Sardinia, Italy. – 2010. – Vol. 9. – P. 249–256.

9. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification / K. He, X. Zhang, S. Ren, J. Sun // Proceedings of the IEEE International Conference on Computer Vision (ICCV). – 2015. – P. 1026–1034. https://doi.org/10.1109/iccv.2015.123

10. Efficient BackProp / Y. LeCun, L. Bottou, G. B. Orr, K.-R. Müller // Neural Networks: Tricks of the Trade. – Berlin; Heidelberg: Springer, 1998. – P. 9–50. – (Lecture Notes in Computer Science; vol 1524). https://doi.org/10.1007/3-540-49430-8_2

11. Roberts, S. W. Control chart tests based on geometric moving averages / S. W. Roberts // Technometrics. – 1958. – Vol. 1, № 3. – P. 239–250. https://doi.org/10.1080/00401706.1959.10489860

12. Kingma, D. P. Adam: A Method for Stochastic Optimization / D. P. Kingma, J. Ba // Arxiv [Preprint]. – 2014. – URL: https://arxiv.org/abs/1412.6980; https://doi.org/10.48550/arXiv.1412.6980

13. Dropout: A simple way to prevent neural networks from overfitting / N. Srivastava, G. Hinton, A. Krizhevsky [et al.] // Journal of Machine Learning Research. – 2014. – Vol. 15, № 1. – P. 1929–1958.

References

1. Behunkou U. I., Kovalyov M. Y. Loan classification using logistic regression. *Informatics*, 2023, vol. 20, no. 1, pp. 55–74 (in Russian). https://doi.org/10.37661/1816-0301-2023-20-1-55-74

2. Behunkou U. I. Loan classification using a feed-forward neural network. *Informatics*, 2024, vol. 21, no. 1, pp. 83–104 (in Russian). https://doi.org/10.37661/1816-0301-2024-21-1-83-104

3. Lessmann S., Baesens B., Seow H.-V., Thomas L. C. Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research. *European Journal of Operational Research*, 2015, vol. 247, no. 1, pp. 124–136. https://doi.org/10.1016/j.ejor.2015.05.030

4. Shalev-Shwartz S., Ben-David S. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014. 397 p. https://doi.org/10.1017/CBO9781107298019.

5. Rumelhart D., Hinton G., Williams R. Learning representations by back-propagating errors. *Nature*, 1986, vol. 323, pp. 533–536. https://doi.org/10.1038/323533a0

6. Geron A. *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow.* 2nd ed. O'Reilly Media, 2019. 483 p.

7. Goodfellow I., Bengio Y., Courville A. *Deep Learning*. MIT Press, 2016. 800 p.

8. Glorot X., Bengio Y. Understanding the difficulty of training deep feedforward neural networks. *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS) 2010, Chia Laguna Resort, Sardinia, Italy. Vol. 9*. 2010, pp. 249–256.

9. He K., Zhang X., Ren S., Sun J. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. *In Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1026–1034. https://doi.org/10.1109/iccv.2015.123

10. LeCun Y., Bottou L., Orr G. B., Müller K.-R. Efficient BackProp. *Neural Networks: Tricks of the Trade. Lecture Notes in Computer Science, vol 1524*. Berlin, Heidelberg, Springer, 1998, pp. 9–50. https://doi.org/10.1007/3-540-49430-8_2

11. Roberts S. W. Control chart tests based on geometric moving averages. *Technometrics*, 1958, vol. 1, no. 3, pp. 239–250. https://doi.org/10.1080/00401706.1959.10489860

12. Kingma D. P., Ba J. Adam: A Method for Stochastic Optimization. *Arxiv* [Preprint], 2010. Available at: https://arxiv.org/abs/1412.6980; https://doi.org/10.48550/arXiv.1412.6980

13. Srivastava N., Hinton G., Krizhevsky A., Sutskever I., Salakhutdinov R. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 2014, vol. 15, no. 1, pp. 1929–1958.

Информация об авторах

Бегунков Владимир Иванович – магистр технических наук, Объединенный институт проблем информатики Национальной академии наук Беларуси (ул. Сурганова, 6, Минск, 220012, Республика Беларусь). E-mail: vbegunkov@gmail.com

Ковалев Михаил Яковлевич – доктор физико-математических наук, профессор, Объединенный институт проблем информатики Национальной академии наук Беларуси (ул. Сурганова, 6, Минск, 220012, Республика Беларусь). kovalyov_my@newman.bas-net.by

Information about the authors

Uladzimir I. Behunkou – Master of Engineering, United Institute of Informatics Problems of the National Academy of Sciences of Belarus (6, Surganov Str., 220012, Minsk, Republic of Belarus). E-mail: vbegunkov@gmail.com

Mikhail Y. Kovalyov – Dr. Sc. (Physics and Mathematics), Professor, United Institute of Informatics Problems of the National Academy of Sciences of Belarus (6, Surganov Str., 220012, Minsk, Republic of Belarus). E-mail: kovalyov_my@newman.bas-net.by