УДК 004.852

ИССЛЕДОВАНИЕ ВЛИЯНИЕ ГИПЕРПАРАМЕТРОВ НА КАЧЕСТВО ОБУЧЕНИЯ МНОГОСЛОЙНОГО ПЕРСЕПТРОНА ПРИ ИСПОЛЬЗОВАНИИ ПОЭЛЕМЕНТНОГО ГРАДИЕНТНОГО СПУСКА

Михалко А.Н., студент гр.453503, Филон Д.А., студент гр. 453502

Белорусский государственный университет информатики и радиоэлектроники, г. Минск, Республика Беларусь

Рыкова О.В. – канд. физ.-мат. наук, доцент

Аннотация. В данной работе проведен анализ различных функций активации и функций ошибок в отношении эффективности обучения нейронной сети на основе многослойного персептрона с малым количеством эпох при прочих равных гиперпараметрах.

Ключевые слова. Персептрон, матрицы, функции активации, функции ошибки.

Введение. Двадцать первый век стал веком нейронных, все чаще они используются как в информационных технология, так и в повседневной и все больше и больше заинтересовывают юное поколение изучать темы с ними связанными. Данная работа создана помочь, всем кто хочет эффективно обучать нейросети с учетом ограниченных вычислительных ресурсов, чтобы достичь максимального эффекта в обучении нейросетей.

Основная часть

Гиперпараметры — это настраиваемые параметры алгоритмов машинного обучения, которые задаются до начала процесса обучения и остаются неизменными в ходе этого процесса. В отличие от параметров модели, которые обучаются на данных (например, веса нейронов в нейронных сетях), гиперпараметры контролируют сам процесс обучения и структуру модели.

Многослойный персептрон (MLP, Multilayer Perceptron) — это искусственная нейронная сеть прямого распространения, состоящая из трех или более слоев (входной, скрытые и выходной слои), которая способна решать нелинейные задачи.

Рассмотрим, как происходит процесс обучения персептрона на примере, состоящем из одного входного слоя, двух скрытых слоев и слоя выхода. Для удобного отображения значения элементов слоя, весов и сдвигов будет использовать транспонированные матрицы. На первый слой подается матрица $[x_1^1, x_2^1, \dots, x_n^1]$из нормализированных чисел ($0 \le x_i^1 \le 1$), где нижний коэффициент $1 \le i \le n$ означает номер элемента в слое, а верхний – номер слоя. Затем высчитывается значения элементов для второго скрытого слоя $\left[x_1^2, x_2^2, \dots, x_k^2\right]$, второй слой состоит из k элементов, получая значения в следствии операции умножения матрицы первого слоя $[x_1^1, x_2^1, \dots, x_n^1]$ и матрицы весов $\begin{bmatrix} w_{11}^1 & \cdots & w_{1k}^1 \\ \vdots & \ddots & \vdots \\ w_{n1}^1 & \cdots & w_{nk}^1 \end{bmatrix}$. Далее к результату прибавляется матрица сдвигов $\left[b_1^1, b_2^1, \dots, b_k^1\right]$. В результате получается матрица размером $1 \times k$, со значениями нейронов второго слоя. Затем дабы избежать линейности нашей функции, необходимо к каждому элементу второго слоя применить функцию активации в качестве нее может выступать:

$$Sigmoidf(x) = \frac{1}{1 + e^{-x}} \quad (1)$$

$$ReLUf(x) = max(0, x) \quad (2)$$

Применив ко второму слою функцию активации F(x) мы получаем матрицу $\left[x_1^2, x_2^2, \dots, x_k^2\right]$, которая содержит элементы второго слоя. Аналогично можно найти значения элементов двух оставшихся слоев.

В общем виде операция нахождения следующего слоя $x^{r+1} = \left[x_1^{r+1}, x_2^{r+1}, \dots, x_l^{r+1}\right]$ по предыдущему $x^r = \left[x_1^r, x_2^r, \dots, x_q^r\right]$, с помощью матрицы весов $w^r = \begin{bmatrix} w_{11}^r & \cdots & w_{1l}^r \\ \vdots & \ddots & \vdots \\ w_{q1}^r & \cdots & w_{ql}^r \end{bmatrix}$ и матрицы сдвигов $b^r = [b_1^r, b_2^r, \dots, b_l^r]$, может быть представлена следующим образом:

$$x^{r+1} = F(x^r * w^r + b^r)\text{ , где } x^r * w^r + b^r = t^r.$$

Получив матрицу $[x_1^4, x_2^4, \dots, x_l^4]$ нашего примера, показывающую прогноз сети, начинается вычисление ошибки данного предсказания. Для этого используется матрица] в задачах на классификацию одного объекта, состоящая из всех единиц и одного нуля, характеризующая верный ответ, а также функция для расчета ошибки $E(x)$, например:

$$E(x) = (y - \mathcal{y})^2$$
(3)

$$E(x) = |y - \mathcal{y}|$$
(4)

Теперь, зная значения функции ошибки, можно вычислить как необходимо изменить веса и сдвиги сети, находя значения градиента ошибки по смещению, и градиента ошибки по сдвигу, и обновляя их значения:

$$w_{new} = w_{old} - \frac{\alpha * \partial E}{\partial w}$$
(5)

$$b_{new} = b_{old} - \frac{\alpha * \partial E}{\partial b}$$
(6)

где коэффициент α, отвечающий за скорость обучения.

Для нахождения градиентов ошибки по весу и смещению используется правила дифференцирование сложной функции. Зная градиент ошибки по каждому элементу слоя, можно посчитать градиент ошибки по весу и сдвигу для элементов матриц, из которых мы получили элементы данного слоя, а также необходимо найти градиент ошибки каждого элемента предыдущего слоя, чтобы после этого повторить операцию. Например, в нашем случае необходимо искать градиент ошибки второго и третьего слоя, зная градиент четвертого. Теперь покажем в общем виде алгоритм нахождения градиентов. Пусть известна матрица градиентов ошибки по элементам слоя

$$x^{r+1} = \left[x_1^{r+1}, x_2^{r+1}, \dots, x_l^{r+1}\right]$$
(7)

а именно:

$$\frac{\partial E}{\partial x^{r+1}} = \left[\frac{\partial E}{\partial x_1^{r+1}}, \frac{\partial E}{\partial x_2^{r+1}}, \dots, \frac{\delta E}{\delta x_l^{r+1}}\right]$$
(8)

зная градиенты можно посчитать матрицу градиентов по ошибке так как:

$$\frac{\partial E}{\partial w_{ij}^r} = \frac{\frac{\partial E}{\partial x_j^{r+1}} * \partial x_i^{r+1}}{\partial t_j^{r+1}} * \partial t_i^{r+1} = \frac{\partial E}{\partial x_j^{r+1}} * F'\left(t_j^{r+1}\right) * x_i^r$$
(9)

в общем виде для матриц:

$$\frac{\partial E}{\partial x^{r+1}} = \left[\frac{\partial E}{\partial x_1^{r+1}}, \frac{\partial E}{\partial x_2^{r+1}}, \dots, \frac{\delta E}{\delta x_l^{r+1}}\right]$$
(10)

$$F'(t^{r+1}) = \left[F'(t_1^{r+1}), F'(t_2^{r+1}), \dots, F'\left(t_l^{r+1}\right)\right] \quad (11)$$

Получаем

$$\frac{\partial E}{\partial w_{ij}^r} = \frac{\partial E}{\partial x^{r+1}} \odot F'(t^{r+1}) * (x^r)^T \quad (12)$$

Таким же образом находится градиент ошибки по сдвигу:

$$\frac{\partial E}{\partial b_{ij}^r} = \frac{\partial E}{\partial x^{r+1}} \odot F'(t^{r+1}) \quad (13)$$

Для нахождения градиента ошибки по элементам предыдущего слоя $\frac{\partial E}{\partial x^r} = \left[\frac{\partial E}{\partial x_1^r}, \frac{\partial E}{\partial x_2^r}, \dots, \frac{\delta E}{\delta x_l^r}\right]$:

$$\frac{\partial E}{\partial x_i^r} = \frac{\frac{\partial E}{\partial t_1^{r+1}} * \partial E}{\partial x_1^r} + \frac{\frac{\partial E}{\partial t_2^{r+1}} * \partial E}{\partial x_2^r}, \dots, \frac{\frac{\delta E}{\delta t_l^{r+1}} * \partial E}{\partial x_l^r} \quad (14)$$

в общем виде:

$$\frac{\partial E}{\partial x^r} = \frac{\partial E}{\partial x^{r+1}} \odot F'(t^{r+1}) * (w^r)^T \quad (15)$$

Метод градиентного спуска, представленный выше, называется поэлементным градиентным спуском. Поэлементный градиентный спуск – это градиентный спуск, где модель обновляет веса отдельно для каждого выхода на одном примере: сначала корректирует параметры по первому выходу, затем по второму и так далее. Такой подход обеспечивает точечную подстройку под каждую компоненту ошибки.

Для корректной проверки результатов были сгенерированы 16 матриц весов и 16 матриц сдвигов соответственно, в каждом обучении использовалось 5000 изображений, переведённых в матрицу 1×784 и 100 изображений для проверки корректности результата обучения персептрона, обучение проходило в течении 2 эпох, если не указаны другие данные. Сеть состояла из двух скрытых слоев, причем первый слой был размера 1×128, второй 1×64, коэффициент обучения стандартно был принят за $\alpha = 0.05$, однако в процессе исследования менялся в некоторых случаях. Причиной стала неработоспособность некоторых функций активации и функций ошибки при больших значениях шага. По той же причине в процессе исследования менялся диапазон входных значений для функции посредствам умножения на константу сгенерированных ранее значений весов.

Рассмотрим подробнее выбор альфа-коэффициента. Даже для современных нейронных сетей, часто производится наугад. Наиболее эффективным методом является наблюдение за тем, как изменяется ошибка с каждой итерацией, и если наблюдается расхождение, значит, альфа-коэффициент слишком велик и его следует уменьшить. Если обучение происходит слишком медленно, значит, альфа-коэффициент слишком мал и его следует увеличить[2]. В связи с этим как начальный коэффициент для обучения персептрона был выбран коэффициент 0.05.

При проверке результат обучения персептрона на выборке из 100 изображений происходит приведения результат работы персептрона к виду, что сумма элементов выходного слоя равна единице и каждый элемент находится в пределах от 0 до 1 посредством использования функции softmax

$$\sigma(x)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad (16)$$

где $z = [z_1, \dots, z_K]$ – выходной слой, причем был использован метод оценки Margin-based evaluation, который для матрицы предсказаний $[x_1^l, x_2^l, \dots, x_n^l]$ и ответа по x_kбудет выглядеть следующим образом:

$$Margin(x^l) = [x_r = max(x_1^l, x_2^l, \dots, x_n^l), \text{где} r \neq k]x_k - x_r$$
(17)

Исследование влияние метода импульса на качество обучения персептрона. Стандартный метод градиентного спуска итеративно обновляет параметры следующим образом:

$$\theta_{new} = \theta_{old} - \alpha * \nabla E$$
(18)

где в качестве θ может выступать веса или сдвиги. Таким образом в ходе обучения общий вид изменения θ выглядит следующим образом на i-ой итерации:

$$\theta_{i+1} = \theta_i - \alpha * \nabla E_i - \alpha * \nabla E_{i-1} - \dots - \alpha * \nabla E_0$$
(19)

Импульсный вектор в свою очередь подразумевает следующий вид изменения параметра веса или сдвига:

$$\theta_{i+1} = \theta_i - \alpha * \nabla E_i - \alpha * \beta * \nabla E_{i-1} - \dots - \alpha * \beta^i * \nabla E_0) = \theta_i - \alpha * \sum_{j=0}^{i} \beta^i * \nabla E_{i-j}$$
(20)

По мере выполнения итераций для уменьшения влияния градиента и предотвращения расхождения суммы градиентов добавляется коэффициент $\beta \epsilon (0,1)$. Согласно определению, метод импульса – это обобщённая форма градиентного спуска. Фактически, градиентный спуск можно рассматривать как частный случай метода импульса, где $\beta = 0$. Чем ближе β к 1, тем больше он отражает предыдущие градиенты, и наоборот, чем ближе он к 0.

Метод градиентного спуска обновляет параметры в направлении наискорейшего текущего градиента и, таким образом, является алгоритм жадного типа. Метод импульса смягчает эту “жадность”, позволяя делать выбор, который может быть не лучшим в краткосрочной перспективе, но более выгодным в долгосрочной. Он также помогает предотвратить резкие и чрезмерные изменения направления градиента.

Очевидно, что, поскольку величина градиента, используемого для обновления параметров, больше по сравнению со стандартным методом градиентного спуска, метод с импульсом обладает преимуществом более быстрой сходимости. Также известно, что он более эффективно позволяет избежать локальных минимумов, подобно тому, как мяч, катящийся вниз по склону, может преодолеть небольшие неровности, если у него достаточно скорости[1].

Также была обучена сеть на одинаковых начальных значениях матрицы весов и сдвигов с коэффициентом $\beta = 0$ (классический градиентный спуск) и $\beta = 0.9$, результаты с использованием метод оценки Margin-based evaluation представлен на графике(рисунок 1), в котором столбцы красного цвета отражают значение функции оценки для $\beta = 0$, с синие - $\beta = 0.9$, для определенного набора весов. Что демонстрирует эффективность метода импульсов (рисунок 1). Можно заметить, что метод импульса дает немного худший результат в среднем по наборам. Это связано с тем, что сети обучались на малом наборе данных. На ранних эпохах метод импульса дает худшие результаты, по той причине, что его природа имеет накопительный эффект, соответственно импульс на малом наборе данных является «случайным» и вносит слабый и скорее отрицательный вклад в процесс обучения сети.

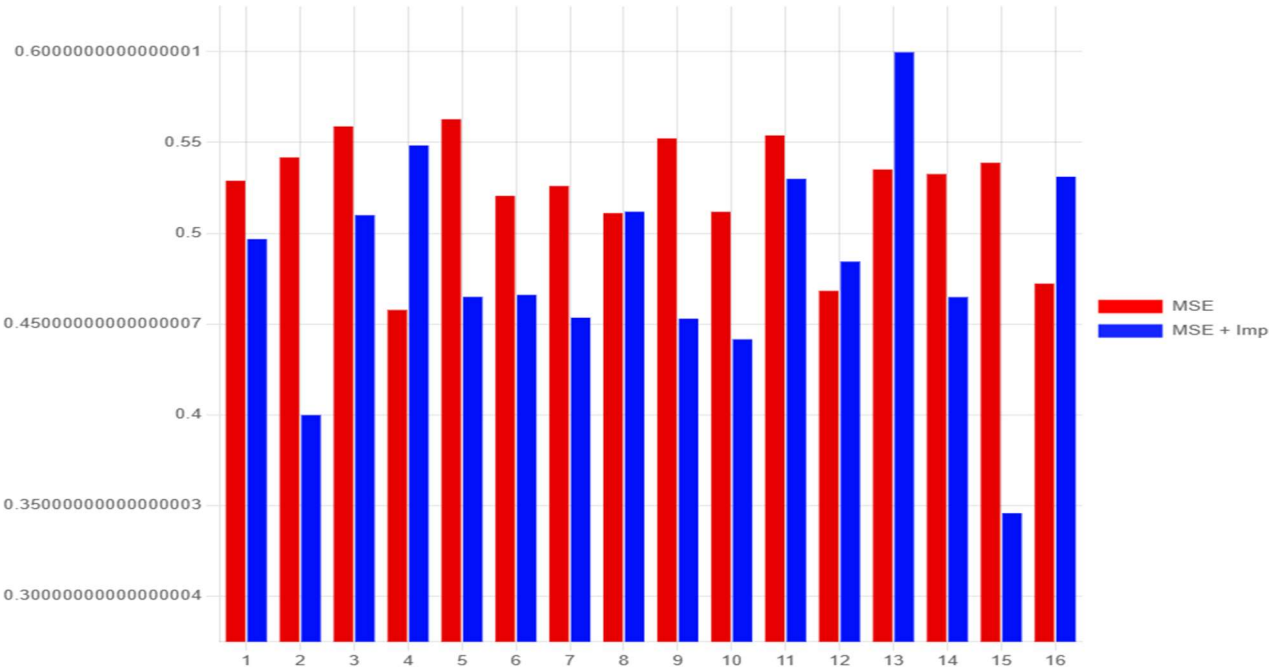

Рисунок 1 – График сравнения обучения с использованием метода импульса и без него

Также были исследованы влияние следующих функций ошибок на качество обучения: среднеквадратичные функции ошибки $(MeanSquaredError)$и$(RootMeanSquaredError)$:

$$E(x,y) = \frac{1}{2} * \sum_{i=1}^{n} (x_i - y_i)^2 \quad (21)$$

$$E(x,y) = \sqrt{\frac{1}{n} * \sum_{i=1}^{n} (x_i - y_i)^2} \quad (22)$$

категориальная кросс-энтропия (Categorical Cross-Entropy):

$$H(x,y) = -\sum_{i=1}^{10} x_i * log(y_i) \quad (23)$$

где x – вектор выхода, а y – вектор, состоящий из всех единиц и одного нуля, характеризующая верный ответ.

Также были исследованы часто используемые функции активации

$$f(x) = \frac{1}{1 + e^{-x}} \quad (24)$$

$$f(x) = max(0, x) \quad (25)$$

Затем были исследованы их попарные комбинации ReLU и MSE, Сигмоида и Categorical Cross-Entropy и т. д.

Сперва были изучены комбинации функции сигмоида и трех функций ошибок, причем все начальные веса были увеличены в 10 для достижения большой нелинейности (рисунок 2). Лучший результат был получен с использованием функции MSE, а наихудший с использованием функции RMSE (рисунок 2).

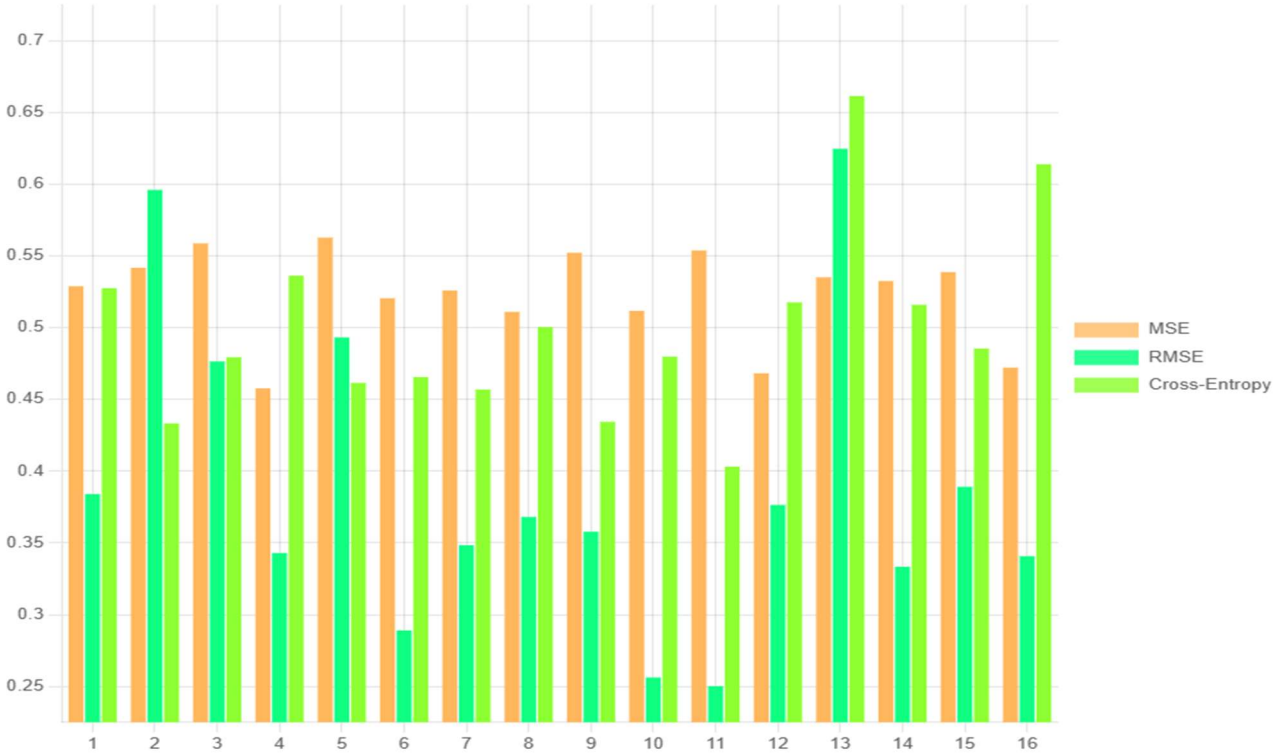

Рисунок 2 – График сравнения функции сигмоида и функций ошибки

Затем был представлен результат использования комбинации ReLU и функций ошибки, причем для того, чтобы получить в конечном итоге после обучения вектор предсказания у которого каждый элемент будет в диапазоне от 0 до 1, высчитывалась сумма элементов вектора и делилась на каждый элемент вектора. Лучший результат был получен с использованием функции MSE а наихудший с использованием функции RMSE (рисунок 3).

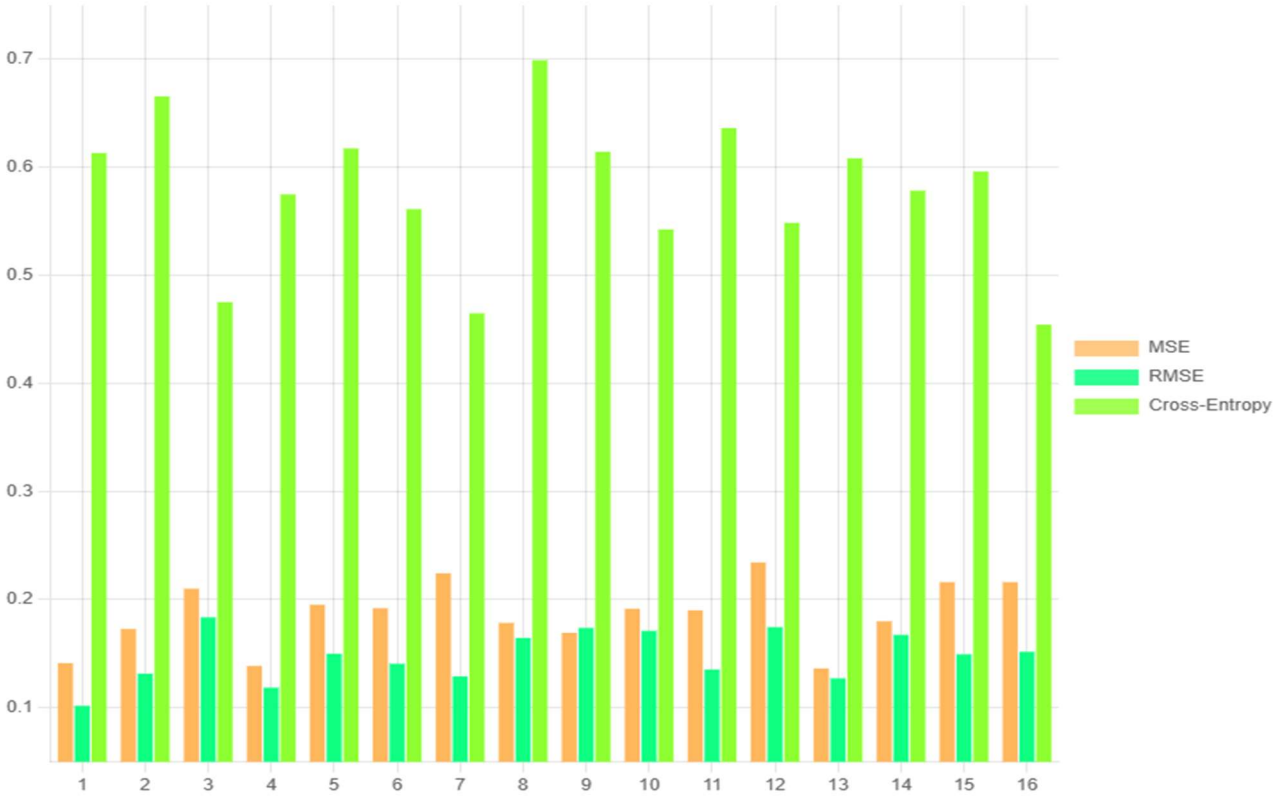

Рисунок 3 – График сравнения функции ReLU и функций ошибки

После чего нами были выбраны следующие функции активации для исследования их эффективности в обучении:

$$f_1(x) = \frac{arctg(x) + \frac{\pi}{2}}{\pi} \quad (26)$$

на всей области определения и

$$f_2(x) = \sqrt{\frac{x}{e^{arctg(0.1)*x}} + 1} \quad (27)$$

при $x \in$ и

$$f_3(x) = \frac{-1}{x - 1} \quad (28)$$

при $x \in (-\infty; 0)$. (рисунки 4, 5).Причем так как коэффициент $\alpha = 0.05$ показал себя неэффективным так как функция “перепрыгивает” локальный минимум, по этой причине был выбран коэффициент $\alpha = 0.005$.

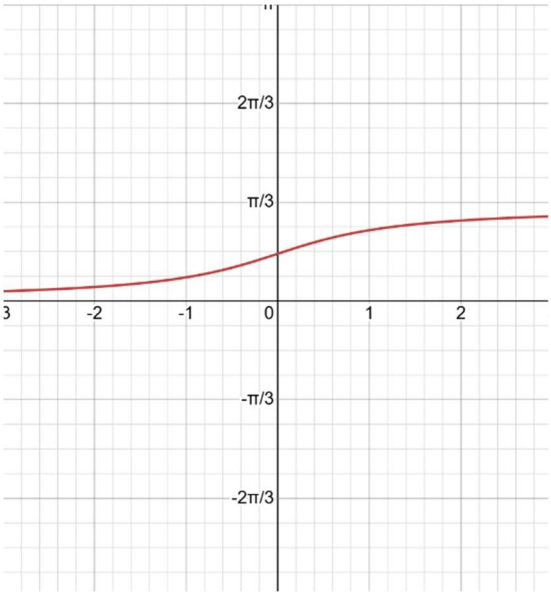

Рисунок 4 – График функции $f_1(x)$

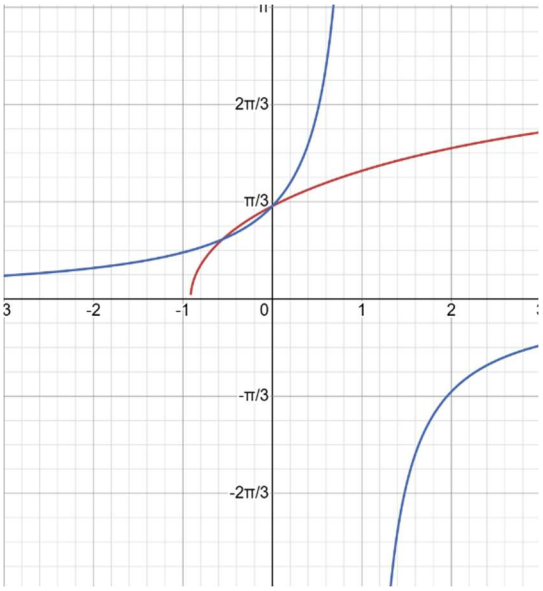

Рисунок 5 – График функции $f_2(x)$и$f_3(x)$

Результат обучения персептрона с использованием комбинации функции активации

$$f_1(x) = \frac{arctg(x) + \frac{\pi}{2}}{\pi} \quad (29)$$

и функций ошибок MSE, RMSE, Categorical Cross-Entropy, причем начальные веса вновь были увеличены в 10 раз (рисунок 6)

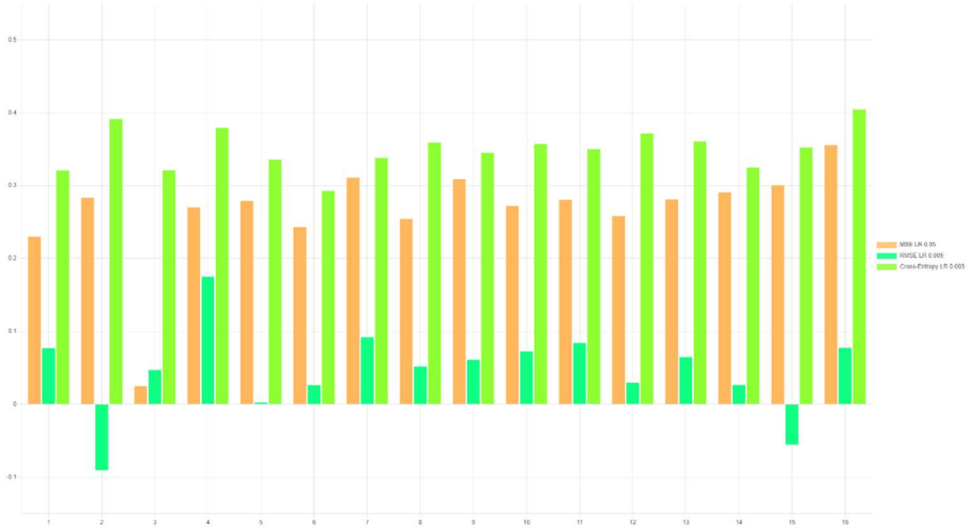

Рисунок 6 – График сравнения функции $f_1(x)$ и функций ошибки

Лучший результат был получен с использованием функции Categorical Cross-Entropy, а наихудший с использованием функции RMSE.

Затем был получен результат обучения персептрона с использованием функций активации

$$f_2(x) = \sqrt{\frac{x}{e^{arctg(0.1)*x}} + 1} \quad (30)$$

при $x \in$ и

$$f_3(x) = \frac{-1}{x-1} \quad (31)$$

при $x \in (-\infty; 0)$(рисунок 7).

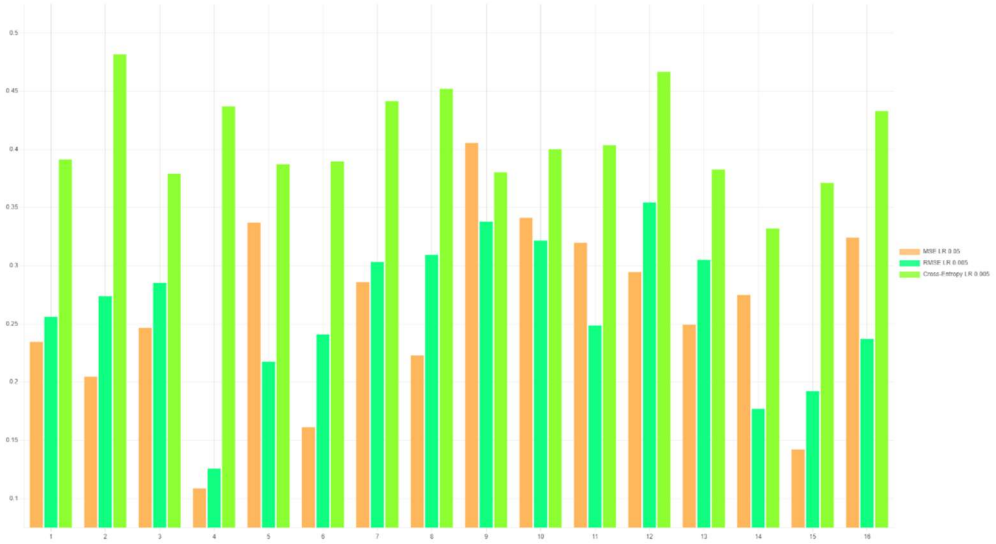

Рисунок 7 – График сравнения функции $f_{23}(x)$ и функций ошибки

Лучший результат был получен с использованием функции Categorical Cross-Entropy, а наихудший с использованием функции MSE. Затем была составлена диаграмма лучших комбинаций функций активации и функций ошибок, а именно Sigmoida и RMSE; ReLU и MSE; $f_1(x)$, $f_{23}(x)$ и Categorical Cross-Entropy. Лучший результат показала функция ReLU и MSE (рисунок 8).

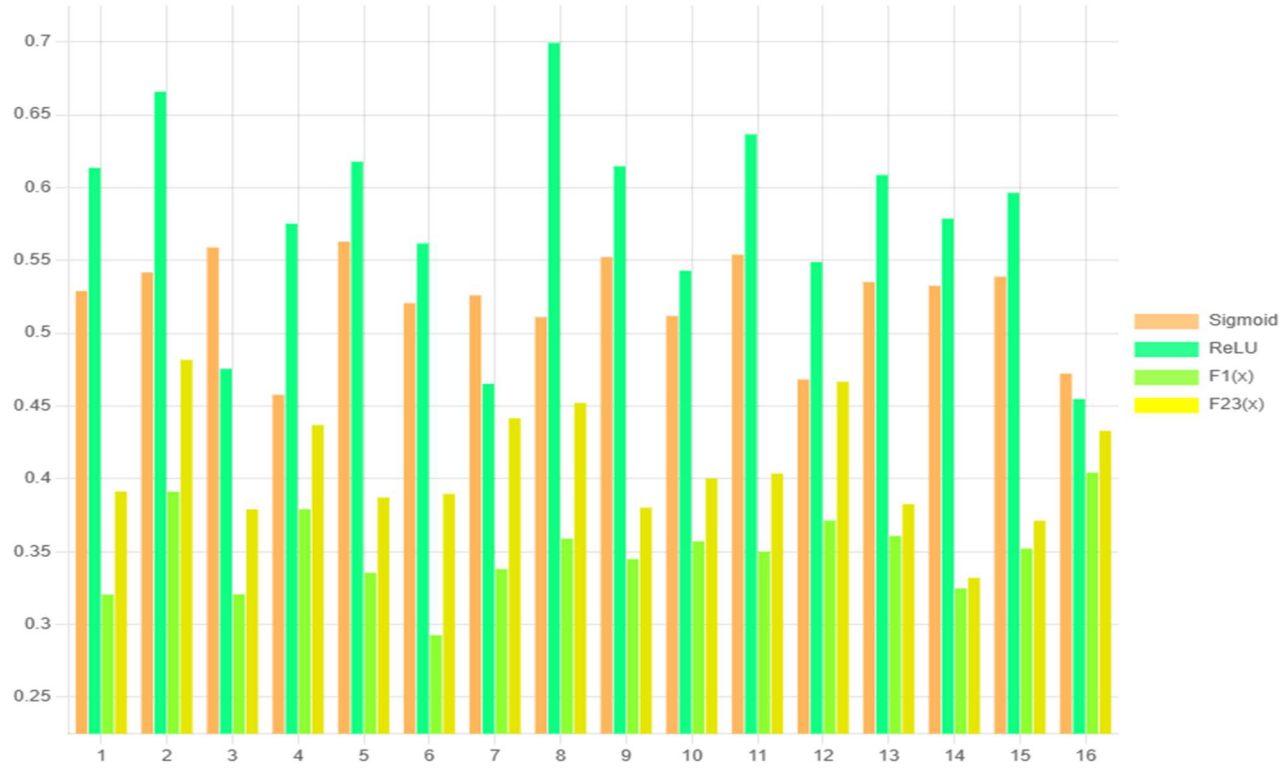

Рисунок 8 – Лучшие комбинации функций активации и функций ошибки

Далее был рассмотрен метод импульса в отношении всех приведенных выше функций. Обучение показало худшею сходимость при использовании метода импульса. Однако некоторые функции активации оказались несовместимы с методом импульса вовсе. На это повлиял следующий фактор: функции активации для применения с импульсом не должны иметь разрывов в производной и должны быть ограничены в своих значениях, например, как сигмоида. Это обусловлено тем что в случае с разрывной функцией резкое изменение градиента функции вызывает резкое изменение градиента, а импульс, из-за своего накопительного эффекта усилит этот скачек, что может привести к взрыву градиентов. В случае же с неограниченной функцией импульс может вызвать ситуацию, при которой градиент растет экспоненциально, в случае если функция сделала несколько шагов в сторону увеличения, для неограниченных функций это приведет к взрыву градиентов.

Заключение. В результате исследования влияния гиперпараметров персептрона был изучено положительное влияние метода импульсов на качество обучение, также были рассмотрены все возможные комбинации приведенных функций активации и ошибки с целью найти наилучшую комбинацию, а именно ReLU и MSE, исследовано несколько функций активации, выведенных эмпирически, а также было исследовано влияние на обучение коэффициент обучения α, для исследованной архитектуры персептрона наилучший результат показала комбинация функции активации ReLU, функции ошибки Cross-Entropy.

Список использованных источников:

1. Хабр [Электронный ресурс]: Методы оптимизации в машинном и глубоком обучении. От простого к сложному. – Режим доступа: https://habr.com/ru/articles/813221/. – Дата доступа: 12.04.2025.

2. Skypro [Электронный ресурс] : Обучение нейронных сетей с обратным распространением ошибки. – Режим доступа: https://sky.pro/wiki/python/obuchenie-nejronnyh-setej-s-obratnym-rasprostraneniem-oshibki/. – Дата доступа: 05.04.2025

3. Грокаем глубокое обучение [Электронный ресурс] – Режим доступа: https://publ.lib.ru/ARCHIVES/B/"Biblioteka_programmista"_(seriya)/%D2%F0%E0%F1%EA%20%DD._%20%C3%F0%EE%EA%E0%E5%EC%20%E3%EB%F3%E1%EE%EA%EE%E5%20%EE%E1%F3%F7%E5%ED%E8%E5.(2020).pdf. – Дата доступа: 15.04.2025

UDC 004.852

INVESTIGATION OF THE EFFECT OF HYPERPARAMETERS ON THE LEARNING QUALITY OF A MULTILAYER PERCEPTRON USING ELEMENT-WISE GRADIENT DESCENT

Mikhalko A.N., Filon D.A.

Belarusian State University of Informatics and Radioelectronics, Minsk, Republic of Belarus

Rykova O. V. – PhD in Physics and Mathematics, Associate Professor

Annotation. In this paper, we analyze various activation functions and error functions in relation to the effectiveness of training a neural network based on a multilayer perceptron with a small number of epochs, all other things being equal, hyperparameters.

Keywords. Perceptron, matrices, activation functions, error functions.