ISSN 2224-087X. Електроніка та інформаційні технології. 2023. Випуск 21. С. 64–80
Electronics and information technologies. 2023. Issue 21. P. 64–80

УДК 004.8 DOI: https://doi.org/10.30970/eli.21.7

АЛГОРИТМ ОПТИМІЗАЦІЇ AMSGrad І ХАОС В БАГАТОШАРОВИХ НЕЙРОННИХ МЕРЕЖАХ ІЗ СТОХАСТИЧНИМ ГРАДІЄНТНИМ СПУСКОМ

С. Свелеба[1], І. Катеринчук[1], І. Куньо[1], О. Семотюк[2], Я. Шмигельський[1], С. Вельгош[1], А. Копач[1], В. Куньо[3]

[1] *Львівський національний університет імені Івана Франка,*
вул. Ген. Тарнавського, 107, 79017 Львів, Україна
incomlviv@gmail.com

[2] *Українська Академія Друкарства*
вул. Під Голоском,19, 79020 Львів, Україна

[3] *Національний університет «Львівська політехніка»*
вул. Степана Бандери, 12, 79013 Львів, Україна

В роботі з допомогою логістичної функції, яка описує процес подвоєння, та Фур'є спектру функції похибки було проведено тестування стохастичного методу оптимізації AMSGrad.

Реалізація алгоритму оптимізації градієнтного спуску за допомогою AMSGrad було здійснено для багатошарової нейронної мережі з прихованими шарами. В середовищі Python була написана програма розпізнавання друкованих цифр. Масив кожної цифри складався з набору «0» і «1» розміром 4x7. Вибірка кожної цифри містила набір з 5 можливих спотворень цифри і набір з 3 масивів які не відповідали жодній із цифр. Нейронна мережа містила 3 прихованих шари з 28 нейронами в кожному шарі.

Встановлено що гіперпараметр *beta1*, що описує вклад лінійного градієнта функції похибки, пов'язаний із подвоєнням кількості локальних і глобальних мінімумів функції похибки в процесі перенавчанні нейронної мережі. Гіперпараметр *beta2*, що описує вклад квадрата градієнта функції похибки, пов'язаний із утворенням блочної структури, яка блокує процес подвоєння кількості локальних мінімумів.

Ключові слова: оптимізаційні методи, функція похибки, AMSGrad, швидкість навчання, діаграма розгалуження.

Градієнтний спуск є одним із найпопулярніших алгоритмів для оптимізації та найпоширенішим способом оптимізації нейронних мереж. Водночас кожна найсучасніша бібліотека Deep Learning містить реалізації різноманітних алгоритмів для оптимізації градієнтного. Однак ці алгоритми часто використовуються як оптимізатори чорної скриньки, оскільки важко знайти практичне пояснення їхніх сильних і слабких сторін.

Обмеження градієнтного спуску полягає в тому, що один розмір кроку (швидкість навчання) використовується для всіх вхідних змінних. Розширення градієнтного спуску, як-от алгоритм Adam, використовує окремий розмір кроку для кожної вхідної змінної, але в результаті розмір кроку може швидко зменшуватися до дуже малих значень [1].

ISSN 2224-087X. Електроніка та інформаційні технології. 2023. Випуск 21

AMSGrad є розширенням до версії Adam градієнтного спуску, в якому намагаються покращити властивості конвергенції алгоритму, уникаючи великих різких змін у швидкості навчання для кожної вхідної змінної. Відмінність AMSGrad від Adam полягає в тому, що він підтримує максимум усіх векторів другого моменту $v(t)$ до поточного кроку часу та використовує це максимальне значення для нормалізації поточного середнього градієнта замість *v(t)* в Adam [1]

На початку обчислюються градієнти (часткові похідні) для поточного кроку за часом:

$g(t) = f'(x(t-1))$, де $f(x(t))$ - цільова функція.

Далі вектор першого моменту $m(t)$ оновлюється за допомогою градієнта та гіперпараметра *beta*1:

$m(t) = beta1(t) \cdot m(t-1) + (1 - beta1(t)) \cdot g(t)$.

Гіперпараметр *beta1* можна підтримувати постійним або експоненціально зменшувати під час пошуку, наприклад: $beta1(t) = (beta1)^t$.

Або, по черзі: *beta1(t) = beta1/t*

Другий вектор моменту оновлюється за допомогою квадрата градієнта та гіперпараметра *beta2*: $v(t) = beta2 \cdot v(t\text{-}1) + (1\text{-}beta2) \cdot (g(t))^2$

Далі оновлюється максимум для вектора другого моменту:

*vhat(t) = max(vhat(t-*1*),v(t))*

Значення параметра *w* можна оновити за допомогою обчислених умов і гіперпараметра *alpha* - розміру кроку: *w(t) = w(t-*1*–alpha(t)·m(t) / sqrt(vhat(t)))*

Розмір кроку також можна підтримувати постійним або зменшувати за експонентою.

Отже для огляду є три гіперпараметри для алгоритму [2, 3], це

alpha: початковий розмір кроку (швидкість навчання), типове значення становить 0,002.

beta1: коефіцієнт розпаду для першого імпульсу, типове значення становить 0,9.

beta2: коефіцієнт розпаду для нескінченної норми, типове значення становить 0,999.

Попередньо, стохастичний метода оптимізації AMSGrad, який був описаний в роботі [3] і практично продемонстровано його роботу в [2], був протестований з допомогою логістичної функції, яка описує процес подвоєння, та Фур'є спектру функції похибки. Було встановлено що процес перенавчання супроводжується зміною швидкості цільової функції похибки, а Фур'є спектру притаманна поява гармонік. Показано, що при незначному наборі вхідних даних коли, значення *beta2*, близьке до 1, а *beta1* = 0.9 спостерігається нестабільність в навчанні, яка зумовлена процесом перенавчання.

Отримані результати дещо є відмінними ніж отриманих праці [3]. Така відмінність може бути пов'язана з малою вибіркою вхідних даних для навчання. Дійсно оптимізаційні методи, до яких відноситься метод AMSGrad, застосовують в згорткових нейромережах для швидкої обробки вхідних даних.

З метою підтвердження даного припущення розглянемо багатошарову нейронну мережу з прихованими шарами.

Для даної системи було проведено дослідження діаграми розгалуження з використанням функції відображення виду:

$y_{n+1} = \eta - y_n - y_n^2$,

де n – індекс ітерацій, η = *alpha* – параметр, який визначає швидкість навчання.

Її нерухомі точки:

$$y^*_{1,2} = -1 \pm \sqrt{\eta + 1},$$

власні значення, яких можна обчислити наступним чином:

$$\rho^*_{1,2} = 1 \mp 2\sqrt{\eta + 1}.$$

Вибір даного логістичного відображення обумовлено тим, що воно описує процес подвоєння частоти коливань [4]. В нашому випадку цей процес зумовлений виникненням локальних мінімумів при підході до глобального мінімуму. Також було проведено дослідження Фур'є спектру функції похибки в залежності від параметрів *beta1, beta2* та *alpha*.

Для цього в програмному середовищі Python була написана програма розпізнавання друкованих цифр. Масив кожної цифри складався з набору «0» і «1» розміром 4x7. Вибірка кожної цифри містила набір з 5 можливих спотворень цифри і набір з 3 масивів які не відповідали жодній із цифр. Наприклад для цифри «0» масив значень *x*:

```
Numt1=[0,0,0,0,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
Numt2=[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1]
Numt3=[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]
Num01=[1,1,1,1,1,0,0,1,1,0,0,1,1,0,0,1,1,0,0,1,1,0,0,1,1,1,1,1]
Num02=[1,1,0,1,1,0,0,1,1,0,0,1,1,0,0,1,1,0,0,1,1,0,0,1,1,1,1,1]
Num03=[1,1,1,1,1,0,0,1,0,0,0,1 ,1,0,0,1,1,0,0,1,1,0,0,1,1,1,1,1]
Num04=[1,1,1,1,1,0,0,1,1,0,0,1,1,0,0,1,1,0,0,1,0,0,0,1,1,1,1,1]
Num05=[1,1,1,1,1,0,0,1,1,0,1,1,1,0,0,1,1,0,0,1,1,0,0,1,1,1,1,1]
```

Масив значень *y*: `Num0Y=[[0],[0],[0],[1],[1],[1],[1],[1]]`

Описана з допомогою даної програми нейронна мережа містила 3 прихованих шари з 28 нейронами в кожному шарі:

```
import numpy as np
import array as arr
from matplotlib import pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
# оголошення за задання необхідних змінних
hiddenSize =28 # кількість нейронів на прихованому шарі
alpha = 0.01 # швидкість навчання (коефіцієнт альфа)
eps = 0.000001 # бажана точність навчання
num =3 # кількість шарів
beta1=0.9
beta2=0.999
```

Вибір кількості прихованих шарів і нейронів у кожному із них визначався найменшою похибкою навчання та розпізнавання цифр. Згідно роботи [5], це трьохшарова нейронна мережа з 28 нейронами в кожному шарі. Значення параметрів *beta1* і *beta2* вибиралися, як в праці [3]. Функцією активації, згідно праці [6] вибиралась сігмоїдальна:

```
def sigmoid(x): # функція активації
    c=1
    return 1 / (1 + np.exp(-x*c))
def sigmoid_output_to_derivative(output): # метод обчислення похідної від функції активації
    c=1
    return output*c*(1 - output)
def gen_synapse(x, y, hiddenSize, num): # генерація початкових ваг
    synapse = []
```

```
np.random.seed(1)
for i in range(num):
    if i == 0:
        synapse.append(2 * np.random.random((len(x[0]),hiddenSize)) - 1)
    elif i == num - 1:
        synapse.append(2 * np.random.random((hiddenSize,len(y[0]))) - 1)
    else:
        synapse.append(2 * np.random.random((hiddenSize,hiddenSize)) - 1)
return synapse
```

Навчання нейронної мережі описувалось функцією training():

```
for j in range(len(yy)):
    x=np.array(xx[j])
    y=np.array(yy[j])
    print('x=',x,'y=',y,len(y), len(x), type(y))
    def training(x, y, alpha, eps, hiddenSize, synapse, num, arr_age, arr_eps, alpha_x, chastota, xx, delta_n, xxx,
epsilon, kilkist,kilkist_alpha, beta1, beta2): # метод навчання нейронної мережі

        for alpha in np.arange(0.000001,0.1,0.00001):
            delta=[]
            age = 1 # age – кількість ітерацій
            while True:
                age += 1
                layers = []
                for i in range (num + 1):
                    if i == 0:
                        layers.append(x)
                    else:
                        layers.append(sigmoid(np.dot(layers[i - 1],synapse[i - 1]))) # функції активації
                layer_errors = []
                layer_deltas = []
                m=[]
                v=[]
                vhat=[]
                layer_errors.append(layers[num] - y) # різниця між виходом і очікуваним значенням
                e = np.mean(np.abs(layer_errors[0]))
                if (age % 1) == 0:
                    arr_age.append(age)
                    arr_eps.append(e)
                    alpha_x.append(alpha)
                    #print("Похибка на " + str(age) + " ітерації: " + str(e))'''
                if(age >10):
                    break
                if (e < eps):
                    #print("Точність " + str(round(e, 4)) + " досягнута за " + str(age) + " епох(и)")
                    break
```

Значення дельта визначалось як:

layer_deltas.append(layer_errors[0] * sigmoid_output_to_derivative(layers[num])) # – g_t=g_{age} – градієнт цільової функції.

Як і в роботі [2], перший і другий вектор моментів, а також максимальний вектор другого моменту для кожного параметра, який оптимізується в рамках пошуку, позначається як *m*, *v* і *vhat* відповідно. Вони ініціалізуються на 0.0 на початку пошуку:

```
m=[0.0]
v=[0.0]
```

```
vhat=[0.0]
m.append( beta1**(age+1) * m[0] + (1.0 - beta1**(age+1)) * layer_deltas[0])
v.append((beta2 * v[0]) + (1.0 - beta2) * layer_deltas[0]**2)
vhat=np.array(v[1])
d=m[1]/np.sqrt(vhat+ 1e-8)
d.shape=(8,1)
d=[d]* int(hiddenSize)
```

Далі визначаємо значення дельта в кожному прихованому шарі. Вектор першого моменту оновлюється за допомогою градієнта та гіперпараметра *beta1*, а другий вектор моменту оновлюється за допомогою квадрата градієнта та гіперпараметра *beta2*.

```
for i in range (num - 1):
        layer_errors.append(layer_deltas[i].dot(synapse[num - 1 - i].T))
        layer_deltas.append(layer_errors[i + 1] * sigmoid_output_to_derivative(layers[num - 1 - i]))
        layer_deltass=layer_errors[i + 1] * sigmoid_output_to_derivative(layers[num - 1 - i])
        # m(t) = beta1(t) * m(t-1) + (1 - beta1(t)) * g(t)
        m = beta1**(age+1) * m[i-1] + (1.0 - beta1**(age+1)) * layer_deltass
        # v(t) = beta2 * v(t-1) + (1 - beta2) * g(t)^2
        v = (beta2 * v[i-1]) + (1.0 - beta2) * layer_deltass**2
        # vhat(t) = max(vhat(t-1), v(t))
        vhat = max( v.reshape(-1,1))
        dd= m / np.sqrt(vhat+ 1e-8)
        d.append(dd)
```

Здійснюємо корекцію ваг та визначаємо похибку навчання:

```
for i in range (num):
        synapse[num - 1 - i] -= alpha * (layers[num - 1 - i].T.dot(d[i]))
        delta.append(np.mean(synapse[num - 1]))
```

На рис.1 наведено результат роботи даної програми. За умови що *beta1* = 0.9, кількість ітерацій рівна 10, на рис.1 наведено залежність значення логістичної функції похибки від параметра *alpha* і Фур'є спектрів при значенях гіперпараметра *beta2* = 0.9; 0.99; 0.999; 0.9999. Отримані діаграми розгалуження при різних значення гіперпараметра *beta2* засвідчують, що весь досліджуваний діапазон зміни *alpha* (0.000001 ÷ 0.009) можна розбити на 4 частини:

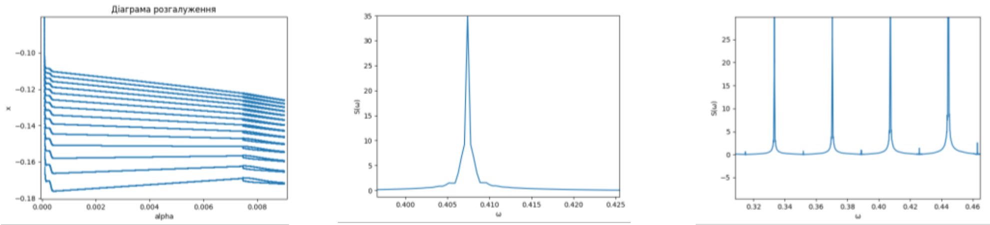
beta2 = 0.9 -- 1)діапазон різкого зменшення величини функції похибки (*alpha* = 0.000001 ÷ 0.0005) – не до навчання; 2) діапазон мало змінної монотонної поведінки функції похибки (*alpha* = 0.0005 ÷ 0.0075) – задовільний процес навчання; 3) діапазон роздвоєння на діаграмі розгалуження (виникнення гармоніки функції похибки, *alpha* = 0.0075 ÷ 0.015) – процес перенавчання; 4) діапазон хаотичної не монотонної поведінки функції похибки від *alpha* (*alpha* > 0.015) – поява хаосу;

beta2 = 0.99 -- 1) діапазон різкого зменшення величини функції похибки (*alpha* = 0.000001 ÷ 0.0002) – не до навчання; 2) діапазон мало змінної, монотонної поведінки функції похибки (*alpha* = 0.0002 ÷ 0.0025)- задовільний процес навчання; 3) діапазон роздвоєння на діаграмі розгалуження (виникнення гармоніки функції похибки, *alpha* = 0.0025 ÷ 0.0057) – процес перенавчання; 4) діапазон хаотичної не монотонної поведінки функції похибки від *alpha* (*alpha* = 0.0057 ÷ 0.009) – поява хаосу;

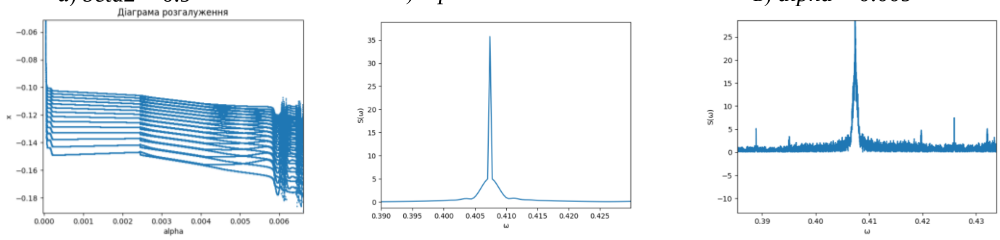
beta2 = 0.999 -- 1) діапазон різкого зменшення величини функції похибки (*alpha* = 0.000001 ÷ 0.0001) – не до навчання; 2) діапазон мало змінної, монотонної поведінки функції похибки (*alpha* = 0.0001 ÷ 0.0008) – задовільний процес навчання; 3) діапазон роздвоєння на діаграмі розгалуження (виникнення гармоніки функції похибки,

alpha = 0.0008 ÷ 0.0018) – процес перенавчання; 4) діапазон хаотичної не монотонної поведінки функції похибки від *alpha* (*alpha* = 0.0018 ÷ 0.009) – поява хаосу;

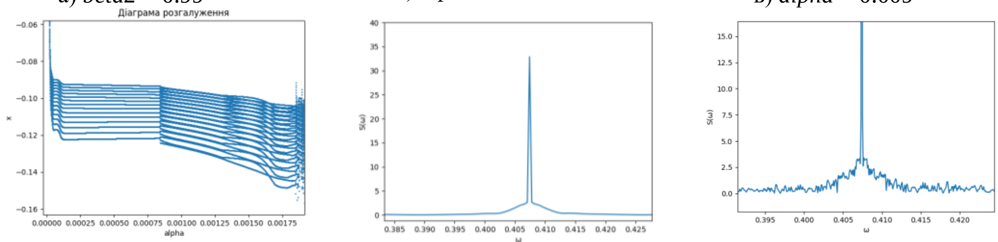
beta2 = 0.9999 -- 1) діапазон різкого зменшення величини функції похибки (*alpha* = 0.000001 ÷ 0.00006) – не до навчання; 2) діапазон мало змінної, монотонної поведінки функції похибки (*alpha* = 0.00006 ÷ 0.00029) – задовільний процес навчання; 3) діапазон роздвоєння на діаграмі розгалуження (виникнення гармоніки функції похибки, *alpha* = 0.00029 ÷ 0.00051) – процес перенавчання; 4) діапазон хаотичної не монотонної поведінки функції похибки від *alpha* (*alpha* = 0.00051 ÷ 0.003) – поява хаосу.



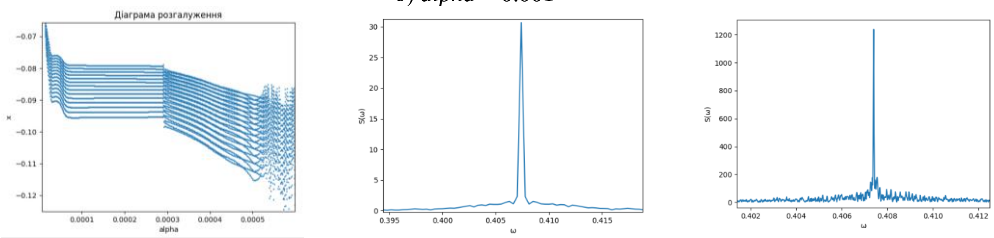
а) *beta2* = 0.9 б) *alpha* = 0.001 в) *alpha* = 0.009



а) *beta2* = 0.99 б) *alpha* = 0.001 в) *alpha* = 0.009



а) *beta2* = 0.999 б) *alpha* = 0.001 в) *alpha* = 0.005



а) *beta2* = 0.9999 б) *alpha* = 0.001 в) *alpha* = 0.003

Рис.1. Діаграма розгалуження (а) від швидкості навчання *alpha*, Фур'є спектри (б) - при задовільному навчанні та перенавчанні, та (в) - в умовах хаосу, за умови 10 ітерацій, *beta1* = 0.9, *beta2* = 0.9; 0.99; 0.999; 0.9999, для цифри «0».

Виходячи із даних результатів можна зазначити, що збільшення величини значень гіперпараметра *beta2* супроводжується зменшення діапазону, як задовільного навчання так і діапазону перенавчання, за рахунок збільшення діапазону хаотичної поведінки функції похибки від швидкості навчання. Слід також зазначити, що діапазон, як задовільного навчання, так і діапазону перенавчання зміщуються в область менших значень швидкості навчання. На це зокрема вказують Фур'є спектри (рис.1,б,в). Так при *alpha* = 0.001, за умови *beta2* = 0.9 і 0.99 простежується процес навчання з відсутністю перенавчання, тобто відсутність в спектрах появи гармоніки (рис.1,б). За умови коли *beta2* = 0.999 при *alpha* = 0.001 система попадає в діапазон перенавчання з появою гарманіки. Цей процес слабо проявляється на Фур'є спектрах при *beta2* = 0.999 і *alpha* = 0.001 (Рис.1, б). Однак він яскраво проявляється себе, за умови *beta2* = 0.9 і *alpha* = 0.009 (Рис.1, в), де даний процес є добре розвинутий.

При *alpha* = 0.001, за умови *beta2* = 0.9999 простежується перехід до хаотичного стану (рис.1), зі слабо вираженою хаотичністю. Більш розвинений хаотичний стан проявляється при *alpha* = 0.003, *beta2* = 0.9999; *alpha* = 0.005, за умови *beta2* = 0.999 (рис.1, в). Також слід зазначити співіснування, як стану перенавчання системи, так і її хаотичного стану (рис.1, в) при *alpha* = 0.009, *beta2* = 0.99.

На перший погляд таке співіснування вказує на те, що процес переходу до хаосу здійснюється через процес подвоєння кількості локальних мінімумів і збільшення кількості проходжень через глобальний мінімум при збільшенні швидкості навчання. Проведені дослідження діаграм розгалуження засвідчують наявність фрактальної структури в даних діаграмах. Детальнішому дослідженню цього питання буде присвячена окрема робота. На відміну від багатошарових нейронних мереж в яких не застосовувались оптимізаційні методи, в розглянутій багатошаровій нейронній мережі при застосуванні оптимізаційного методу AMSGrad зі збільшенням швидкості навчання простежуються каскади переходу до хаотичного стану так і виходу із нього. Їх кількість збільшується зі збільшенням швидкості навчання *alpha*.

Відомо [2, 3], що алгоритм AMSGrad оновлює експоненціальні ковзні середні градієнта (*m(t)*) і квадрат градієнта (*v(t)*), де гіперпараметри *beta1, beta2* [0, 1) контролюють експоненціальні швидкості спаду цих ковзних середніх. Ковзні середні є оцінками 1-го моменту (середнє) і 2-го моменту (нецентрована дисперсія) градієнта. За умови, коли *beta1* = 0 величина ковзного середнього градієнта *m(t)* = *layer_deltass*, тобто для даної багатошарової нейронної мережі буде рівний величині корекції ваг дельта. А корекція ваг буде описуватись виразом:

*synapse[num - 1 - i] -= alpha * (layers[num - 1 - i].T.dot(layer_deltas[i] / np.sqrt(vhat+ 1e-8)))*

Тобто відмінність від звичайної багатошарової нейронної мережі зі зворотнім методом поширенням похибки навчання багатошарова нейромережа з застосуванням оптимізаційного методу AMSGrad полягає в наявності множника *1/ np.sqrt((vhat)+ 1e-8)*. В звичайній багатошаровій нейромережі опис корекції ваг здійснюється за допомогою виразу [5]:

*synapse[num - 1 - i] -= alpha * (layers[num - 1 - i].T.dot(layer_deltas[i]))*

На рис.2 наведено залежність значення логістичної функції похибки від параметра alpha і Фур'є спектрів при значень гіперпараметра *beta1* = 0.0, кількісті ітерацій рівних 10 та при значень гіперпараметра *beta2* = 0.9; 0.99; 0.999; 0.9999. Як і при *beta1* = 0.9 так і при *beta1* = 0.0 на діаграмі розгалуження простежуються чотири інтервали поведінки функції похибки від швидкості навчання *alpha*: не до навчання, навчання з мінімальною похибкою, перенавчання, та хаотична поведінка функції похибки (рис.2, а). Це підтверджується відповідною поведінкою Фур'є спектрів функції похибки (рис.2, б, в). Відмінність діаграм розгалуження і Фур'є спектрів при *beta1* = 0.9 і *beta1* = 0.0 полягала у пригнічувані процесів подвоєння кількості локальних і глобальних мінімумів для *beta1* = 0.0. Отже, гіперпараметр *beta2* зменшує виродженість системи, тобто блокує процеси утворення локальних мінімумів, а отже перенавчання.

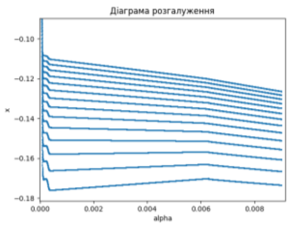
В праці [3] здійснено дослідження впливу гіперпараметрів *beta1* і *beta2* на процес навчання згорткової нейромережі, при зміні їх в широкому діапазоні вибору, тобто *beta1* [0, 0,9] і *beta2* [0,99, 0,999, 0,9999]. Значення *beta2*, близькі до 1, необхідні для стійкості до розріджених градієнтів, призводять до більшого зміщення ініціалізації; тому слід очікувати, що член корекції зміщення є важливим у випадках повільного затухання, запобігаючи несприятливому впливу на оптимізацію. Значення *beta2*, близькі до 1, дійсно призводять до нестабільності в навчанні, коли не було поправки на зміщення (*beta1* = 0). Особливо це проявляється в перших кілька ітераціях навчання [3]. Найкращі результати були досягнуті при малих значеннях (1 - *beta2*) і корекції зсуву (*beta1*). Це стає більш очевидним наприкінці оптимізації, коли градієнти стають проріодженими, оскільки приховані блоки спеціалізуються на певних шаблонах.

Отримані нами результати (рис.1 і рис.2) також засвідчують, що за умови відсутності гіперпараметра *beta1*, діапазон задовільного процесу навчання є дещо вужчий ніж при *beta1* > 0, за умови малих значеннях (1 - *beta2*). Як зазначалось в роботі [3], а також в наших дослідженях, коли значення *beta2*, близькі до 1, це призводять до нестабільності в навчанні (Рис.1, а і Рис.2, а).

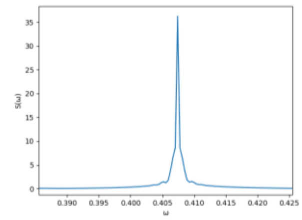
Член корекції зміщення є спадною степеневою функцією ($m_t = beta1^N * m_{t-1} + (1 - beta1^N) * g_t$, де *N*- кількість ітерацій), де в ролі значення степені виступає кількість ітерацій. Отже, самі ковзні середні є оцінками 1-го моменту [3] і залежать від кількості ітерацій.

На рис.3 наведено діаграму розгалуження при *beta1* = 0.9, *beta2* = 0.999 при 3, 5, 10 і 100 ітераціях (Рис.3, а – діаграма розгалуження в сьому діапазоні можливих змін *alpha*; рис.3, б – діапазон зміни *alpha*, що відповідає не довчання, задовільного навчання і перенавчання нейронної мережі). Процес перенавчання супроводжується переходом через глобальний мінімум та подвоєнням кількості локальних мінімумів. Цей процес особливо добре проявляється при малих значеннях ітерації. Хоча і тут починає прослідковуватись процес блокування подвоєння кількості локальних мінімумів, що унеможливлює перехід системи до хаотичного стану (рис.3, б).

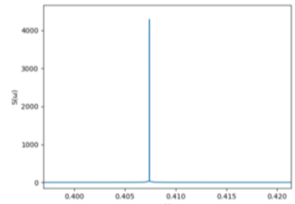
Зі збільшенням кількості ітерацій (*N*), зменшення градієнту функції похибки та збільшення величини виразу (1 - $beta1^N$), за умови малих значеннях (1 - *beta2*), зумовлює те, що градієнти стають проріодженими, оскільки співвідношення векторів першого і другого моментів утворюють блочну структуру, яка спеціалізуються на певних шаблонах. Це яскраво проявляється при малих значеннях ітерацій (*N* = 3 при *alpha* > 0.002; *N* = 5 при *alpha* > 0.0017; *N* = 10 при *alpha* > 0.0018 (Рис.3, б)). Отже процес оптимізації особливо проявляє себе при збільшенні кількості епох, приводячи до зменшення градієнту при наближенні до глобального мінімуму. Це проявляється в наближенні значення логістичної функції похибки до нуля при збільшенні кількості ітерацій (Рис.3).



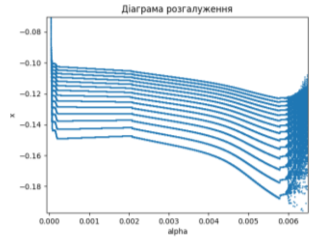
а) *beta2* = 0.9



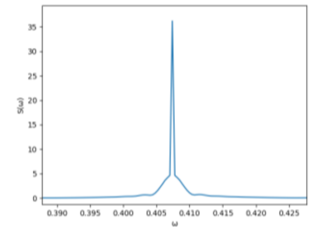
б) *alpha* = 0.001



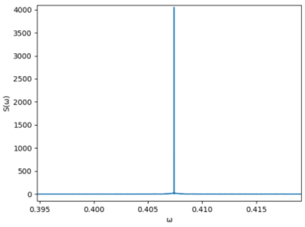
в) *alpha* = 0.009



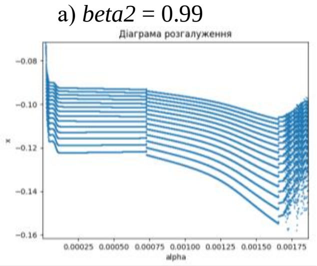
а) *beta2* = 0.99



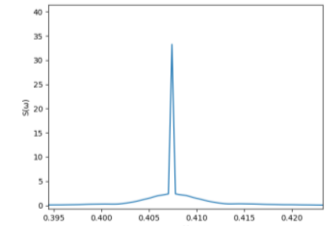
б) *alpha* = 0.001



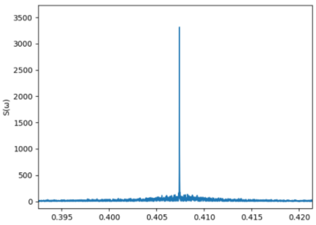
в) *alpha* = 0.009



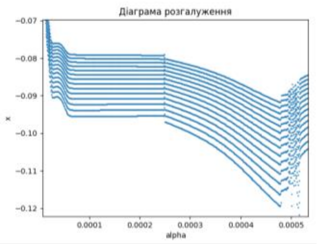
а) *beta2* = 0.999



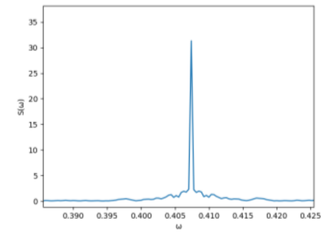
б) *alpha* = 0.001



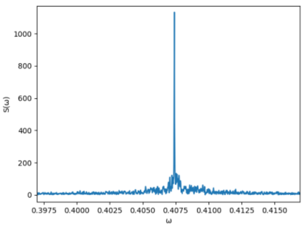
в) *alpha* = 0.008



а) *beta2* = 0.9999



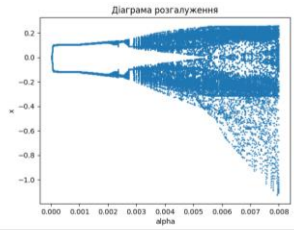
б) *alpha* = 0.001



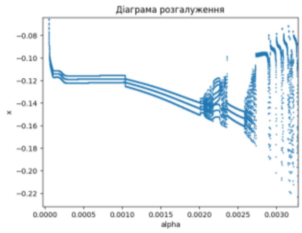
в) *alpha* = 0.0027

Рис.2. Діаграма розгалуження (а) від швидкості навчання *alpha*, Фур'є спектри (б) - при задовільному навчанні та перенавчанні, та (в) - в умовах хаосу, за умови 10 ітерацій, *beta1* = 0.0, *beta2* = 0.9; 0.99; 0.999; 0.9999, для цифри «0».

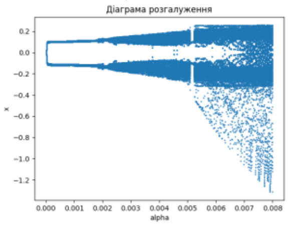
Розглянемо, як впливає на процес навчання значення величини гіперпараметра *beta2* при 100 ітераціях. На рис.4 наведено залежність значення логістичної функції похибки від параметра *alpha* і Фур'є спектрів при значенні гіперпараметра *beta1* = 0.9 кількість ітерацій рівна 100, та при значенні гіперпараметра *beta2* = 0.9; 0.99; 0.999; 0.9999.



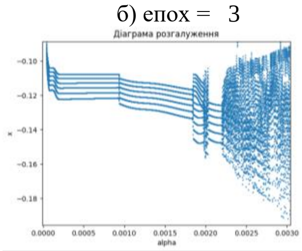
а) епох =3



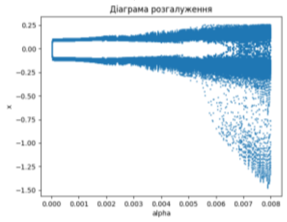
б) епох = 3



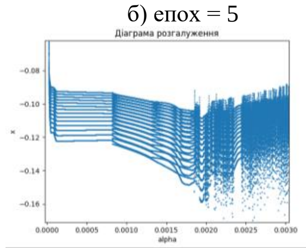
а) епох =5



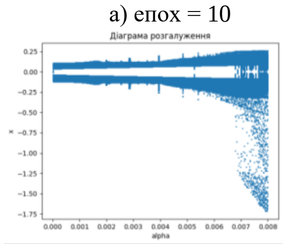
б) епох = 5



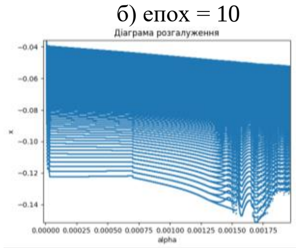
а) епох = 10



б) епох = 10



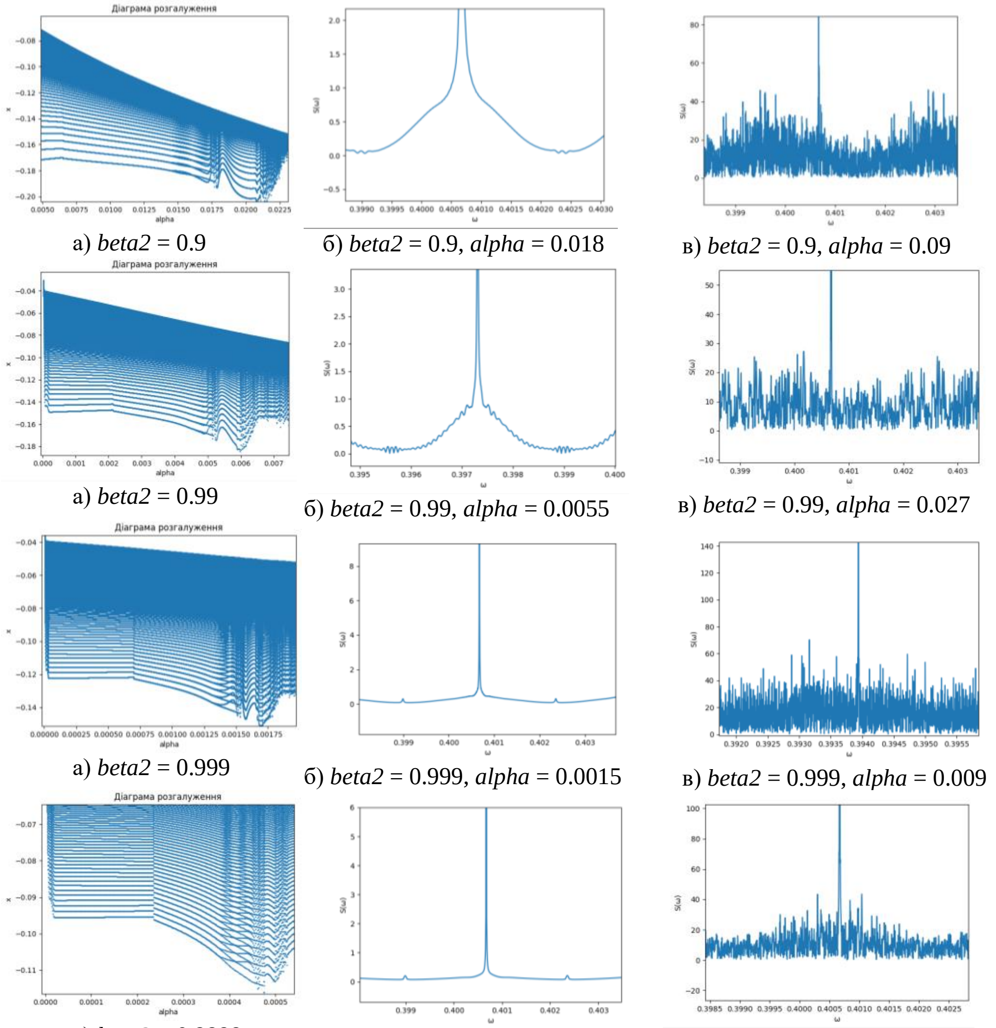
а) епох = 100



б) епох = 100

Рис.3. Діаграма розгалуження від швидкості навчання *alpha*, за умови 3, 5, 10 і 100 ітерацій, *beta1* = 0.9, *beta2* = 0.999, для цифри «0» у всьому інтервалі можливих змін *alpha* (а) та в діапазоні зміни *alpha*, що відповідає недонавчанню, задовільному навчанню і перенавчанню нейронної мережі (б).

Хоча процес подвоєння і проявляється на діаграмі розгалуження в області значень *alpha*, що відповідає за перенавчання нейронної мережі, але подальшого розвитку механізм переходу до хаотичного стану не набуває. Як відзначалось в роботі [3], при зменшені величини виразу 1 - *beta2* спостерігається блокування процесу подвоєння кількості локальних мінімумів та процесу перенавчання, тобто проходить процес зняття виродження системи (рис.4, б). При максимально можливих значеннях швидкості навчання *alpha* для кожного значення гіперпараметра *beta2*, Фур'є спектри характеризуються наявністю значної кількості локальних мінімумів та переходів системи через глобальний мінімум (рис.4, в). Як було показано вище, за певних значеннях швидкості навчання *alpha*, в досліджуваній системі спостерігається процес перенавчання, який супроводжується переходом системи через глобальний мінімум, причому не одноразово.



а) *beta2* = 0.9 б) *beta2* = 0.9, *alpha* = 0.018 в) *beta2* = 0.9, *alpha* = 0.09

а) *beta2* = 0.99 б) *beta2* = 0.99, *alpha* = 0.0055 в) *beta2* = 0.99, *alpha* = 0.027

а) *beta2* = 0.999 б) *beta2* = 0.999, *alpha* = 0.0015 в) *beta2* = 0.999, *alpha* = 0.009

а) *beta2* = 0.9999 б) *beta2* = 0.9999, *alpha* = 0.00045 в) *beta2* = 0.9999, *alpha* = 0.0025

Рис.4. Діаграма розгалуження (а) від параметра *alpha* і Фур'є спектрів (б, в) (Фур'є спектри в діапазоні значень параметра *alpha*, що відповідає перенавчанню системи (б) та в діапазоні значень параметра *alpha*, що відповідає хаотичному стану системи (в)), при значеннях гіперпараметра *beta1* = 0.9, кількісті ітерацій – 100, та при значеннях гіперпараметра *beta2* = 0.9; 0.99; 0.999; 0.9999, для цифри «0».

Тобто за умови коли швидкість навчання *alpha* більша за швидкість перенавчання відбувається перехід до хаотичного стану, який супроводжується, як багатократним проходженням через глобальний мінімум, так і мабуть виникненням локальних мінімумів.

За такої швидкості навчання оптимізатор практично не працює, але за наявності гіперпараметра *beta1*, тобто лінійного градієнту, загальна картина переходу до хаосу описується процесом подвоєння кількості локальних мінімумів. Як було показано вище, за умови нульового вкладу гіперпараметра *beta1*, процес подвоєння є відсутній. Тому розглянемо вид діаграми розгалуження при 100 ітераціях за нульового значення гіперпараметра *beta1*.

На рис.5 наведено діаграму розгалуження при *beta1* = 0.0, *beta2* = 0.999 при 3, 5, 10 і 100 ітераціях (рис.5, а – діаграма розгалуження в сьому діапазоні можливих змін *alpha*; Рис.5, б – діапазон зміни *alpha*, що відповідає недонавчанню, задовільному навчанню і перенавчанню нейронної мережі).

Як і при 10 ітераціях так і при 100 ітераціях, за нульового значення гіперпараметра *beta1*, простежується відсутність процесу подвоєння кількості локальних та глобальних мінімумів. Діапазон існування недонавчання, задовільного навчання та перенавчання від кількості ітерацій практично не залежить (рис.5, б). Перехід до хаосу супроводжується утворенням блочної структури, і за умови збільшення величини швидкості навчання *alpha* (*alpha* > 0.0017) спостерігається збільшенням їх кількості. На це зокрема вказують і Фур’є спектри, згідно яких, при перенавчанні системи, спостерігається збільшення кількості існуючих блоків. Більш детальний розгляд переходу до хаосу в таких системах буде розглянуто в окремій роботі.

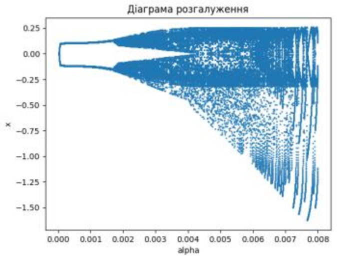
Розглянемо, як на процес навчання впливає значення гіперпараметра *beta2* (*beta2* = 0.9; 0.99; 0.999; 0.9999) за умови, що гіперпараметр *beta1* = 0 при 100 ітераціях. На Рис.6 наведено залежність значення логістичної функції похибки від параметра *alpha* і Фур’є спектрів при значенні гіперпараметра *beta1* = 0.0, кількісті ітерацій рівною 100, та при значеннях гіперпараметра beta2 = 0.9; 0.99; 0.999; 0.9999.

Як при 10 ітераціях так і при 100 ітераціях, за умови *beta1* = 0.0, при зміні *beta2* не спостерігаються процеси подвоєння на діаграмі розгалуження, так і на Фур’є спектрах не спостерігається поява гармоніки в діапазоні зміни *alpha*, що відповідає процесу перенавчання. Але на Фур’є спектрах що відповідають максимально можливій швидкості навчання *alpha* прослідковується співіснування хаотичної та періодичної структури. Це особливо яскраво проявляється при *beta2* = 0.999. Тобто згідно діаграми розгалуження, цільова функція похибки характеризується існуванням структури у вигляді блоків, які (блоки) в свою чергу характеризується відповідною кількістю локальних і глобальних мінімумів. В загальному, дана система демонструє хаотичну поведінку, але при цьому складається із різних блоків, які характеризуються різною кількістю локальних і глобальних мінімумів. Тобто процес подвоєння кількості локальних і глобальних мінімумів пов'язаний із гіперпараметра *beta1*, а гіперпараметер *beta2* спричиняє появу блочної структури, тобто здійснює процес прорідження градієнтів.

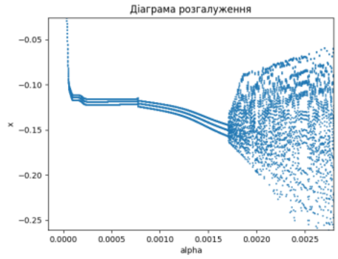
Отже, при застосуванні стохастичного методу оптимізації AMSGrad до багатошарової нейронної мережі (трьохшарова мережа з 28 нейронами в шарі) для розпізнавання друкованих цифр встановлено, що гіперпараметер *beta1*, що описує вклад лінійного градієнта функції похибки і є основою степеневої функції від кількості ітерацій, пов'язаний із подвоєнням кількості локальних і глобальних мінімумів функції похибки в процесі перенавчання нейронної мережі. А гіперпараметер *beta2*, що описує вклад квадрата градієнта функції похибки, пов'язаний із утворенням блочної структури, яка блокує процес подвоєння і тим самим приводить до розрідження градієнтів.

ISSN 2224-087X. Електроніка та інформаційні технології. 2023. Випуск 21

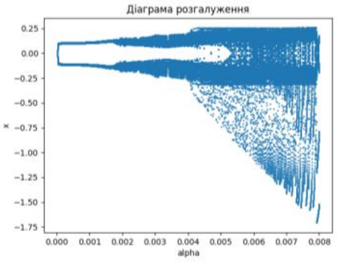
Що стосується похибки навчання то вона практично в 3 рази більша при умові не використання стохастичного метода оптимізації AMSGrad (Таблиця 1, Таблиця 2).



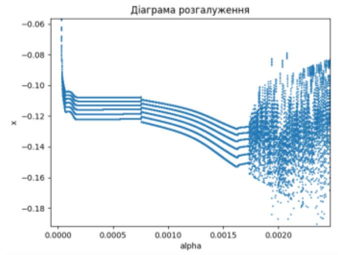
а) епох = 3



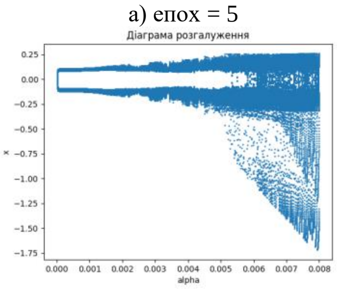
б) епох = 3



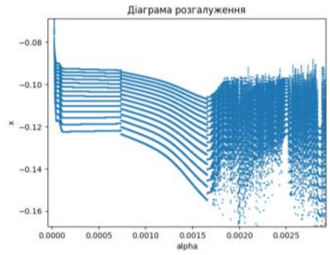
а) епох = 5



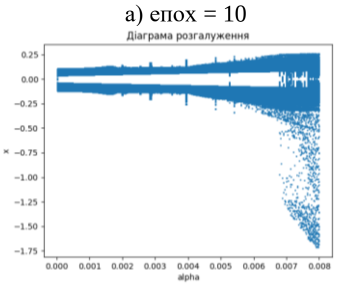
б) епох = 5



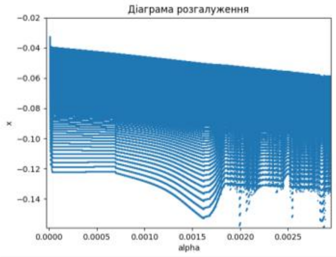
а) епох = 10



б) епох = 10



а) епох = 100

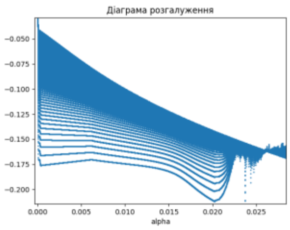


б) епох = 100

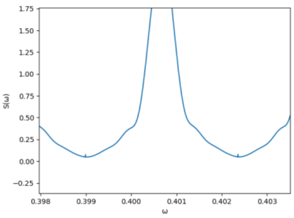
Рис.5. Діаграма розгалуження від швидкості навчання *alpha*, за умови 3, 5, 10 і 100 ітераціях, *beta1* = 0.0, *beta2* = 0.999, для цифри «0» у всьому інтервалі можливих змін *alpha* (а), та в діапазоні зміни *alpha*, що відповідає не довчання, задовільного навчання і перенавчання нейронної мережі (б).

ISSN 2224-087X. Електроніка та інформаційні технології. 2023. Випуск 21

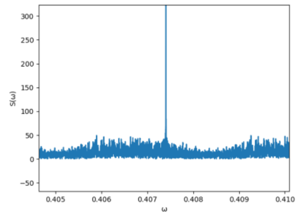
Як при *beta1* = 0.0 так і при *beta1* = 0.9 простежується та сама закономірність при збільшенні величини гіперпараметра *beta2* (beta2=0.0; 0.9; 0.99; 0.999; 0.9999). А саме, при *beta2* = 0.99 спостерігається найменша величина похибки навчання.



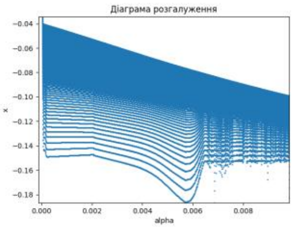
а) *beta2* = 0.9



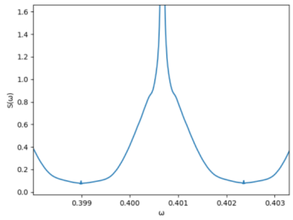
б) *beta2* = 0.9, *alpha* = 0.02



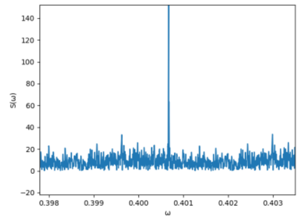
в) *beta2* = 0.9, *alpha* = 0.09



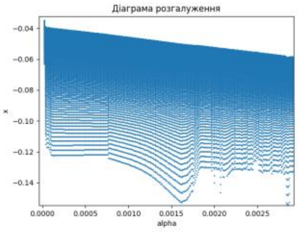
а) *beta2* = 0.99



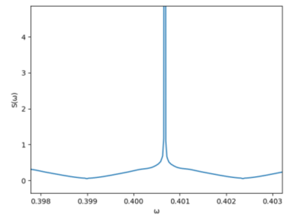
б) *beta2* = 0.99, *alpha* = 0.006



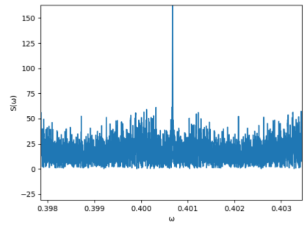
в) *beta2* = 0.99, *alpha* = 0.027



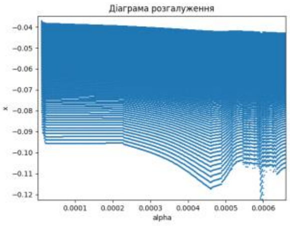
а) *beta2* = 0.999



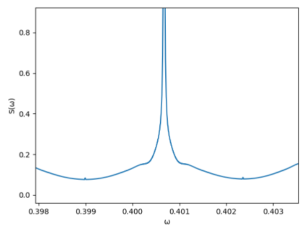
б) *beta2* = 0.999, *alpha* = 0.0015



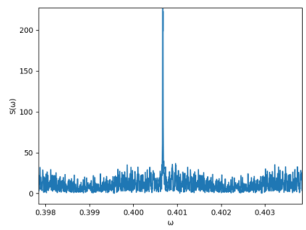
в) *beta2* = 0.999, *alpha* = 0.008



а) *beta2* = 0.9999



б) *beta2* = 0.9999, *alpha* = 0.00045



в) *beta2* = 0.9999, *alpha* = 0.0025

Рис.6. Діаграма розгалуження (а) від параметра *alpha* і Фур'є спектри (б, в) (Фур'є спектри в діапазоні значень параметра *alpha*, що відповідає перенавчанні системи (б) та в діапазоні значень параметра *alpha*, що відповідає хаотичному стану системи (в)), при значеннях гіперпараметра *beta1* = 0.0, кількісті ітерацій рівних 100, та при значеннях гіперпараметра *beta2* = 0.9; 0.99; 0.999; 0.9999, для цифри «0».

ISSN 2224-087X. Електроніка та інформаційні технології. 2023. Випуск 21

Порівнюючи величину похибки навчання при *beta1* = 0.0 і при *beta1* = 0.9, можна зазначити, що похибка навчання при *beta1* = 0.9 є дещо меншою. Це добре корелює із висновками робіти [3], де зазначалось, що найкращі результати були досягнуті при малих значеннях величини (1 - *beta2*) і наявності корекції зсуву (*beta1* > 0.0). Спостережувана залежність похибки навчання від величини гіперпараметра *beta2* (Таблиця 1) зумовлена зміною стану навчання нейромережі. Тобто при збільшенні значення параметра *beta2* нейромережа переходить від стану недонавчання, до задовільного навчання, перенавчання та до хаотичного її стану при сталому значенні швидкості навчання.

Отже, застосування стохастичного методу оптимізації AMSGrad при навчанні багатошарової нейромережі приводить до кращого навчання (Таблиця 1), у тому числі і при оптимальній швидкості навчання (швидкість навчання при якій проявляється подвоєння кількості існуючих локальних і глобальних мінімумів)(Таблиця 2).

Таблиця 1.

Похибка навчання багатошарової нейронної мережі (з трьома прихованими шарами з 28 нейронами в кожному шарі) при 100 ітераціях, *alpha* = 0.001, за умови застосування стохастичного метода оптимізації AMSGrad в процесі розпізнавання друкованих цифр, які задані масивом 4x7 нулів і одиничок.

	beta2 = 0	*beta2* = 0.9	*beta2* = 0.99	*beta2* = 0.999	*beta2* = 0.9999
Beta1 = 0.0	0 похибка= 0.000985 1 похибка= 0.001013 2 похибка= 0.000986 3 похибка= 0.001057 4 похибка= 0.000974 5 похибка= 0.00106 6 похибка= 0.001075 7 похибка= 0.000888 8 похибка= 0.000943 9 похибка= 0.001018	0 похибка= 0.001081 1 похибка= 0.0011 2 похибка= 0.001093 3 похибка= 0.00111 4 похибка= 0.001087 5 похибка= 0.001109 6 похибка= 0.001109 7 похибка= 0.001087 8 похибка= 0.001078 9 похибка= 0.001088	0 похибка= 0.000917 1 похибка= 0.001024 2 похибка= 0.000962 3 похибка= 0.000975 4 похибка= 0.000936 5 похибка= 0.000978 6 похибка= 0.000981 7 похибка= 0.000974 8 похибка= 0.000909 9 похибка= 0.00095	0 похибка= 0.001351 1 похибка= 0.001466 2 похибка= 0.001359 3 похибка= 0.001392 4 похибка= 0.001476 5 похибка= 0.001398 6 похибка= 0.001405 7 похибка= 0.00142 8 похибка= 0.001332 9 похибка= 0.0014	0 похибка= 0.001473 1 похибка= 0.002069 2 похибка= 0.001981 3 похибка= 0.001637 4 похибка= 0.001846 5 похибка= 0.00163 6 похибка= 0.001727 7 похибка= 0.002325 8 похибка= 0.001475 9 похибка= 0.001477
Beta1 = 0.9	0 похибка= 0.000915 1 похибка= 0.00094 2 похибка= 0.000914 3 похибка= 0.000992 4 похибка= 0.000903 5 похибка= 0.000995 6 похибка= 0.001011 7 похибка= 0.000811 8 похибка= 0.00087 9 похибка= 0.000949	0 похибка= 0.001071 1 похибка= 0.001089 2 похибка= 0.001083 3 похибка= 0.001098 4 похибка= 0.001076 5 похибка= 0.001097 6 похибка= 0.001097 7 похибка= 0.001077 8 похибка= 0.001069 9 похибка= 0.001077	0 похибка= 0.000884 1 похибка= 0.000983 2 похибка= 0.00093 3 похибка= 0.00094 4 похибка= 0.000897 5 похибка= 0.000941 6 похибка= 0.000943 7 похибка= 0.000937 8 похибка= 0.000878 9 похибка= 0.000911	0 похибка= 0.001258 1 похибка= 0.00134 2 похибка= 0.001264 3 похибка= 0.001288 4 похибка= 0.00137 5 похибка= 0.001293 6 похибка= 0.001298 7 похибка= 0.001317 8 похибка= 0.001244 9 похибка= 0.001294	0 похибка= 0.00158 1 похибка= 0.002175 2 похибка= 0.002137 3 похибка= 0.001723 4 похибка= 0.001965 5 похибка= 0.001724 6 похибка= 0.001711 7 похибка= 0.002359 8 похибка= 0.001707 9 похибка= 0.001677

Таблиця 2.

Похибка навчання багатошарової нейронної мережі (з трьома прихованими шарами з 28 нейронами в кожному шарі) при 100 ітераціях, і оптимальній швидкості навчання в процесі розпізнавання друкованих цифр, які задані масивом 4x7 нулів і одиничок.

цифра= 0 ; alpha оптимальне= 0.46 ; мінімальна похибка= 0.004829
цифра= 1 ; alpha оптимальне= 0.45 ; мінімальна похибка= 0.004975
цифра= 2 ; alpha оптимальне= 0.45 ; мінімальна похибка= 0.005025
цифра= 3 ; alpha оптимальне= 0.45 ; мінімальна похибка= 0.005236
цифра= 4 ; alpha оптимальне= 0.45 ; мінімальна похибка= 0.005088
цифра= 5 ; alpha оптимальне= 0.45 ; мінімальна похибка= 0.00493
цифра= 6 ; alpha оптимальне= 0.45 ; мінімальна похибка= 0.00496
цифра= 7 ; alpha оптимальне= 0.46 ; мінімальна похибка= 0.00471
цифра= 8 ; alpha оптимальне= 0.45 ; мінімальна похибка= 0.005261
цифра= 9 ; alpha оптимальне= 0.46 ; мінімальна похибка= 0.004996

Список використаних джерел

[1] *Sashank J.* On the Convergence of Adam and Beyond /Sashank J. Reddi, Satyen Kale, Sanjiv Kumar. – Published as a conference paper at ICLR 2018. – 2019. – P. 1-23. https://doi.org/10.48550/arXiv.1904.09237

[2] *Jason Brownlee.* Optimization for Machine Learning. Finding Function Optima with Python. – The MIT Press. – 2021. – 403p.

[3] *Diederik P.* Adam: a method for stochastic optimization / Diederik P. Kingma, Jimmy Lei Ba – Published as a conference paper at ICLR 2015. – 2015. – P. 1-15. https://doi.org/10.48550/arXiv.1412.6980

[4] *Yu. Taranenko* Information entropy of chaos URL*:* https://habr.com/ru/post/447874/

[5] *Свелеба С.* Хаотичні стани багатошарової нейронної мережі / С. Свелеба, І. Катеринчук, І. Куньо, І. Карпа, О. Семотюк, Я. Шмигельський, Н. Свелеба, В. Куньо // Electronics and information technologies. – 2021. – Issue 13. – P. 96–107.

[6] *Sveleba S.* Specifics of the learning error dependence of multilayered neural networks from the activation function during the process of printed digits identification / S. Sveleba, I. Katerynchuk, I. Kuno, O. Semotiuk, Ya. Shmyhelskyy, N. Sveleba // Electronics and Information Technologies. – 2022. – Issue 17. – P. 36–53.

ALGORITHM OF AMSGrad AND CHAOS OPTIMIZATION IN MULTILAYERED NEURON NETWORKS WITH STOCHASTIC GRADIENT DESCENT

S. Sveleba[1], I. Katerynchuk[1], I. Kuno[1], O. Semotyuk[2], Ya. Shmygelsky[1], S. Velgosh[1], A. Kopach[1], V. Kuno[3]

Lviv National University named after Ivan Franko,
107 Gen. Tarnavskoho St., 79017 Lviv, Ukraine
incomlviv@gmail.com

[2] Ukrainian Academy of Printing,
19 Pid Goloskom str., 79020 Lviv, Ukraine

[3] Lviv Polytechnic National University,
12 Stepan Bandera Str 79013 Lviv, Ukraine

In this paper, the AMSGrad stochastic optimization method was tested using the logistic function that describes the doubling process and the Fourier spectra of the error function.

The implementation of the gradient descent optimization algorithm, using AMSGrad, was done for a multilayer neural network with hidden layers. The program for recognizing printed digits was written using the Python software environment. The array of each digit consisted of a set of "0" and "1" of size 4x7. The sample of each digit contained a set of 5 possible distortions and a set of 3 arrays that did not correspond to any of the digits. The analysis of the influence of the value of hyperparameters *beta1*, *beta2*, and learning rate on the optimizing process for teaching a multilayer neural network, which contained 3 hidden layers of 28 neurons each, was carried out. We constructed branching diagrams based on these parameters.

We found that the hyperparameter *beta1*, which describes the contribution of the linear gradient of the error function, is associated with a doubling of the number of local and global minima of the error function in the process of retraining the neural network. The hyperparameter *beta2*, which describes the error function gradient square contribution, is associated with the block structure formation that blocks the number of local minima doubling processes.

If the *alpha* learning rate is greater than the retraining rate, there is a transition to a chaotic state, which is accompanied by both multiple passage through the global minimum and, apparently, the appearance of local minima.

At such a speed of learning, the optimizer practically does not work, but in the presence of the hyperparameter *beta1*, i.e. a linear gradient, the general picture of the transition to chaos is described by the process of doubling the number of local minima.

The application of the AMSGrad stochastic optimization method for teaching a multilayer neural network is shown to lead to better learning, compared to a conventional multilayer neural network, even at the optimal learning rate (the learning rate when the number of existing local and global minima doubles).

Keywords: optimization methods, error function, AMSGrad, learning rate, branching diagrams.

Стаття надійшла до редакції 17.02.2023
Прийнята до друку 03.03.2023