Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе № 1

по курсу «Основы глубокого обучения»

Выполнил:  
студент группы 22ВВИм1:

Милованов А.С.

Приняли:

Митрохин М.А  
Панков А.А.

Пенза 2023

# Порядок выполнения работы:

# Задание:

# Реализовать правила остановки алгоритма обучения перцептрона при его зацикливании или сходимости.

# Теорема о сходимости персептрона. Если существует вектор параметров α, при котором персептрон правильно решает все примеры обучающей выборки, то при обучении персептрона по правилу Хебба решение будет найдено за конечное число шагов.

# Теорема о «зацикливании» персептрона. Если не существует вектора параметров α, при котором персептрон правильно решает все примеры обучающей выборки, то при обучении персептрона по правилу Хебба через конечное число шагов вектор весов начнет повторяться.

# Листинг программы:

import pandas as pd  
import numpy as np  
  
  
df = pd.read\_csv('data.csv')  
df = df.iloc[np.random.permutation(len(df))]  
y = df.iloc[0:100, 4].values  
y = np.where(y == "Iris-setosa", 1, -1)  
  
X = df.iloc[0:100, [0, 2]].values  
  
input\_size = X.shape[1]  
neurons\_hidden\_layer= 10  
neurons\_output\_layer = 1 if len(y.shape) else y.shape[1] # количество выходных сигналов равно количеству классов задачи  
print('input\_size:', input\_size)  
print('neurons\_hidden\_layer:',neurons\_hidden\_layer)  
print('neurons\_output\_layer:',neurons\_output\_layer)  
  
# матрица весов и порого первого слоя  
W\_1 = np.zeros((1 + input\_size, neurons\_hidden\_layer))  
# пороговые значения  
W\_1[0, :] = (np.random.randint(0, 3, size = (neurons\_hidden\_layer)))  
# веса  
W\_1[1:, :] = (np.random.randint(-1, 2, size = (input\_size, neurons\_hidden\_layer)))  
print('W\_1:',W\_1)  
  
# матрица весов и порогов второго слоя  
W\_2 = np.random.randint(0, 2, size = (1 + neurons\_hidden\_layer, neurons\_output\_layer)).astype(np.float64)  
print('W\_2:', W\_2)  
  
def predict(X):  
 W\_1\_out = np.where((np.dot(X, W\_1[1:, :]) + W\_1[0, :]) >= 0.0, 1, -1).astype(np.float64)  
 W\_2\_out = np.where((np.dot(W\_1\_out, W\_2[1:, :]) + W\_2[0, :]) >= 0.0, 1, -1).astype(np.float64)  
 return W\_2\_out, W\_1\_out  
  
# количетсво обучающих итераций  
n\_iter=0  
# шаг обучения  
step = 0.01  
#количество  
check\_iter = 5  
  
# список для хранения матрицы весов второго слоя  
list\_w\_2\_weights= [];  
  
#обучение  
while(True):  
 print('iteration:',n\_iter)  
 n\_iter+=1  
  
  
  
 for x\_input, expected, j in zip(X, y, range(X.shape[0])):  
 W\_2\_out, W\_1\_out = predict(x\_input)  
 W\_2[1:] += ((step \* (expected - W\_2\_out)) \* W\_1\_out).reshape(-1, 1)  
 W\_2[0] += step \* (expected - W\_2\_out)  
  
 #список матрицы весов второго слоя  
 list\_w\_2\_weights.append(W\_2.tobytes())  
  
 W\_2\_out, W\_1\_out = predict(X)  
 sum\_errors =sum(W\_2\_out.reshape(-1) - y)  
 print('sum\_errors', sum\_errors)  
 if (sum\_errors == 0):  
 print('Все примеры обучающей выборки решены:')  
 break  
  
 #проверка наличия дубликатов в списке весов  
 if ((n\_iter % check\_iter )==0):  
 break\_out\_flag = False  
 for item in list\_w\_2\_weights:  
 if list\_w\_2\_weights.count(item) > 1:  
 print('Повторение весов:')  
 break\_out\_flag = True  
 break  
 if break\_out\_flag:  
 break  
  
  
W\_2\_out, W\_1\_out = predict(X)  
sum\_errors =sum( W\_2\_out.reshape(-1) - y)  
print('sum\_errors', sum\_errors)

# Результат работы программы:

# 

Рисунок 1 Остановка обучения по первой теореме

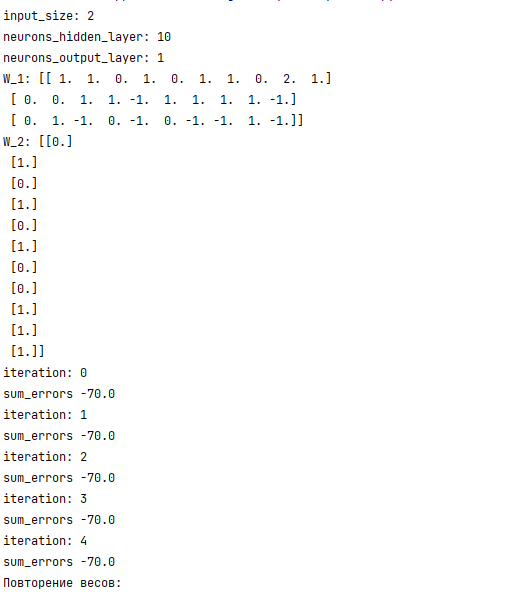


Рисунок Остановка обучения по второй теореме

# Вывод:

# В ходе выполнения лабораторной работы был реализованы правила остановки алгоритма обучения перцептрона при его зацикливании или сходимости.

# 