**Лабораторная работа № 1**

**«Персептрон Румельхарта»**

Задание № 1: Классификация цифр.

Обучающая выборка:

1,1,1,1,0,1,1,0,1,1,0,1,1,1,1,0

0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,1

1,1,1,0,0,1,1,1,1,1,0,0,1,1,1,2

1,1,1,0,0,1,1,1,1,0,0,1,1,1,1,3

1,0,1,1,0,1,1,1,1,0,0,1,0,0,1,4

1,1,1,1,0,0,1,1,1,0,0,1,1,1,1,5

1,1,1,1,0,0,1,1,1,1,0,1,1,1,1,6

1,1,1,0,0,1,0,0,1,0,0,1,0,0,1,7

1,1,1,1,0,1,1,1,1,1,0,1,1,1,1,8

1,1,1,1,0,1,1,1,1,0,0,1,1,1,1,9

Тестовая выборка с зашумлёнными данными:

1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,0

0,0,1,1,0,1,0,0,1,0,0,1,0,1,1,1

1,1,1,0,1,1,1,1,1,1,1,0,1,1,1,2

1,1,1,0,0,1,1,1,1,0,1,0,1,1,1,3

1,1,1,1,1,1,1,1,1,0,0,1,0,0,1,4

1,1,1,0,0,0,1,1,1,0,0,1,1,1,1,5

1,1,1,1,1,0,1,1,1,1,0,0,1,1,1,6

1,1,1,0,0,1,0,0,1,1,1,1,0,0,1,7

1,1,1,1,0,1,0,1,1,1,0,1,1,1,1,8

1,1,1,1,0,1,0,1,1,0,0,1,1,0,1,9

Реализация персептрона Румельхарта на языке Python:

%matplotlib notebook

import numpy as np

from numpy import loadtxt

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

from RBFN import RBFN

import matplotlib.pyplot as plt

# Задание 1

dataset = loadtxt('123.txt', delimiter=',')

X = dataset[:,0:15]

Y = np.array([

[1, 0, 0, 0, 0, 0, 0, 0, 0, 0],

[0, 1, 0, 0, 0, 0, 0, 0, 0, 0],

[0, 0, 1, 0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 1, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 1, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 1, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 1, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0, 1, 0, 0],

[0, 0, 0, 0, 0, 0, 0, 0, 1, 0],

[0, 0, 0, 0, 0, 0, 0, 0, 0, 1]

])

model = Sequential()

model.add(Dense(15, input\_dim=15, activation='sigmoid'))

#model.add(Dense(15, activation='sigmoid'))

model.add(Dense(10, activation='sigmoid'))

model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])

# Обучение и оценка точности на обучающей выборке

model.fit(X, Y, epochs=350, batch\_size=1, verbose=0)

\_, accuracy = model.evaluate(X, Y, verbose=0)

print('Точность: %.2f' % (accuracy\*100), '%')

# Тестирование на обучающей выборке

predictions = model.predict\_classes(X)

for i in range(10):

print('Сеть предсказала:', predictions[i], 'Правильный ответ:', i, Y[i])

string = ''

for j in range(len(X[i])):

string += str(int(X[i][j])) + ' '

if ((j + 1) % 3 == 0):

print(string)

string = ''

Результаты работы программы:

Точность: 100.00 %

Сеть предсказала: 0 Правильный ответ: 0 [1 0 0 0 0 0 0 0 0 0]

1 1 1

1 0 1

1 0 1

1 0 1

1 1 1

Сеть предсказала: 1 Правильный ответ: 1 [0 1 0 0 0 0 0 0 0 0]

0 0 1

0 0 1

0 0 1

0 0 1

0 0 1

Сеть предсказала: 2 Правильный ответ: 2 [0 0 1 0 0 0 0 0 0 0]

1 1 1

0 0 1

1 1 1

1 0 0

1 1 1

Сеть предсказала: 3 Правильный ответ: 3 [0 0 0 1 0 0 0 0 0 0]

1 1 1

0 0 1

1 1 1

0 0 1

1 1 1

Сеть предсказала: 4 Правильный ответ: 4 [0 0 0 0 1 0 0 0 0 0]

1 0 1

1 0 1

1 1 1

0 0 1

0 0 1

Сеть предсказала: 5 Правильный ответ: 5 [0 0 0 0 0 1 0 0 0 0]

1 1 1

1 0 0

1 1 1

0 0 1

1 1 1

Сеть предсказала: 6 Правильный ответ: 6 [0 0 0 0 0 0 1 0 0 0]

1 1 1

1 0 0

1 1 1

1 0 1

1 1 1

Сеть предсказала: 7 Правильный ответ: 7 [0 0 0 0 0 0 0 1 0 0]

1 1 1

0 0 1

0 0 1

0 0 1

0 0 1

Сеть предсказала: 8 Правильный ответ: 8 [0 0 0 0 0 0 0 0 1 0]

1 1 1

1 0 1

1 1 1

1 0 1

1 1 1

Сеть предсказала: 9 Правильный ответ: 9 [0 0 0 0 0 0 0 0 0 1]

1 1 1

1 0 1

1 1 1

0 0 1

1 1 1

Тестирование на зашумлённой выборке:

# Оценка точности и тестирование на тестовой выборке

dataset = loadtxt('1232.txt', delimiter=',')

X = dataset[:,0:15]

\_, accuracy = model.evaluate(X, Y, verbose=0)

print('Точность: %.2f' % (accuracy\*100), '%')

predictions = model.predict\_classes(X)

for i in range(10):

print('Сеть предсказала:', predictions[i], 'Правильный ответ:', i, Y[i])

string = ''

for j in range(len(X[i])):

string += str(int(X[i][j])) + ' '

if ((j + 1) % 3 == 0):

print(string)

string = ''

Результаты работы программы:

Точность: 80.00 %

Сеть предсказала: 0 Правильный ответ: 0 [1 0 0 0 0 0 0 0 0 0]

1 1 1

1 1 1

1 0 1

1 1 1

1 1 1

Сеть предсказала: 4 Правильный ответ: 1 [0 1 0 0 0 0 0 0 0 0]

0 0 1

1 0 1

0 0 1

0 0 1

0 1 1

Сеть предсказала: 2 Правильный ответ: 2 [0 0 1 0 0 0 0 0 0 0]

1 1 1

0 1 1

1 1 1

1 1 0

1 1 1

Сеть предсказала: 3 Правильный ответ: 3 [0 0 0 1 0 0 0 0 0 0]

1 1 1

0 0 1

1 1 1

0 1 0

1 1 1

Сеть предсказала: 4 Правильный ответ: 4 [0 0 0 0 1 0 0 0 0 0]

1 1 1

1 1 1

1 1 1

0 0 1

0 0 1

Сеть предсказала: 5 Правильный ответ: 5 [0 0 0 0 0 1 0 0 0 0]

1 1 1

0 0 0

1 1 1

0 0 1

1 1 1

Сеть предсказала: 6 Правильный ответ: 6 [0 0 0 0 0 0 1 0 0 0]

1 1 1

1 1 0

1 1 1

1 0 0

1 1 1

Сеть предсказала: 7 Правильный ответ: 7 [0 0 0 0 0 0 0 1 0 0]

1 1 1

0 0 1

0 0 1

1 1 1

0 0 1

Сеть предсказала: 8 Правильный ответ: 8 [0 0 0 0 0 0 0 0 1 0]

1 1 1

1 0 1

0 1 1

1 0 1

1 1 1

Сеть предсказала: 4 Правильный ответ: 9 [0 0 0 0 0 0 0 0 0 1]

1 1 1

1 0 1

0 1 1

0 0 1

1 0 1

Результаты работы сети не зависят от того, в каком порядке подаются примеры из обучающей выборки, однако, если какие-то цифры будут подаваться чаще, чем остальные, то те цифры, которые подавались реже, будут распознаваться с меньшей точностью.

Эффект паралича коэффициентов появляется при возникновении больших приращений весовых коэффициентов нейронов. Это может привести к тому, что веса примут очень большие значения, на несколько порядков большие, чем величина, на которую они корректируются каждую эпоху обучения. Таким образом, обучение может затянуться на неопределённо долгий промежуток времени или даже никогда не закончиться.

Эффект переобучения сети представляет собой ситуацию, когда обученная нейронная сеть хорошо распознаёт примеры из обучающей выборки, однако плохо распознаёт или не распознаёт вообще любые другие примеры. То есть сеть теряет способность к обобщению. Переобучение возникает из-за большого числа излишних промежуточных слоёв, нейронов в слоях и эпох обучения.

Пример переобучения:

model = Sequential()

model.add(Dense(15, input\_dim=15, activation='sigmoid'))

model.add(Dense(50, activation='sigmoid'))

model.add(Dense(50, activation='sigmoid'))

model.add(Dense(50, activation='sigmoid'))

model.add(Dense(50, activation='sigmoid'))

model.add(Dense(50, activation='sigmoid'))

model.add(Dense(10, activation='sigmoid'))

model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])

# Обучение и оценка точности на обучающей выборке

model.fit(X, Y, epochs=1000, batch\_size=1, verbose=0)

\_, accuracy = model.evaluate(X, Y, verbose=0)

print('Точность: %.2f' % (accuracy\*100), '%')

# Тестирование на обучающей выборке

predictions = model.predict\_classes(X)

for i in range(10):

print('Сеть предсказала:', predictions[i], 'Правильный ответ:', i, Y[i])

string = ''

for j in range(len(X[i])):

string += str(int(X[i][j])) + ' '

if ((j + 1) % 3 == 0):

print(string)

string = ''

Результаты работы программы (проверка на обучающей выборке):

Точность: 20.00 %

Сеть предсказала: 8 Правильный ответ: 0 [1 0 0 0 0 0 0 0 0 0]

1 1 1

1 0 1

1 0 1

1 0 1

1 1 1

Сеть предсказала: 7 Правильный ответ: 1 [0 1 0 0 0 0 0 0 0 0]

0 0 1

0 0 1

0 0 1

0 0 1

0 0 1

Сеть предсказала: 8 Правильный ответ: 2 [0 0 1 0 0 0 0 0 0 0]

1 1 1

0 0 1

1 1 1

1 0 0

1 1 1

Сеть предсказала: 8 Правильный ответ: 3 [0 0 0 1 0 0 0 0 0 0]

1 1 1

0 0 1

1 1 1

0 0 1

1 1 1

Сеть предсказала: 7 Правильный ответ: 4 [0 0 0 0 1 0 0 0 0 0]

1 0 1

1 0 1

1 1 1

0 0 1

0 0 1

Сеть предсказала: 8 Правильный ответ: 5 [0 0 0 0 0 1 0 0 0 0]

1 1 1

1 0 0

1 1 1

0 0 1

1 1 1

Сеть предсказала: 8 Правильный ответ: 6 [0 0 0 0 0 0 1 0 0 0]

1 1 1

1 0 0

1 1 1

1 0 1

1 1 1

Сеть предсказала: 7 Правильный ответ: 7 [0 0 0 0 0 0 0 1 0 0]

1 1 1

0 0 1

0 0 1

0 0 1

0 0 1

Сеть предсказала: 8 Правильный ответ: 8 [0 0 0 0 0 0 0 0 1 0]

1 1 1

1 0 1

1 1 1

1 0 1

1 1 1

Сеть предсказала: 8 Правильный ответ: 9 [0 0 0 0 0 0 0 0 0 1]

1 1 1

1 0 1

1 1 1

0 0 1

1 1 1

# Оценка точности и тестирование на тестовой выборке

dataset = loadtxt('1232.txt', delimiter=',')

X = dataset[:,0:15]

\_, accuracy = model.evaluate(X, Y, verbose=0)

print('Точность: %.2f' % (accuracy\*100), '%')

predictions = model.predict\_classes(X)

for i in range(10):

print('Сеть предсказала:', predictions[i], 'Правильный ответ:', i, Y[i])

string = ''

for j in range(len(X[i])):

string += str(int(X[i][j])) + ' '

if ((j + 1) % 3 == 0):

print(string)

string = ''

Результаты работы программы (проверка на тестовой зашумлённой выборке):

Точность: 20.00 %

Сеть предсказала: 8 Правильный ответ: 0 [1 0 0 0 0 0 0 0 0 0]

1 1 1

1 1 1

1 0 1

1 1 1

1 1 1

Сеть предсказала: 7 Правильный ответ: 1 [0 1 0 0 0 0 0 0 0 0]

0 0 1

1 0 1

0 0 1

0 0 1

0 1 1

Сеть предсказала: 8 Правильный ответ: 2 [0 0 1 0 0 0 0 0 0 0]

1 1 1

0 1 1

1 1 1

1 1 0

1 1 1

Сеть предсказала: 8 Правильный ответ: 3 [0 0 0 1 0 0 0 0 0 0]

1 1 1

0 0 1

1 1 1

0 1 0

1 1 1

Сеть предсказала: 7 Правильный ответ: 4 [0 0 0 0 1 0 0 0 0 0]

1 1 1

1 1 1

1 1 1

0 0 1

0 0 1

Сеть предсказала: 8 Правильный ответ: 5 [0 0 0 0 0 1 0 0 0 0]

1 1 1

0 0 0

1 1 1

0 0 1

1 1 1

Сеть предсказала: 8 Правильный ответ: 6 [0 0 0 0 0 0 1 0 0 0]

1 1 1

1 1 0

1 1 1

1 0 0

1 1 1

Сеть предсказала: 7 Правильный ответ: 7 [0 0 0 0 0 0 0 1 0 0]

1 1 1

0 0 1

0 0 1

1 1 1

0 0 1

Сеть предсказала: 8 Правильный ответ: 8 [0 0 0 0 0 0 0 0 1 0]

1 1 1

1 0 1

0 1 1

1 0 1

1 1 1

Сеть предсказала: 7 Правильный ответ: 9 [0 0 0 0 0 0 0 0 0 1]

1 1 1

1 0 1

0 1 1

0 0 1

1 0 1

Как можно видеть по результатам экспериментов, сеть научилась определять только цифры 7 и 8 и не способна различать другие.

Задание № 2: Аппроксимация хаотических последовательностей.

Вариант 1: .

Реализация радиально-базисной нейронной сети на языке Python:

# Задание 2

def sequence(num, start=0.5):

x = [start]

if (num == 1):

return start

else:

for i in range(num-1):

x.append(x[-1]\*(1-2\*x[-1]))

return np.array(x)

x = np.arange(15)

x0 = 0.3

y = sequence(len(x), x0)

model = RBFN(hidden\_shape=12, sigma=0.01)

# Обучение и предсказание значений в пределах обучающей выборки

model.fit(x, y)

y\_pred = model.predict(x)

#print(y)

#print(y\_pred)

plt.plot(x, y, 'bo')

plt.plot(x, y\_pred, 'ro')

plt.grid()

# Предсказание значений за пределами обучающей выборки

x = np.arange(20)

y = sequence(len(x), x0)

model.fit(x, y)

y\_pred = model.predict(x)

#print(y)

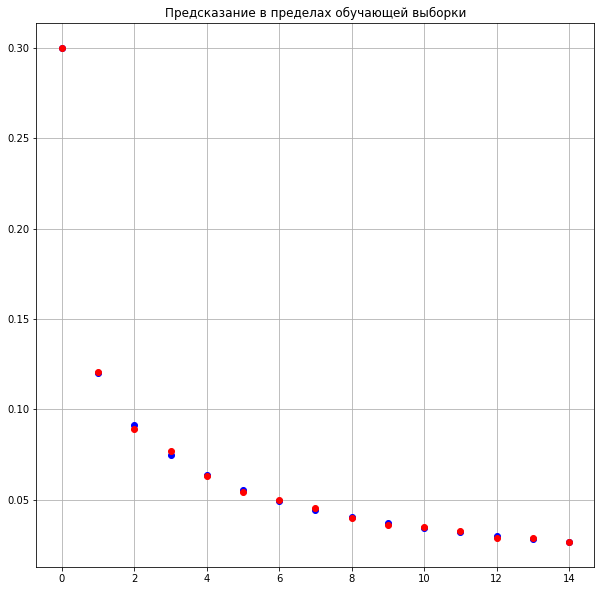
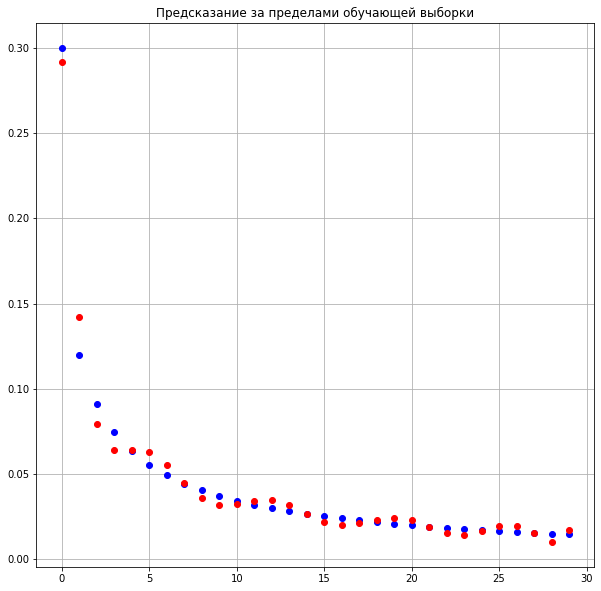
#print(y\_pred)

plt.figure()

plt.plot(x, y, 'bo')

plt.plot(x, y\_pred, 'ro')

plt.grid()

Результаты работы программы:

Как можно видеть по результатам экспериментов, сеть способна предсказывать значения последовательности за пределами обучающей выборки. Точность предсказания будет зависеть от количества нейронов в сети, так как добавление очередного нейрона – добавление базисной функций (в нашем случае Гауссова импульса). Чем больше базисных функций – тем больше точность аппроксимации последовательности. Отсюда же следует, что переобучить такую сеть нельзя. Паралич коэффициентов возможен и приводит к увеличению времени обучения.