Федеральное Государственное бюджетное учреждение

высшего профессионального образования

"Заполярный Государственный Университет имени Н. М. Федоровского"

Кафедра «Информационные системы и технологии»

Нейронные сети

Отчет по лабораторной работе №1

Тема: Персептрон.

Выполнил:

студент группы ИЭ-21

Антошкин Никита

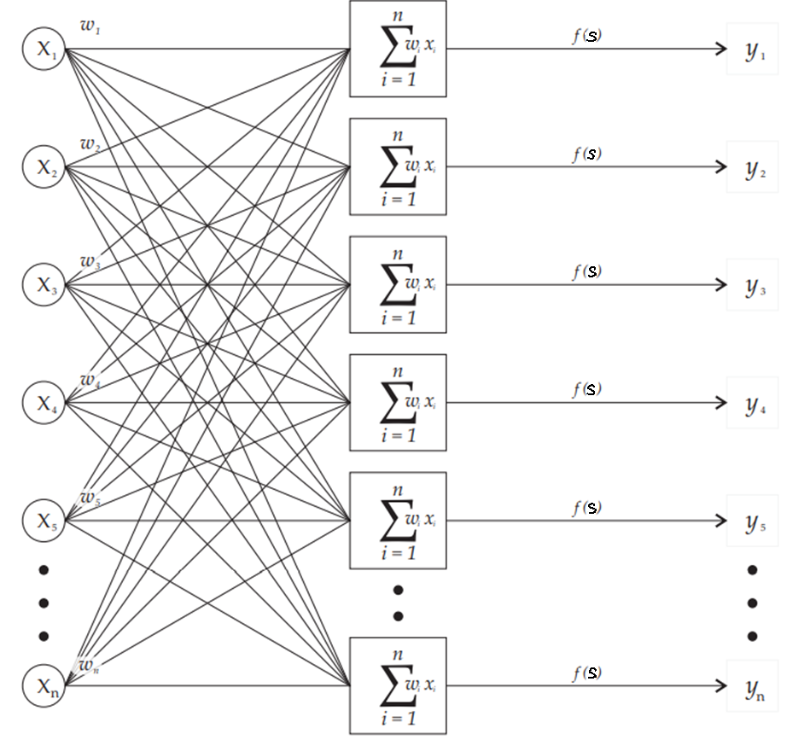
Проверила:

Дыптан Елена Арнольдовна

Норильск 2024

**Цель.** Ознакомиться с основными понятиями нейронных сетей, принципами их работы, построением архитектуры, а также получить практические навыки в создании, настройке и обучении персептрона.

**Архитектура сети**

****

**, ,**

**,**

N = 12 (0,1,2,3,4,5,6,7,8,9)

*, где порог функции*

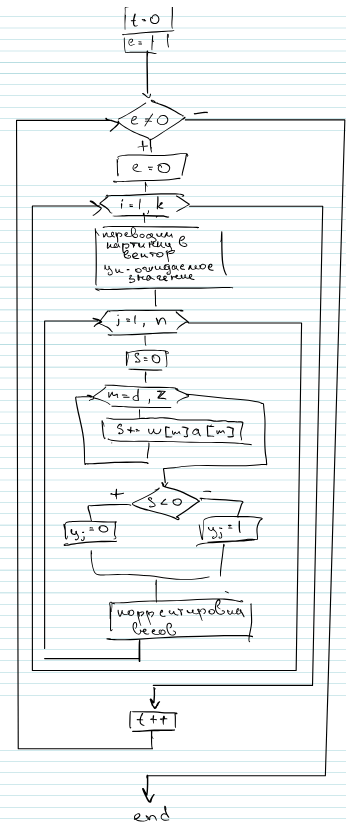
В отчете представлена однослойная сеть, с активационной функцией вида: пороговая, порог имеет значение 0. Весовые коэффициенты инициализируется случайными значениями [-0.3, 0.3]. Количество нейронов равно 10 по количеству цифр: 0, 1, 2, 3, 4, 5, 6, 8, 9. На вход подается изображение 100\*100рх.

**Алгоритм обучения**

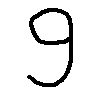
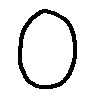
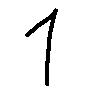
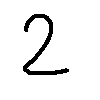
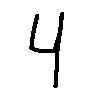
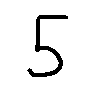
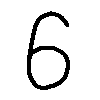
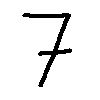
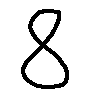
1. Инициализация весов
2. Преобразуем фото в массив данных
3. Обучаем каждый нейрон: пока есть ошибка мы не прекращаем обучение.

Функция обучения:

1. Подаем на вход образы.
2. Задаем счетчик ошибки. После чего мы отправляем образы на проверку нейронам, проверяем ошиблись ли они, если нет, то завершаем обучение и переходим к шагу 4, иначе переходим дальше.
3. Отправляем образы в функцию «обновления весов», которая работает опирается на формулу: . Возвращаемся к шагу 2.
4. Сохраняем веса и переходим к режиму работы и тестирования.

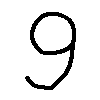
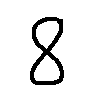
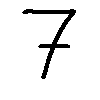
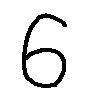
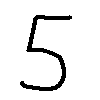
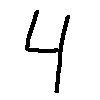
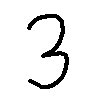
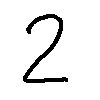
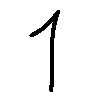
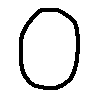


**Обучающая выборка**



Размер обучающей выборки:

**Тестовая выборка**



*Эксперименты*

|  |  |
| --- | --- |
|  | Количество эпох |
| 0,001 | 62 |
| 0,01 | 55 |
| 0,1 | 43 |
| 0,2 | 51 |
| 0,3 | 37 |
| 0,4 | 26 |
| 0,5 | 37 |
| 0,6 | 17 |
| 0,7 | 32 |
| 0,8 | 47 |
| 0,9 | 22 |
| 1 | 52 |

**Вывод:** Проведя тестирование и проанализировав полученные данные, был сделан вывод, чтопоказательскорости обучения напрямую влияет на эффективность нейронной сети.

**Листинг**

def get\_object\_bounds(image):

image\_array = np.array(image)

non\_empty\_columns = np.where(image\_array.min(axis=0) < 255)[0]

non\_empty\_rows = np.where(image\_array.min(axis=1) < 255)[0]

if non\_empty\_columns.any() and non\_empty\_rows.any():

upper, lower = non\_empty\_rows[0], non\_empty\_rows[-1

left, right = non\_empty\_columns[0], non\_empty\_columns[-1]

return left, upper, right, lower

else:

return None

def center\_object(image):

bounds = get\_object\_bounds(image)

if bounds:

left, upper, right, lower = bounds

object\_width = right - left

object\_height = lower - upper

horizontal\_padding = (image.width - object\_width) // 2

vertical\_padding = (image.height - object\_height) // 2

cropped\_image = image.crop(bounds)

centered\_image = Image.new("L", (image.width, image.height), "white")

centered\_image.paste(cropped\_image, (horizontal\_padding, vertical\_padding))

return centered\_image

return image

def load\_images(folder):

images = []

labels = []

for filename in os.listdir(folder):

img\_path = os.path.join(folder, filename)

try:

with Image.open(img\_path) as img:

img = img.convert("L")

img = center\_object(img)

img = img.resize((100, 100))

images.append(np.asarray(img).flatten() / 255.0)

label = int(filename[0])

labels.append(label)

except Exception as e:

print(f"Ошибка загрузки {img\_path}: {e}")

return np.array(images), np.array(labels)

def threshold(value, threshold=0):

return 1 if value > threshold else 0

def guess(image, weights, x):

sum = [np.dot(w, image) for w in weights]

output = [threshold(s, x) for s in sum]

if np.array(output).sum() == 1:

return output, output.index(1)

else:

return output, None

def calculate\_errors(images, labels, weights):

wrong\_count = 0

for img, label in zip(images, labels):

flag = guess(img, weights, 0)

if flag is None or flag != label:

wrong\_count += 1

return wrong\_count

class TestCanvas(tk.Tk):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.canvas = Canvas(self, width=300, height=300, bg="white")

self.canvas.pack()

self.bind("<B1-Motion>", self.draw)

self.image = Image.new("L", (280, 280), "white")

self.draw\_image = ImageDraw.Draw(self.image)

tk.Button(self, text="Проверить", command=self.test\_image).pack(side=tk.BOTTOM)

tk.Button(self, text="Очистить", command=self.clear\_canvas).pack(side=tk.BOTTOM)

tk.Button(self, text="Показать графики", command=self.show\_weights\_graph).pack(side=tk.BOTTOM)

def draw(self, event):

x, y = event.x, event.y

self.canvas.create\_oval(x-7, y-7, x+7, y+7, fill='black')

self.draw\_image.ellipse([x-7, y-7, x+7, y+7], fill='black')

def test\_image(self):

centered\_image = center\_object(self.image)

resized\_image = centered\_image.resize((100, 100))

img\_array = np.array(resized\_image).flatten() / 255.0

result, flag = guess(img\_array, weights, x=0)

if flag is None:

messagebox.showinfo("Результат", "Не похоже на одну из цифр!")

else:

messagebox.showinfo("Результат", f"Похоже на цифру {flag}!")

self.clear\_canvas()

def clear\_canvas(self):

self.canvas.delete("all")

self.image = Image.new("L", (280, 280), "white")

self.draw\_image = ImageDraw.Draw(self.image

def show\_weights\_graph(self):

fig, axes = plt.subplots(nrows=2, ncols=5, figsize=(20, 8))

for i, ax in enumerate(axes.flat):

ax.imshow(weights[i].reshape((100, 100)), cmap='GnBu')

ax.set\_title(f"Веса класса {i}")

plt.tight\_layout()

plt.show()

def train(images, labels, weights, learning\_rate, tolerance = 0):

epoch = 0

while True:

indices = np.random.permutation(len(images))

images\_shuffled = images[indices]

labels\_shuffled = labels[indices]

errors = 0

for img, label in zip(images\_shuffled, labels\_shuffled

predictions, predicted\_label = guess(img, weights, x=0)

if predicted\_label != label

errors += 1

weights[label] += learning\_rate \* img

for i in range(len(weights)):

if predictions[i] == 1 and i != label:

weights[i] -= learning\_rate \* img

epoch += 1

if errors <= tolerance:

print(f"Обучение завершено на эпохе {epoch}. Количество ошибок: {errors}")

return

print(f"Эпоха {epoch}; Ошибок: {errors}")

if \_\_name\_\_ == "\_\_main\_\_":

folder\_path = "learn"

weights = np.random.uniform(-0.3, 0.3, (10, 10000))

learning\_rate = 0.001

images, labels = load\_images(folder\_path)

train(images, labels, weights, learning\_rate)

print("Обучение завершено.")

app = TestCanvas()

app.title("Тест нейронной сети")

app.mainloop()