Санкт-Петербургский Политехнический университет имени Петра Великого
Институт компьютерных наук и технологий
Высшая школа программной инженерии

ЛАБОРАТОРНАЯ РАБОТА №2

«НЕЙРОННЫЕ СЕТИ»

по дисциплине «Статистическое моделирование случайных процессов и систем»

Выполнил студент гр. 3530904/70103 Русаков Е.С.

Преподаватель Селин И.

Санкт-Петербург
2020

Оглавление

Задание

1. Постройте нейронную сеть из одного нейрона и обучите её на датасетах nn_0.csv и nn_1.csv. Насколько отличается результат обучения и почему? Сколько потребовалось эпох для обучения? Попробуйте различные функции активации и оптимизаторы.
2. Модифицируйте нейронную сеть из пункта 1 так, чтобы достичь минимальной ошибки на датасете nn_1.csv. Почему были сделаны именно такие изменения?
3. Создайте классификатор на базе нейронной сети для набора данных MNIST (так же можно загрузить с помощью torchvision.datasets.MNIST, tensorflow.keras.datasets.mnist.load_data и пр.). Оцените качество классификации.

Ход работы

1. Нейросеть из одного нейрона

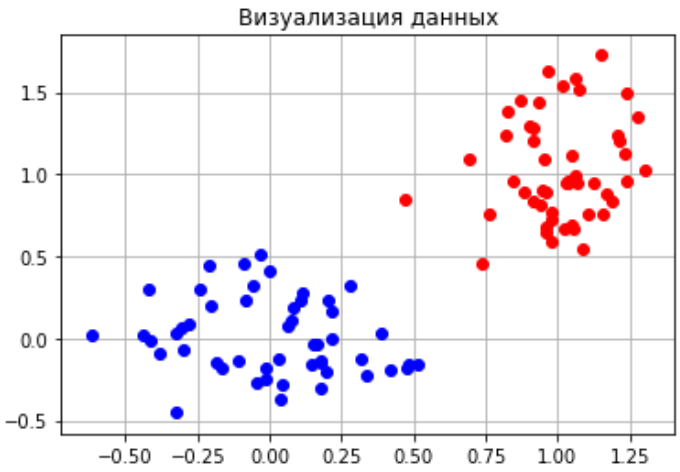
Для построения нейросети используется фреймворк Keras. Обучаться она будет на двух датасетах.

На каждом будут тестироваться несколько типов оптимизаторов и функций активации: SDG, Adagrad, Adadelta, Adam, RMSprop и Softmax, tanh, ReLU, Sigmoid, Softplus.

Каждая модель проходила обучение 50 эпох, результаты которого, а именно достигнутая максимальная точность и требуемое для неё число эпох, занесены в таблицу. Под прочерком в таблице понимается, что на протяжении всего обучения, всех эпох показатель loss-функции не изменялся.

Датасет nn_0.csv

Это сбалансированный датасет. Его визуализация выгладит следующим образом:



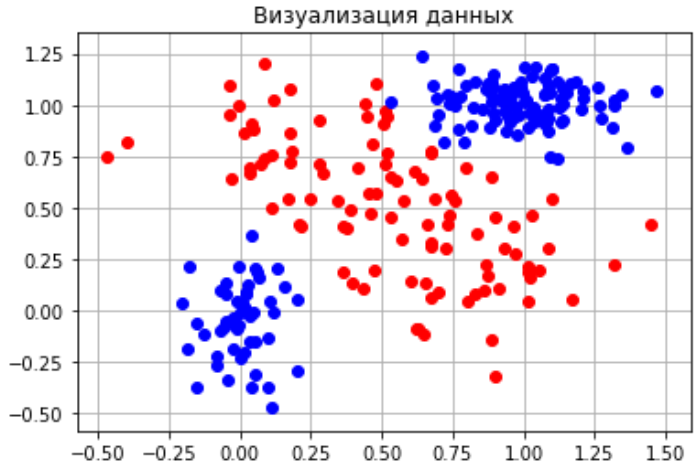
Из него выделены две выборки: обучающая и тестовая в пропорции 80:20.

Оптимизатор	Функция активации	Точность обуч/тест, %	Достигнуто за, эпохи
SGD	Softmax	50/50	–
	tanh	92/90	12
	ReLU	65/70	24
	Sigmoid	95/95	41
	Softplus	98/100	24
Adagrad	Softmax	51/25	–
	tanh	53/50	31
	ReLU	49/55	1
	Sigmoid	63/60	50
	Softplus	79/90	50
Adadelta	Softmax	51/45	–
	tanh	51/45	35

	ReLU	56/60	49
	Sigmoid	90/81	48
	Softplus	96/100	49
Adam	Softmax	51/45	–
	tanh	68/70	50
	ReLU	73/65	50
	Sigmoid	28/30	1
	Softplus	54/05	50
RMSprop	Softmax	50/50	–
	tanh	96/100	48
	ReLU	91/95	50
	Sigmoid	99/100	37
	Softplus	100/100	15

Вывод
Хорошо себя показали функции активации сигмоидальная, гиперболический тангенс и Softplus. Они добиваются большой точности при всех оптимизаторах, хотя для этого может потребоваться много эпох. Функция ReLU показала себя не так хорошо, в нескольких случаях в процессе обучения её точность только падала. Softmax же во всех опытах не демонстрировал никаких улучшений в ни в точности, ни в функции потерь.
Важно привести метки классов к разумительному для фреймворка виду. Например, он не может адекватно обучать нейросети, когда одна из меток классов есть отрицательное число. С этой целью в данном случае используется LabelEncoder() из библиотеки Sklearn.

Датасет nn_1.csv
Начнём с анализа данных.



Во-первых, датасет немного разбалансирован: экземпляров класса «+1» на 50 штук больше, чем экземпляров «-1». В общей сложности классы распределены как 3:2, что не слишком критично.

Во-вторых, по расположению данных можно сразу сказать, что результаты обучения той-же однонейронной модели окажется куда скуднее, чем на предыдущем датасете. Дело в том, что один нейрон, да и вообще все однослойные сети, в двумерном пространстве может разделить данные только прямой линией. В нынешнем же случае простой линейной функции окажется мало, нужна нелинейная, поэтому поменяем модель.

2. Сеть для второго датасета

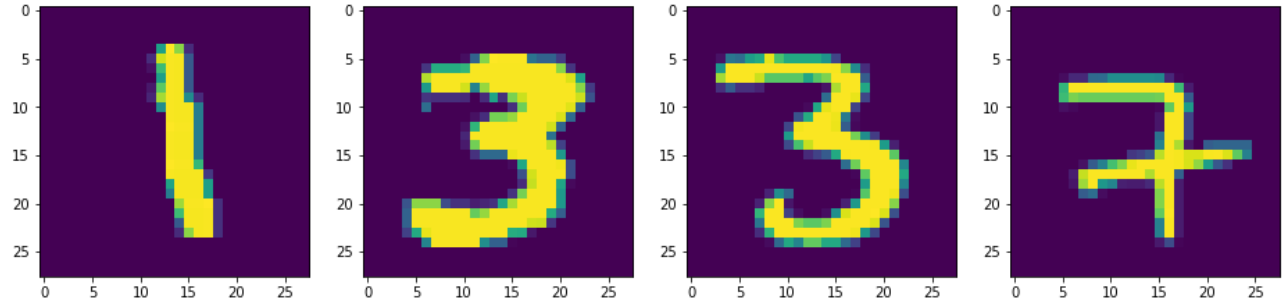
Конструируем новую сеть с несколькими слоями, чтобы обеспечить нелинейность классифицирующей функции:

1. Первый слой будет шириной в три нейрона, второй – в два, третий – в один
2. Функция активации на всех трёх – tanh
3. Оптимизатор – RMSprop

С такой трёхслойной сетью удалось за сто эпох достичь точности 98/98.

3. Классификация MNIST

Датасет для распознавания рукописных цифр MNIST предлагает 60 000 монохромных изображений 28x28 пикселей.



Некоторые из них

Сеть будет иметь следующую архитектуру:

1) Первый слой: 784 нейрона, сигмоидальная функция активации
2) Второй слой: 196 нейронов, сигмоидальная функция активации
3) Третий слой: 10 нейронов, сигмоидальная функция активации
4) Оптимизатор – RMSprop

За 20 эпох удалось достигнуть точности 99,9% на обучающем и 98,4 на тестовом датасетах.
О качестве классификации говорит матрица ошибок. Наш классификатор ошибается в основном только в определении схожих по начертанию цифр.

	0	1	2	3	4	5	6	7	8	9
0	971	0	0	1	1	2	3	0	1	1
1	0	1128	1	0	0	1	2	1	2	0
2	2	0	1018	2	1	0	2	3	4	0
3	0	1	1	997	0	2	0	2	1	6
4	0	1	0	1	951	0	7	2	0	20
5	1	0	0	7	1	880	1	0	1	1
6	3	2	2	0	1	6	943	0	1	0
7	0	4	7	0	1	1	0	1005	4	6
8	2	0	2	4	0	7	1	3	949	6
9	0	3	0	2	2	7	0	1	0	994

Приложение

1. Однонейронная нейросеть

```
# создание тренировочной и тестовой выборок
def prepare_dataset(file_path):
    test_size = 0.2
    pd_data = pd.read_csv(file_path)
    X_all = pd_data.iloc[:, :-1].to_numpy()
    Y_all = LabelEncoder().fit_transform(pd_data.iloc[:, -1].to_numpy())

    X_tr, X_t, Y_tr, Y_t = train_test_split(X_all, Y_all, test_size=test_size,
shuffle=True)

    return X_tr, Y_tr, X_t, Y_t, pd_data

#
def plot_points(data):
    colors = ['red', 'blue', 'green']
    classes = np.unique(data.iloc[:, -1])
    for idx, _ in enumerate(classes):
        plt.scatter(x=data[data.iloc[:, -1] == classes[idx]].iloc[:, 0],
                    y=data[data.iloc[:, -1] == classes[idx]].iloc[:, 1],
                    color=colors[idx])
    plt.title('Визуализация данных')
    plt.grid(True)
    plt.show()

# визуализация данных
dat1 = pd.read_csv('nn_1.csv')
plot_points(dat1)


# оптимизаторы
optims = [ 'sgd', 'adagrad', 'adadelta','adam', 'rmsprop' ]

# функции активации
activations = [ 'softmax', 'tanh', 'relu', 'sigmoid', 'softplus' ]

max_epochs = 50

X, Y,  X_t, Y_t, _ = prepare_dataset('nn_0.csv')

for opt in optims:
    for act in activations:
        print(f'\nOptimizer: {opt}, activation: {act}')
        model_n_0 = Sequential()
        model_n_0.add(Dense(1, input_dim=2, activation=act))
        model_n_0.compile(loss='mse', optimizer=opt, metrics=['accuracy'])

        for ep in range(max_epochs ):
            model_n_0.fit(X, Y, initial_epoch=ep, epochs=ep+1, batch_size=4)
            scores = model_n_0.evaluate(X_t, Y_t)
            print(f'Accuracy: {scores[1]:.3}')
```

2. Сеть для нелинейной классификации

```
max_epochs = 100
```

```
X, Y,  X_t, Y_t, _ = prepare_dataset('nn_1.csv')

model_n_0 = Sequential([
    Dense(3, input_dim=2, activation='tanh'),
    Dense(2, activation='tanh'),
    Dense(1, activation='tanh')
])
model_n_0.compile(loss='mse', optimizer='rmsprop', metrics=['accuracy'])

for ep in range(max_epochs ):
    model_n_0.fit(X, Y, initial_epoch=ep, epochs=ep+1, batch_size=4)
    scores = model_n_0.evaluate(X_t, Y_t)
    print(f'Accuracy: {scores[1]:.3}')
```

3. Классификация рукописных цифр

```
from keras.datasets import mnist
from keras.utils import to_categorical

num_classes = 10
max_epochs = 20


(X, Y), (X_t, Y_t) = mnist.load_data()
num_pix = 28 * 28

# демонстрация примеров датасета
fig, (ax1, ax2, ax3, ax4) = plt.subplots(1, 4, figsize=(16, 4))
ax1.imshow(X[6])
ax2.imshow(X[27])
ax3.imshow(X[12])
ax4.imshow(X[38])
plt.show()

# обработка данных. нормализация
X = X / 255.
X = X.reshape(len(X), num_pix)
X_t = X_t / 255.
X_t = X_t.reshape(len(X_t), num_pix)
# приведение номеров классов в унарную позиционную(?) запись
Y = to_categorical(Y, num_classes)
Y_t = to_categorical(Y_t, num_classes)

mnist_model = Sequential([
    Dense(num_pix, input_shape=(num_pix, ), activation='sigmoid'),
    Dense(196, activation='sigmoid'),
    Dense(10, activation='sigmoid')
])
mnist_model.compile(loss='categorical_crossentropy', optimizer='rmsprop',
metrics=['accuracy'])

mnist_model.summary()

mnist_model.fit(X, Y, batch_size=128, epochs=max_epochs, validation_data=(X_t,
Y_t))

# матрица ошибок
```

```
from sklearn.metrics import confusion_matrix

conf_matr = pd.DataFrame(confusion_matrix(Y_t.argmax(axis=1),
mnist_model.predict(X_t).argmax(axis=1)))
conf_matr
```