ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО»

Институт компьютерных наук и технологий

Высшая школа программной инженерии

ЛАБОРАТОРНАЯ РАБОТА №2

по дисциплине «Машинное обучение»

Студент А. М. Потапова
гр. 3530202/90202

Руководитель И. А. Селин

Санкт-Петербург
2022 г

Содержание

Задание 1 3

Задание 2 7

Задание 3 8

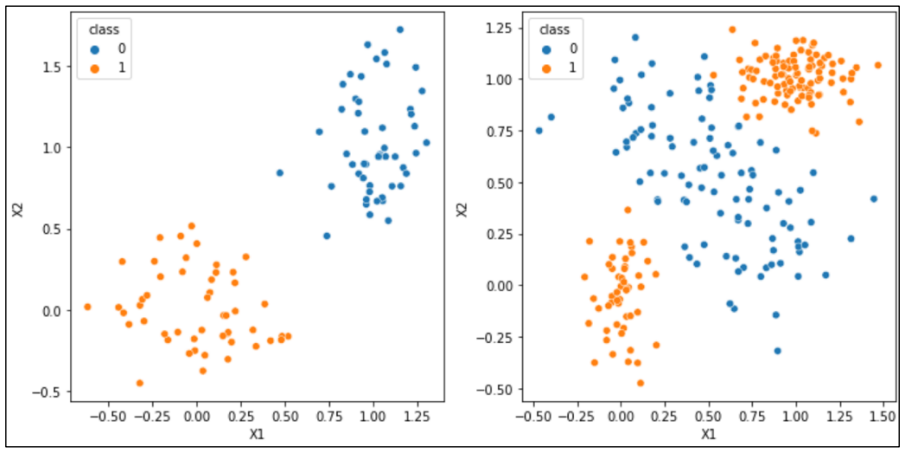
Задание 1

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.

Ход работы

- Получим данные из датасета.

```
[16] data_0 = pd.read_csv('/content/drive/MyDrive/nn_0.csv').replace({'class' : {-1 : 0}})
     X_0 = data_0.iloc[:, 0:-1]
     y_0 = data_0.iloc[:, -1]

[17] data_1 = pd.read_csv('/content/drive/MyDrive/nn_1.csv').replace({'class' : {-1 : 0}})
     X_1 = data_1.iloc[:, 0:-1]
     y_1 = data_1.iloc[:, -1]

[18] _, axes = plt.subplots(1, 2, figsize=(10, 5))
     sns.scatterplot(x='X1', y='X2', data=data_0, hue='class', ax=axes[0])
     sns.scatterplot(x='X1', y='X2', data=data_1, hue='class', ax=axes[1])
     plt.tight_layout()
```



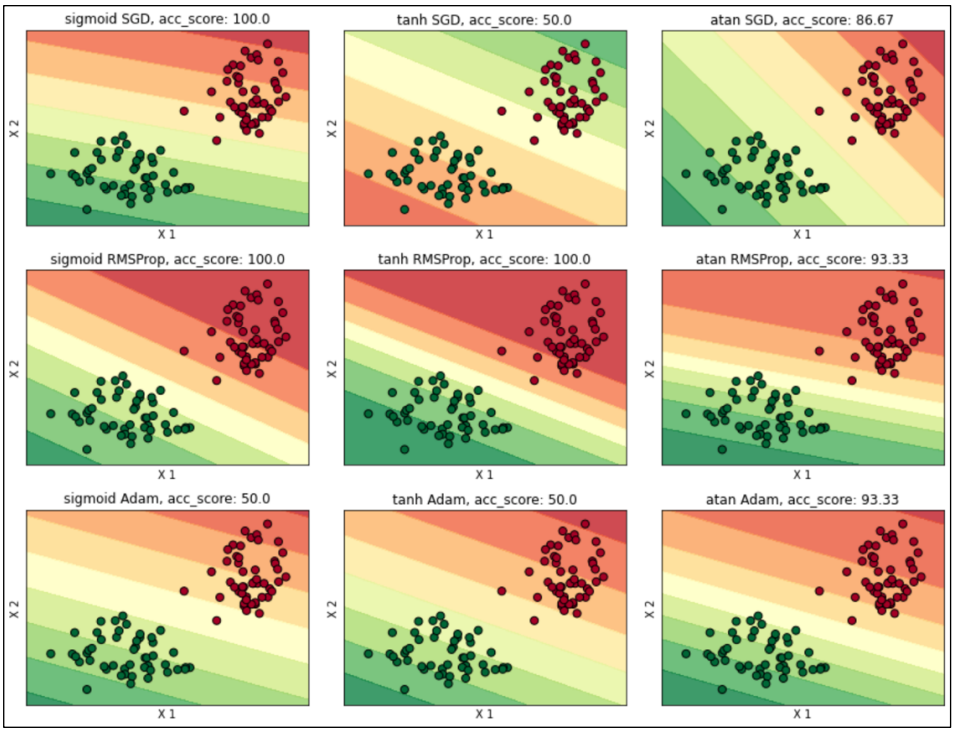
- Построим нейронную сеть из одного нейрона и обучим ее на двух датасетах, указанных выше. В качестве функции активации возьмем функции sigmoid, tanh и atan, а в качестве оптимизаторов SGD, RMSProp и Adam:

```
act_funcs = ['sigmoid', 'tanh', 'atan']
```

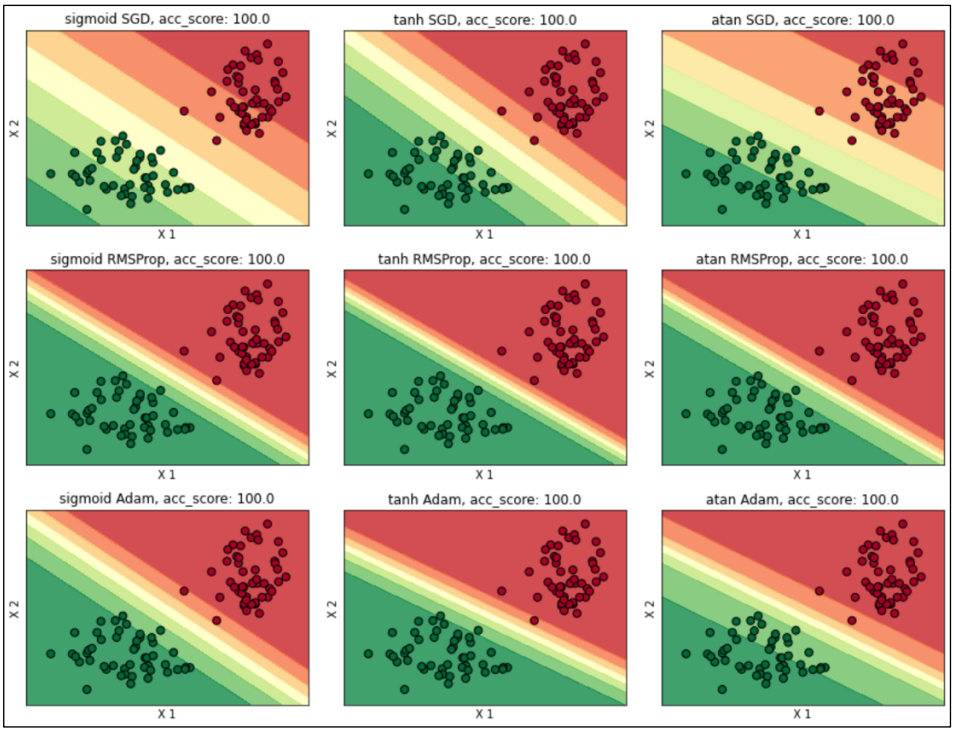
```
optims = ['SGD', 'RMSProp', 'Adam']
```

- Для датасета nn_0.csv были получены следующие результаты (обучающая и тестовая выборки в соотношении 80% на 20%):

Графическое разбиение пространства признаков для различных функций активации и оптимизаторов для 10 эпох (точность сходимости моделей в среднем 80%)



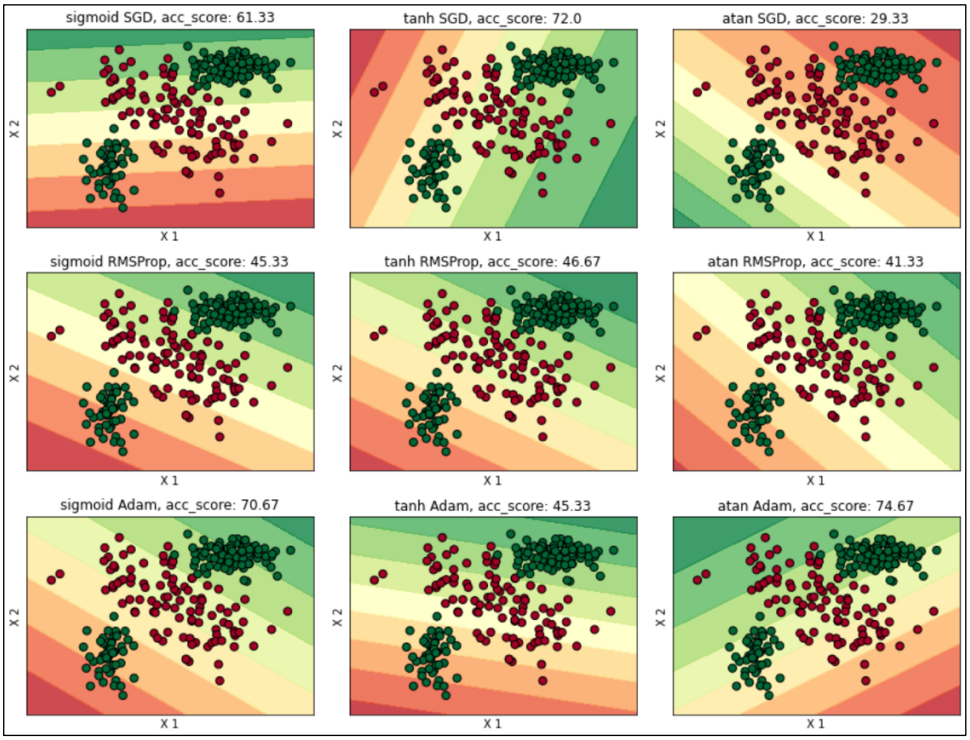
Графическое разбиение пространства признаков для различных функций активации и оптимизаторов для 100 эпох (точность сходимости моделей в среднем 100%)



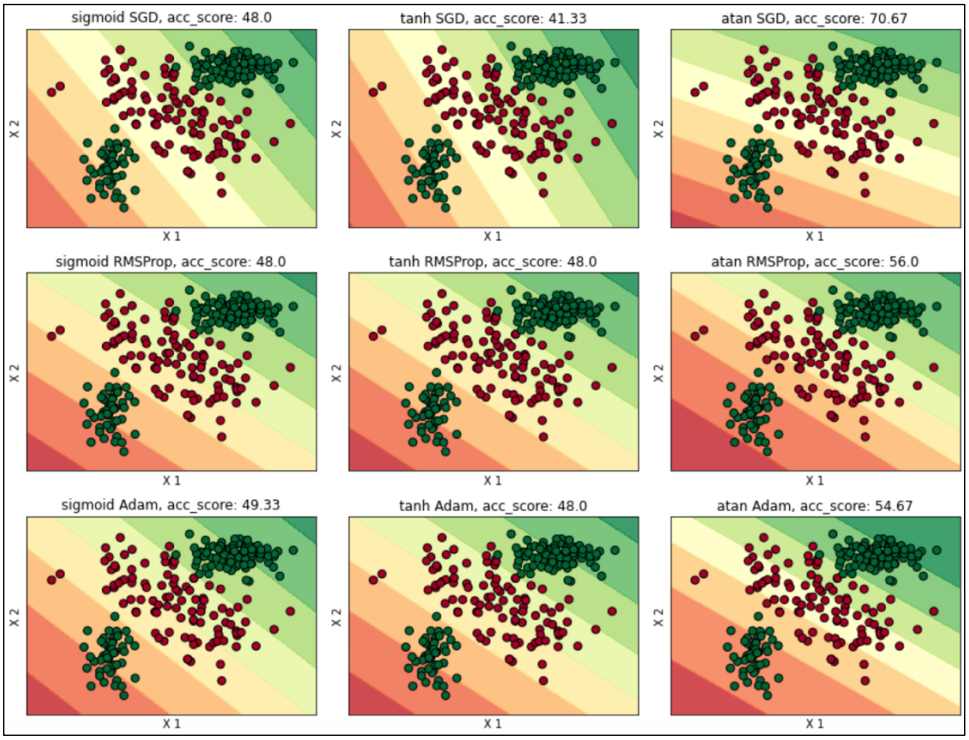
Исходя из полученных результатов можно отметить, что лучшее разбиение происходит при функциях активации tanh и sigmoid с оптимизатором RMSProp.

- Для датасета nn_1.csv были получены следующие результаты (обучающая и тестовая выборки в соотношении 80% на 20%):

Графическое разбиение пространства признаков для различных функций активации и оптимизаторов для 10 эпох (точность сходимости моделей в среднем 54%):



Графическое разбиение пространства признаков для различных функций активации и оптимизаторов для 100 эпох (точность сходимости моделей в среднем 52%):



Точность моделей при данном датасете сильно хуже, чем при предыдущем. Исходя из полученных результатов можно отметить, что лучшее разбиение происходит при функции активации atan с оптимизатором Adam.

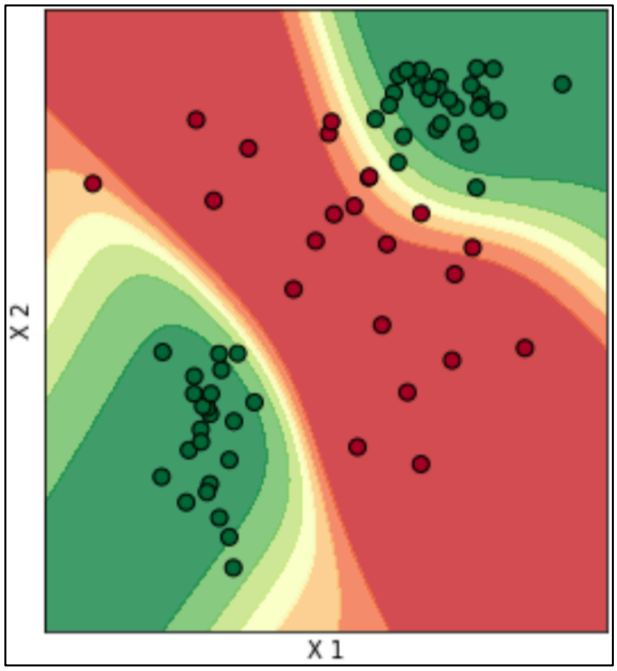
Вывод

Как было упомянуто выше, точность моделей при втором датасете сильно хуже, чтобы понять почему результаты так сильно отличаются, достаточно обратить внимание на изначальные данные. Данные второго датасета (nn_1) невозможно разделить линейно, а первого (nn_0) возможно. Следовательно, для второго датасета следует использовать нейронную сеть с больше, чем с 1 нейроном.

Задание 2

Модифицируйте нейронную сеть из пункта 1, чтобы достичь минимальной ошибки на датасете nn_1.csv. Почему были выбраны именно такие гиперпараметы?

- Увеличим количество нейронов до 10, количество эпох – 100, функция активации – atan, оптимизатор – Adam. Полученный график разбиения пространства:



Вывод

Поскольку нейронная сеть из 1 нейрона не справлялась со своей задачей, я увеличила количество нейронов. После изменений, мы доказали, что результат улучшился, а точнее точность модели увеличилась.

Задание 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.

Используем библиотеку Tensorflor

```
import tensorflow as tf

from tensorflow.keras import datasets, layers, models
import matplotlib.pyplot as plt
```

```
(train_images, train_labels), (test_images, test_labels) = datasets.mnist.load_data()

# Normalize pixel values to be between 0 and 1
train_images, test_images = train_images / 255.0, test_images / 255.0
```

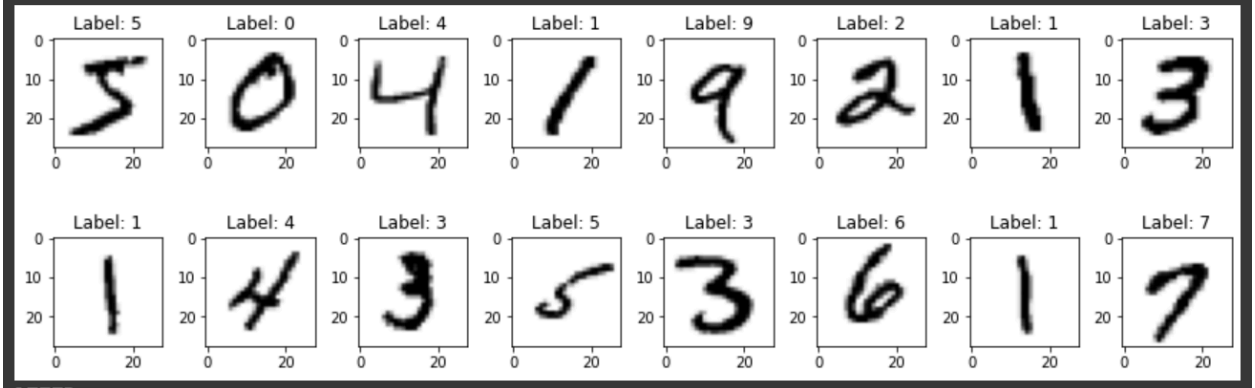
```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 [==============================] - 0s 0us/step
```

Используем Shear (сдвиг) преобразование. Преобразования сдвига изображения выполняются с использованием аргумента shear_range. Преобразование сдвига используется для смещения пикселей в фиксированном направлении на величину, пропорциональную их расстоянию со знаком от линии, параллельной этому направлению и проходящей через начало координат.

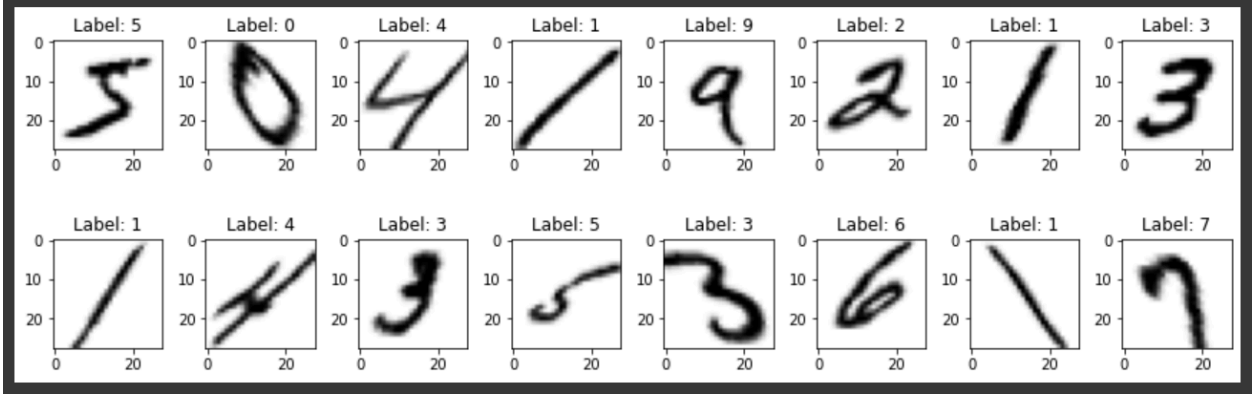
```
shear_range_val=45

from tensorflow.keras.preprocessing.image import ImageDataGenerator
datagen = ImageDataGenerator(shear_range=shear_range_val)
datagen.fit(train_images.reshape(train_images.shape[0], 28, 28, 1))

num_row = 2
num_col = 8
num= num_row*num_col
# plot before
print('BEFORE:\n')
fig1, axes1 = plt.subplots(num_row, num_col, figsize=(1.5*num_col,2*num_row))
for i in range(num):
    ax = axes1[i//num_col, i%num_col]
    ax.imshow(train_images[i], cmap='gray_r')
    ax.set_title('Label: {}'.format(train_labels[i]))
plt.tight_layout()
plt.show()
# plot after
print('AFTER:\n')
fig2, axes2 = plt.subplots(num_row, num_col, figsize=(1.5*num_col,2*num_row))
for X, Y in datagen.flow(train_images.reshape(train_images.shape[0], 28, 28, 1)
                         ,train_labels.reshape(train_labels.shape[0], 1),batch_size=num,shuffle=False):
    for i in range(0, num):
        ax = axes2[i//num_col, i%num_col]
        ax.imshow(X[i].reshape(28,28), cmap='gray_r')
        ax.set_title('Label: {}'.format(int(Y[i])))
    break
plt.tight_layout()
plt.show()
```

Преобразования выглядят следующим образом:

BEFORE:



AFTER:



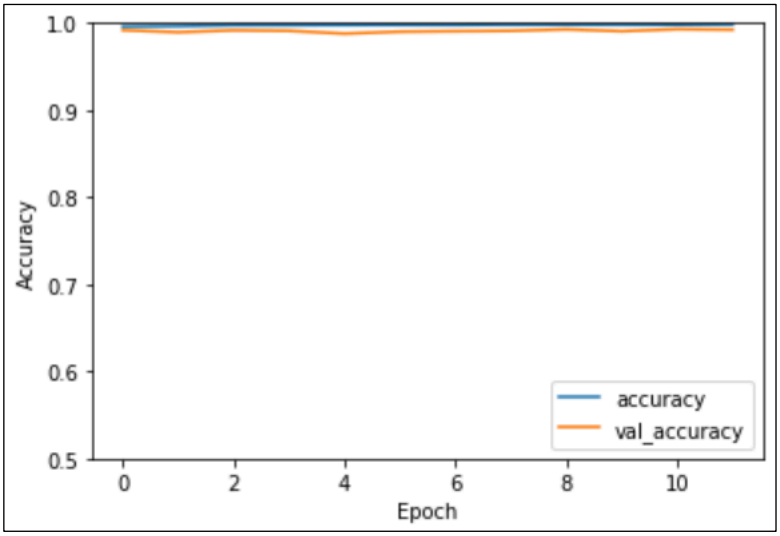
Создадим модель со следующими параметрами:

```
model = models.Sequential()
model.add(layers.Conv2D(32, (3,3) ,activation='relu', input_shape=(28,28,1)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))

model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10))
```

В качестве оптимизатора используем adam с 12 эпохами:

```
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])

history = model.fit(train_images, train_labels, epochs=12,
                    validation_data=(test_images, test_labels))
```

Изменение точности в зависимости от эпохи:



Окончательная точность:

```
Accuracy: 0.9924
```

Вывод

В результате нам удалось создать и обучить нейронную сеть, которая способна классифицировать рукописные цифры с точностью 0,9924.