Министерство образования и науки Российской Федерации

САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО

ПОЛИТЕХ
Санкт-Петербургский
политехнический университет
Петра Великого

**Отчет по Лабораторной работе № 2.**
по дисциплине “Машинное обучение”

Выполнила
студентка гр. 3530202/00201 Козлова Е. А.

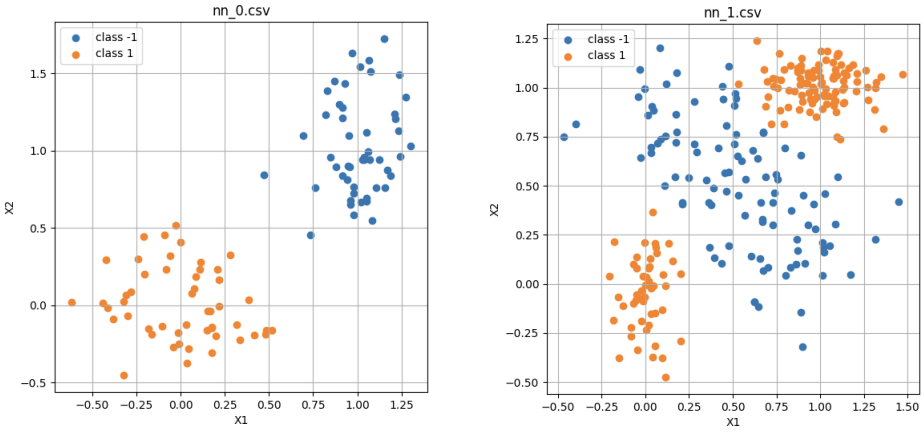
Руководитель Селин И. А.

Санкт-Петербург
2023

# 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.

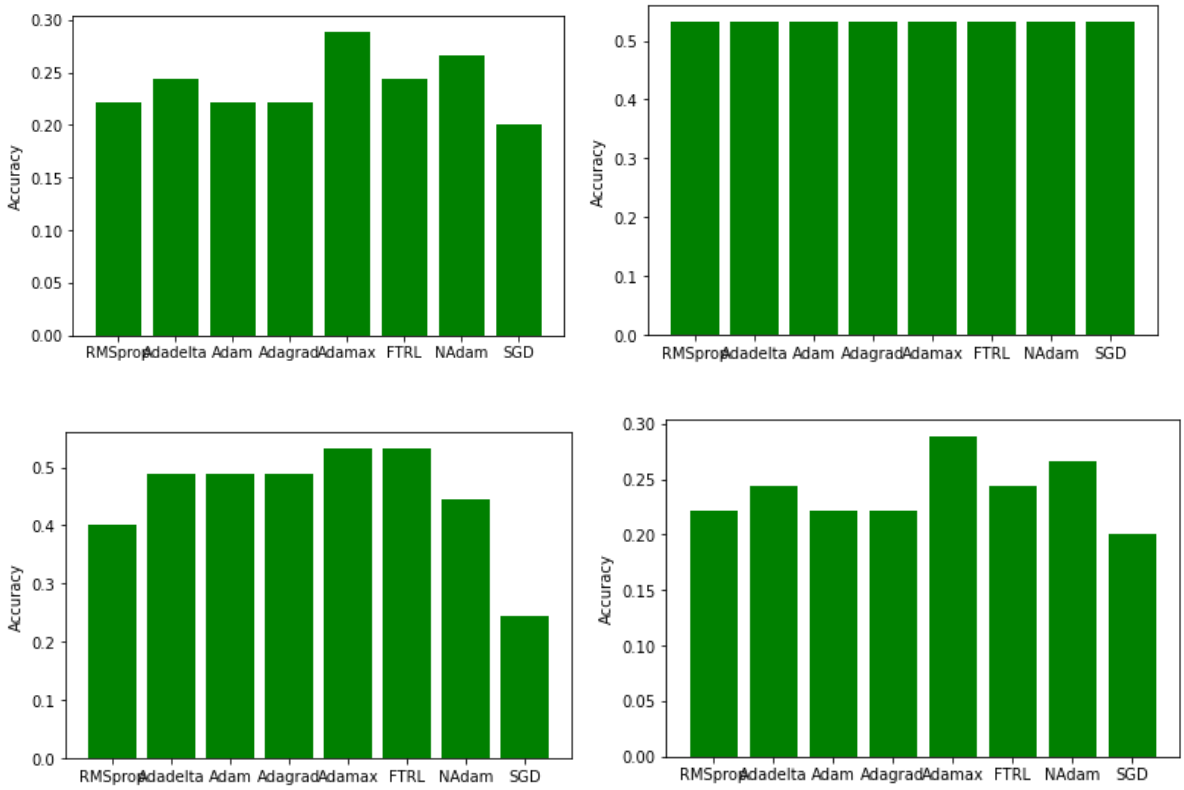



Видно, что в первом наборе оба класса хорошо обособлены друг от друга (линейно разделимы), а во втором наборе классы линейно неразделимы, что позволит более детально оценить возможности нейронных сетей.

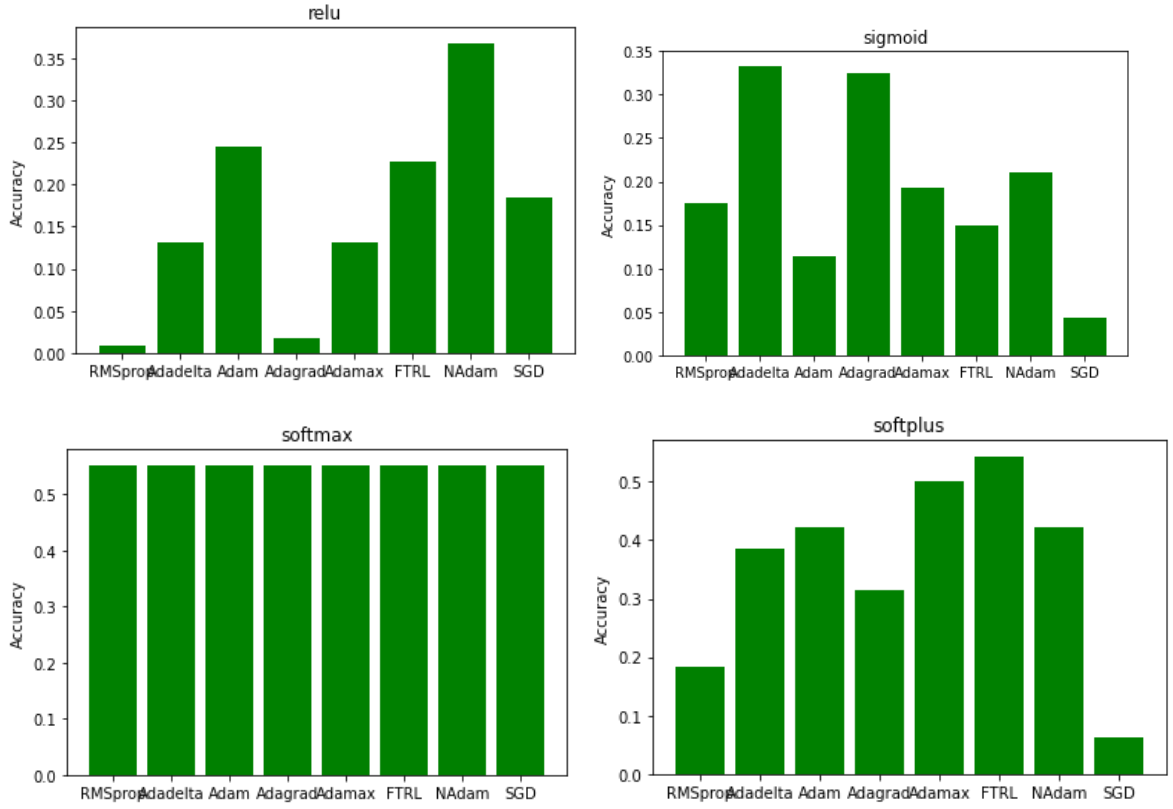
Точность возрастает с количеством эпох, но главное не переучить нашу нейронную сеть, иначе эффект будет обратным.

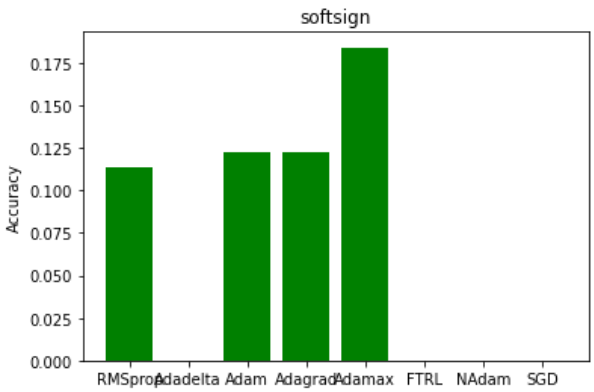
```
activations = ('relu', 'sigmoid', 'softmax', 'softplus', 'softsign',
                   'exponential', 'elu', 'selu', 'tanh')

optimizers = ('RMSprop', 'Adadelta', 'Adam',
                  'Adagrad', 'Adamax', 'FTRL', 'NAdam', 'SGD')

act = {}

for activation in activations:
  opt = {}
  for optimizer in optimizers:
    temp = []
    for _ in range(3):
      tf.keras.backend.clear_session()
      model = keras.Sequential([
          tf.keras.layers.Input(2),
          tf.keras.layers.Dense(1, activation=activation)
      ])

      model.compile(
          optimizer=optimizer,
          loss='binary_crossentropy',
          metrics=['accuracy']
      )

      model.fit(X_train_0, y_train_0, epochs = 10)

      _, test_acc = model.evaluate(X_test_0, y_test_0, verbose=0)
      temp.append(test_acc)

    opt[optimizer] = np.mean(temp)
  act[activation] = opt
```

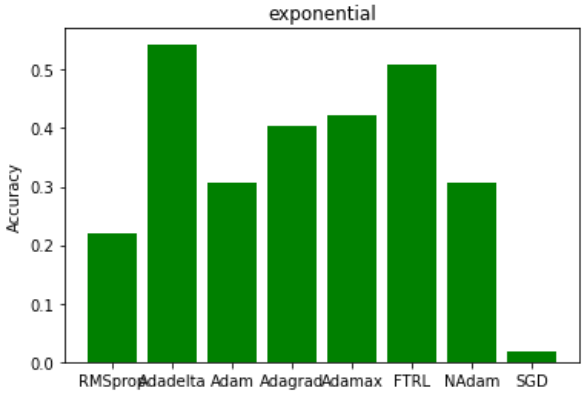
Ниже представленырезультаты точности для разных активаторов и оптимизаторов для датасета n00 (sigmoid, softmax, softplus, exponential):

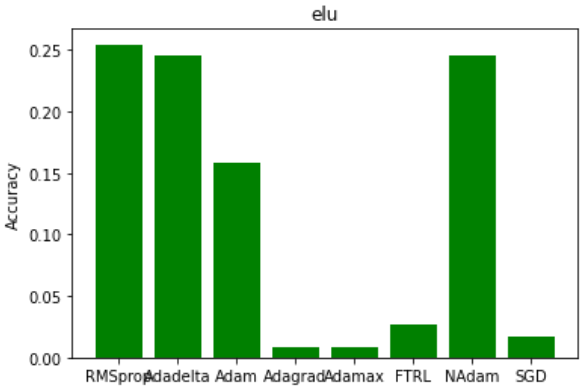


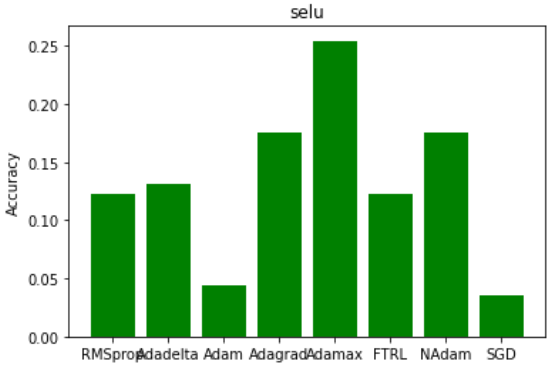

Для датасета nn1:

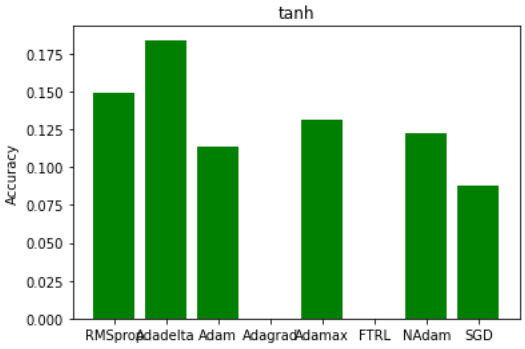



softsign
Accuracy
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
RMSprop
Adadelta
Adam
Adagrad
Adamax
FTRL
NAdam
SGD


exponential
Accuracy
0.5
0.4
0.3
0.2
0.1
0.0
RMSprop
Adadelta
Adam
Adagrad
Adamax
FTRL
NAdam
SGD


elu
Accuracy
0.25
0.20
0.15
0.10
0.05
0.00
RMSprop
Adadelta
Adam
Adagrad
Adamax
FTRL
NAdam
SGD


selu
Accuracy
0.25
0.20
0.15
0.10
0.05
0.00
RMSprop
Adadelta
Adam
Adagrad
Adamax
FTRL
NAdam
SGD


tanh
Accuracy
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
RMSprop
Adadelta
Adam
Adagrad
Adamax
FTRL
NAdam
SGD

## Задание 2

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

Для достижения минимальной ошибки на датасете nn_1.csv, мы внесли изменения в нейронную сеть из первого пункта. Для достижения этой цели мы оставили функции активации и оптимизаторы во входном слое неизменными, но добавили новые функции активации, такие как sigmoid, tanh и relu, в скрытый слой. Это привело к достижению точности в 99.06% и потерь на уровне 5% в результате обучения данной нейронной сети.

## 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.

```
[155] mnist = tf.keras.datasets.mnist

      (x_train, y_train), (x_test, y_test) = mnist.load_data()
      x_train, x_test = x_train / 255.0, x_test / 255.0

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

```
[156] model = tf.keras.models.Sequential([
        tf.keras.layers.Flatten(input_shape=(28, 28)),
        tf.keras.layers.Dense(128, activation='relu'),
        tf.keras.layers.Dropout(0.2),
        tf.keras.layers.Dense(10)
      ])
```

```
[158] predictions = model(x_train[:1]).numpy()
      tf.nn.softmax(predictions).numpy()

      array([[0.05516737, 0.07475657, 0.11931919, 0.05206195, 0.1258966 ,
              0.08169834, 0.07776548, 0.1484546 , 0.18796131, 0.07691851]],
            dtype=float32)
```

```
[159] loss_fn = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
      loss_fn(y_train[:1], predictions).numpy()

      2.5047216
```

```
      model.compile(optimizer='adam',
                    loss=loss_fn,
                    metrics=['accuracy'])
```

```
[161] model.fit(x_train, y_train, epochs=5)

      Epoch 1/5
      1875/1875 [==============================] - 9s 5ms/step - loss: 0.2992 - accuracy: 0.9122
      Epoch 2/5
      1875/1875 [==============================] - 8s 4ms/step - loss: 0.1464 - accuracy: 0.9562
      Epoch 3/5
      1875/1875 [==============================] - 10s 5ms/step - loss: 0.1082 - accuracy: 0.9675
      Epoch 4/5
      1875/1875 [==============================] - 7s 4ms/step - loss: 0.0888 - accuracy: 0.9732
      Epoch 5/5
      1875/1875 [==============================] - 7s 4ms/step - loss: 0.0766 - accuracy: 0.9755
      <keras.callbacks.History at 0x7fa8b6a848d0>
```

```
      test_loss, test_acc = model.evaluate(x_test, y_test, verbose=2)
      print(test_acc)

      313/313 - 1s - loss: 0.0742 - accuracy: 0.9768 - 638ms/epoch - 2ms/step
      0.9768000245094299
```

Полученное значение свидетельствует о высокой точности классификации.