Лабораторная работа 1

Вариант 13

Аминов С.С. М8О-408Б-19

Целью работы является исследование свойств персептрона Розенблатта и его применение для решения задачи распознавания образов.

In [1]:
```
import tensorflow as tf
from tensorflow import keras
from keras import layers
import numpy as np
import matplotlib.pyplot as plt
```

Датасет

In [2]:
```
P = np.array([[0.7, -4.5], [-2.7, -1.5], [3.2, -4.9],[-4.6, -3.4],[1.4, 2.3],[1.4, -0.5]
T = np.array([0, 1, 0, 1, 1, 0])
```

Создаём модель

In [3]:
```
model = keras.Sequential(
    layers.Dense(1, input_dim=2, activation='sigmoid')
)
```

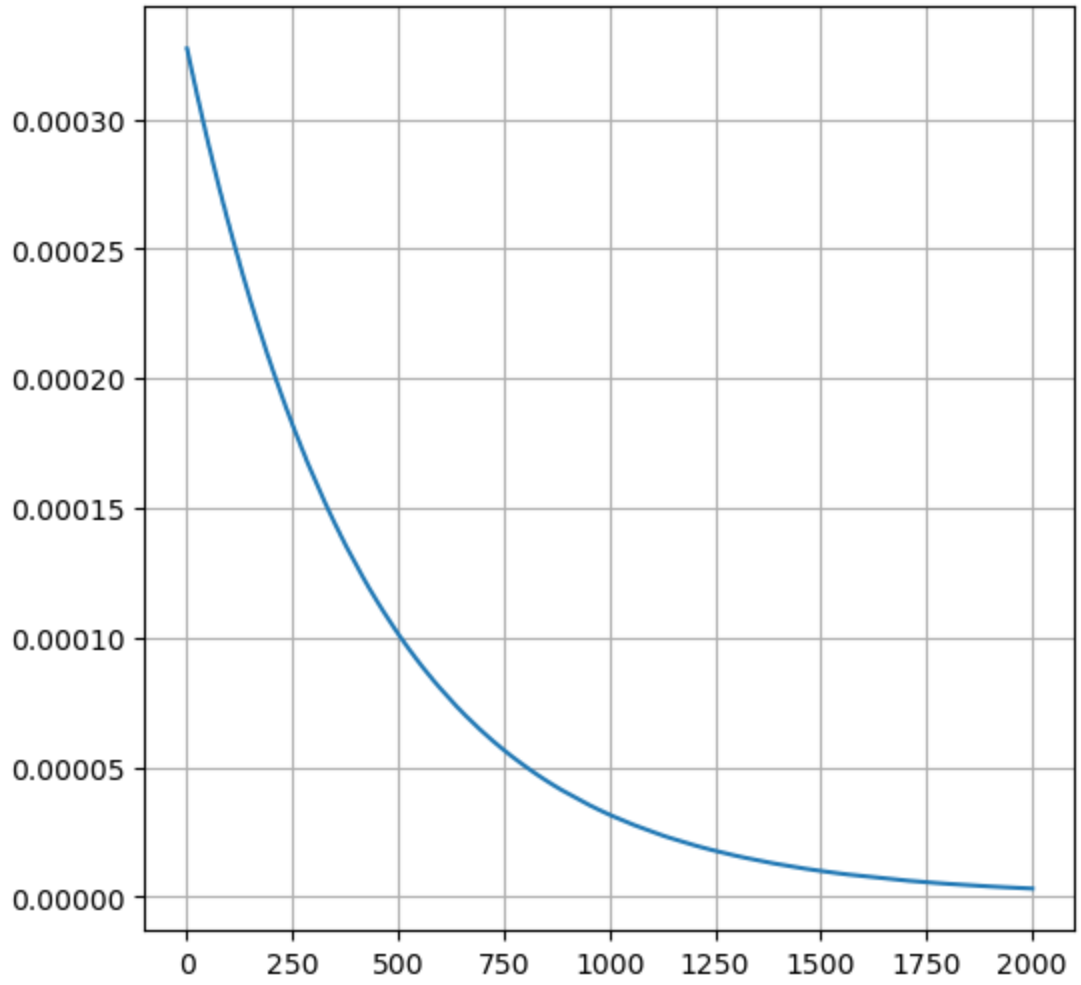
Компилируем

In [4]:
```
model.compile(loss='mse', optimizer='adam', metrics=['mae'])
```

Тренеруем

In [7]:
```
def plot_loss(train):
    plt.figure(figsize=(6, 6))
    loss_history = train.history['loss']
    plt.plot(range(1, len(loss_history) + 1), loss_history)
    plt.grid()
    plt.show()
```

In [23]:
```
plot_loss(train)
```



Получаем веса и строим дискриминантную линию

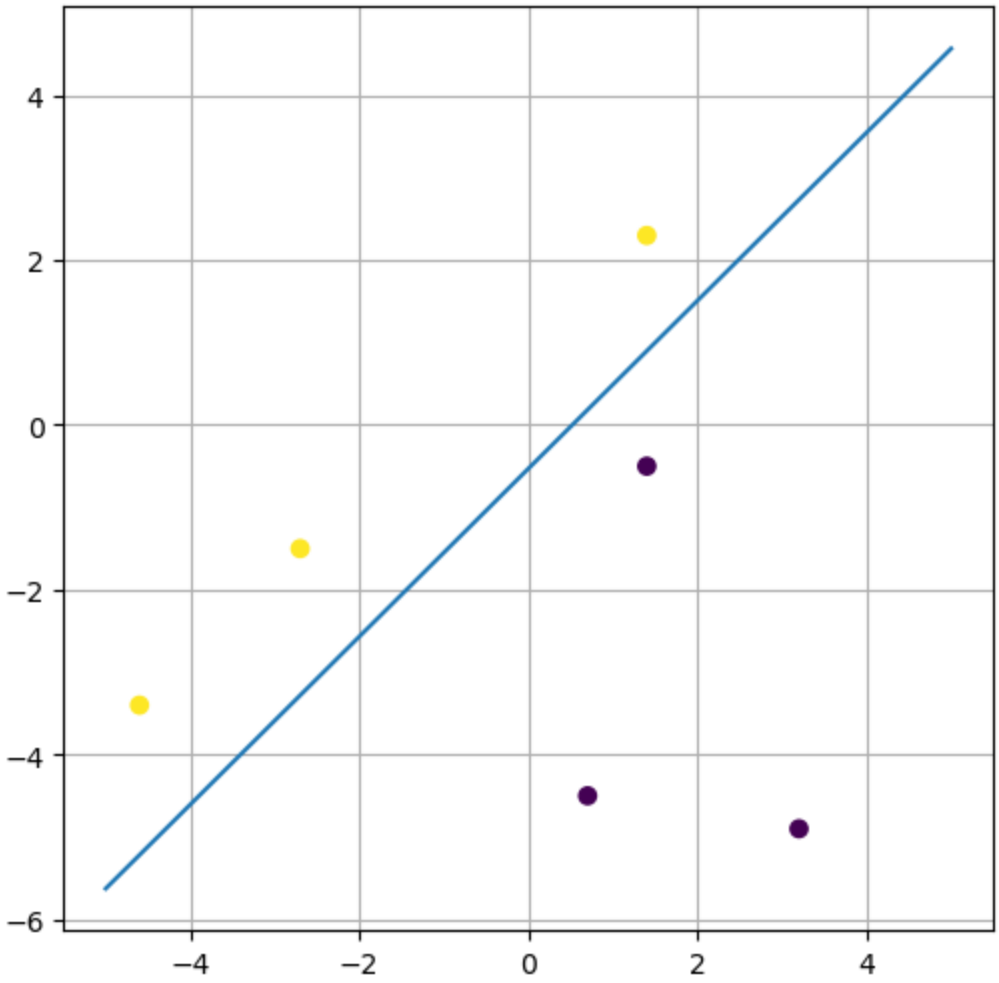
In [19]:
```
plt.figure(figsize=(6,6))

plt.scatter(P[:, 0], P[:, 1], c=T)

A, b = model.layers[0].get_weights()

x_disc = np.linspace(-5, 5, 5)
plt.plot(x_disc, (-A[0] * x_disc - b) / A[1])

plt.grid()
plt.show()
```



Часть 2

Датасет

In [12]:
```
P_2 = np.array([[0.6, -2.4], [2.4, 0], [1.4, -2],[-3.7, -0.3],[1.4, 2.8],[2.8, 1.6],[-3.
T_2 = np.array([[1, 0],[1, 1],[1, 0],[0, 0],[0, 1],[1, 1], [1, 0],[1, 0]])
```

Создаём модель

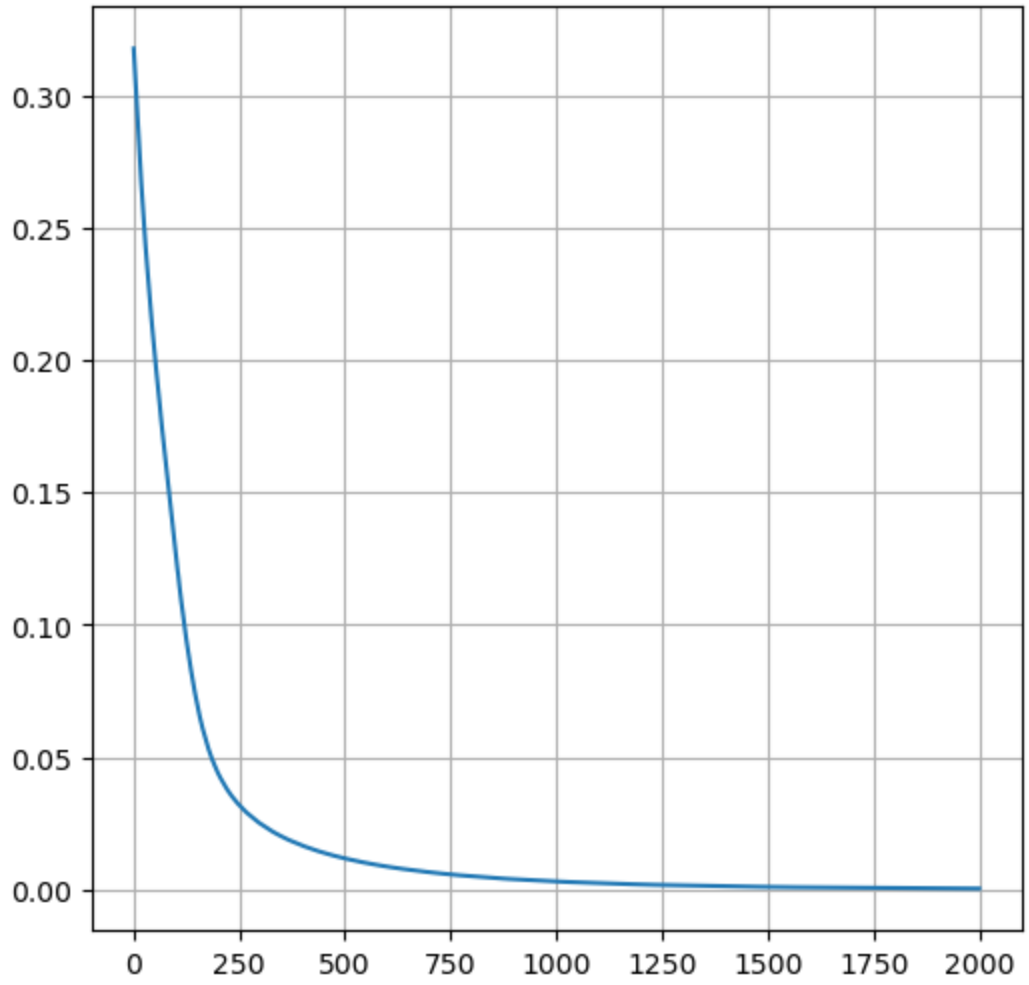
In [13]:
```
model_2 = keras.Sequential(
    layers.Dense(2, input_dim=2, activation='sigmoid')
)
```

Компилируем

In [14]:
```
model_2.compile(loss='mse', optimizer='adam', metrics=['mae'])
```

Тренеруем

In [19]:
```
plot_loss(train)
```



Получаем веса и строим дискриминантную линию

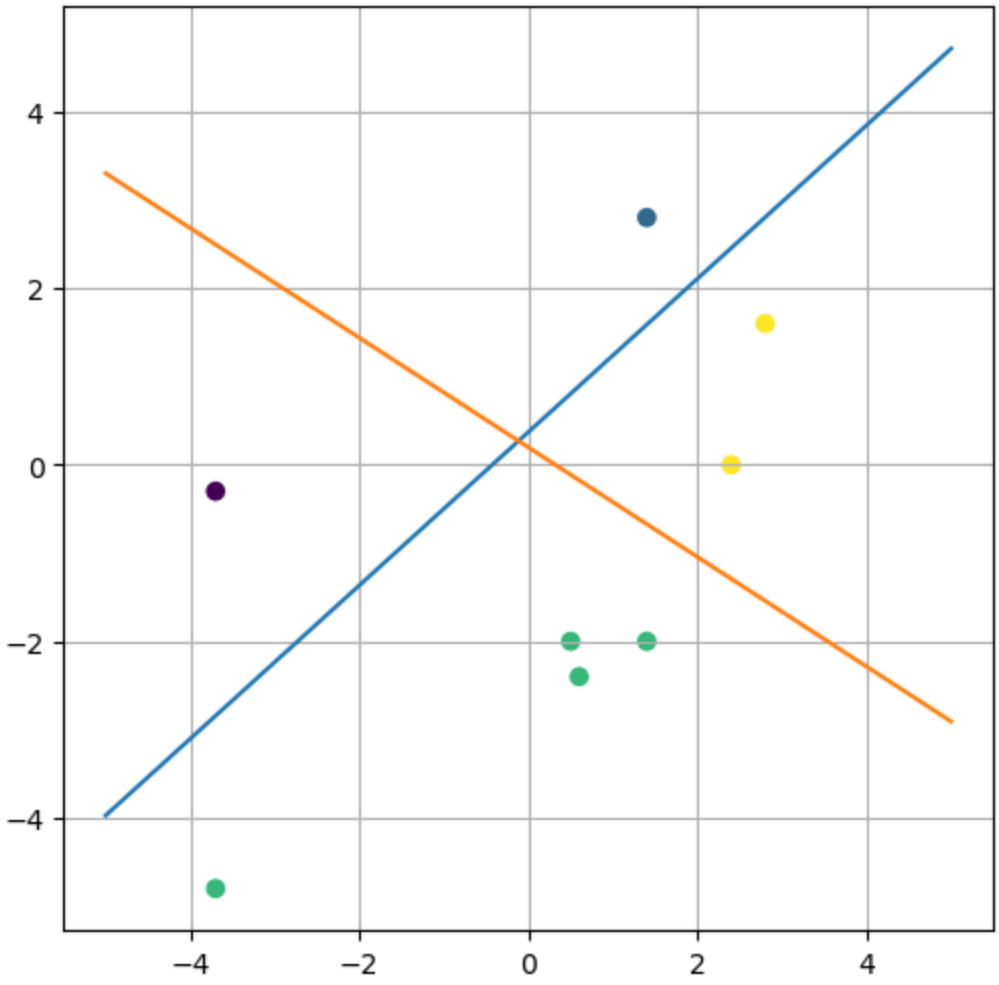
In [18]:
```
plt.figure(figsize=(6,6))

plt.scatter(P_2[:, 0], P_2[:, 1], c=[int(str(i*10 + j), 2) for i, j in T_2])

A, b = model_2.layers[0].get_weights()

x_disc = np.linspace(-5, 5, 5)
plt.plot(x_disc, (-A[0][0] * x_disc - b[0]) / A[1][0])
plt.plot(x_disc, (-A[0][1] * x_disc - b[1]) / A[1][1])

plt.grid()
plt.show()
```



Вывод

В данной работе я решил задачу классификации на 2 и 4 класса. Обученный перцептрон довольно неплохо справляется со своей задачей, хотя мы, конечно, не можем гарантировать стопроцентный результат для настолько небольших датасетов при обучении.