

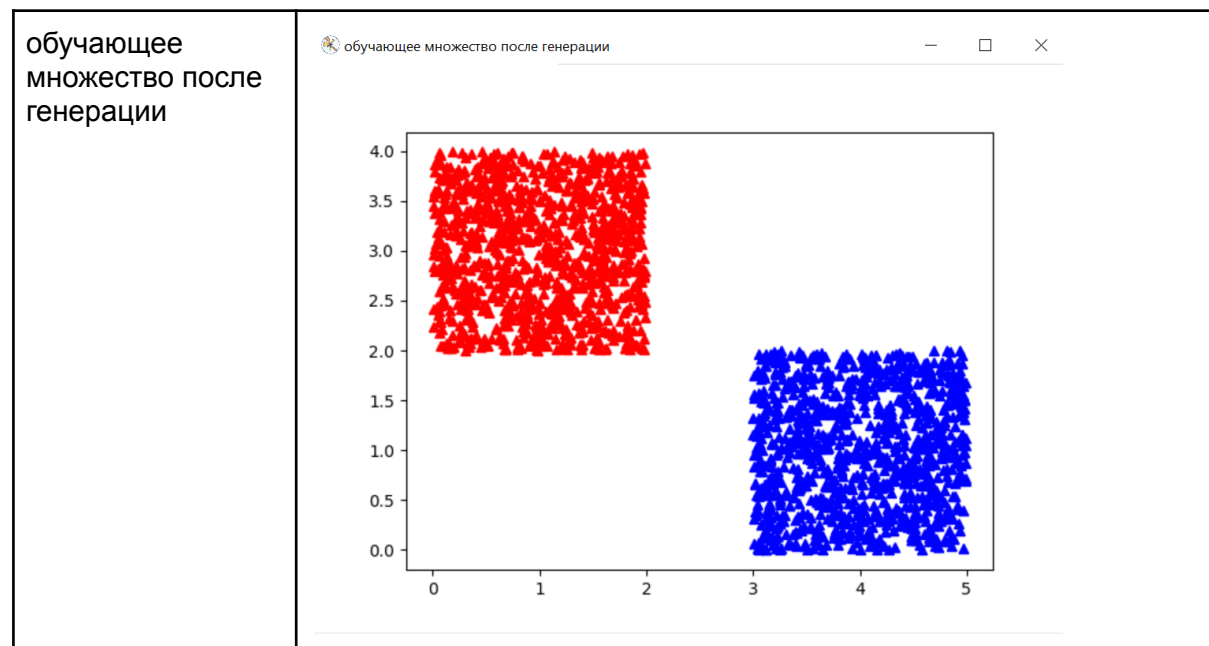
Задача. Обучить нейронную сеть с одним нейроном и последующей функцией активации разделять два непересекающихся множества. Обучение выполнить с использованием смещения и без него.

Последовательность решения задачи:

1. Генерация набора данных.
2. Подготовка обучающих и проверочных множеств.
3. Отображение набора данных, обучающих и проверочных множеств на рисунках.
4. Формирование и обучение модели НС.
5. Прогноз модели на проверочном множестве.
6. Вывод весов обученного нейрона и смещения (при наличии последнего).
7. Оформление отчета.

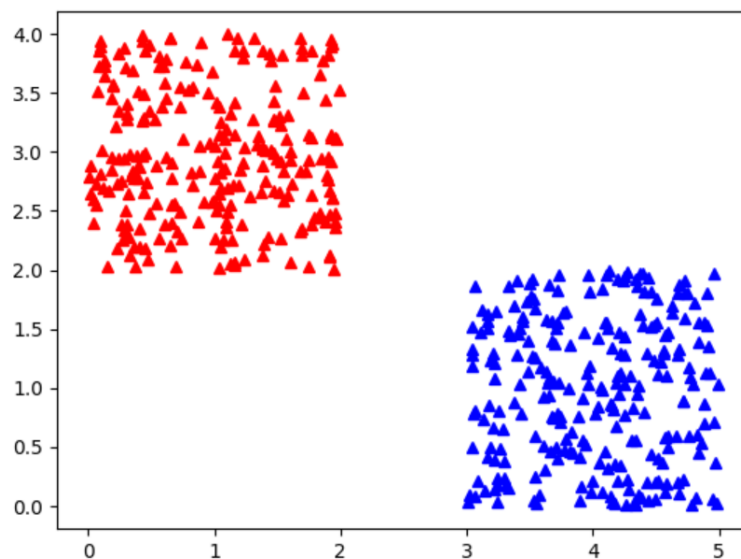
Решение задачи

1. Генерация набора данных.



проверочное
множество после
генерации

тестовое множество после генерации



2. Подготовка обучающих и проверочных множеств.

```
def generateData( n, x1, x2, y1, y2, myclass):
    set = np.zeros(shape=(n, 2), dtype=float)
    for i in range(n):
        set[i][0] = random.uniform(x1, x2)
        set[i][1] = random.uniform(y1, y2)
    y = np.array([myclass] * n)
    return set, y

def generateDataSet(bufn1, bufn2):
    set1, y1 = generateData(bufn1, 0, 2, 2, 4, myclass1)
    set2, y2 = generateData(bufn2, 3, 5, 0, 2, myclass2)
    set = np.concatenate((set1, set2))
    y = np.concatenate((y1, y2))
    return set, y

def Standart(set):
    scaler = StandardScaler(copy=False).fit(set) # fit- вычисляет
    # среднее значение и стандартное отклонение (для стандартизации)
    set = scaler.transform(set) # Стандартизация за счет центрирования и
    # масштабирования
    return set

set, y = generateDataSet(n1, n2)
```

```

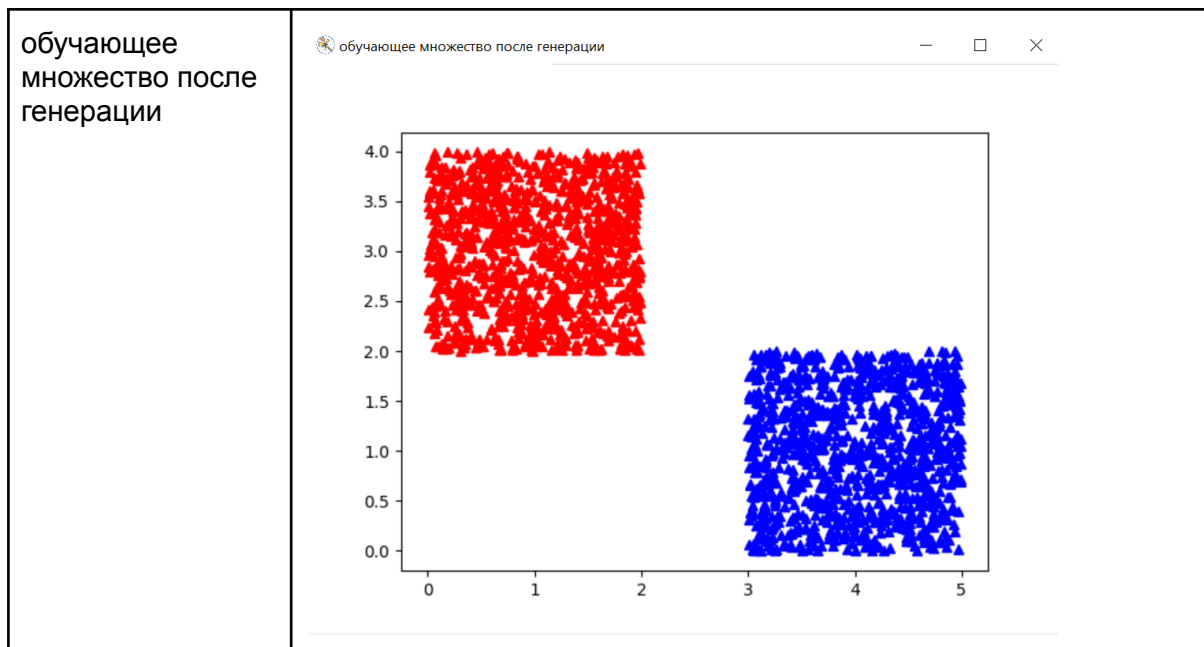
#PaintSET(set, n1, n2, "обучающее множество после генерации")
num_elem_set = len (set[:, 0])
print("количество данных в обучающем множестве ", num_elem_set )
set = Standart(set)
#PaintSET(set, n1, n2, "обучающее множество после стандартизации")

n1_test = int(1/4 * n1)
n2_test = int(1/4 * n2)
set_test, y_test = generateDataSet(n1_test, n2_test)
#PaintSET(set_test, n1_test, n2_test, "тестовое множество после
генерации")
set_test = Standart(set_test)
#PaintSET(set_test, n1_test, n2_test, "тестовое множество после
стандартизации")
print ("количество данных в тестовом множестве", len (set_test[:, 0]))

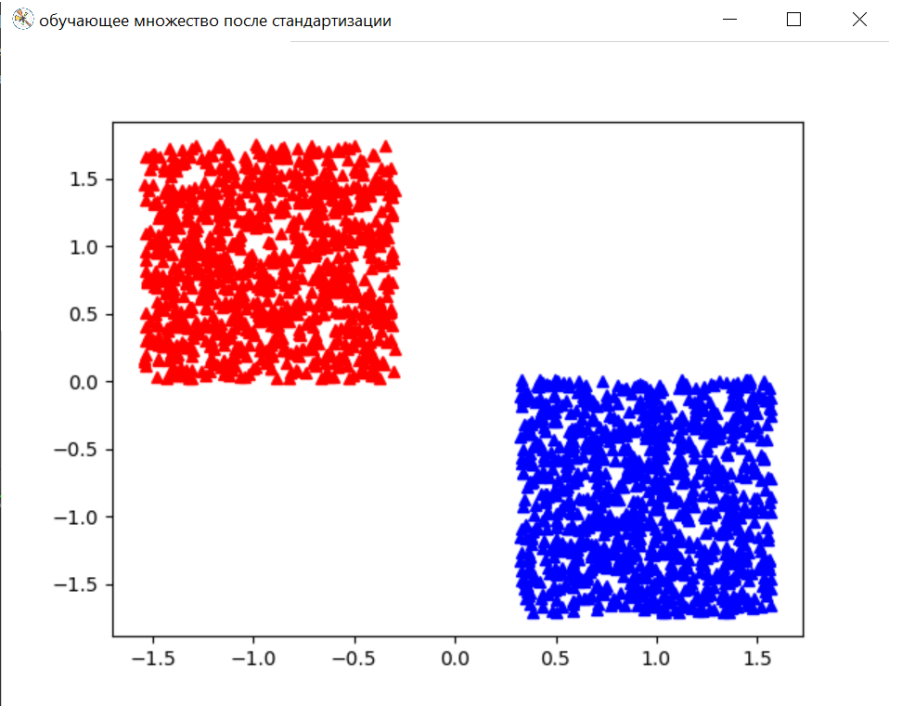
x_train, y_train = set, y
x_test, y_test = set_test, y_test

```

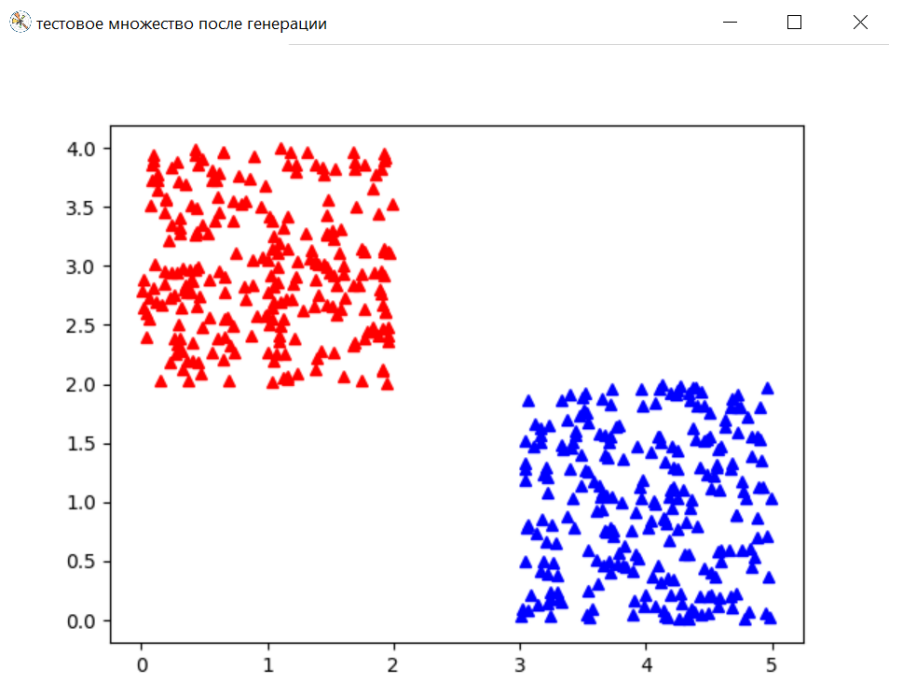
3. Отображение набора данных, обучающих и проверочных множеств на рисунках.



обучающее
множество после
стандартизации

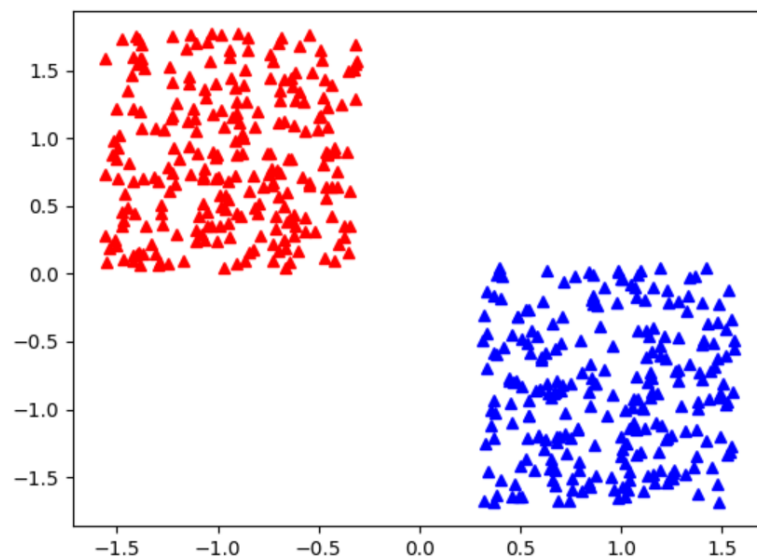


проверочное
множество после
генерации



проверочное
множество после
стандартизации

тестовое множество после стандартизации



4. Формирование и обучение модели НС.

```
K.clear_session()
#isBiasUse = False
isBiasUse = True
size = 2
x_train = x_train.reshape(-1, 2)
x_test = x_test.reshape(-1, 2)
input_shape = (2) # 784

model1 = keras.models.Sequential()
model1.add(keras.layers.Dense(1, input_dim=2, activation='sigmoid',
use_bias=isBiasUse))
print(model1.summary())
model1.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])

start = time.time()
history = model1.fit(
x_train,
y_train,
batch_size=32,
epochs=numberepochs,
validation_data=(x_train.reshape(-1, 2 * 1), y_train)

)
```

```
end = time.time()
print("время обучения: ", end - start)
```

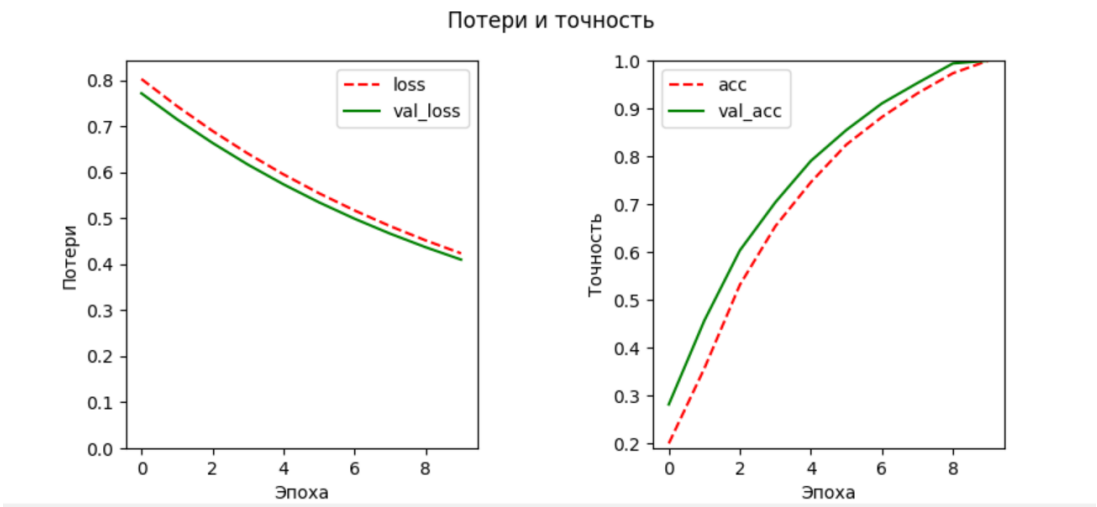
```
Model: "sequential"
-----
Layer (type)                Output Shape              Param #
-----
dense (Dense)               (None, 1)                 3
-----
Total params: 3
Trainable params: 3
Non-trainable params: 0
-----
```

Свойства\вариант	1	2	3	4
символьное описание НС	DS1	DS1	DS1	DS1
имя оптимизатора	adam	adam	adam	adam
имя функции потерь	'binary_crossentropy'	'binary_crossentropy'	'binary_crossentropy'	'binary_crossentropy'
число эпох	10	100	100	100
размер обучающего пакета	32	32	32	32
время обучения	1.343749761581421	7.393176078796387	7.12186336517334	7.2829461097717285
точность на обучающем множестве	1.0000	1.0000	1.0000	1.0000
точность на оценочном множестве	1.0000	1.0000	1.0000	1.0000
функция активации	sigmoid	sigmoid	sigmoid	sigmoid

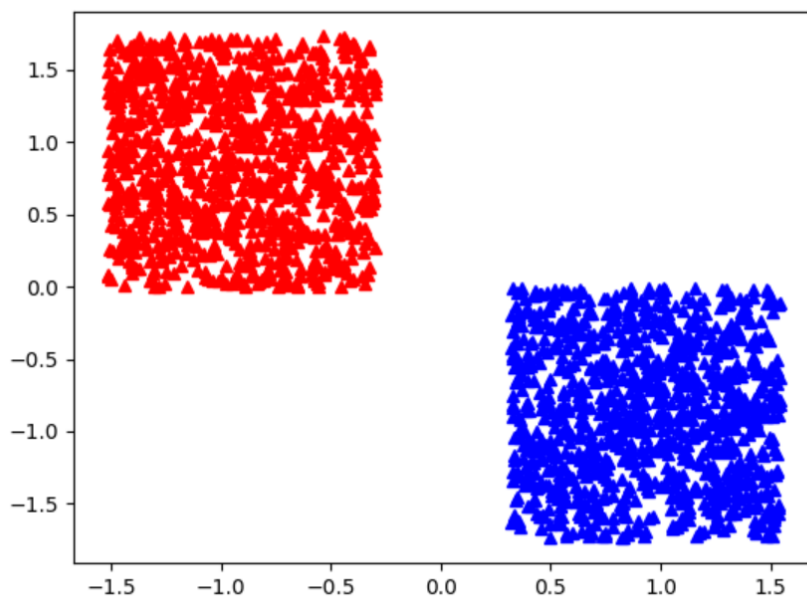
use_bias	True		False	
Использовалась стандартизация	да	нет	да	нет
веса	bias: -0.0020637563 w1: 0.039312556 w2: -0.79847044	bias: 0.6833514 w1: 1.7054116 w2: -2.4504328	w1: 2.7537584 w2: -2.566365	w1: 1.59587 w2: -1.8247474

Графики:

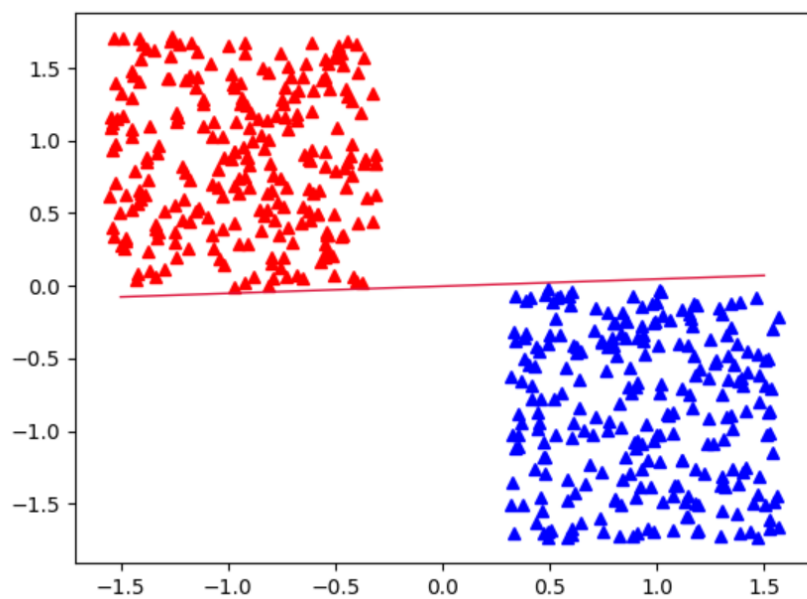
1) вариант 1: с использованием стандартизации и смещения



Обучающее множество:



Прогноз на проверочном множестве:



Гиперплоскость — это подпространство с размерностью, на единицу меньшей, чем объемлющее пространство. Так как у нас двумерное пространство, то гиперплоскостью является прямая. На графике она нарисована розовым цветом.

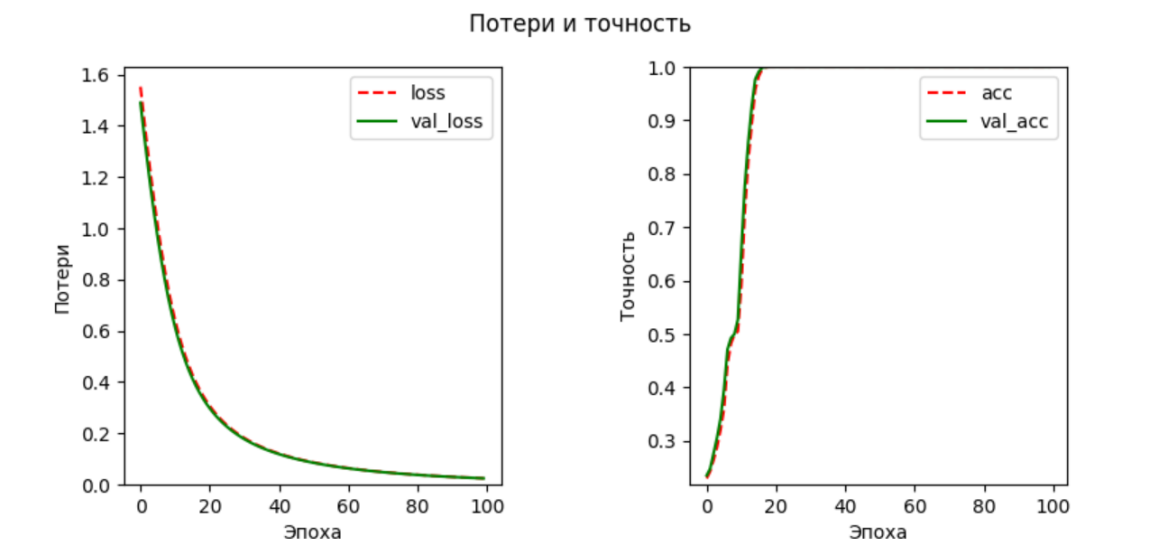
Уравнение прямой в общем виде $Ax + By + C = 0$

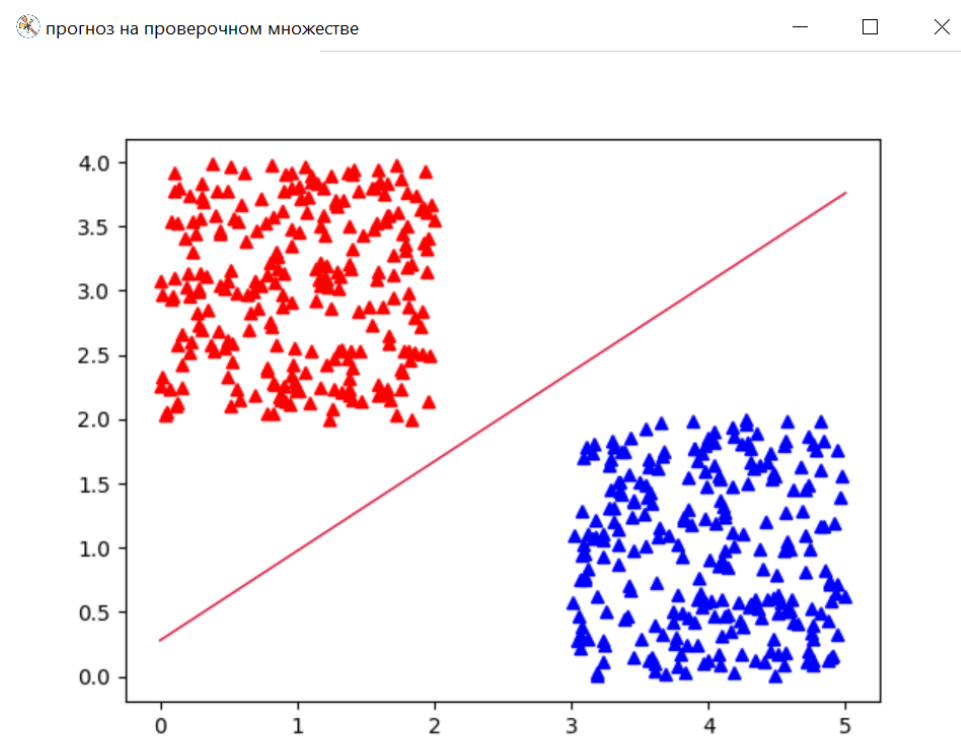
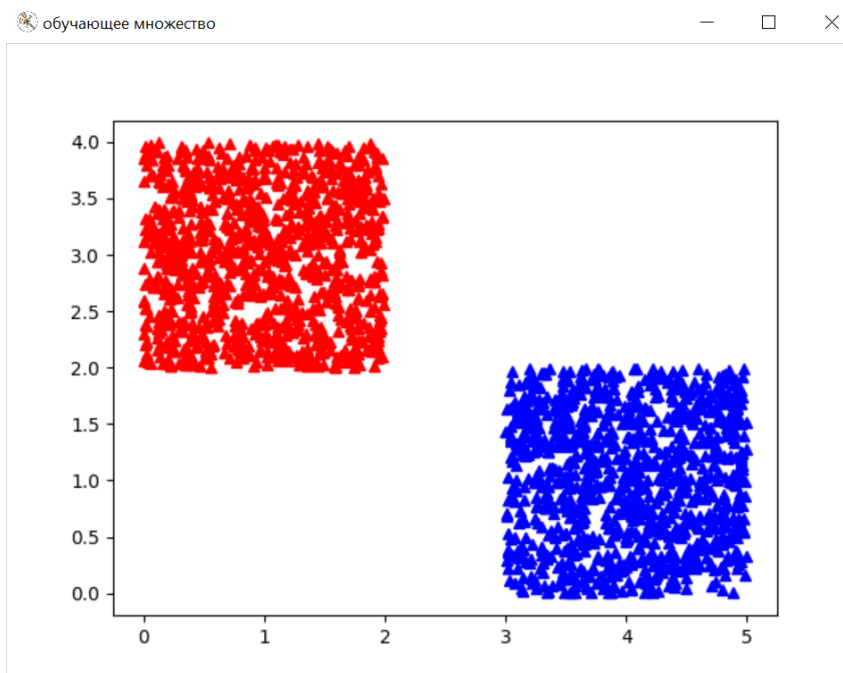
В нашей задаче $w_1 \cdot x + w_2 \cdot y + \text{bias} = 0$, где w_1, w_2 - веса

Гиперплоскость делит гиперпространство на два гиперподпространства. Точки, находящиеся в одном из этих подпространств (условно говоря «выше» гиперплоскости), и точки, находящиеся в другом из этих подпространств (условно говоря «ниже» гиперплоскости), будут в сумме $w_1 * x + w_2 * y + bias$ давать разный знак. То есть точки разных множеств (множеств 1 и 2), будут лежать по разные стороны от гиперплоскости.

2) вариант 2: с использованием смещения , но без стандартизации.

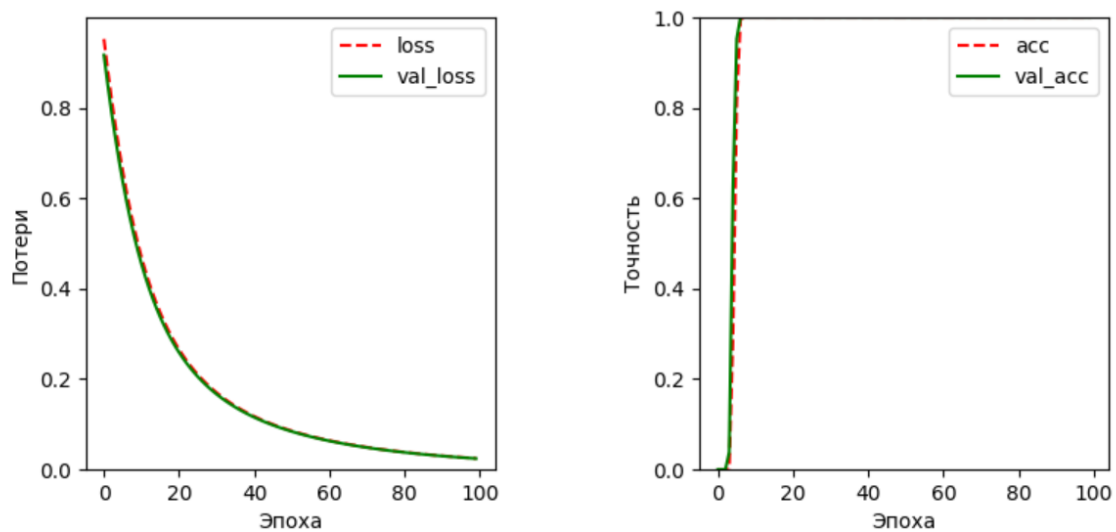
В этом варианте, при 10 эпохах точность не так высока в сравнении с 1 вариантом, поэтому произвели увеличение числа эпох до 100. Время обучения тоже увеличилось.






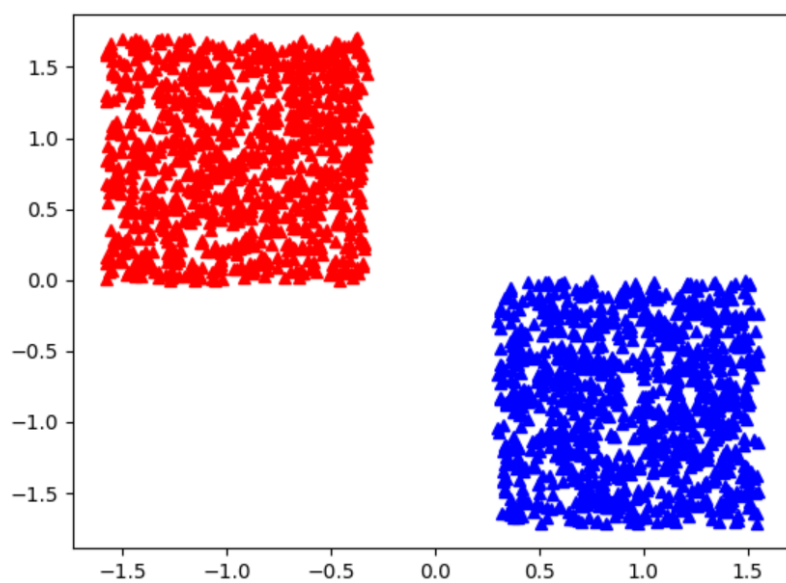
- 3) вариант 3: без использования смещения, но со стандартизацией.
- В этом варианте, при 10 эпохах точность не так высока в сравнении с 1 вариантом, поэтому произвели увеличение числа эпох до 100.
- Время обучения тоже увеличилось, оно сопоставимо со временем обучения во 2 варианте.

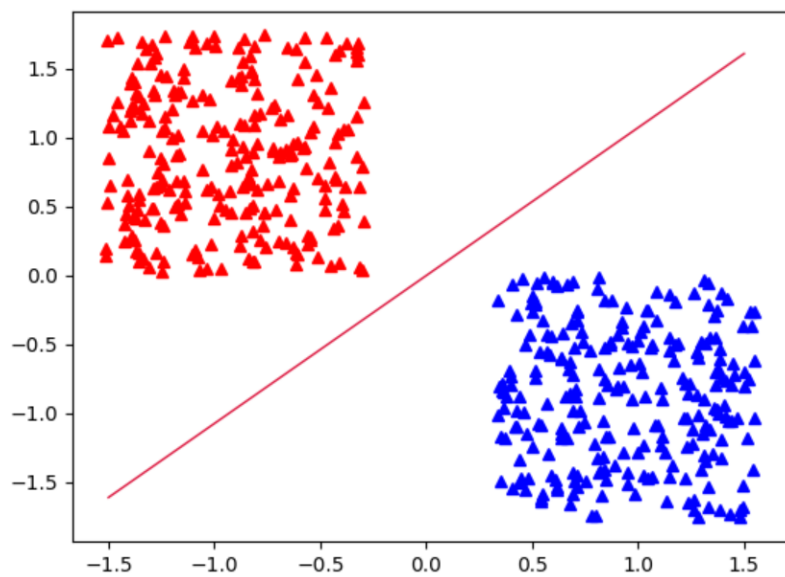
Потери и точность



 обучающее множество

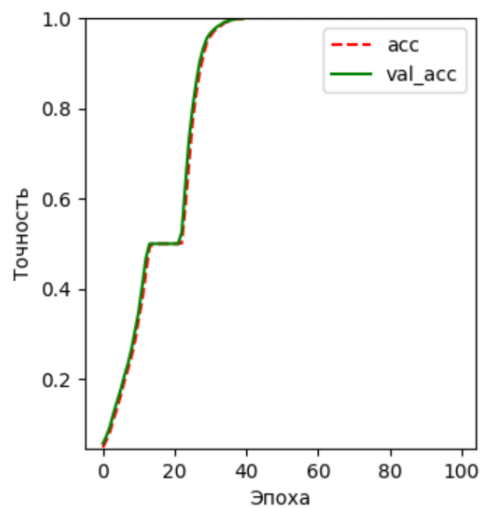
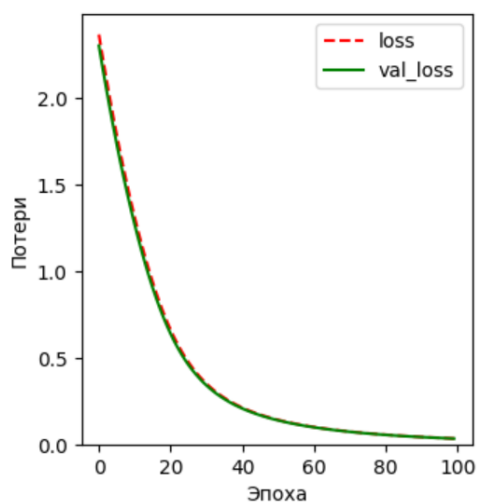
— □ ×





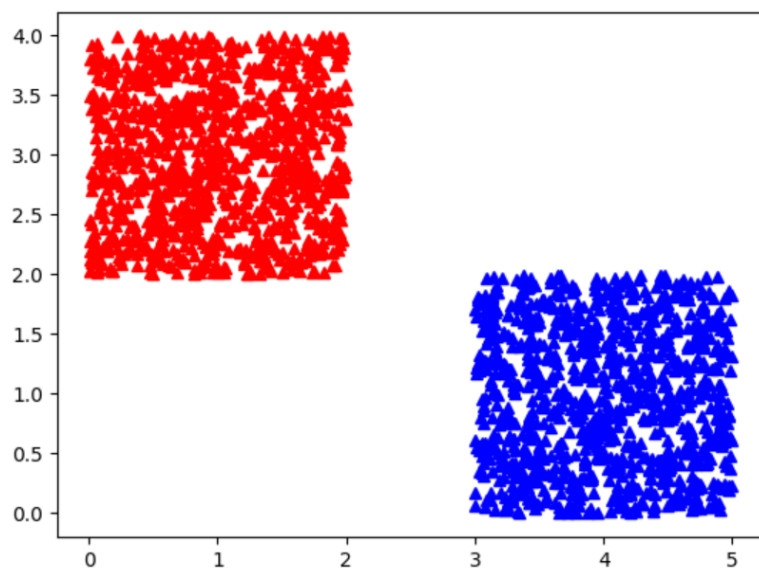
4)

Потери и точность



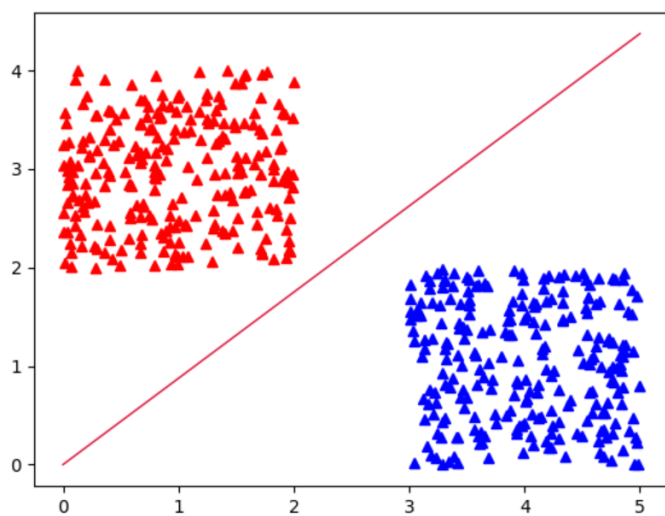
обучающее множество

— □ ×



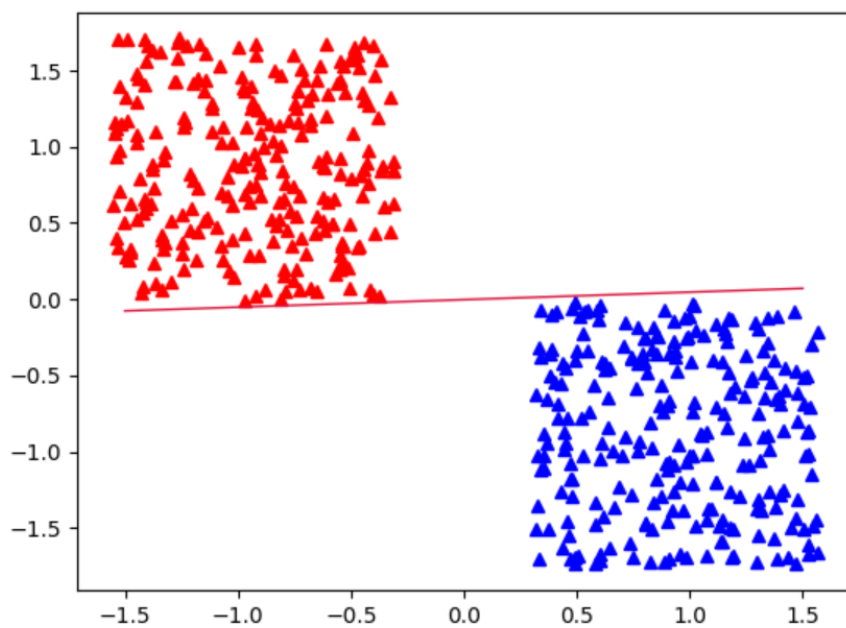
прогноз на проверочном множестве

— □ ×



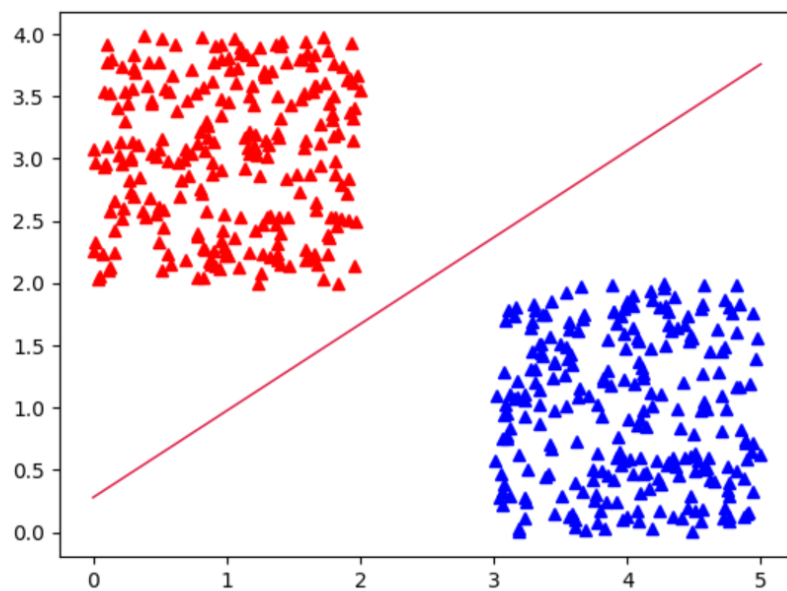
5. Прогноз модели на проверочном множестве.

1)



2)

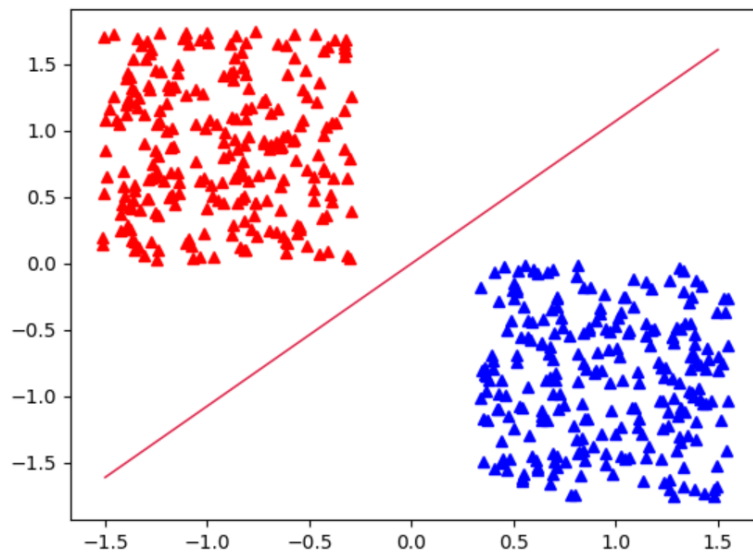
прогноз на проверочном множестве



3)

прогноз на проверочном множестве

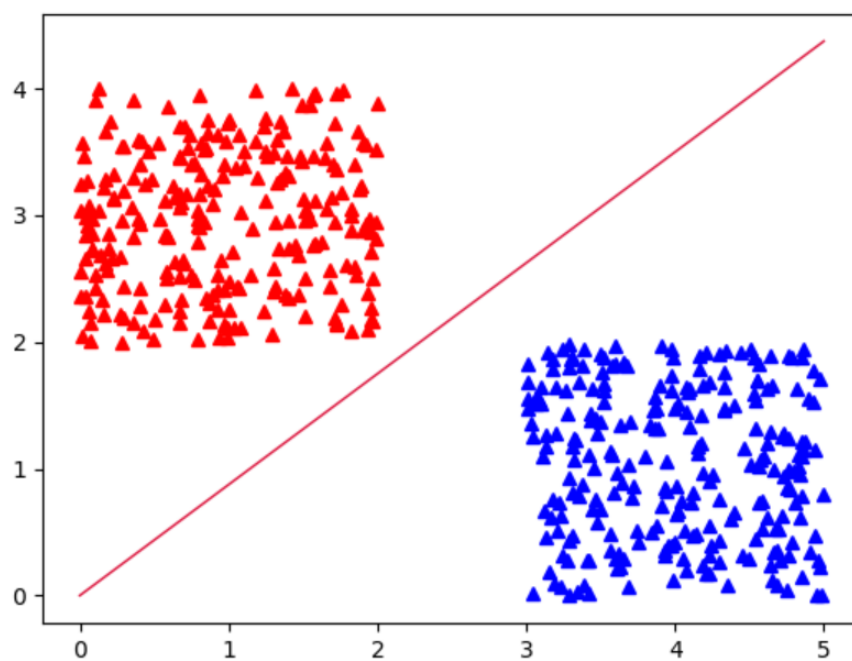
— □ ×



4)

прогноз на проверочном множестве

— □ ×



6. Вывод весов обученного нейрона и смещения (при наличии последнего).

см. строку веса в таблице п.4