```
import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np


import keras
from keras.models import Sequential
from keras.layers import Dense
```

Regresja **softmax**

```
s = [0.2, 0.1, 0.6, 0.1]
exps = [np.exp(i) for i in s]
sum_of_exps = sum(exps)
softmax = [j/sum_of_exps for j in exps]
```

3 gangi

Zbiór danych:

```
x_label0 = np.random.normal(1, 1.5, (1000, 1))
y_label0 = np.random.normal(1, 1.5, (1000, 1))
x_label1 = np.random.normal(5, 1.5, (1000, 1))
y_label1 = np.random.normal(4, 1.5, (1000, 1))
x_label2 = np.random.normal(8, 1.5, (1000, 1))
y_label2 = np.random.normal(0, 1.5, (1000, 1))


data_label0 = np.concatenate([x_label0, y_label0],axis=1)
data_label1 = np.concatenate([x_label1, y_label1],axis=1)
data_label2 = np.concatenate([x_label2, y_label2],axis=1)
points = np.concatenate([data_label0, data_label1, data_label2],axis=0)
```

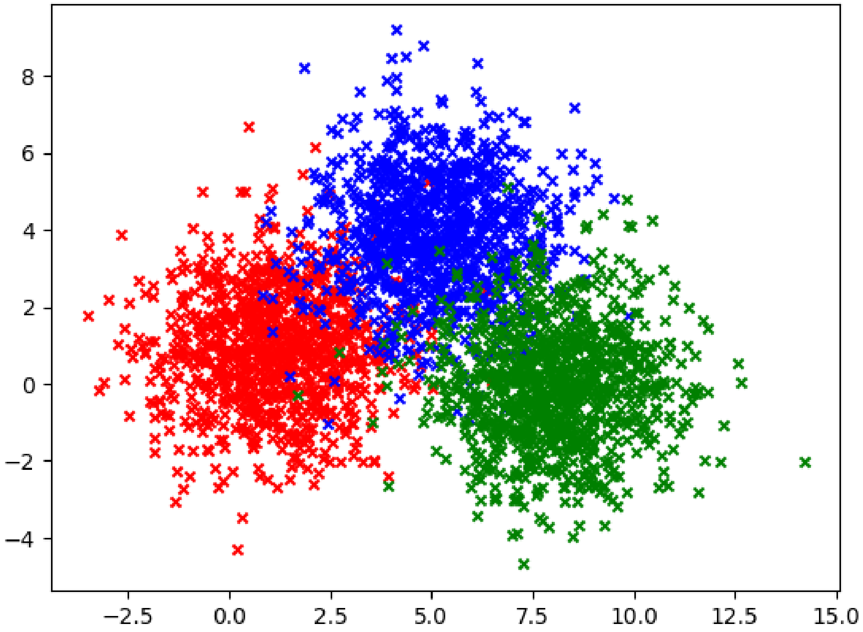
Kodowanie one-hot

```
labels = np.array([[1., 0., 0.]] * len(data_label0) + [[0., 1., 0.]] * len(data_label1) + [[0.,0., 1.]] * len(data_label2))
```

```
points.shape,labels.shape
```

```
((3000, 2), (3000, 3))
```

```
plt.scatter(x_label0, y_label0, c='r', marker='x', s=20)
plt.scatter(x_label1, y_label1, c='b', marker='x', s=20)
plt.scatter(x_label2, y_label2, c='g', marker='x', s=20)
plt.show()
```



```
model = Sequential()
```

```
model.add(Dense(units = 3, use_bias=True, input_dim=2, activation = "softmax"))
```

```
#opt = tf.keras.optimizers.Adam(learning_rate=0.1)
opt = tf.keras.optimizers.Adam()
#opt = tf.keras.optimizers.SGD(learning_rate=0.1)
```

```
model.compile(loss='binary_crossentropy',optimizer=opt)
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 dense (Dense)               (None, 3)                 9         
                                                                 
=================================================================
Total params: 9 (36.00 Byte)
Trainable params: 9 (36.00 Byte)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
epochs = 1000
#h = model.fit(data_points,labels, verbose=1, epochs=epochs,validation_split=0.2)
h = model.fit(points,labels, verbose=1, epochs=epochs, batch_size= 70, validation_split=0.2)
```

```
    Epoch 989/1000
    35/35 [==============================] - 0s 3ms/step - loss: 0.1567 - val_loss: 0.2449
    Epoch 990/1000
    35/35 [==============================] - 0s 2ms/step - loss: 0.1567 - val_loss: 0.2451
    Epoch 991/1000
    35/35 [==============================] - 0s 3ms/step - loss: 0.1567 - val_loss: 0.2431
    Epoch 992/1000
    35/35 [==============================] - 0s 3ms/step - loss: 0.1567 - val_loss: 0.2419
    Epoch 993/1000
    35/35 [==============================] - 0s 3ms/step - loss: 0.1567 - val_loss: 0.2430
    Epoch 994/1000
    35/35 [==============================] - 0s 3ms/step - loss: 0.1567 - val_loss: 0.2433
    Epoch 995/1000
    35/35 [==============================] - 0s 3ms/step - loss: 0.1567 - val_loss: 0.2472
    Epoch 996/1000
    35/35 [==============================] - 0s 4ms/step - loss: 0.1567 - val_loss: 0.2428
    Epoch 997/1000
    35/35 [==============================] - 0s 3ms/step - loss: 0.1567 - val_loss: 0.2434
    Epoch 998/1000
    35/35 [==============================] - 0s 4ms/step - loss: 0.1567 - val_loss: 0.2433
    Epoch 999/1000
    35/35 [==============================] - 0s 3ms/step - loss: 0.1567 - val_loss: 0.2450
    Epoch 1000/1000
    35/35 [==============================] - 0s 3ms/step - loss: 0.1568 - val_loss: 0.2440
```

```
Loss = h.history['loss']
Loss
```

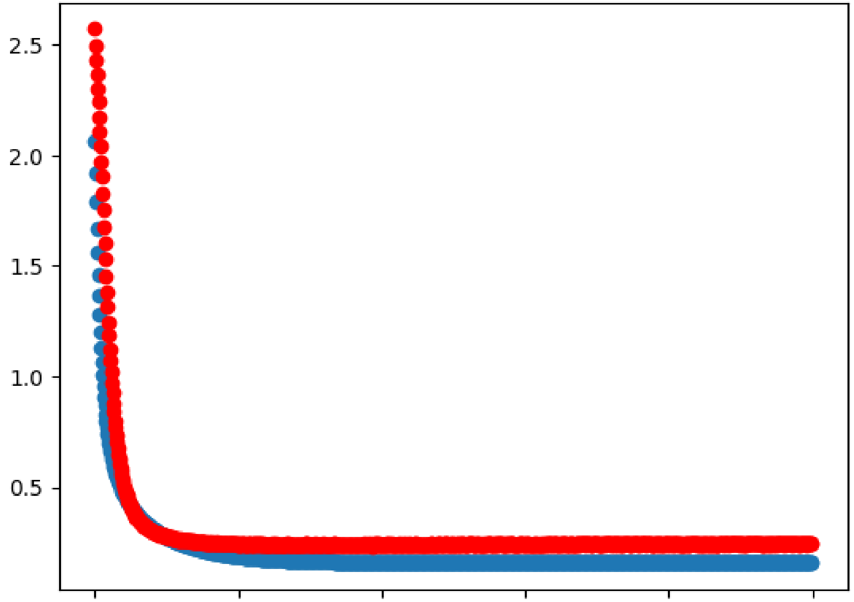
```
      0.15674307942390442,
      0.15672850608825684,
      0.15678222477436066,
      0.15671847760677338,
      0.15672115981578827,
      0.15673989057540894,
      0.15675140917301178,
      0.15671634674072266,
      0.15681302547454834,
      0.15670068562030792,
      0.1567232459783554,
      0.15674148499965668,
      0.15671007335186005,
      0.15672567486763,
      0.15671215951442719,
      0.15676255524158478,
      0.15670548379421234,
      0.15671847760677338,
      0.15670041739940643,
      0.1567092388868332,
      0.15673935413360596,
      0.15671317279338837,
      0.15672899782657623,
      0.15669238567352295,
      0.15672625601291656,
      0.15672796964645386,
      0.15671248733997345,
      0.1567256599664688,
      0.15672150254249573,
      0.15669448673725128,
      0.15669465065002441,
      0.1567067801952362,
      0.1567171961069107,
      0.15676064789295197]
```

```
weights = model.get_weights()

print(weights[0])
print(weights[1])    #bias
```

```
[[-1.8542533   0.37775218  1.5208719 ]
 [-0.9898408   1.4547888  -1.4014187 ]]
[ 7.970148  -5.1614227 -8.469914 ]
```

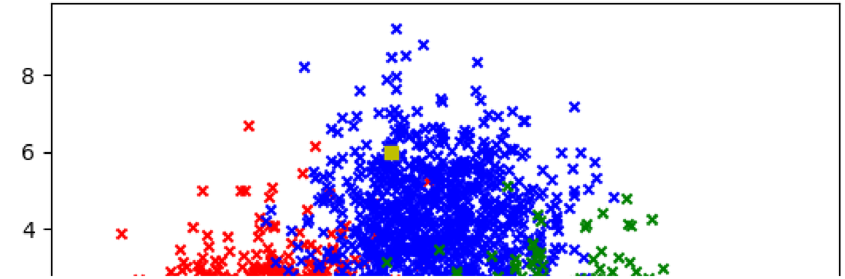
```
plt.scatter(np.arange(epochs),h.history['loss'])
plt.scatter(np.arange(epochs),h.history['val_loss'],c='r')
plt.show()
```



```
model.predict([[4,6]])
```

```
1/1 [==============================] - 0s 65ms/step
array([[2.8538452e-05, 9.9997139e-01, 1.2773766e-07]], dtype=float32)
```

```
x=4.0
y=6.0
plt.scatter(x_label0, y_label0, c='r', marker='x', s=20)
plt.scatter(x_label1, y_label1, c='b', marker='x', s=20)
plt.scatter(x_label2, y_label2, c='g', marker='x', s=20)
plt.scatter(x,y,c='y', marker='s')
plt.show()
```



Number of epochs 100

```
model = Sequential()

model.add(Dense(units = 3, use_bias=True, input_dim=2, activation = "softmax"))


#opt = tf.keras.optimizers.Adam(learning_rate=0.1)
opt = tf.keras.optimizers.Adam()
#opt = tf.keras.optimizers.SGD(learning_rate=0.1)


model.compile(loss='binary_crossentropy',optimizer=opt)


model.summary()
```

```
    Model: "sequential_1"
    _________________________________________________________________
     Layer (type)                Output Shape              Param #   
    =================================================================
     dense_1 (Dense)             (None, 3)                 9         
                                                                     
    =================================================================
    Total params: 9 (36.00 Byte)
    Trainable params: 9 (36.00 Byte)
    Non-trainable params: 0 (0.00 Byte)
    _________________________________________________________________
```

```
epochs = 100
#h = model.fit(data_points,labels, verbose=1, epochs=epochs,validation_split=0.2)
h = model.fit(points,labels, verbose=1, epochs=epochs, batch_size= 70, validation_split=0.2)
```

```
    35/35 [==============================] - 0s 2ms/step - loss: 0.2496 - val_loss: 0.2688
    Epoch 98/100
    35/35 [==============================] - 0s 2ms/step - loss: 0.2483 - val_loss: 0.2692
    Epoch 99/100
    35/35 [==============================] - 0s 2ms/step - loss: 0.2470 - val_loss: 0.2691
    Epoch 100/100
    35/35 [==============================] - 0s 2ms/step - loss: 0.2457 - val_loss: 0.2656
```

```
Loss = h.history['loss']
Loss
```

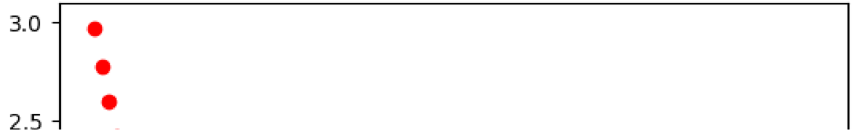
```
     0.26903593354019165,
     0.26736921072006226,
     0.265764057636261,
     0.2641422152519226,
     0.26258763670921326,
     0.2610386908054352,
     0.25954481959342957,
     0.2580412030220032,
     0.2566033601760864,
     0.25513237714767456,
     0.25371623039245605,
     0.2523210048675537,
     0.25096815824508667,
     0.24962598085403442,
     0.24828501045703888,
     0.24699103832244873,
     0.2457052618265152]
```

```
weights = model.get_weights()

print(weights[0])
print(weights[1])    #bias
```

```
[[-0.6192839   0.12776168  0.46283087]
 [-0.25504512  0.8719286  -0.91608995]]
[ 1.8876922 -2.4891806 -2.2122905]
```

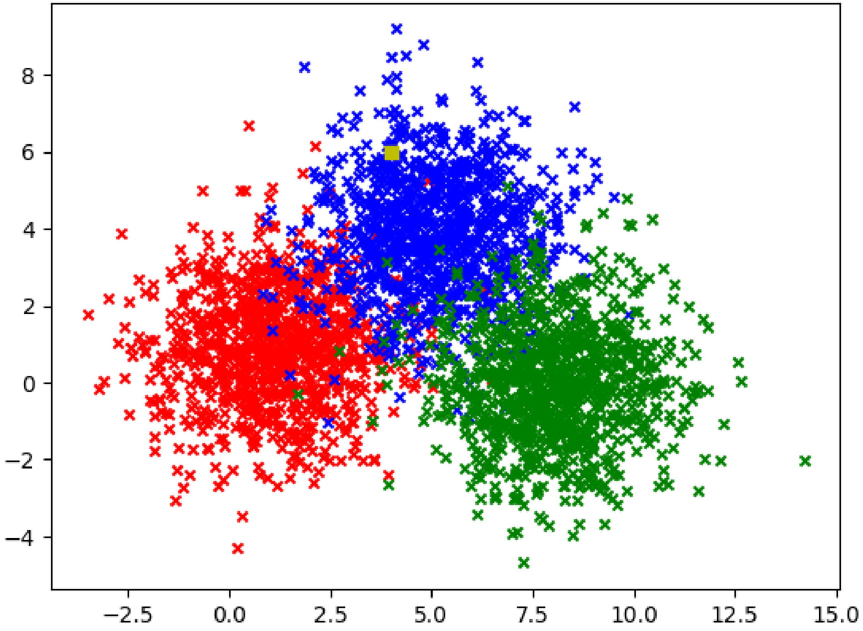
```
plt.scatter(np.arange(epochs),h.history['loss'])
plt.scatter(np.arange(epochs),h.history['val_loss'],c='r')
plt.show()
```



```
model.predict([[4,6]])
```

```
    1/1 [==============================] - 0s 59ms/step
    array([[4.6175518e-03, 9.9527246e-01, 1.0992816e-04]], dtype=float32)
```

```
x=4.0
y=6.0
plt.scatter(x_label0, y_label0, c='r', marker='x', s=20)
plt.scatter(x_label1, y_label1, c='b', marker='x', s=20)
plt.scatter(x_label2, y_label2, c='g', marker='x', s=20)
plt.scatter(x,y,c='y', marker='s')
plt.show()
```



Number of epochs 2000

```
model = Sequential()
```

```
model.add(Dense(units = 3, use_bias=True, input_dim=2, activation = "softmax"))
```

```
#opt = tf.keras.optimizers.Adam(learning_rate=0.1)
opt = tf.keras.optimizers.Adam()
#opt = tf.keras.optimizers.SGD(learning_rate=0.1)
```

```
model.compile(loss='binary_crossentropy',optimizer=opt)
```

```
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 dense_2 (Dense)             (None, 3)                 9         
                                                                 
=================================================================
Total params: 9 (36.00 Byte)
Trainable params: 9 (36.00 Byte)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
epochs = 2000
#h = model.fit(data_points,labels, verbose=1, epochs=epochs,validation_split=0.2)
h = model.fit(points,labels, verbose=1, epochs=epochs, batch_size= 70, validation_split=0.2)
```

```
    Epoch 1980/2000
    35/35 [==============================] - 0s 3ms/step - loss: 0.1566 - val_loss: 0.2443
    Epoch 1981/2000
    35/35 [==============================] - 0s 3ms/step - loss: 0.1566 - val_loss: 0.2444
    Epoch 1982/2000
    35/35 [==============================] - 0s 3ms/step - loss: 0.1566 - val_loss: 0.2419
    Epoch 1983/2000
    35/35 [==============================] - 0s 3ms/step - loss: 0.1566 - val_loss: 0.2434
    Epoch 1984/2000
    35/35 [==============================] - 0s 2ms/step - loss: 0.1566 - val_loss: 0.2454
    Epoch 1985/2000
    35/35 [==============================] - 0s 3ms/step - loss: 0.1566 - val_loss: 0.2434
    Epoch 1986/2000
    35/35 [==============================] - 0s 2ms/step - loss: 0.1566 - val_loss: 0.2448
    Epoch 1987/2000
    35/35 [==============================] - 0s 3ms/step - loss: 0.1566 - val_loss: 0.2445
    Epoch 1988/2000
    35/35 [==============================] - 0s 2ms/step - loss: 0.1566 - val_loss: 0.2450
    Epoch 1989/2000
    35/35 [==============================] - 0s 3ms/step - loss: 0.1566 - val_loss: 0.2447
    Epoch 1990/2000
    35/35 [==============================] - 0s 3ms/step - loss: 0.1566 - val_loss: 0.2470
    Epoch 1991/2000
    35/35 [==============================] - 0s 3ms/step - loss: 0.1566 - val_loss: 0.2474
    Epoch 1992/2000
    35/35 [==============================] - 0s 2ms/step - loss: 0.1566 - val_loss: 0.2466
    Epoch 1993/2000
    35/35 [==============================] - 0s 2ms/step - loss: 0.1566 - val_loss: 0.2487
    Epoch 1994/2000
    35/35 [==============================] - 0s 2ms/step - loss: 0.1566 - val_loss: 0.2475
    Epoch 1995/2000
    35/35 [==============================] - 0s 2ms/step - loss: 0.1566 - val_loss: 0.2439
    Epoch 1996/2000
    35/35 [==============================] - 0s 2ms/step - loss: 0.1566 - val_loss: 0.2469
    Epoch 1997/2000
    35/35 [==============================] - 0s 2ms/step - loss: 0.1566 - val_loss: 0.2461
    Epoch 1998/2000
    35/35 [==============================] - 0s 3ms/step - loss: 0.1566 - val_loss: 0.2453
    Epoch 1999/2000
    35/35 [==============================] - 0s 3ms/step - loss: 0.1566 - val_loss: 0.2455
    Epoch 2000/2000
    35/35 [==============================] - 0s 2ms/step - loss: 0.1566 - val_loss: 0.2444
```

```
Loss = h.history['loss']
Loss
```

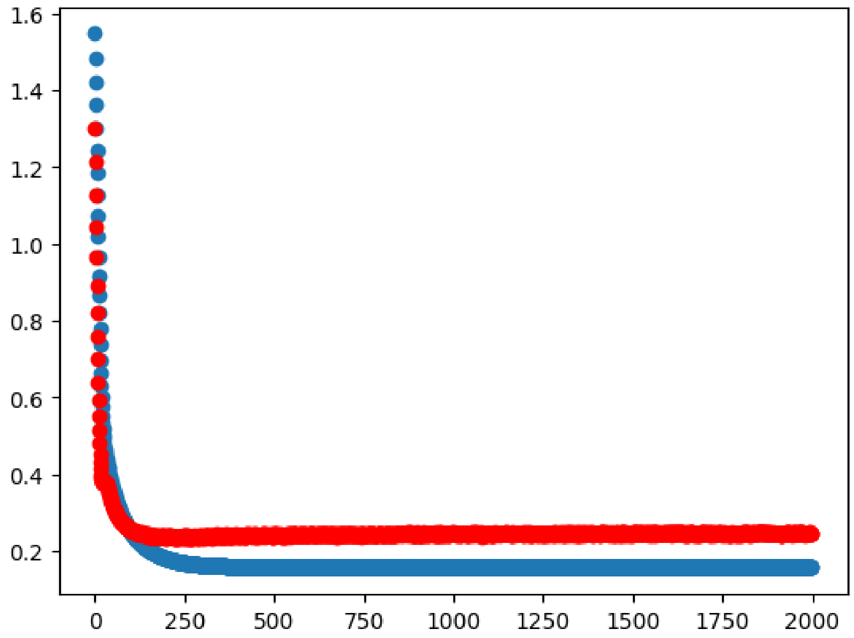
```
     ...]
```

```
weights = model.get_weights()

print(weights[0])
print(weights[1])    #bias
```

```
[[-1.9186313   0.37638855  1.6253421 ]
 [-1.0266261   1.4578031  -1.4661033 ]]
[ 8.280908 -5.164768 -9.07732 ]
```

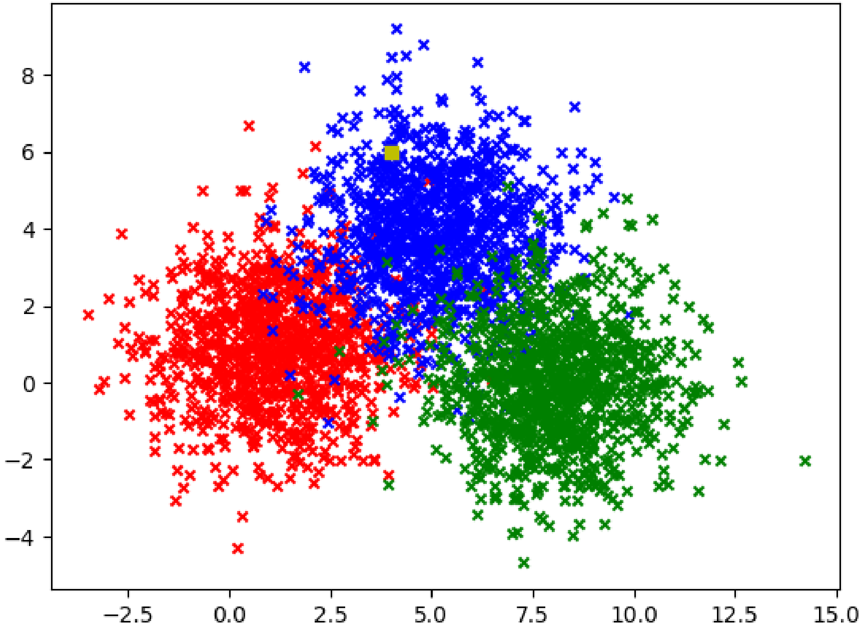
```
plt.scatter(np.arange(epochs),h.history['loss'])
plt.scatter(np.arange(epochs),h.history['val_loss'],c='r')
plt.show()
```



```
model.predict([[4,6]])
```

```
1/1 [==============================] - 0s 43ms/step
array([[2.3915041e-05, 9.9997592e-01, 7.1027500e-08]], dtype=float32)
```

```
x=4.0
y=6.0
plt.scatter(x_label0, y_label0, c='r', marker='x', s=20)
plt.scatter(x_label1, y_label1, c='b', marker='x', s=20)
plt.scatter(x_label2, y_label2, c='g', marker='x', s=20)
plt.scatter(x,y,c='y', marker='s')
plt.show()
```



Learning rate 0.1

```
model = Sequential()


model.add(Dense(units = 3, use_bias=True, input_dim=2, activation = "softmax"))


opt = tf.keras.optimizers.Adam(learning_rate=0.1)
#opt = tf.keras.optimizers.Adam()
#opt = tf.keras.optimizers.SGD(learning_rate=0.1)
```

```
model.compile(loss='binary_crossentropy',optimizer=opt)
```

```
model.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 dense_3 (Dense)             (None, 3)                 9         
                                                                 
=================================================================
Total params: 9 (36.00 Byte)
Trainable params: 9 (36.00 Byte)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
epochs = 1000
#h = model.fit(data_points,labels, verbose=1, epochs=epochs,validation_split=0.2)
h = model.fit(points,labels, verbose=1, epochs=epochs, batch_size= 70, validation_split=0.2)
```

```
    35/35 [==============================] - 0s 2ms/step - loss: 0.1602 - val_loss: 0.2664
    Epoch 987/1000
    35/35 [==============================] - 0s 2ms/step - loss: 0.1636 - val_loss: 0.2565
    Epoch 988/1000
    35/35 [==============================] - 0s 2ms/step - loss: 0.1628 - val_loss: 0.2772
    Epoch 989/1000
    35/35 [==============================] - 0s 2ms/step - loss: 0.1630 - val_loss: 0.2362
    Epoch 990/1000
    35/35 [==============================] - 0s 2ms/step - loss: 0.1616 - val_loss: 0.2718
    Epoch 991/1000
    35/35 [==============================] - 0s 2ms/step - loss: 0.1605 - val_loss: 0.2094
    Epoch 992/1000
    35/35 [==============================] - 0s 2ms/step - loss: 0.1604 - val_loss: 0.2640
    Epoch 993/1000
    35/35 [==============================] - 0s 2ms/step - loss: 0.1603 - val_loss: 0.2967
    Epoch 994/1000
    35/35 [==============================] - 0s 2ms/step - loss: 0.1599 - val_loss: 0.2399
    Epoch 995/1000
    35/35 [==============================] - 0s 2ms/step - loss: 0.1610 - val_loss: 0.2613
    Epoch 996/1000
    35/35 [==============================] - 0s 3ms/step - loss: 0.1598 - val_loss: 0.2741
    Epoch 997/1000
    35/35 [==============================] - 0s 2ms/step - loss: 0.1600 - val_loss: 0.3083
    Epoch 998/1000
    35/35 [==============================] - 0s 2ms/step - loss: 0.1602 - val_loss: 0.3103
    Epoch 999/1000
    35/35 [==============================] - 0s 2ms/step - loss: 0.1608 - val_loss: 0.2647
    Epoch 1000/1000
    35/35 [==============================] - 0s 2ms/step - loss: 0.1598 - val_loss: 0.2681
```

```
Loss = h.history['loss']
Loss
```

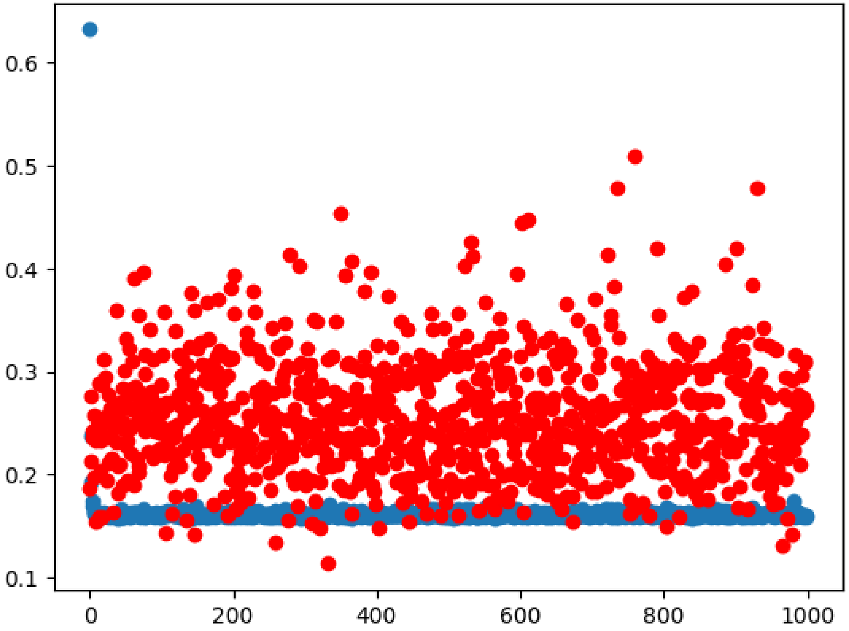
```
     0.16379012167453766,
     0.16042982041835785,
     0.16075921058654785,
     0.16320320963859558,
     0.16297510266304016,
     0.16055388748645782,
     0.1608719378709793,
     0.16025833785533905,
     0.1604868471622467,
     0.1593562811613083,
     0.16037946939468384,
     0.16341416537761688,
     0.15937060117721558,
     0.16094814240932465,
     0.15983568131923676,
     0.1607755869626999,
     0.1638484001159668,
     0.16113193333148956,
     0.16345638036727905,
     0.17447996139526367,
     0.16293780505657196,
     0.16053014993667603,
     0.16081421077251434,
     0.1586494892835617,
     0.16016899049282074,
     0.16359543800354004,
     0.16283239424228668,
     0.16302351653575897,
     0.16162802278995514,
     0.16053351759910583,
     0.16038376092910767,
     0.16029150784015656,
     0.15989957749843597,
     0.1610257625579834,
     0.1598190814256668,
     0.15997277200222015,
     0.1602095067501068,
     0.16083262860774994,
     0.1598118096590042]
```

```
weights = model.get_weights()

print(weights[0])
print(weights[1])    #bias
```

```
[[-1.9795942  0.4349273  1.706098 ]
 [-0.8989535  1.4161886 -1.5113696]]
[ 8.54315  -5.302328 -9.565231]
```

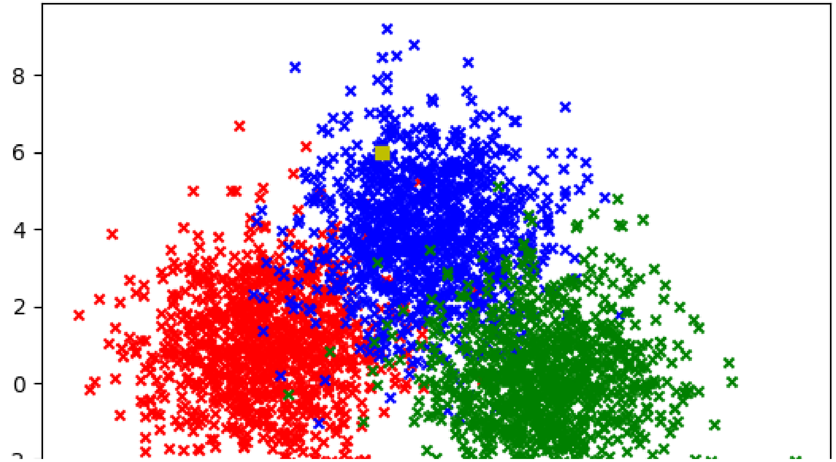
```
plt.scatter(np.arange(epochs),h.history['loss'])
plt.scatter(np.arange(epochs),h.history['val_loss'],c='r')
plt.show()
```



```
model.predict([[4,6]])
```

```
1/1 [==============================] - 0s 42ms/step
array([[6.1068109e-05, 9.9993896e-01, 5.3497580e-08]], dtype=float32)
```

```
x=4.0
y=6.0
plt.scatter(x_label0, y_label0, c='r', marker='x', s=20)
plt.scatter(x_label1, y_label1, c='b', marker='x', s=20)
plt.scatter(x_label2, y_label2, c='g', marker='x', s=20)
plt.scatter(x,y,c='y', marker='s')
plt.show()
```



Learning rate 0.0001

```
model = Sequential()


model.add(Dense(units = 3, use_bias=True, input_dim=2, activation = "softmax"))


opt = tf.keras.optimizers.Adam(learning_rate=0.0001)
#opt = tf.keras.optimizers.Adam()
#opt = tf.keras.optimizers.SGD(learning_rate=0.1)


model.compile(loss='binary_crossentropy',optimizer=opt)


model.summary()
```

```
    Model: "sequential_4"
    _________________________________________________________________
     Layer (type)                Output Shape              Param #   
    =================================================================
     dense_4 (Dense)             (None, 3)                 9         
                                                                     
    =================================================================
    Total params: 9 (36.00 Byte)
    Trainable params: 9 (36.00 Byte)
    Non-trainable params: 0 (0.00 Byte)
    _________________________________________________________________
```

```
epochs = 1000
#h = model.fit(data_points,labels, verbose=1, epochs=epochs,validation_split=0.2)
h = model.fit(points,labels, verbose=1, epochs=epochs, batch_size= 70, validation_split=0.2)
```

```
    Epoch 993/1000
    35/35 [==============================] - 0s 2ms/step - loss: 0.2106 - val_loss: 0.2484
    Epoch 994/1000
    35/35 [==============================] - 0s 2ms/step - loss: 0.2105 - val_loss: 0.2485
    Epoch 995/1000
    35/35 [==============================] - 0s 2ms/step - loss: 0.2104 - val_loss: 0.2480
    Epoch 996/1000
    35/35 [==============================] - 0s 2ms/step - loss: 0.2103 - val_loss: 0.2483
    Epoch 997/1000
    35/35 [==============================] - 0s 2ms/step - loss: 0.2102 - val_loss: 0.2483
    Epoch 998/1000
    35/35 [==============================] - 0s 2ms/step - loss: 0.2101 - val_loss: 0.2485
    Epoch 999/1000
    35/35 [==============================] - 0s 2ms/step - loss: 0.2100 - val_loss: 0.2483
    Epoch 1000/1000
    35/35 [==============================] - 0s 2ms/step - loss: 0.2099 - val_loss: 0.2481
```

```
Loss = h.history['loss']
Loss
```

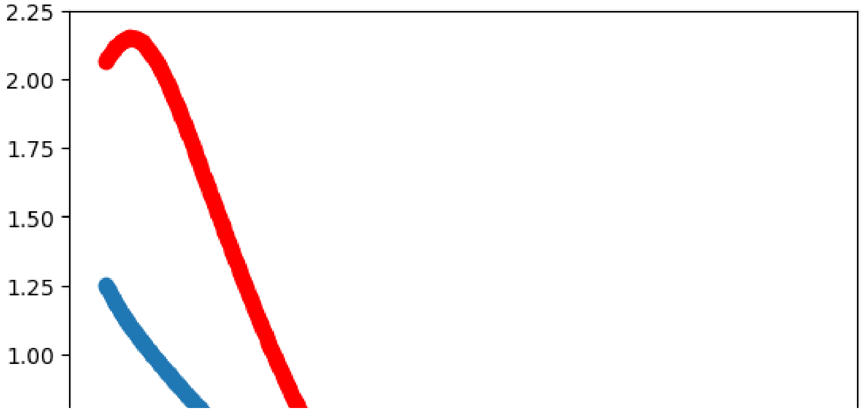
```
      0.2124360203742981,
      0.21233676373958588,
      0.21223615109920502,
      0.21213240921497345,
      0.21203115582466125,
      0.21193036437034607,
      0.21183323860168457,
      0.21174253523349762,
      0.21163320541381836,
      0.21153868734836578,
      0.2114371806383133,
      0.21134229004383087,
      0.21124067902565002,
      0.21115081012248993,
      0.21105551719665527,
      0.21095231175422668,
      0.21085263788700104,
      0.21075329184532166,
      0.21066242456436157,
      0.2105596661567688,
      0.21046391129493713,
      0.21037010848522186,
      0.2102724313735962,
      0.21017520129680634,
      0.21008704602718353,
      0.20998099446296692,
      0.20989182591438293]
```

```
weights = model.get_weights()

print(weights[0])
print(weights[1])    #bias
```

```
[[-0.86395454  0.15022355  0.5943968 ]
 [-0.38220945  0.91327024 -0.95378107]]
[ 3.0835235 -2.6992445 -2.9725344]
```

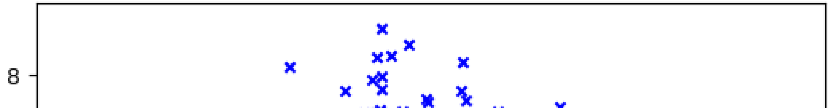
```
plt.scatter(np.arange(epochs),h.history['loss'])
plt.scatter(np.arange(epochs),h.history['val_loss'],c='r')
plt.show()
```



```
model.predict([[4,6]])
```

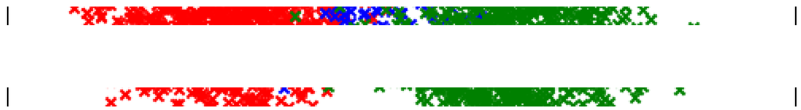
```
WARNING:tensorflow:5 out of the last 5 calls to <function Model.make_predict_function.<locals>.predict_function at 0x788676b92440> triggered tf.funct
1/1 [==============================] - 0s 48ms/step
array([[2.3596396e-03, 9.9757916e-01, 6.1202118e-05]], dtype=float32)
```

```
x=4.0
y=6.0
plt.scatter(x_label0, y_label0, c='r', marker='x', s=20)
plt.scatter(x_label1, y_label1, c='b', marker='x', s=20)
plt.scatter(x_label2, y_label2, c='g', marker='x', s=20)
plt.scatter(x,y,c='y', marker='s')
plt.show()
```



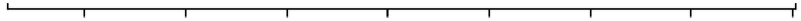
Minibatch

Minibatch-20



```
model = Sequential()
```

```
model.add(Dense(units = 3, use_bias=True, input_dim=2, activation = "softmax"))
```



```
opt = tf.keras.optimizers.Adam(learning_rate=0.0001)
#opt = tf.keras.optimizers.Adam()
#opt = tf.keras.optimizers.SGD(learning_rate=0.1)


model.compile(loss='binary_crossentropy',optimizer=opt)


model.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_5 (Dense)             (None, 3)                 9

=================================================================
Total params: 9 (36.00 Byte)
Trainable params: 9 (36.00 Byte)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
epochs = 1000
#h = model.fit(data_points,labels, verbose=1, epochs=epochs,validation_split=0.2)
h = model.fit(points,labels, verbose=1, epochs=epochs,  validation_split=0.2,batch_size=20)
```

```
    120/120 [==============================] - 0s 2ms/step - loss: 0.1643 - val_loss: 0.2391
    Epoch 998/1000
    120/120 [==============================] - 0s 2ms/step - loss: 0.1643 - val_loss: 0.2388
    Epoch 999/1000
    120/120 [==============================] - 0s 2ms/step - loss: 0.1643 - val_loss: 0.2385
    Epoch 1000/1000
    120/120 [==============================] - 0s 2ms/step - loss: 0.1643 - val_loss: 0.2391
```

```
Loss = h.history['loss']
Loss
```

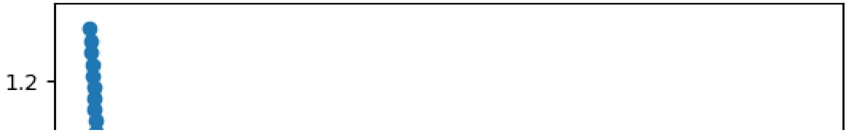
```
     0.16455519199371338,
     0.16453391313552856,
     0.1645217388868332,
     0.16450464725494385,
     0.1644873470067978,
     0.16446520388126373,
     0.16444945335388184,
     0.1644359529018402,
     0.1644258201122284,
     0.1644003540277481,
     0.16438671946525574,
     0.16437384486198425,
     0.16434895992279053,
     0.16434694826602936,
     0.16432400047779083,
     0.16430972516536713,
     0.1642874926328659]
```

```
weights = model.get_weights()

print(weights[0])
print(weights[1])    #bias
```

```
    [[-1.376739   0.3119349  1.0393476]
     [-0.7011174  1.2859708 -1.1403483]]
    [ 5.670027 -4.416901 -5.620191]
```

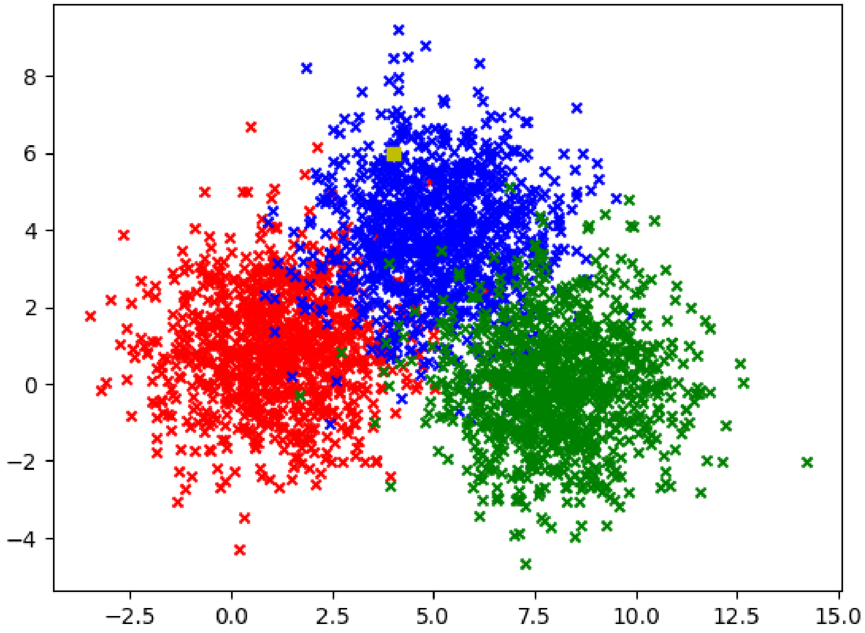
```
plt.scatter(np.arange(epochs),h.history['loss'])
plt.scatter(np.arange(epochs),h.history['val_loss'],c='r')
plt.show()
```



```
model.predict([[4,6]])
```

```
WARNING:tensorflow:6 out of the last 6 calls to <function Model.make_predict_function.<locals>.predict_function at 0x788710239ea0> triggered tf.funct
1/1 [==============================] - 0s 64ms/step
array([[1.858657e-04, 9.998116e-01, 2.621559e-06]], dtype=float32)
```

```
x=4.0
y=6.0
plt.scatter(x_label0, y_label0, c='r', marker='x', s=20)
plt.scatter(x_label1, y_label1, c='b', marker='x', s=20)
plt.scatter(x_label2, y_label2, c='g', marker='x', s=20)
plt.scatter(x,y,c='y', marker='s')
plt.show()
```



mini batch - 50

```
model = Sequential()
```

```
model.add(Dense(units = 3, use_bias=True, input_dim=2, activation = "softmax"))
```

```
opt = tf.keras.optimizers.Adam(learning_rate=0.0001)
#opt = tf.keras.optimizers.Adam()
#opt = tf.keras.optimizers.SGD(learning_rate=0.1)
```

```
model.compile(loss='binary_crossentropy',optimizer=opt)
```

```
model.summary()
```

```
Model: "sequential_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 dense_6 (Dense)             (None, 3)                 9         
                                                                 
=================================================================
Total params: 9 (36.00 Byte)
Trainable params: 9 (36.00 Byte)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
epochs = 1000
#h = model.fit(data_points,labels, verbose=1, epochs=epochs,validation_split=0.2)
h = model.fit(points,labels, verbose=1, epochs=epochs, batch_size= 50, validation_split=0.2)
```

```
Epoch 980/1000
48/48 [==============================] - 0s 2ms/step - loss: 0.1898 - val_loss: 0.2435
Epoch 981/1000
48/48 [==============================] - 0s 2ms/step - loss: 0.1897 - val_loss: 0.2442
Epoch 982/1000
48/48 [==============================] - 0s 2ms/step - loss: 0.1896 - val_loss: 0.2440
Epoch 983/1000
48/48 [==============================] - 0s 2ms/step - loss: 0.1896 - val_loss: 0.2440
Epoch 984/1000
48/48 [==============================] - 0s 2ms/step - loss: 0.1895 - val_loss: 0.2439
Epoch 985/1000
48/48 [==============================] - 0s 2ms/step - loss: 0.1894 - val_loss: 0.2440
Epoch 986/1000
48/48 [==============================] - 0s 2ms/step - loss: 0.1894 - val_loss: 0.2440
Epoch 987/1000
48/48 [==============================] - 0s 2ms/step - loss: 0.1893 - val_loss: 0.2439
Epoch 988/1000
48/48 [==============================] - 0s 2ms/step - loss: 0.1892 - val_loss: 0.2437
Epoch 989/1000
48/48 [==============================] - 0s 2ms/step - loss: 0.1892 - val_loss: 0.2441
Epoch 990/1000
48/48 [==============================] - 0s 2ms/step - loss: 0.1891 - val_loss: 0.2439
Epoch 991/1000
48/48 [==============================] - 0s 2ms/step - loss: 0.1890 - val_loss: 0.2436
Epoch 992/1000
48/48 [==============================] - 0s 2ms/step - loss: 0.1890 - val_loss: 0.2435
Epoch 993/1000
48/48 [==============================] - 0s 2ms/step - loss: 0.1889 - val_loss: 0.2437
Epoch 994/1000
48/48 [==============================] - 0s 2ms/step - loss: 0.1888 - val_loss: 0.2435
Epoch 995/1000
48/48 [==============================] - 0s 2ms/step - loss: 0.1888 - val_loss: 0.2440
Epoch 996/1000
48/48 [==============================] - 0s 2ms/step - loss: 0.1887 - val_loss: 0.2439
Epoch 997/1000
48/48 [==============================] - 0s 2ms/step - loss: 0.1886 - val_loss: 0.2436
Epoch 998/1000
48/48 [==============================] - 0s 2ms/step - loss: 0.1886 - val_loss: 0.2437
Epoch 999/1000
48/48 [==============================] - 0s 2ms/step - loss: 0.1885 - val_loss: 0.2438
Epoch 1000/1000
48/48 [==============================] - 0s 2ms/step - loss: 0.1884 - val_loss: 0.2437
```

```
Loss = h.history['loss']
Loss
```

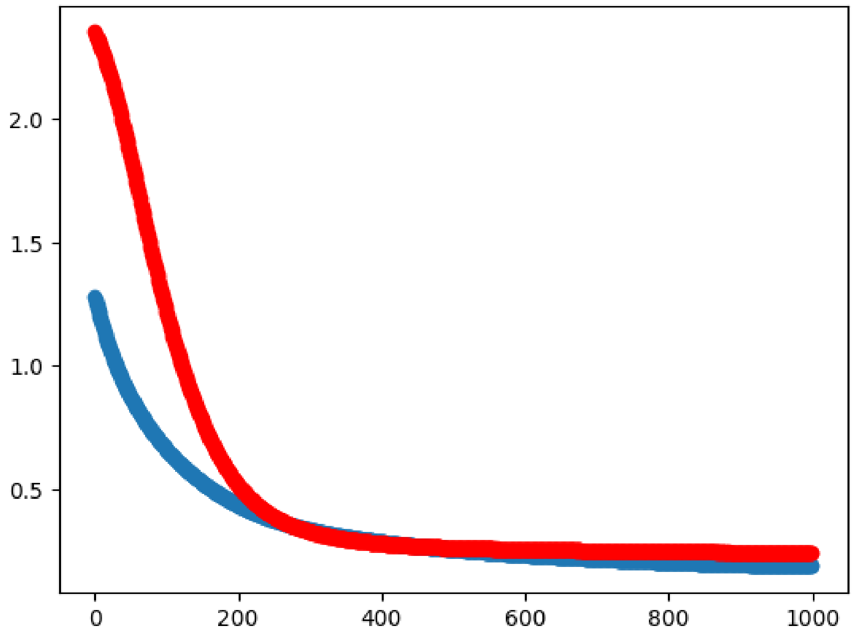
```
     0.18844370543956757]
```

```
weights = model.get_weights()

print(weights[0])
print(weights[1])    #bias
```

```
[[-0.98609793  0.21861346  0.7294145 ]
 [-0.46031675  1.0607656  -0.997672  ]]
[ 3.722764  -3.4068813 -3.7914543]
```

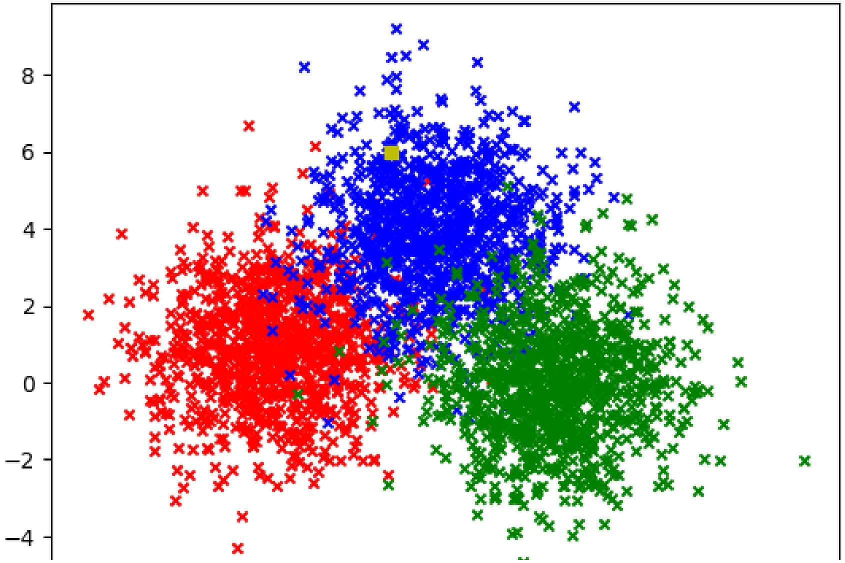
```
plt.scatter(np.arange(epochs),h.history['loss'])
plt.scatter(np.arange(epochs),h.history['val_loss'],c='r')
plt.show()
```



```
model.predict([[4,6]])
```

```
1/1 [==============================] - 0s 47ms/step
array([[1.0952058e-03, 9.9888200e-01, 2.2700757e-05]], dtype=float32)
```

```
x=4.0
y=6.0
plt.scatter(x_label0, y_label0, c='r', marker='x', s=20)
plt.scatter(x_label1, y_label1, c='b', marker='x', s=20)
plt.scatter(x_label2, y_label2, c='g', marker='x', s=20)
plt.scatter(x,y,c='y', marker='s')
plt.show()
```



Najlepsze wyniki otrzymałem dla współczynnika uczenia 0.1, liczby epok 2000,batcha równego 20, najgorsze dla współczynnika uczenia 0.0001, liczby epok 100, batcha równego 50.