```
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)
```

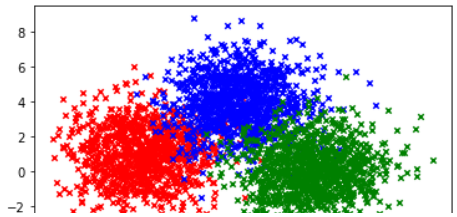
Kodowanie one-hot

```
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
Trainable params: 9
Non-trainable params: 0
_________________________________________________________________
```

```
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 983/1000
35/35 [==============================] - 0s 3ms/step - loss: 0.1529 - val_loss: 0.1990
Epoch 984/1000
35/35 [==============================] - 0s 3ms/step - loss: 0.1529 - val_loss: 0.1986
Epoch 985/1000
35/35 [==============================] - 0s 3ms/step - loss: 0.1529 - val_loss: 0.1974
Epoch 986/1000
35/35 [==============================] - 0s 3ms/step - loss: 0.1529 - val_loss: 0.1967
Epoch 987/1000
35/35 [==============================] - 0s 3ms/step - loss: 0.1529 - val_loss: 0.1996
Epoch 988/1000
35/35 [==============================] - 0s 3ms/step - loss: 0.1529 - val_loss: 0.1984
Epoch 989/1000
35/35 [==============================] - 0s 3ms/step - loss: 0.1530 - val_loss: 0.1957
Epoch 990/1000
35/35 [==============================] - 0s 3ms/step - loss: 0.1529 - val_loss: 0.1974
Epoch 991/1000
35/35 [==============================] - 0s 4ms/step - loss: 0.1529 - val_loss: 0.1986
Epoch 992/1000
35/35 [==============================] - 0s 3ms/step - loss: 0.1529 - val_loss: 0.1952
Epoch 993/1000
35/35 [==============================] - 0s 3ms/step - loss: 0.1529 - val_loss: 0.1966
Epoch 994/1000
35/35 [==============================] - 0s 3ms/step - loss: 0.1529 - val_loss: 0.1953
Epoch 995/1000
35/35 [==============================] - 0s 3ms/step - loss: 0.1529 - val_loss: 0.1941
Epoch 996/1000
35/35 [==============================] - 0s 3ms/step - loss: 0.1529 - val_loss: 0.1951
Epoch 997/1000
35/35 [==============================] - 0s 3ms/step - loss: 0.1529 - val_loss: 0.1959
Epoch 998/1000
35/35 [==============================] - 0s 3ms/step - loss: 0.1529 - val_loss: 0.1950
Epoch 999/1000
35/35 [==============================] - 0s 3ms/step - loss: 0.1529 - val_loss: 0.1960
Epoch 1000/1000
35/35 [==============================] - 0s 3ms/step - loss: 0.1529 - val_loss: 0.1987
```

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

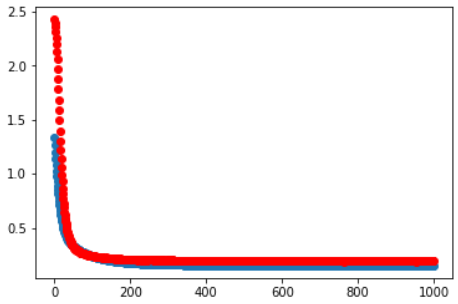
```
        0.15288247168064117,
        0.15288357436656952,
        0.15290099382400513,
        0.15291644632816315,
        0.15294672548770905,
        0.15289084613323212,
        0.15292371809482574,
        0.15288567543029785,
        0.15290939807891846,
        0.15288037061691284,
        0.15292233228683472,
        0.15287461876869202,
        0.15291030704975128,
        0.1528879553079605,
        0.1528937667608261,
        0.15289969742298126,
        0.1529001146554947,
        0.1529145985841751,
        0.15286779403686523,
        0.1528824418783188,
        0.15296931564807892,
        0.15288585424423218,
        0.15290039777755737,
        0.1529109627008438,
        0.15287992358207703,
        0.15286143124103546,
        0.15291322767734528,
        0.15287907421588898,
        0.15293265879154205,
        0.15290330350399017,
        0.15286415815353394,
        0.1528845578432083]
```

```
weights = model.get_weights()

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

```
[[-1.963214   0.392408   1.4140855]
 [-0.9900861  1.5471411 -1.2999952]]
[ 8.10579   -5.5480375 -7.5953736]
```

```
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 86ms/step
array([[1.6835167e-05, 9.9998295e-01, 2.9289262e-07]], dtype=float32)
```

✓ 0 s ukończono o 16:24