***Несколько выходов сети (нейронов в выходном слое больше одного)***

 Ранее мы использовали нейросеть для классифиции данных на 2 класса. Использовали нейросеть с одним нейроном в выходном слое:



Значение 0 или 1 на выходе сети указывало принадлежность объекта первому или второму классу.

 Используем сеть с несколькими выходами (в данном случае с двумя):



В этом случае номер выхода со значением 1 указывает номер класса.

 Для обучения сети можно использовать целевую функцию вида:

$$F\left(w, b\right)=\sum\_{train}^{}\sum\_{n}^{}I\left(t\_{n}-x\_{n}^{(L)}=0\right)$$

здесь $L$ – номер последнего слоя, n – номер нейрона в выходном слое

Значение функция $F\left(w, b\right)$ равно числу правильно классифицированных точек в тренирочном наборе данных.

 Правильные значения весовых множетелей *w* и смещения *b* обеспечивают максимальному значению целевой функции $F\left(w, b\right)$.

 Как и прежде, заменим индикаторную функцию на гладкую, дифференцируемую функцию:

$$I\left(z>0\right)\rightarrow σ\left(z\right)=\frac{1}{1+e^{-az}} \frac{dσ(z)}{dz}=aσ\left(z\right)\left(1-σ\left(z\right)\right)$$

$$I\left(z=0\right)\rightarrow δ\left(z\right)=e^{-az^{2}} \frac{dδ(z)}{dz}=-2azδ(z)$$



***Softmax – активационная функция***

 Выход сети с максимальным значением указывает номер класса, которому принадлежат входные данные. Для того чтобы увеличить значение на «правильном» выходе и придать выходным значениям смысл вероятности вхождения объекта в тот или иной класс в качестве активационной функции выходного слоя используется Softmax функция. Например:

$$x\_{n}^{(L)}=S\left(z\_{n}^{(L)}\right)={exp\left(α∙z\_{n}^{(L)}\right)}/{\sum\_{i}^{}exp\left(α∙z\_{i}^{(L)}\right)}$$

$L$ – номер последнего слоя,

n – номер нейрона в выходном слое.

import numpy as np

import matplotlib.pyplot as plt

from sklearn import datasets

POINT\_N = 300

DIM\_N = 2

CLUST\_N = 3

data, targ = datasets.make\_blobs(n\_samples=POINT\_N, centers=CLUST\_N, cluster\_std=[0.5,1.5,1.5], n\_features=DIM\_N, center\_box=(-10,10), random\_state=0)

plt.figure(figsize=(5,5))

for i in range(POINT\_N):

 if targ[i]==0:

 plt.scatter(data[i][0], data[i][1], c='blue', marker='o')

 else:

 if targ[i]==1:

 plt.scatter(data[i][0], data[i][1], c='green', marker='o')

 else:

 plt.scatter(data[i][0], data[i][1], c='magenta', marker='o')

plt.show()

# Neural network module

import keras

from keras.models import Sequential

from keras.layers import Dense

# Один нейрон в выходном слое

model=Sequential()

model.add(Dense(20, input\_dim=DIM\_N, activation='sigmoid'))

model.add(Dense(20, activation='sigmoid'))

model.add(Dense(1,activation='linear'))

model.compile(loss='mse')

#model.compile(loss='mse', optimizer='adam')

callback = keras.callbacks.EarlyStopping(monitor="loss", mode="min", verbose=1, patience=300, min\_delta=0.003)

model.fit(data, targ, batch\_size=int(data.shape[0]/5), epochs=5000, verbose=1, callbacks=[callback])

res = model.predict(data)

print("")

for n in range(10):

 print (targ[n],"-",res[n])

print("")

plt.figure(figsize=(5,5))

for i in range(POINT\_N):

 if targ[i]==0:

 plt.scatter(data[i][0], data[i][1], c='blue', marker='o')

 else:

 if targ[i]==1:

 plt.scatter(data[i][0], data[i][1], c='green', marker='o')

 else:

 plt.scatter(data[i][0], data[i][1], c='magenta', marker='o')

 if res[i]<0.5:

 plt.scatter(data[i][0], data[i][1], c='blue', marker='x')

 else:

 if res[i]<1.5:

 plt.scatter(data[i][0], data[i][1], c='green', marker='x')

 else:

 plt.scatter(data[i][0], data[i][1], c='magenta', marker='x')

plt.show()

plt.figure(figsize=(7,5))

plt.hist(res, bins=15\*CLUST\_N, color='blue')

plt.show()

"""

#---------------------------------------------------------------------

# три нейрона в выходном слое - SoftMax

from sklearn import preprocessing

targ = targ.reshape(-1, 1)

enc = preprocessing.OneHotEncoder(handle\_unknown='ignore')

targ = enc.fit\_transform(targ).toarray ().astype(int)

model=Sequential()

model.add(Dense(20, input\_dim=DIM\_N, activation='sigmoid'))

model.add(Dense(20, activation='sigmoid'))

model.add(Dense(3,activation='linear'))

#model.add(Dense(3,activation='softmax'))

model.compile(loss='mse')

callback = keras.callbacks.EarlyStopping(monitor="loss", mode="min", verbose=1, patience=300, min\_delta=0.003)

model.fit(data, targ, batch\_size=int(data.shape[0]/5), epochs=5000, verbose=1, callbacks=[callback])

res = model.predict(data)

print("")

for n in range(10):

 print (targ[n],"-",res[n])

print("")

plt.figure(figsize=(5,5))

for i in range(POINT\_N):

 if np.argmax(targ[i])==0:

 plt.scatter(data[i][0], data[i][1], c='blue', marker='o')

 else:

 if np.argmax(targ[i])==1:

 plt.scatter(data[i][0], data[i][1], c='green', marker='o')

 else:

 plt.scatter(data[i][0], data[i][1], c='magenta', marker='o')

 if np.argmax(res[i])==0:

 plt.scatter(data[i][0], data[i][1], c='blue', marker='x')

 else:

 if np.argmax(res[i])==1:

 plt.scatter(data[i][0], data[i][1], c='green', marker='x')

 else:

 plt.scatter(data[i][0], data[i][1], c='magenta', marker='x')

plt.show()

res1 = []

for n in range(res.shape[0]):

 res1.append(res[n,np.argmax(res[n])]+np.argmax(res[n]))

plt.figure(figsize=(7,5))

plt.hist(res1, bins=15\*CLUST\_N, color='blue')

plt.show()

"""

Синтетический набор входных данных:



Выходные данные для 3-х случаев.

1. Сеть с одним нейроном в выходном слое
2. Сеть с тремя нейронами в выходном слое
3. Сеть с тремя нейронами и SoftMax активационной функцией

















