Blobs

March 13, 2019

1 Sample Blob Multi-Class Classification - CA2

1.1 First is to create the data set:

1.2 These values are set for the CA:

1.2.1 5 Classes in predictor (Centers)

1.2.2 N = 100000

1.3 These are the student assigned values for the CA:

1.3.1 STD = 100+ and will be assigned per student (range of 100-150)

1.3.2 Features = 50+ and will be assigned per student (range of 50 - 100)

1.3.3 Random sate = is assigned per student (and this must be the same as the mumpy seed value)

In [93]:
```
# https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_blobs.html
import numpy
from keras.utils import np_utils
from keras.models import Sequential
from keras.layers import Dense
from sklearn.datasets.samples_generator import make_blobs
import matplotlib.pyplot as plt



# Blob dataset
X, y = make_blobs(n_samples=100000, centers=5,cluster_std=100,

 n_features=50, random_state = 1)

print(X[:5])
print(y[:10])
```

```
[[ 218.95755936  110.60013998   -4.32514839   26.62303312   32.36567397
    -4.75487925  -75.96196727   32.92782986 -131.73042545   -1.28667689
   134.4826006   -85.01234956  -70.50158856   42.88372428  -84.34907656
  -102.86726242  -80.15235803  112.97490008  -19.99237307 -164.11183033
    61.50861637  -64.49556875    9.90967315   -3.14163272   28.33100832
   -72.49460052  117.50208285  159.8530953  -232.27853026  -16.41924185
    -4.24274607 -145.95247294  -29.89241078   70.92244104  -59.32501521
   -55.13038399  -73.48876674  151.93652037   17.87306901  -17.08669193
   -39.48442911  -27.65791095 -224.5675604    66.99843702   33.14473049
   107.30089048   67.06010421  -33.08753327  -41.99771769  -69.49072687]
 [  67.22299112   59.50254151  112.340049    183.39548841   59.56934778
    58.36795877  100.62377037   32.21570431   22.36147018   97.02249565
   -24.90562505  -19.16376711  -82.36087667 -152.14613525   86.38669267
   109.84216147 -143.26345352   90.89851637  -29.04417491 -178.66462622
   261.275181    132.65301973   81.69910228 -109.97606642  -54.61836811
   169.47145471  -39.29434297   90.42988589  -94.79751531   29.60238276
    34.6878297    19.82441097   28.49793801   26.86630875    5.95530099
   157.19219891 -227.83479049   -7.84613828  -68.91233873  -25.73433126
   -27.24605144 -140.37576235 -144.81265876  -44.73985475 -113.77805299
   -81.72352626  -88.19342895  132.69930309  236.03834398  -69.2368123 ]
 [ -29.1077525    -7.96767707  157.9134502    73.81168792    3.98305488
   126.22020741  -34.28802333  -13.10040071  -34.83186599  151.30447314
    31.97200991  135.16132751 -128.44925037    3.32815441 -172.75098452
  -142.97012362   30.48258797  192.67519378 -138.72578887 -103.0958636
   115.84613551   -2.69163566 -131.06644826  -93.33403195    6.26375195
  -145.66378899  -39.4637296   -62.42265222 -103.05642565  -25.2285845
    15.7052077    96.17991538   98.69032325   -8.10707449  -84.55528855
    60.41863821  -45.78862239   17.5039992    57.66213683   -9.76731769
   151.46828631  141.78827147   16.14247626 -102.97898959   74.47983035
    42.79636954   44.50330302   -1.88491809 -124.84890157  -22.01063258]
```

```
[ -69.47218541  204.62580297  -45.4105192     8.77322204  -37.12959195
   58.04609517   86.06569539 -171.15658039  206.90629832  -51.31651723
   52.26955433   19.60049016   39.58672601   76.29604419 -125.26191615
  -77.20896494  -31.1499599   -68.38680378  -35.36106481  109.62195619
  -62.25785647   15.11718188   41.79600592 -106.65130478  147.84068218
  101.04512519   34.32224783  -53.63160431  -79.93417705  128.78254581
  -20.535925    -31.80530322   96.79202855  262.7199787   -45.90911337
   73.72741534  -22.73622214  224.61182002   14.38763736   27.84051151
   48.45729222  -57.60605416 -171.01809724   12.38893788  119.27711239
  -84.419302   -119.31846787 -178.05131095 -102.01121908  -59.35759838]
 [ -91.39800254  -96.64797228  -50.60946761  -84.69961444   68.03078563
  -23.29597741   48.15265084   60.07275323    1.27849296  144.38727392
  -23.92704831  -47.33813098  -33.29143073  -61.97521331  -75.23220946
  -33.07843842   27.59725538  -42.2219507  -105.18713278  -99.76085964
 -105.67563922  -92.10898237  -55.70585098 -131.03842502  205.86076442
  -13.18380635  -94.09005633   22.96246156  -74.73237515  -29.81097527
    1.25483387    1.7797931    66.76065938   53.6112064    89.12199876
  -20.42824938  -39.50317153   27.94303935  -62.30298113  195.33554818
  -19.34463306  -51.22838781  -41.59385367  -58.5488858    10.25275547
   89.95309473  -85.48487045   71.85480944  122.48378661   25.10306946]]
[3 4 1 3 1 4 2 0 1 2]
```

1.4 Sample Keras code

1.4.1 Notes:

Must use softmax and layer must have same number of neurons as class's

y must be encoded to categorical values

1.4.2 Keras Code

```
In [95]: seed = 1
         numpy.random.seed(seed)

         # enocde values to categorical from Int.
         encoded_y = np_utils.to_categorical(y)


         # define and fit the final model
         model = Sequential()
         model.add(Dense(50, input_dim=50, activation='relu'))
         model.add(Dense(5, activation='softmax'))


         model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

         history = model.fit(X, encoded_y, validation_split=0.33,epochs=10)

         # summarize history for accuracy
         plt.plot(history.history['acc'])
         plt.plot(history.history['val_acc'])
         plt.title('model accuracy')
         plt.ylabel('accuracy')
         plt.xlabel('epoch')
         plt.legend(['train', 'test'], loc='upper left')
         plt.show()
```

1.4.3 Results

```
Train on 67000 samples, validate on 33000 samples
Epoch 1/10
67000/67000 [==============================] - 4s - loss: 4.5537 - acc: 0.6901 - val_loss: 4.373
Epoch 2/10
67000/67000 [==============================] - 4s - loss: 4.1870 - acc: 0.7065 - val_loss: 3.995
Epoch 3/10
67000/67000 [==============================] - 4s - loss: 2.4479 - acc: 0.6898 - val_loss: 2.232
Epoch 4/10
67000/67000 [==============================] - 4s - loss: 2.2429 - acc: 0.6860 - val_loss: 2.221
Epoch 5/10
67000/67000 [==============================] - 4s - loss: 2.2363 - acc: 0.6857 - val_loss: 2.243
Epoch 6/10
67000/67000 [==============================] - 4s - loss: 2.0563 - acc: 0.7017 - val_loss: 1.638
Epoch 7/10
67000/67000 [==============================] - 4s - loss: 1.6447 - acc: 0.7459 - val_loss: 1.640
Epoch 8/10
67000/67000 [==============================] - 4s - loss: 1.6401 - acc: 0.7493 - val_loss: 1.663
Epoch 9/10
67000/67000 [==============================] - 4s - loss: 1.6337 - acc: 0.7533 - val_loss: 1.633
Epoch 10/10
67000/67000 [==============================] - 4s - loss: 1.1789 - acc: 0.7687 - val_loss: 0.536
```