CAP5615.RBFNetwork

June 17, 2021

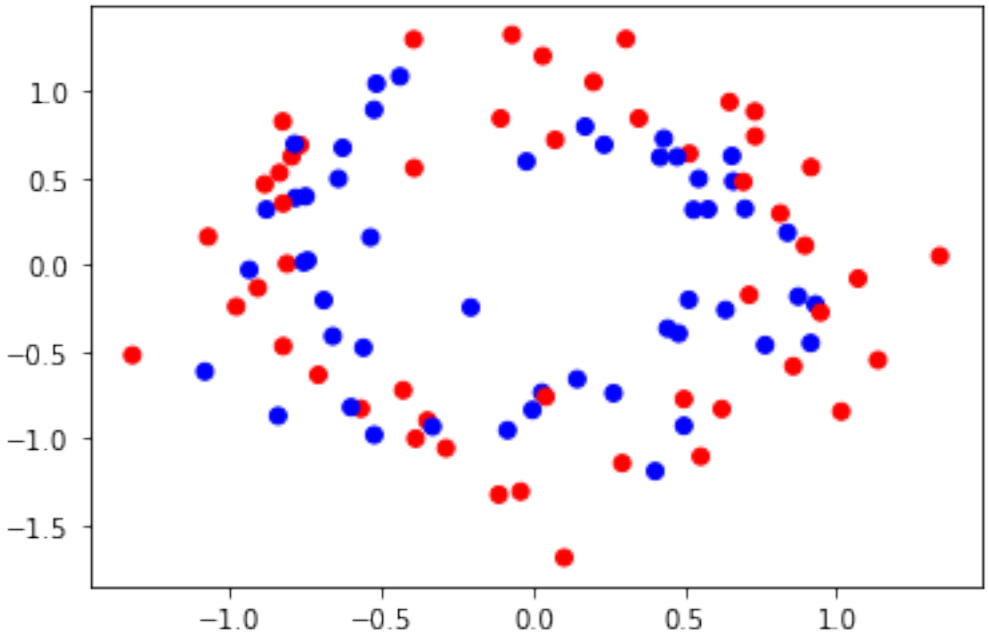
[1]:
```
                                # CAP 5615, X. Zhu, June 17 2021
                                # RBF Neural Network Learning
# some codes were adopted from https://towardsdatascience.com/
 ↪most-effective-way-to-implement-radial-basis-function-neural-network-for-classification-pro
%matplotlib inline
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from sklearn.utils import shuffle
```

[25]:
```
# create circle shaped dataset
from sklearn.datasets import make_circles, make_classification
circle =make_circles(noise=0.2, random_state=0)
features,labels=circle
#print(features)
print(labels)
```

```
[0 1 0 1 1 1 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 0 1 0 0 0 1 1 1 1 1 1 1 1 0 0
 1 1 1 0 1 0 0 0 0 0 0 0 0 1 1 0 0 0 1 0 0 0 1 0 0 1 1 1 1 0 1 0 1 0 0 0 1
 1 1 0 1 1 1 0 0 1 0 0 1 1 0 1 1 1 0 0 1 0 1 1 1 0 0]
```

[26]:
```
colors=["red","blue"]
plt.scatter(features[:,0],features[:,1],color=[colors[idx] for idx in labels])
```

[26]:
```
<matplotlib.collections.PathCollection at 0x22da3cab4c8>
```



[28]:
```
from sklearn.model_selection import train_test_split

X, y=features,labels
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.2,␣
↪random_state=42)
```

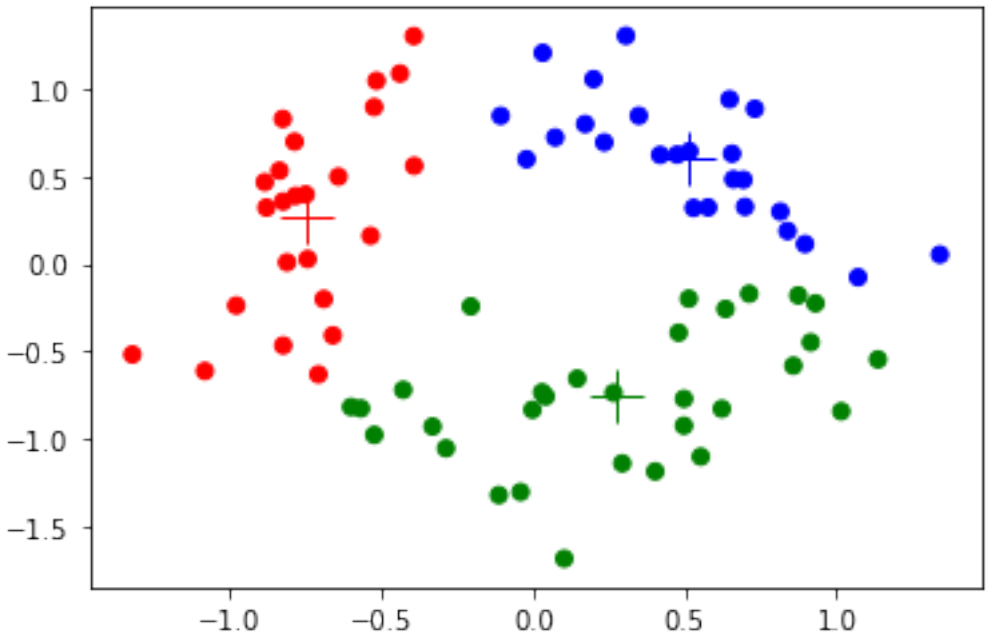
[29]:
```
def get_distance(x1, x2):
    sum = 0
    for i in range(len(x1)):
        sum += (x1[i] - x2[i]) ** 2
    return np.sqrt(sum)
```

[30]:
```
def kmeans(X, k, max_iters):
    centroids = X[np.random.choice(range(len(X)), k, replace=False)]
    converged = False
    current_iter = 0
    memberships=[0]*X.shape[0]
    while (not converged) and (current_iter < max_iters):
        cluster_list = [[] for i in range(len(centroids))]
        for i, x in enumerate(X):  # Go through each data point
            distances_list = []
            member=0
            smallest=9999999
            for j, c in enumerate(centroids):
                thisDistance=get_distance(c, x)
                distances_list.append(thisDistance)
```

```
                if(thisDistance<smallest):
                    smallest=thisDistance
                    member=j
            cluster_list[int(np.argmin(distances_list))].append(x)
            memberships[i]=member
        cluster_list = list((filter(None, cluster_list)))
        prev_centroids = centroids.copy()
        centroids = []
        for j in range(len(cluster_list)):
            centroids.append(np.mean(cluster_list[j], axis=0))
        pattern = np.abs(np.sum(prev_centroids) - np.sum(centroids))
        print("Iteration: %d, cluster center change: %f" %␣
↪(current_iter,pattern))
        converged = (pattern == 0)
        current_iter += 1
    return np.array(centroids), [np.std(x) for x in cluster_list], memberships
```

[63]:
```
k=3
centroids, std, clusters=kmeans(X_train,k,100)
colors=["red","blue","green","pink","brown","black","yellow"]
plt.scatter(X_train[:,0],X_train[:,1],color=[colors[idx] for idx in clusters])
for i in range(centroids.shape[0]):
    plt.plot(centroids[i][0],centroids[i][1],"+",markersize=20,color=colors[i])
```

```
Iteration: 0, cluster center change: 1.500340
Iteration: 1, cluster center change: 0.442540
Iteration: 2, cluster center change: 0.061341
Iteration: 3, cluster center change: 0.000000
```



[64]:
```
dMax = np.max([get_distance(c1, c2) for c1 in centroids for c2 in centroids])
std = np.repeat(dMax / np.sqrt(2 * k), k)
```

[65]:
```
def rbf_single(x, c, s):
    distance = get_distance(x, c)
    return 1 / np.exp(-distance / s ** 2)

def rbf_list(X, centroids, std_list):
    RBF_list = []
    for x in X:
        RBF_list.append([rbf_single(x, c, s) for (c, s) in zip(centroids,␣
 ↪std_list)])
    return np.array(RBF_list)
```

[66]:
```
rbf_values=rbf_list(X_train,centroids,std)
# add bias 1s to the first column
rbf_values=np.c_[np.ones(rbf_values.shape[0]),rbf_values]
rbf_values
```

[66]:
```
array([[  1.        , 153.44263141,  14.34127693,  11.44860553],
       [  1.        ,  46.34933521,  80.34351136,   2.27410745],
       [  1.        ,  43.93890979,   1.14264701,  59.98743924],
       [  1.        , 231.40501693, 134.77208601,   3.62716784],
       [  1.        , 209.80780283, 173.35361255,   3.67380182],
       [  1.        ,   8.49859288, 137.54531068,  26.20174894],
       [  1.        , 420.1793111 ,  17.45567456,  47.70149585],
       [  1.        ,   2.0525037 ,  28.16934916,  87.66805232],
       [  1.        ,  55.27643203,  17.45250402,   3.3297819 ],
       [  1.        , 205.46197953,  12.49475945,  20.57352259],
       [  1.        , 143.84386211,  80.80301936,   2.2597271 ],
       [  1.        , 156.6720668 ,  61.85489726,   2.80879639],
       [  1.        ,  10.28885453,  25.41833664, 290.48364571],
       [  1.        ,   8.54204979,  24.20298083,   7.51791695],
       [  1.        ,  38.74476898, 229.59939642,  10.81498909],
       [  1.        ,  70.3172479 ,  12.07607214,   5.89986951],
       [  1.        ,  20.06899544, 106.44713707,   7.53911735],
       [  1.        ,  23.76969949, 179.69709702,  12.42685827],
       [  1.        ,   5.04386002, 137.92006503,  49.40984134],
       [  1.        ,   2.2290096 ,  48.94285434, 134.06441634],
       [  1.        ,  65.62259337,   1.86062723,  43.20865854],
       [  1.        ,  96.1420784 ,   4.52920556,  23.88104931],
       [  1.        ,   2.01699225,  53.22220799,  40.27585616],
       [  1.        ,  47.78755726,   9.94238597,   5.74553501],
       [  1.        , 183.29101429,  25.34536207,   7.80901396],
```

```
[  1.        , 100.00119931,   2.84717423, 136.51586715],
[  1.        ,  41.7534427 ,  62.69791294,   1.94123663],
[  1.        , 368.60266239,  42.88509337,  13.01473634],
[  1.        ,   6.75855023,  22.39213116, 197.62282405],
[  1.        ,  69.78566717,   8.35596051, 379.78255111],
[  1.        , 404.39319334,  81.50035515,   8.675344  ],
[  1.        ,  45.04089536,   2.25656735,  25.29271101],
[  1.        ,   2.18512521,  65.07540623,  45.64151394],
[  1.        ,  12.3664611 ,  21.86805816, 298.02905856],
[  1.        ,  59.81144967,   1.71322977,  42.28262283],
[  1.        ,   1.52077547,  59.64016879,  94.1007001 ],
[  1.        ,   3.71231749,  13.57293405,  70.03052451],
[  1.        ,  15.32389037, 320.59268484,  50.60692516],
[  1.        ,   5.08490717,  50.01686979, 258.6966013 ],
[  1.        ,   3.45425004,  42.7139314 , 178.49996529],
[  1.        ,  33.19759328,   9.35931915, 298.60016113],
[  1.        , 106.89733098,  51.84988605,   1.92070445],
[  1.        ,  24.67381553, 173.03049447,  11.34547521],
[  1.        , 156.08953879, 156.3487016 ,   3.04692459],
[  1.        ,  38.69969196,  59.46000059,   2.01564549],
[  1.        ,  38.91162242, 153.22840159,   6.11506386],
[  1.        ,  23.10733144,  27.33474309, 510.41064791],
[  1.        ,  11.85600874,   6.84267561, 115.07484033],
[  1.        ,  13.54311488, 147.47360549,  17.22135195],
[  1.        ,  21.56658207,   2.33448837,  65.52297386],
[  1.        ,   3.93470982,  64.11500887,  24.53119032],
[  1.        ,  33.11036183,   1.32070249,  54.42524775],
[  1.        ,   2.00942421,  57.00590691, 127.90154043],
[  1.        , 139.86743686, 337.54854314,   7.21229701],
[  1.        ,  80.21657256,   9.8712591 ,   8.25178945],
[  1.        , 144.5412402 , 302.51329535,   6.19199471],
[  1.        ,   1.43037474,  43.93184609,  88.7599807 ],
[  1.        ,  61.06706052,  50.16038825,   1.05502271],
[  1.        ,  85.65302518,   2.90193605, 149.91343094],
[  1.        ,  63.96480409,   2.58424019,  28.55936616],
[  1.        ,  39.24365827,   2.22956814,  24.23494248],
[  1.        , 466.83837306, 802.91863053,  15.28338466],
[  1.        ,  64.24877925,   1.52939686,  63.13215554],
[  1.        ,   1.9669886 ,  26.39393743,  33.78892996],
[  1.        ,  40.53815537,  43.23269933,   1.58805232],
[  1.        ,  89.26950845,   3.4091782 ,  30.54302048],
[  1.        ,  21.01734364,   3.13172379,  89.38403087],
[  1.        , 115.28514667,   5.94090421,  21.87083702],
[  1.        ,   1.43807782,  39.78769151,  84.71666764],
[  1.        ,  14.70995053,   3.73073855,  73.73554147],
[  1.        ,  56.97544439, 198.84741173,   6.22405756],
[  1.        , 126.43331726,  11.87200019,  11.02747933],
```

```
       [  1.        ,  38.67853749,   1.14263173,  55.81552205],
       [  1.        ,   7.20836601,  86.01347614,  17.56622106],
       [  1.        , 186.78315949,  34.64354534,   5.82524823],
       [  1.        ,  34.33318183,   4.95968268, 186.39912186],
       [  1.        ,  16.79083431, 486.19985965, 103.7620828 ],
       [  1.        ,  35.08294312,   2.36304039, 102.01069872],
       [  1.        ,   1.40535167,  50.10920251,  90.08954109],
       [  1.        ,   9.81004897,   4.65798476,  54.26692919]])
```

[67]:
```
rbf_pseudo_inverse=np.linalg.pinv(rbf_values)
print(rbf_pseudo_inverse.shape)
rbf_pseudo_inverse
```

```
(4, 80)
```

[67]:
```
array([[ 1.44513006e-02,  2.41046540e-02,  2.06018710e-02,
         2.66739653e-04,  5.69047029e-04,  2.18589946e-02,
        -2.06085898e-02,  1.99269468e-02,  2.63896562e-02,
         7.45960625e-03,  1.31869932e-02,  1.27100914e-02,
        -9.28161100e-03,  3.06565557e-02,  1.56152118e-02,
         2.46418112e-02,  2.48792223e-02,  1.97846837e-02,
         1.89704300e-02,  1.22684953e-02,  2.04923184e-02,
         1.96463895e-02,  2.52095292e-02,  2.72974404e-02,
         1.10259470e-02,  3.51355730e-03,  2.56274362e-02,
        -1.13685595e-02,  4.29864803e-03, -2.75206585e-02,
        -1.68659400e-02,  2.52827653e-02,  2.37918125e-02,
        -1.03781106e-02,  2.12795762e-02,  1.73677201e-02,
         2.30107633e-02,  7.68936152e-03, -5.58480585e-03,
         6.24072645e-03, -1.21036528e-02,  1.89424945e-02,
         2.01989323e-02,  7.58732835e-03,  2.61348692e-02,
         2.04214787e-02, -4.16561164e-02,  1.61518734e-02,
         2.20129257e-02,  2.22606963e-02,  2.66087229e-02,
         2.25821191e-02,  1.27173494e-02, -1.06810377e-03,
         2.33260913e-02,  4.63792210e-04,  1.89835439e-02,
         2.42773810e-02,  3.23535011e-03,  2.26921811e-02,
         2.60803728e-02, -6.41225731e-02,  1.78702868e-02,
         2.75881301e-02,  2.68745548e-02,  1.95413431e-02,
         1.89328521e-02,  1.77120912e-02,  1.97756816e-02,
         2.17991901e-02,  1.58992025e-02,  1.76632240e-02,
         2.17746715e-02,  2.60249917e-02,  1.04066730e-02,
         3.74196202e-03, -8.96140884e-03,  1.56325331e-02,
         1.84629491e-02,  2.50259893e-02],
       [ 1.27298485e-04, -6.72244063e-05, -2.18358376e-05,
         2.03273447e-04,  1.55664103e-04, -1.38351943e-04,
         5.56545100e-04, -8.71062900e-05, -3.03239217e-05,
         2.12629782e-04,  8.51414420e-05,  1.12204522e-04,
        -2.89577312e-07, -1.04357110e-04, -1.29576253e-04,
```

```
 -3.93926468e-06, -1.15809149e-04, -1.34530031e-04,
 -1.35544529e-04, -7.75931449e-05,  5.79746077e-06,
  4.56386582e-05, -1.13191157e-04, -3.84785937e-05,
  1.68742736e-04,  9.27895030e-05, -6.82066351e-05,
  4.54255701e-04, -3.81256690e-05,  1.31032405e-04,
  4.94843232e-04, -3.29893669e-05, -1.15249081e-04,
  6.94790692e-06, -3.57460376e-06, -9.69106695e-05,
 -8.56183263e-05, -1.84539979e-04, -2.86853402e-05,
 -5.74608569e-05,  4.42306738e-05,  3.75976470e-05,
 -1.31113652e-04,  7.74915885e-05, -7.17964689e-05,
 -1.03617546e-04,  9.81765412e-05, -5.42629884e-05,
 -1.37249819e-04, -5.52751601e-05, -1.19759800e-04,
 -4.08420563e-05, -8.30447717e-05, -1.13722470e-05,
  1.31853679e-05,  8.13750001e-06, -9.33396427e-05,
 -3.38119169e-05,  7.51398369e-05, -2.32445694e-06,
 -4.24302292e-05,  3.36205363e-04,  1.09324456e-05,
 -1.05981354e-04, -6.32544712e-05,  3.76838081e-05,
 -4.78385990e-05,  7.43598461e-05, -9.32956254e-05,
 -6.35497743e-05, -9.16762529e-05,  8.57749221e-05,
 -3.15664442e-05, -1.24996156e-04,  1.70158429e-04,
  7.23134655e-06, -2.22516987e-04, -2.10152334e-05,
 -9.51159192e-05, -7.85504888e-05],
[-9.40487931e-05,  7.38073673e-06, -6.10146159e-05,
 -5.79511743e-06,  3.95280967e-05,  7.99213911e-05,
 -1.81730093e-04, -1.59003582e-05, -5.69216793e-05,
 -1.13258046e-04, -2.71370347e-05, -5.01147851e-05,
  6.11697224e-06, -3.30171242e-05,  1.56621570e-04,
 -6.72009151e-05,  4.29435217e-05,  1.13615473e-04,
  8.46892475e-05,  1.05913906e-05, -7.03790542e-05,
 -8.13601102e-05,  2.04848761e-06, -6.12202429e-05,
 -9.45330813e-05, -6.90264551e-05, -8.20022265e-06,
 -1.43198820e-04, -8.22379974e-06, -1.96612423e-05,
 -1.19020983e-04, -6.50550081e-05,  1.42626540e-05,
  2.94322372e-06, -6.85648603e-05,  1.58144113e-05,
 -3.31145117e-05,  2.59057252e-04,  2.76043430e-05,
  1.01440487e-05, -1.66306758e-05, -4.21288869e-05,
  1.06651702e-04,  4.21471400e-05, -1.02481128e-05,
  8.15489016e-05,  3.33691399e-05, -3.64481545e-05,
  8.65564637e-05, -5.10761854e-05,  9.82197076e-06,
 -5.77162075e-05,  1.76819601e-05,  2.24989726e-04,
 -7.25774566e-05,  1.89056270e-04, -1.78453174e-07,
 -2.74566238e-05, -6.20011106e-05, -7.10770197e-05,
 -6.31465137e-05,  5.62022052e-04, -6.74927513e-05,
 -2.49440255e-05, -2.67682864e-05, -7.90779622e-05,
 -4.68498492e-05, -8.71244638e-05, -4.76813124e-06,
 -4.61379416e-05,  1.19510743e-04, -8.68248262e-05,
 -5.96969281e-05,  2.90237421e-05, -8.70010603e-05,
```

```
 -3.66188744e-05,  4.27050502e-04, -5.09211848e-05,
  6.02745319e-06, -4.61319463e-05],
[-6.51546190e-05, -1.09034371e-04, -3.03583861e-05,
 -3.29396449e-05, -3.53743651e-05, -7.73945861e-05,
  8.79512502e-05,  1.59078850e-06, -1.12745196e-04,
 -3.24137595e-05, -7.39297160e-05, -7.10397121e-05,
  3.21651431e-04, -1.22077396e-04, -7.80519022e-05,
 -1.04045610e-04, -1.06686048e-04, -8.77179000e-05,
 -4.22576768e-05,  7.71119857e-05, -4.87261672e-05,
 -6.76399289e-05, -6.91909538e-05, -1.12681026e-04,
 -5.86164266e-05,  1.09829432e-04, -1.13614007e-04,
  1.85719579e-05,  1.74611293e-04,  4.80506005e-04,
  2.99161197e-05, -8.41189047e-05, -5.91154981e-05,
  3.33725708e-04, -5.22858284e-05,  1.57592156e-05,
 -2.74107525e-05, -1.17826356e-05,  2.73376110e-04,
  1.46259776e-04,  3.40406392e-04, -9.16956591e-05,
 -8.99943152e-05, -5.79940425e-05, -1.15037261e-04,
 -9.57605589e-05,  6.70781389e-04,  4.51098765e-05,
 -8.82840475e-05, -2.95773084e-05, -9.16618145e-05,
 -4.29353350e-05,  6.84853163e-05, -3.26050925e-05,
 -9.71043073e-05, -3.72976287e-05,  5.22518089e-06,
 -1.09764213e-04,  1.25648102e-04, -7.21547350e-05,
 -8.78633557e-05,  1.61070328e-04, -1.80785674e-05,
 -8.30206450e-05, -1.17257654e-04, -5.98361996e-05,
  7.68428284e-06, -6.37091924e-05, -1.66614646e-06,
 -1.89976044e-05, -8.28738975e-05, -7.58645293e-05,
 -3.87807434e-05, -9.84014635e-05, -5.91980735e-05,
  1.64583297e-04,  9.45277333e-05,  3.24031772e-05,
  8.13954749e-06, -5.11082852e-05]])
```

[68]:
```
w=np.matmul(rbf_pseudo_inverse, y_train)
w
```

[68]:
```
array([ 7.46576989e-01, -1.54515747e-03, -5.83893980e-04, -1.36687172e-03])
```

[69]:
```
testRBF=rbf_list(X_test, centroids, std)
test_rbf_values=np.c_[np.ones(testRBF.shape[0]),testRBF]
testY=np.matmul(test_rbf_values,np.matrix(w).T)
testY
```

[69]:
```
matrix([[0.46361482],
        [0.55187273],
        [0.5069854 ],
        [0.07727423],
        [0.45102711],
        [0.6552704 ],
        [0.48529688],
```

```
       [0.52689401],
       [0.62316234],
       [0.61899794],
       [0.60032356],
       [0.58568582],
       [0.53218807],
       [0.54638714],
       [0.58764404],
       [0.60756956],
       [0.48401216],
       [0.65762069],
       [0.66909713],
       [0.48686146]])
```

[70]:
```
pred = [1 if a_ > 0.5 else 0 for a_ in testY]
print(pred)
```

```
[0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0]
```

[71]:
```
# use scikit-learn to calculate accuracy.
from sklearn.metrics import accuracy_score
accuracy_score(pred, y_test)
```

[71]: 0.7

[72]:
```
# now let's compare the RBF classifier with a Multi-Layer NN.
# It shows that RNF netowrk is much more accurate than Multi-Layer NN.
from sklearn.neural_network import MLPClassifier
clf = MLPClassifier(solver='lbfgs', hidden_layer_sizes=200,␣
 ↪random_state=12,activation='logistic',max_iter=500)
clf.fit(X_train, y_train)
y_pred=clf.predict(X_test)
accuracy_score(y_pred, y_test)
```

[72]: 0.55