8E_F_Solution

April 10, 2022

0.0.1 Task E : Implementing Decision Function of SVM RBF Kernel

[1]:
```
import numpy as np
import pandas as pd
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
```

[2]:
```
X, y = make_classification(n_samples=5000, n_features=5, n_redundant=2,
                           n_classes=2, weights=[0.7], class_sep=0.7,␣
 ↪random_state=15)
```

[3]:
```
# split test train and cross validation data
X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.4,␣
 ↪random_state=42)
X_cv,X_test, y_cv, y_test = train_test_split(X_test, y_test, test_size = 0.5,␣
 ↪random_state=43)

print("XTrain shape",X_train.shape )
print("XTest shape",X_test.shape )
print("XCv shape",X_cv.shape )

# fit RBF svc to Xtrain data
svc_clf = SVC(gamma=0.001, C= 100)
svc_clf.fit(X_train,y_train)

dual_coefs = svc_clf.dual_coef_
intercept = svc_clf.intercept_
support_vectors = svc_clf.support_vectors_
gamma = 0.001
```

```
XTrain shape (3000, 5)
XTest shape (1000, 5)
XCv shape (1000, 5)
```

[4]:
```
def rbf(xi,xq, gamma):
    """
    Function to get RBF kernel value
```

```
    """
    #   (- || - ||2)

    X_norm = np.sum((xi-xq) **2,axis= -1)
    return np.exp(-gamma * X_norm)
```

[5]:
```
def getKernel(supportVectors, X, gamma):
    """
    Function to get Kernel Matrix
    Kernel used : RBF

    """
    Kernels = np.zeros((X.shape[0], supportVectors.shape[0]))
    for id, point in enumerate(X):
        for idx, vector in enumerate(supportVectors):
            rbf_ = rbf(point, vector, gamma)
            Kernels[id][idx] = rbf_

    return Kernels
```

[6]:
```
def decision_function_custom(X, intercept, dual_coeff, support_vector, gamma ) :
    """
    return decison function for sum rbf kernel.

    parameters:
      X : Data
      intercept :  intercept value of classfier
      dual_coeff : dual values( alpha * y)
      support_vector : array of  support vector

    returns:
      decision function : sum_all_supoort_vectors(yi*alpha_i * Kernel(xi,xq) )␣
 ↪+ intecept
     """
    Kernels = getKernel(support_vector, X,  gamma)
    decision_custom = np.sum(dual_coeff * Kernels, axis = -1) + intercept
    return decision_custom
```

[7]:
```
f_cv = decision_function_custom(X_cv,intercept, dual_coefs, support_vectors, 0.
 ↪001 )

f_cv
```

[7]:
```
array([-4.54631892e+00, -3.18769119e+00,  1.62139697e+00,  8.61360038e-02,
        1.75246241e+00, -9.76830808e-01, -3.20246796e+00, -2.62082863e+00,
       -2.42937891e+00,  1.57993649e+00, -2.06329703e+00,  9.02139954e-01,
```

```
-2.49633212e+00, -3.12153041e+00,  2.98793441e-01, -9.79851167e-02,
-2.10058057e+00, -3.05987657e+00,  6.09426723e-01, -2.28247363e+00,
 1.83705137e+00, -1.49963660e+00,  1.64502533e+00,  1.74774640e+00,
 8.64699998e-01, -2.44402367e+00, -2.77765859e+00,  2.81547371e+00,
 5.69770433e-01, -2.75397962e+00, -3.38300646e+00, -2.98418538e+00,
-3.96622930e+00, -3.17731359e+00, -2.46000821e+00, -2.51646933e+00,
-3.78091581e+00, -2.92728890e+00,  1.55489810e+00,  1.38727315e+00,
 1.44977701e+00,  9.80246803e-01, -3.50913018e+00,  3.68527290e+00,
 9.93865029e-01,  1.64063405e+00,  1.44693130e+00, -3.46194602e+00,
-2.92257250e+00, -3.04151815e-01, -2.88716371e+00, -2.01001616e+00,
-2.96415173e+00, -3.93243751e+00, -3.16152581e+00, -3.16365721e+00,
 2.31341730e+00, -3.37939147e+00, -7.22598505e-01,  2.08339272e+00,
-2.13517154e+00,  1.48945168e+00, -1.96099648e+00,  1.81353239e+00,
-1.72974263e+00, -4.05107267e+00, -1.25869622e+00, -3.50988882e+00,
 2.22264304e+00, -1.55737048e+00,  1.82902013e+00, -3.13187903e+00,
-2.06958284e+00,  3.06721490e-01, -3.38649015e+00, -2.16142201e+00,
 1.75969650e+00, -2.31379348e+00,  1.98058589e+00,  1.73120299e+00,
 1.75251759e+00,  1.89470290e+00,  9.37506327e-01, -4.33277042e+00,
-8.84867047e-01, -3.83688018e+00, -2.32607846e+00,  1.75091747e+00,
 1.81344348e+00, -5.26804576e+00, -2.10552492e+00, -3.01532258e+00,
 9.77028455e-01, -2.99306948e+00, -2.43721097e+00,  1.71194516e+00,
-2.68862936e+00,  1.76309636e+00,  1.82568324e+00, -2.32946397e+00,
-2.78524770e+00, -2.46524122e+00,  1.78353974e+00,  1.44476848e-01,
-1.86125683e+00, -4.89620508e-01, -1.71939959e-01, -1.68411036e-01,
-1.66184081e+00, -1.88403070e+00,  2.84558085e-01,  7.60389202e-01,
-3.38448657e+00, -1.78960898e+00, -4.49608093e+00, -1.67717426e+00,
 2.01078485e+00, -3.24420133e+00, -2.44425550e+00, -8.32302953e-01,
-3.80067326e+00,  2.92187317e+00, -2.27471347e+00,  1.44519856e+00,
-6.10299350e-01, -8.10399386e-01,  1.39930514e+00, -2.44363306e+00,
-4.33871064e+00,  1.26413080e+00, -2.32283229e+00, -3.17756645e+00,
 1.29668096e+00, -1.89433211e+00, -2.46762695e+00, -4.64510217e+00,
-4.12305208e-01, -2.30221061e+00, -1.28337917e+00, -1.13341996e+00,
 1.88327632e+00, -2.77483417e+00, -2.74351358e+00, -2.92405747e+00,
 2.15576025e+00, -2.17133620e+00, -2.89823110e+00,  2.34848980e-01,
-2.36128079e+00, -1.20595954e-01,  2.26492574e-01,  1.79860602e-02,
-3.09164897e+00,  1.98784221e+00,  3.00951053e+00, -2.57173094e+00,
-4.08687293e+00, -1.38144148e+00, -3.68924631e+00, -2.52663321e+00,
-2.27129864e+00,  1.83053884e+00, -2.96901209e+00,  2.33615089e-01,
-2.87944791e+00, -2.86476077e+00,  1.34041328e-02, -4.22685012e+00,
-2.92031302e+00, -1.78774590e+00,  3.40975902e-01, -3.66875724e-01,
-2.71829230e+00, -6.61163302e-01, -4.76658862e-01, -2.68972218e+00,
-3.75746824e+00, -1.79295670e+00,  1.37937263e+00, -1.40369265e+00,
-2.52007920e+00, -2.13958712e+00, -2.07428714e-01,  2.68274654e+00,
-2.85560108e+00,  5.25195524e-01, -2.04438572e+00, -1.93410546e+00,
-5.80208441e-02, -8.30118864e-01, -3.41838481e+00, -1.53916924e+00,
-1.93610575e+00,  1.10226541e+00, -7.72263440e-01,  1.10945083e+00,
-1.48060670e-01, -1.73872187e+00, -9.84777293e-01, -2.79466785e+00,
```

```
 1.82479028e+00, -2.37274019e+00,  1.73351606e+00, -2.66416873e+00,
-2.44470515e+00,  2.50004313e+00, -1.75564036e+00, -3.04880509e+00,
 1.57049054e+00,  1.23346888e+00,  1.65834290e+00, -2.94252342e+00,
-2.37528423e-01, -2.19200870e+00,  2.96635753e+00, -6.68194993e-01,
-2.39978766e+00, -2.90690886e+00,  1.82604402e-01,  1.81687850e+00,
-1.56902548e+00, -2.22859840e+00,  1.81698322e+00, -2.46036751e+00,
-4.79780015e-01, -3.33846107e+00, -2.40029149e+00, -3.52491139e+00,
-3.16735485e+00, -6.80718770e-01, -2.76595359e+00, -3.31962980e+00,
-1.94469317e+00,  9.39096907e-01, -2.41026998e+00, -3.53264473e+00,
-1.52707034e+00, -1.92893631e+00, -2.52987398e+00, -2.85032955e+00,
-5.34712291e-01, -2.78008703e+00, -1.48777741e+00, -3.01310854e-01,
-1.30444992e+00, -1.30274552e+00, -7.44971137e-01,  5.15627258e-02,
-1.64445728e+00,  1.87282731e+00, -2.15121414e+00, -1.30560938e+00,
-1.97840469e+00, -2.95560858e-01, -2.16564440e+00, -7.58381669e-01,
 1.64260132e+00, -1.87402295e+00, -2.10491370e-03, -3.82842356e+00,
 1.69281933e+00, -3.32872585e+00, -1.68103038e+00, -1.75443379e+00,
 9.21764764e-01, -1.52593144e+00, -1.75470831e+00,  1.96423413e+00,
-2.35225233e+00, -2.19599501e+00, -2.00260604e+00, -2.96102699e+00,
 1.59344293e+00, -3.26142963e-01, -3.72478152e+00,  1.79203133e+00,
-3.60259332e+00,  1.28090101e+00, -2.64684348e+00, -2.43057072e+00,
-4.02361158e+00, -3.21749646e+00,  1.23887126e-01,  1.80762390e+00,
 6.62413698e-01, -2.81953595e+00, -1.58968072e+00, -1.53967081e+00,
-4.50322365e+00, -1.70427854e+00, -1.85578405e+00, -1.34068256e+00,
-3.43100398e+00,  7.98384557e-01, -2.28419240e+00, -3.77082766e+00,
-2.01040500e+00, -1.80044358e+00, -5.52219073e-01, -7.62619087e-01,
-2.01756085e+00,  1.03889617e+00, -3.22549864e+00, -5.51423949e-01,
-3.24159684e+00, -2.40683080e+00, -1.55033474e+00,  2.07864248e-01,
-1.20285240e+00,  3.25281077e-01, -5.34529437e+00,  1.77726140e+00,
 4.04199906e-02, -2.73072293e+00,  1.64055352e+00, -2.73431955e+00,
 2.12489789e+00, -3.58611141e+00, -6.17008444e-02,  6.05813087e-01,
-3.71219198e+00,  1.25207707e+00,  1.18429418e+00, -1.84837016e+00,
-4.18853872e-01, -3.72096151e+00, -2.06877948e+00, -1.91195983e+00,
-3.25252106e+00, -2.80613616e+00, -3.19895369e-01, -2.36674318e+00,
-2.81646163e+00, -1.79742715e+00,  1.54081305e+00,  6.88782910e-01,
-5.10481929e-01, -2.50990580e+00, -3.91212814e+00, -2.45423153e+00,
-6.85421676e-01, -2.89223785e+00, -2.32775110e+00, -7.47198770e-01,
 1.47472073e+00,  1.84830140e+00, -1.09362115e+00, -9.42317690e-01,
-2.99428467e+00,  4.32773525e-01, -3.58481569e+00, -1.76324902e+00,
-2.63077891e+00, -2.91103377e+00,  2.23371041e+00, -3.75129300e+00,
-3.80078641e-01,  1.81005227e+00, -2.10229778e+00, -4.29005602e+00,
-1.30267126e+00, -1.89181695e+00, -3.13779527e+00, -2.61379250e+00,
-2.20336036e+00,  1.72168922e+00,  2.37955219e+00, -2.30547772e+00,
-1.74705476e+00, -3.48216615e+00, -2.07998495e+00, -1.98229211e+00,
-2.97829337e+00,  1.68477402e+00,  7.65206496e-01, -1.30756006e+00,
-2.83549472e+00,  1.09874870e+00, -1.39177302e+00, -1.66907800e+00,
 1.58557650e+00,  4.03992688e-01, -2.16709701e+00, -7.53486537e-01,
-2.95711784e+00,  1.45819645e+00,  9.65456768e-01, -1.13975876e+00,
```

```
-8.11838816e-01, -3.17095908e+00,  1.93038695e+00, -3.62598180e+00,
 1.25639610e+00, -2.98321004e+00, -2.92128702e+00, -3.42941102e+00,
-3.86284114e+00, -4.78368449e+00, -2.54171540e+00, -1.22475282e+00,
-4.35622229e+00, -3.05221486e+00, -2.49422774e+00, -2.54720692e+00,
 1.32751096e+00, -1.31965974e+00, -2.31709391e+00, -1.05876547e+00,
-2.36370348e+00,  1.51524755e+00,  1.87286517e+00, -3.53528338e+00,
-1.60709826e+00, -1.45866564e+00, -2.76034336e+00, -3.64579332e+00,
-2.14636798e+00, -3.49618838e+00,  1.07398967e+00, -3.58968697e+00,
 1.77868215e+00, -1.91034931e+00,  1.69862289e+00, -3.03165880e+00,
-1.64682919e+00, -4.45967337e+00, -3.41419216e-02,  5.52911670e-01,
-2.82225582e+00,  2.10298272e+00, -2.93216117e+00, -2.55616872e+00,
-2.89731760e+00, -1.81756178e+00,  3.54705420e-01, -2.43112434e+00,
-5.61401544e-01,  1.71806899e+00, -1.44505902e+00, -1.82164118e+00,
-1.56398166e+00, -1.64280990e+00, -2.41638001e+00,  2.43697920e+00,
-2.99343122e+00, -1.93438887e+00, -2.71088439e+00, -1.67345875e+00,
-1.86944096e+00, -2.25710762e+00, -2.00161749e+00, -1.89469115e+00,
-3.41949887e+00, -2.50229845e+00, -3.25967351e+00, -1.07480503e+00,
-3.41416378e+00,  2.33639072e+00, -2.44709942e-01,  7.97894898e-01,
-3.66288643e-03, -6.35759948e-01, -1.80647541e+00,  1.18919045e+00,
-2.67457989e+00, -1.33695416e+00,  2.35752685e+00,  1.48780415e+00,
-1.43531917e+00, -2.72964053e+00, -2.13266993e-02, -1.08073218e+00,
-3.01869027e+00,  1.19871477e+00, -1.77454115e+00, -2.75148690e+00,
-1.83729625e+00,  1.59674517e+00,  3.32550033e-01, -4.58518812e-01,
-1.62141228e+00, -4.01820784e+00,  1.74943003e+00, -7.12222868e-01,
-2.72389261e+00, -1.93729550e+00, -2.00448345e-04, -3.43170557e+00,
 1.45118265e+00, -2.22058122e+00,  2.39633559e+00,  8.49061719e-01,
-1.26292866e+00, -2.66109621e+00,  8.52445153e-01,  1.05834618e+00,
-3.13462257e+00, -1.24727167e+00,  1.50665227e+00, -6.36807038e-01,
 1.75580944e+00,  2.66657775e+00, -3.03408055e+00, -1.63248780e-01,
-3.26709092e+00, -3.92158794e+00, -1.18892593e+00, -1.87040532e+00,
-2.86526958e+00,  1.99037794e+00, -2.55196545e+00,  1.72932059e+00,
 1.99260181e+00, -3.14441691e+00,  9.92716673e-01, -3.25166265e+00,
 9.18897600e-02, -2.98090170e+00, -3.44504772e+00, -2.58442937e+00,
-9.51065875e-01, -8.09729341e-01, -1.57583176e+00, -3.76072116e-01,
-2.85257678e+00, -1.88770800e+00, -1.75383651e+00,  1.06008099e+00,
-2.87075778e+00, -4.04604819e+00,  1.01394350e+00,  1.84867529e+00,
 7.89333472e-01,  2.07576238e+00, -2.62512408e+00,  1.29024045e+00,
-2.54202735e+00,  1.80156181e+00, -2.07771943e+00,  1.34747713e+00,
-3.02288176e+00, -7.77275620e-01, -8.30681294e-01, -3.13849700e+00,
 2.17167703e+00,  1.10070039e+00,  3.53867103e+00, -2.73126333e+00,
-2.05065375e+00,  1.36906054e+00, -2.14768136e+00, -2.66889832e+00,
 4.34565156e-01, -3.45612820e+00, -3.03607734e+00, -1.32572145e+00,
-3.17747113e+00, -2.32006812e+00, -3.17623158e+00,  7.47181519e-01,
-3.58719601e-01, -2.54908301e+00, -3.53258150e-01, -3.37735582e+00,
-3.42738872e+00, -2.73502748e+00, -1.67832030e+00, -4.57087922e+00,
 9.67039810e-01, -2.87230861e+00, -7.65204842e-01, -3.59403044e+00,
-1.71219037e+00, -2.13336602e+00,  4.13254577e-01, -3.97512625e+00,
```

```
-2.58386978e+00, -2.96788233e+00,  6.35579838e-01, -3.64223584e+00,
-3.65382935e+00, -2.19467239e+00, -2.44588475e+00, -3.44518453e+00,
 1.88614198e+00, -2.44199799e+00, -3.11045817e+00, -2.00910731e+00,
-5.16253447e+00, -6.41486536e-01, -3.00252577e+00, -7.53621352e-01,
-2.74089896e+00, -1.28262565e+00, -3.04354172e+00, -1.78329090e+00,
-1.66755562e+00, -2.64814843e+00, -2.40259951e+00,  1.75487036e+00,
-3.12857015e-01,  2.20644629e+00,  1.22813526e+00, -1.87725993e+00,
-6.08448784e-01,  1.75603265e+00,  2.31147552e+00, -1.57536800e+00,
 4.59300756e-01, -3.03052048e-01,  1.54555106e+00,  1.91148154e+00,
-4.04793185e+00, -2.32267250e+00, -1.32629300e+00, -2.00323940e+00,
-1.44365097e+00,  4.28186127e-01,  1.92103636e+00,  1.50816099e+00,
 6.59779614e-01,  2.15838653e+00, -1.63049275e+00,  5.69849137e-01,
-2.35621402e+00, -9.23419924e-01, -3.78069468e+00,  1.09004549e-01,
 2.08930318e+00,  1.12675140e+00,  1.60862133e+00, -3.83461125e+00,
-2.63219475e+00,  6.38285420e-01,  1.59935092e+00, -7.02602246e-01,
 1.62259762e+00,  2.21652996e+00, -1.48002156e+00, -1.44617674e+00,
 1.47026887e+00, -1.25698939e+00, -7.87747063e-01,  1.64232780e+00,
 1.47013352e+00, -2.92021833e+00,  3.96634429e-01,  2.47769940e+00,
-4.90263694e+00, -2.18390668e+00, -2.36522775e+00, -2.66779758e+00,
 1.88190780e+00,  1.40708781e+00, -5.25238477e-01, -3.36837041e-01,
 2.25571597e+00, -1.88670262e+00, -2.95632739e+00, -2.28651213e+00,
 8.28309796e-01, -2.54513113e+00, -2.57592289e+00,  2.30388964e+00,
-2.45583004e+00,  1.84775886e+00,  1.05241161e+00, -2.12214718e+00,
-2.84355245e+00, -2.17200033e+00, -3.25246559e+00, -1.58379364e+00,
-2.80475988e+00, -1.46435551e+00, -2.74131943e+00, -2.41333685e+00,
 1.98576933e+00, -2.35342003e+00, -1.73040902e+00, -3.19191204e+00,
 9.93261580e-01, -1.89552250e+00, -1.15878192e-01,  1.86374549e+00,
 1.37136099e+00,  5.64042745e-01,  1.08615971e+00,  1.74560623e+00,
 1.89983889e+00,  2.23499903e+00,  6.70075551e-01, -2.93231003e+00,
 1.37797763e+00, -2.87850322e+00, -2.10472728e+00, -2.45713506e+00,
-2.06002927e+00, -2.64844361e+00, -3.11876985e+00,  2.02883690e+00,
-3.81084700e+00, -2.79791391e+00, -2.68297725e+00, -2.04283941e+00,
-3.41344725e+00, -1.12837248e+00, -2.92544358e+00,  5.72953786e-01,
-4.11969984e+00, -2.84735645e+00, -3.99338260e+00, -5.65970347e-01,
-1.52839419e+00, -2.99676386e+00, -7.50625346e-01, -2.40662297e+00,
-2.18777351e+00,  1.04316025e+00, -2.10473491e+00, -1.60097003e+00,
 2.10724986e+00,  2.38808400e+00, -1.74353107e+00, -1.26198504e+00,
 4.55980659e-01, -4.11625464e+00,  1.11586229e+00,  3.72447138e-01,
 1.98815888e+00,  2.11537137e+00,  4.54380920e-01, -1.19687415e+00,
 1.64637398e+00, -4.44422654e+00,  1.29241276e+00,  2.08132316e+00,
 1.06743324e+00, -2.45924889e+00, -1.57728847e+00, -3.43267491e+00,
-9.92665468e-01, -3.72652996e+00, -2.40205907e+00, -2.59081966e-01,
-1.83036684e+00, -1.24183607e-01, -3.57305430e+00,  2.99646825e-01,
-1.29763384e+00, -3.64153230e+00, -8.45249212e-01, -3.59152473e+00,
-3.37595370e+00,  1.00460969e+00, -2.89842398e+00, -1.58989812e+00,
-3.48424034e+00, -2.76406976e+00, -5.53869136e-01, -2.28248336e+00,
-2.21794832e+00, -2.34250955e+00, -3.05796791e+00, -9.11618313e-01,
```

```
-9.04951273e-01, -3.94604360e-01, -1.48459143e+00,  1.73228268e+00,
-2.71658115e+00, -2.83564696e+00, -2.41526964e-01, -1.15281637e+00,
 1.03051871e+00, -1.63763778e+00, -2.90076355e+00, -3.75639817e+00,
-2.33988104e+00, -4.84715436e+00,  2.10176877e+00, -2.44563737e+00,
-1.88821377e+00,  2.57382219e-02,  1.66595715e+00, -2.90993002e+00,
-2.15368806e+00,  1.30325042e+00, -1.11908751e+00,  1.26247833e+00,
-2.87844097e+00, -1.83579384e+00, -3.49982820e+00,  2.02201171e+00,
-1.36354180e+00, -6.16951395e-01,  1.19255510e+00, -5.22285933e+00,
-1.92185220e+00, -1.84592286e+00, -1.63444412e+00, -2.39023809e+00,
-2.14890432e+00, -2.58782948e+00, -1.80667560e+00, -3.36935199e+00,
 2.73294081e+00,  6.40433031e-01, -2.70359065e+00,  1.75138061e+00,
-2.32838089e+00, -9.17345095e-01, -1.54943669e-01,  2.07232900e+00,
-3.39673603e+00, -8.31414911e-01, -2.07138162e+00, -2.79672543e+00,
 1.06370573e-01, -2.49697161e+00, -2.51356927e+00, -2.66544253e+00,
 1.28874382e+00, -1.02899871e+00, -1.63700358e+00, -3.62653799e+00,
-2.31765229e+00,  1.42456874e+00, -2.08002726e+00,  2.86612971e+00,
-2.84625550e+00,  7.15114991e-01, -3.02893700e-01, -4.56933302e+00,
-2.45924515e+00, -2.76477918e+00, -3.08989843e+00, -1.23865887e+00,
 1.75720134e+00, -2.38383892e+00,  2.92709540e+00, -3.94916285e+00,
 1.50574879e+00, -2.84983161e+00, -3.05813479e+00, -1.55225691e+00,
 1.15884049e-01, -3.22621588e-01, -3.27741282e+00,  1.77450050e-01,
-2.93137128e+00,  1.01230649e+00, -2.39780995e+00, -1.93942900e+00,
-3.12690651e+00, -3.35211622e-01, -1.59680855e+00, -1.98553993e+00,
 7.98670492e-01,  1.07582820e+00, -1.96688593e+00, -3.49950953e-01,
-1.88818608e+00, -8.04811338e-03, -2.72192350e+00, -1.66371603e+00,
-2.94378582e+00, -1.72036069e+00, -3.74309192e+00,  1.42571565e+00,
-1.63353139e+00, -1.41280853e+00, -3.99353666e+00,  8.74225512e-01,
-2.44342691e+00,  5.41220260e-01, -2.26799021e+00, -1.94438673e+00,
-1.96145486e+00,  2.48082852e-01, -1.25848792e+00, -3.59629199e+00,
-2.37953032e+00,  1.54680630e+00, -1.95644714e+00, -4.08291360e+00,
-1.98655759e+00,  3.50783482e-01, -3.86678256e+00, -8.19952798e-01,
-3.99134809e-01, -1.89181350e+00,  1.87667056e+00, -3.37680309e+00,
 8.36775815e-01, -2.65084913e+00, -3.33640440e+00, -1.81891650e+00,
-2.38121828e-01, -1.82375322e+00, -2.88218317e+00,  9.70616148e-01,
-2.16930100e+00, -5.56261810e-01, -1.23150552e+00,  8.72553250e-01,
 2.33267587e+00, -3.40812394e+00, -3.15676760e+00, -2.37292708e+00,
 2.06508415e+00,  6.79003732e-01, -2.89077899e+00, -2.60942335e+00,
-8.41267841e-01, -1.13885723e+00, -5.14995187e-01, -2.80416300e+00,
 3.68974845e-01, -6.90472620e+00, -2.26010498e+00, -1.92884034e+00,
-1.80569419e+00, -2.44735307e+00, -1.21653605e+00, -1.60985559e+00,
-1.55503336e+00, -2.14363928e+00,  1.51243280e+00, -3.66753990e+00,
 1.37131778e+00, -2.56560115e+00, -1.26729544e-01, -2.87401878e+00,
-1.31136853e-01, -2.16882955e+00, -2.31781979e+00, -2.76623132e+00,
-2.62846733e+00,  1.58017206e+00,  1.66030109e+00, -1.44394253e+00,
-2.45796481e+00, -6.97780849e-01, -1.67280226e+00,  1.41933306e+00,
-9.84504309e-01, -2.82352414e+00, -1.58082227e+00, -1.46133429e+00,
 1.34730584e+00, -2.37425104e+00,  1.06097986e+00, -9.88883923e-01,
```

```
-2.81958845e+00,  2.04844481e+00, -2.28175556e+00, -2.52019024e+00,
-2.29366192e+00,  1.69469933e+00, -3.37280965e+00, -3.06582859e+00,
-9.26915348e-01, -9.17572537e-01, -3.86802964e+00, -1.58304307e+00,
-1.58485362e+00, -2.61102624e+00, -3.19575122e+00,  1.19425402e+00,
-4.78224804e-01, -1.92807468e+00,  1.93668011e+00, -2.75198434e+00,
-2.05874093e+00, -7.60846426e-01, -4.34859609e+00, -2.49275205e+00,
-2.54250204e+00, -2.29175031e+00, -1.14690180e+00, -3.96919214e+00,
-2.20528986e+00, -3.00482944e+00, -3.05215806e+00, -2.33776095e+00,
 1.39398288e+00, -2.34465643e+00, -1.71808056e+00, -3.57900825e+00,
-1.03122648e+00,  2.29758770e+00, -2.30210102e+00,  7.99383228e-01,
-3.16379395e+00, -2.95946308e+00, -2.15460789e+00, -3.95708015e+00,
 1.65305547e+00, -3.47183462e+00,  6.84983063e-01, -1.61425733e+00])
```

[8]:
```
# difference between custom decision value and sklearn decision and check for
 ↪indices where difference is greater than 10 ^ -6 ,
# we can see we get array of length zero
np.where(f_cv - svc_clf.decision_function(X_cv) > 10e-6)
```

[8]:
```
(array([], dtype=int64),)
```

0.0.2 Task F : Implementing Platt Scaling to find P(Y==1|X)

[9]:
```
def sigmoid(w,x,b):
    return 1/(1+np.exp(-(np.dot(x,w.T)+b))) #return 1/1+e(-x)
```

[10]:
```
def logloss(w,x,y,b,reg=0):
    val=sigmoid(w,x,b)
    return -np.mean(y*np.log10(val)+(1-y)*np.log10(1-val))+reg  # cost function
 ↪of logistic regression
```

[11]:
```
count_one=list(y_train).count(1)
count_zero=list(y_train).count(0)       # calculating y+ and y_
y_plus=(count_one+1)/(count_one+2)
y_minus=1/(count_zero+2)
```

[12]:
```
def update(y_cv,y_plus,y_minus):
    u_cv=[]
    for point in y_cv:    # update function convert y_cv into y+,y_
        if point==1:
                u_cv.append(y_plus)
        else:
              u_cv.append(y_minus)
    return(np.array(u_cv))
```

[13]:
```
y_cv_updated=update(y_cv,y_plus,y_minus)
```

[14]:
```
w = np.zeros_like(f_cv[0])# initial weight vector
b = 0      # initial intercept value
eta0  = 0.0001  # learning rate
alpha = 0.0001 # lambda value
N = len(f_cv)
print(len(y_cv_updated))
print(N)
```

```
1000
1000
```

[15]:
```
ini=logloss(w,f_cv,y_cv_updated,b)
print("Initial log loss =",ini)
```

```
Initial log loss = 0.3010299956639812
```

0.0.3 SGD alorithm for calculating optimal w and b

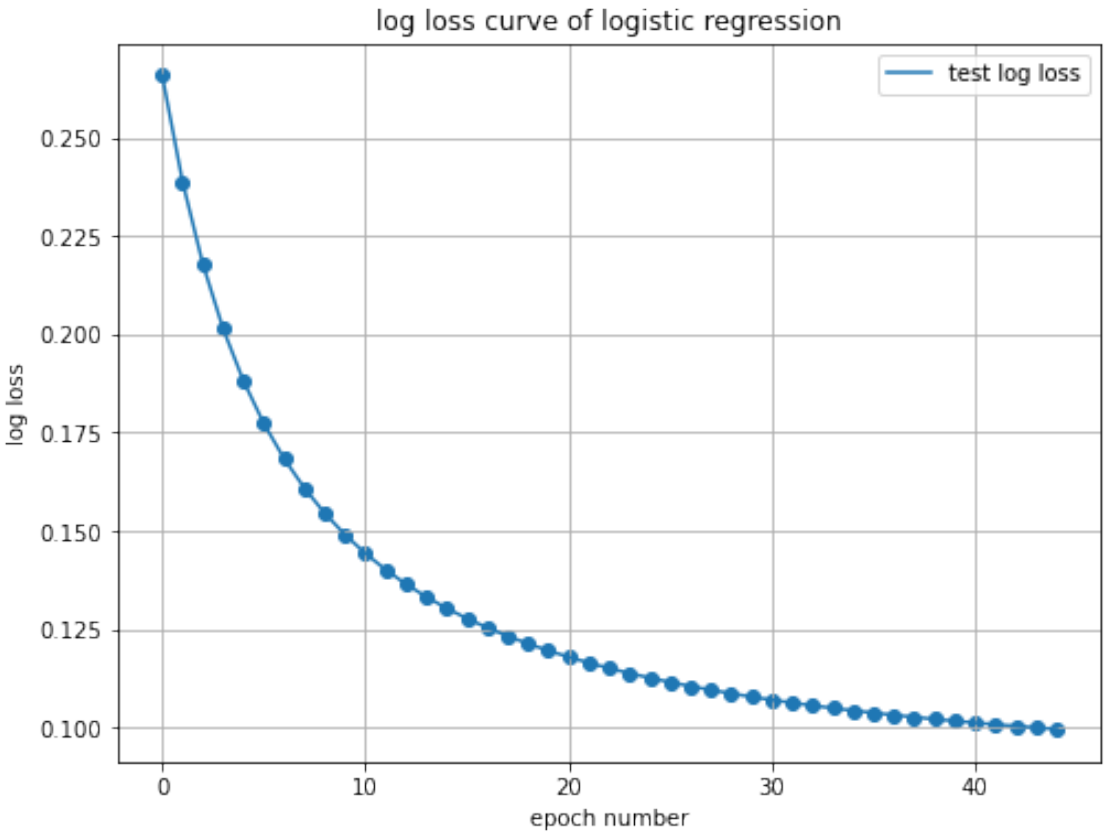
[16]:
```
def sgd_algo(f_cv,y_cv_updated,eta0,alpha,w,b,epoch):
    t=0.001 # tolerence
    test_loss=[]
    epoc=[]
    for i in  range(0,epoch):
        epoc.append(i)
        for j in range(0,N):
            reg=alpha/2*np.dot(w.T,w) #regulrization term
            w = ((1-eta0*(alpha/
 ↪N))*w)+((eta0*f_cv[j])*(y_cv_updated[j]-sigmoid(w,f_cv[j],b))) # updating␣
 ↪weight vector
            b = b+(eta0*(y_cv_updated[j]-sigmoid(w,f_cv[j],b)))       #␣
 ↪updatind intercept
        test=logloss(w,f_cv,y_cv_updated,b,reg)
        test_loss.append(test)
        if i<=t :
            continue
            if abs(test_loss[i]-test_loss[i-1])>t: # block to check convergence
                continue
            else:
                break
    return w,b,epoc,test_loss
```

[17]:
```
epoch=45
we,be,epo,loss=sgd_algo(f_cv,y_cv_updated,eta0, alpha,w,b,epoch)
print("optimal weight = ",we)
print("optimal intercept = ",be)
```

```
optimal weight =  1.1449699301415204
optimal intercept =  -0.11335493592139145
```

[18]:
```
%matplotlib inline
import matplotlib.pyplot as plt
plt.figure(figsize=(8,6))
plt.grid()
plt.plot(epo,loss, label='test log loss')
plt.scatter(epo,loss)
plt.title('log loss curve of logistic regression')
plt.xlabel('epoch number')
plt.ylabel("log loss")
plt.legend()
```

[18]:
```
<matplotlib.legend.Legend at 0x206874288e0>
```



[19]:
```
f_test=decision_function_custom(X_test,intercept, dual_coefs, support_vectors,␣
↪gamma)

def probability(f_test,w,b):
    p=1/(1+np.exp(-w*f_test+b)) # to calculate probilty P(Y=1|X)
    return p
```

```
prob = probability(f_test,we,be)
print("The top 10 probabilities are:",prob[:10])
```

```
The top 10 probabilities are: [0.03349361 0.8924317  0.29748392 0.05499249
0.42894947 0.04339435
 0.03226516 0.14468386 0.8812426  0.08964745]
```