parth.pandey13103347@gmail.com_10_E_F

June 14, 2020

8E and 8F: Finding the Probability P(Y==1|X)

8E: Implementing Decision Function of SVM RBF Kernel

After we train a kernel SVM model, we will be getting support vectors and their corresponsing coefficients α_i

Check the documentation for better understanding of these attributes:

https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

As a part of this assignment you will be implementing the `decision_function()` of kernel SVM, here decision_function() means based on the value return by `decision_function()` model will classify the data point either as positive or negative

Ex 1: In logistic regression After traning the models with the optimal weights w we get, we will find the value $\frac{1}{1+\exp(-(wx+b))}$, if this value comes out to be < 0.5 we will mark it as negative class, else its positive class

Ex 2: In Linear SVM After traning the models with the optimal weights w we get, we will find the value of $sign(wx + b)$, if this value comes out to be -ve we will mark it as negative class, else its positive class.

Similarly in Kernel SVM After traning the models with the coefficients α_i we get, we will find the value of $sign(\sum_{i=1}^{n}(y_i\alpha_i K(x_i, x_q)) + intercept)$, here $K(x_i, x_q)$ is the RBF kernel. If this value comes out to be -ve we will mark x_q as negative class, else its positive class.

RBF kernel is defined as: $K(x_i, x_q) = exp(-\gamma||x_i - x_q||^2)$

For better understanding check this link: https://scikit-learn.org/stable/modules/svm.html#svm-mathematical-formulation

0.1 Task E

1. Split the data into $X_{train}(60)$, $X_{cv}(20)$, $X_{test}(20)$
2. Train $SVC(gamma = 0.001, C = 100.)$ on the (X_{train}, y_{train})
3. Get the decision boundry values f_{cv} on the X_{cv} data i.e. f_{cv} = `decision_function(`X_{cv}`)` you need to implement this decision_function()

0.2 Imports

[2]:
```
import numpy as np
import pandas as pd
from sklearn.datasets import make_classification
import numpy as np
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix

import matplotlib.pyplot as plt
```

[3]:
```
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)
```

0.2.1 Pseudo code

clf = SVC(gamma=0.001, C=100.) clf.fit(Xtrain, ytrain)

def decision_function(Xcv, …): #use appropriate parameters for a data point x_q in Xcv: #write code to implement $(\sum_{i=1}^{\text{all the support vectors}}(y_i \alpha_i K(x_i, x_q)) + intercept)$, here the values y_i, α_i, and $intercept$ can be obtained from the trained model return # the decision_function output for all the data points in the Xcv

fcv = decision_function(Xcv, …) # based on your requirement you can pass any other parameters

Note: Make sure the values you get as fcv, should be equal to outputs of clf.decision_function(Xcv)

0.3 Splitting Train and Test and CV data

[4]:
```
print(X.shape)
print(y.shape)
```

```
(5000, 5)
(5000,)
```

[5]:
```
x_train, x_test , y_train , y_test = train_test_split(X, y, stratify=y,␣
 ↪test_size=0.2)
# print(x_train.shape, y_train.shape)
print(x_test.shape,  y_test.shape)
x_train, x_cv, y_train, y_cv   = train_test_split(x_train, y_train,␣
 ↪stratify=y_train, test_size=0.2)
print(x_train.shape, y_train.shape)
print(x_cv.shape,  y_cv.shape)
```

```
(1000, 5) (1000,)
(3200, 5) (3200,)
(800, 5) (800,)
```

0.4 Training Model

[6]:
```
# you can write your code here

clf = SVC(gamma=0.001, C =100, kernel='rbf', verbose=2, random_state=42)
clf.fit(x_train, y_train)
```

```
[LibSVM]
```

[6]:
```
SVC(C=100, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma=0.001, kernel='rbf',
    max_iter=-1, probability=False, random_state=42, shrinking=True, tol=0.001,
    verbose=2)
```

0.4.1 Implementing Decision Function

[7]:
```
def decision_function(data, clf):
    support_vec = clf.support_vectors_

    alpha_i = clf.dual_coef_.reshape(-1,1)
    output_list = np.array([])
    for query in data:
        sum_proba = 0
        for a_i,x_i in zip(alpha_i, support_vec):
            sum_proba += a_i *  np.exp(-1 * clf.gamma * np.sum(np.
 ↪power(x_i-query, 2)) )

        output_list = np.append(output_list,sum_proba + clf.intercept_)
    return np.array(output_list)
```

0.4.2 Checking Decision Function Results

[8]:
```
# Applying Check for decision function
for a,b in zip(clf.decision_function(x_cv),decision_function(x_cv, clf)):
    if a != b:
        print('No match')
```

[9]:
```
f_cv = decision_function(x_cv, clf)
```

[10]:
```
f_cv.reshape(-1,1)
```

[10]:
```
array([[-1.03160929e+00],
       [ 1.29772677e+00],
       [-1.58540449e+00],
       [-3.07220395e+00],
       [-1.35006214e-01],
       [-1.11699386e+00],
```

```
[ 7.13586476e-01],
[ 1.76424884e+00],
[ 1.01277999e+00],
[-2.44835327e+00],
[-7.28314976e-01],
[-5.03451526e+00],
[-3.80426935e+00],
[-3.15261636e+00],
[-1.75886428e+00],
[-2.29178640e+00],
[ 2.07259850e+00],
[ 8.91493536e-01],
[-2.40038464e+00],
[-2.02115972e+00],
[-2.54825505e+00],
[-2.80991901e+00],
[ 1.54329679e+00],
[-3.25095983e+00],
[-3.90851079e+00],
[ 2.43234374e+00],
[-2.46667716e+00],
[-2.08549043e+00],
[ 2.29920463e+00],
[-3.12897547e+00],
[-1.00241705e+00],
[-4.47497791e+00],
[ 6.31710249e-01],
[-2.83023596e+00],
[-1.13228789e+00],
[-2.39261109e+00],
[-7.80268086e-01],
[-3.67424244e+00],
[ 2.26804986e-01],
[-7.05943775e-01],
[-2.42502642e+00],
[-9.85873277e-01],
[ 1.82667881e+00],
[-1.47386683e+00],
[-1.42086435e+00],
[-3.26429527e+00],
[-1.49731369e+00],
[-3.26652616e+00],
[-2.65154338e+00],
[-1.77428186e+00],
[ 1.85491270e+00],
[-3.32871031e-01],
[ 1.89114164e+00],
```

```
[-3.71046526e+00],
[-3.00531454e+00],
[ 2.21844773e+00],
[-3.03563879e+00],
[-3.77738943e-01],
[-3.01928902e+00],
[ 1.67817239e+00],
[ 1.33654340e-01],
[-4.34690776e+00],
[-7.24011444e-01],
[-3.32945443e+00],
[-1.91778604e+00],
[-1.65967752e-02],
[ 1.13741379e+00],
[-2.12330104e+00],
[-2.09504268e+00],
[-2.68077382e+00],
[ 2.52675633e+00],
[-2.92774304e+00],
[ 6.31776246e-01],
[-3.35526393e+00],
[ 1.59792013e+00],
[-3.27864028e+00],
[-2.45268609e-02],
[-1.78675592e+00],
[-1.96238276e+00],
[-3.09333202e+00],
[-9.86001771e-01],
[-3.07154305e+00],
[-4.26453496e+00],
[-2.95556783e+00],
[-3.07097063e+00],
[-4.57834262e+00],
[-2.18635130e+00],
[-2.61280739e+00],
[ 1.24956885e+00],
[-1.03899184e+00],
[ 1.56175746e+00],
[-3.20806179e+00],
[ 2.16345034e+00],
[-2.67924577e+00],
[-3.32495177e+00],
[ 2.27923087e+00],
[-1.56043744e+00],
[ 1.79846600e+00],
[-2.74329875e+00],
[-2.77937858e+00],
```

```
[-1.33099555e+00],
[-3.36113264e+00],
[ 9.69954344e-01],
[-2.54085447e+00],
[ 1.61851542e+00],
[-1.42389504e+00],
[-3.06753372e+00],
[-1.52987260e+00],
[-2.26500012e+00],
[ 3.21047355e+00],
[-1.74504561e+00],
[-2.56565713e+00],
[ 6.50661830e-01],
[ 1.59868197e+00],
[-2.32688746e+00],
[-1.05208448e+00],
[-3.43788371e-01],
[-2.02586409e+00],
[ 2.08356992e+00],
[-2.40886585e+00],
[-3.52513563e+00],
[-4.50768733e-01],
[-2.56996221e+00],
[-2.84885721e+00],
[-3.15439876e+00],
[ 2.07454558e+00],
[-1.97855022e+00],
[-3.51947875e+00],
[ 1.66290851e+00],
[-3.31674702e+00],
[ 2.67212546e-01],
[-3.18144274e+00],
[-2.67702227e+00],
[-1.72939062e+00],
[ 1.66987457e+00],
[-8.39697957e-01],
[-1.88020254e+00],
[-2.85660933e+00],
[ 2.70488486e+00],
[-3.33209595e+00],
[-2.31698604e+00],
[-3.16843764e+00],
[-2.93713180e+00],
[ 2.69148198e+00],
[-3.82710439e+00],
[ 1.19508590e+00],
[ 2.77622011e+00],
```

```
[-1.83036713e+00],
[-1.88042940e-01],
[-3.84969558e+00],
[-2.51765447e+00],
[-4.23734773e+00],
[-2.20194220e+00],
[-1.91316726e+00],
[-2.12336285e+00],
[-2.27216965e+00],
[ 1.59151525e+00],
[-8.43013854e-01],
[ 1.89539665e+00],
[ 2.20600978e+00],
[-2.04481885e+00],
[ 1.71374948e+00],
[-1.75087199e+00],
[ 2.02539130e+00],
[-3.15777034e+00],
[ 2.09691554e+00],
[-2.96249332e+00],
[-2.83085284e+00],
[ 2.21328659e+00],
[ 1.56219396e+00],
[-3.13944966e+00],
[-7.75365755e-01],
[ 1.53422165e+00],
[ 7.13865989e-01],
[-1.85475049e+00],
[ 1.78578113e+00],
[-3.55014363e+00],
[ 2.79143860e+00],
[-3.52878842e+00],
[-1.73680909e+00],
[-2.28266142e+00],
[ 1.90999110e+00],
[ 1.95829487e+00],
[-4.24157412e+00],
[-2.55638864e+00],
[-1.64752031e+00],
[ 1.75946689e+00],
[-1.84284932e+00],
[-2.95437209e+00],
[-2.24908275e+00],
[-3.72529871e+00],
[-2.47338766e+00],
[ 2.36655272e+00],
[-1.74899368e+00],
```

```
[ 2.68826946e-02],
[-2.30567769e+00],
[ 6.78905846e-01],
[ 2.03244905e+00],
[ 1.94310678e+00],
[-2.32936379e+00],
[ 1.88700967e+00],
[-3.41844073e+00],
[-3.61461845e+00],
[ 1.43514931e+00],
[ 1.71835622e+00],
[-3.32666326e+00],
[ 2.09430273e+00],
[-2.58516618e+00],
[-2.04074667e+00],
[-4.50130424e+00],
[-2.70852990e+00],
[-2.18576764e+00],
[-2.43500352e+00],
[-2.38936722e+00],
[-2.35515335e+00],
[-2.31211689e+00],
[-2.65561205e+00],
[-1.64241173e+00],
[ 2.18011427e+00],
[ 2.29796571e+00],
[ 2.24065837e+00],
[-3.14949394e+00],
[-9.70710853e-01],
[ 3.03580960e+00],
[-3.52989421e-01],
[ 2.58807219e+00],
[-9.58336738e-01],
[ 2.73352240e+00],
[-6.90939559e-01],
[-2.30858023e+00],
[-3.86412989e+00],
[-3.64595094e+00],
[-2.49939652e+00],
[-2.34224702e+00],
[-3.25292618e+00],
[-3.21705963e+00],
[ 2.22870072e+00],
[ 1.42354725e+00],
[-2.61175282e+00],
[-4.20780762e-03],
[-2.89497549e+00],
```

```
[ 2.43381156e+00],
[-2.61169391e+00],
[ 4.93661430e-01],
[-1.66954834e+00],
[-3.11912680e+00],
[-1.28926121e+00],
[-2.29354119e+00],
[-9.42032647e-01],
[-1.72725144e+00],
[ 2.18835208e+00],
[-3.29630276e+00],
[-2.85681475e+00],
[-2.45024061e-01],
[-3.71679715e+00],
[-4.67165163e+00],
[ 2.44327411e+00],
[ 1.00713833e+00],
[ 4.01252139e+00],
[ 1.92406272e+00],
[-2.04531964e+00],
[-3.48677289e+00],
[-1.22600407e+00],
[-2.02151610e+00],
[-4.43532901e+00],
[-3.30926867e+00],
[-1.93219413e+00],
[-1.92777520e+00],
[ 1.11776612e+00],
[ 9.00928728e-01],
[-3.60545775e-01],
[ 1.45904430e+00],
[-2.99197945e+00],
[ 2.62907317e+00],
[ 2.24152328e+00],
[-1.55359435e+00],
[-3.60312372e+00],
[-2.30429017e+00],
[-2.87035500e+00],
[ 1.97072753e+00],
[-1.76209582e+00],
[-3.06441942e+00],
[-2.86852615e+00],
[-2.19939844e+00],
[-1.60752782e+00],
[ 1.83208019e+00],
[-2.79211338e+00],
[-3.03025816e+00],
```

```
[-2.57824917e+00],
[-4.35586048e+00],
[ 1.87178053e+00],
[-1.90430604e+00],
[ 6.29167985e-01],
[-7.88931797e-01],
[-1.49923758e+00],
[-1.08786097e+00],
[ 2.24566809e+00],
[ 2.18247839e+00],
[ 1.01701273e+00],
[ 2.51078580e+00],
[-1.79681283e+00],
[-1.59671154e-01],
[-3.42614147e+00],
[-2.32870189e+00],
[ 1.61199503e+00],
[-9.12740917e-01],
[-3.00260890e+00],
[-1.53115314e+00],
[-3.84456432e+00],
[-2.69033256e+00],
[ 2.72824195e-02],
[-3.23937496e+00],
[-2.14028165e+00],
[ 1.71433540e+00],
[-1.96630260e+00],
[ 1.88546780e+00],
[-2.69955700e+00],
[ 1.30235086e+00],
[ 1.52042266e+00],
[ 1.83782271e+00],
[-3.18249137e+00],
[ 4.12469028e-01],
[-2.83087699e+00],
[-1.63013492e+00],
[ 2.28697150e+00],
[-2.93884941e+00],
[ 5.34787171e-01],
[-2.68002940e+00],
[-2.56835233e+00],
[-3.96001015e+00],
[ 7.59070984e-01],
[-2.88502826e+00],
[-2.79937397e+00],
[ 1.64302267e+00],
[ 2.51848295e-01],
```

```
[-4.37886490e+00],
[-2.44677295e+00],
[ 2.15469020e-02],
[-1.67696723e+00],
[ 2.14942180e+00],
[-1.22958565e+00],
[-9.22617111e-01],
[-7.89642333e-01],
[-4.59025324e+00],
[-1.68478843e+00],
[-3.91301541e+00],
[-3.03453831e+00],
[ 1.82873390e+00],
[-3.06197731e+00],
[ 6.11635013e-01],
[ 1.11914992e+00],
[-3.94580723e+00],
[-2.82237842e+00],
[-3.05368495e+00],
[-1.06129298e+00],
[-2.84635453e+00],
[-3.12164938e+00],
[ 1.51840556e+00],
[ 1.56780963e+00],
[-2.74845778e-01],
[-2.38445308e+00],
[-2.23837851e+00],
[-1.24404505e+00],
[-2.75846257e+00],
[-1.83649734e+00],
[-2.40577259e+00],
[-2.96613341e+00],
[-3.25446495e-01],
[-7.72195522e-01],
[-2.96912648e+00],
[ 1.09815456e+00],
[-2.50321186e+00],
[ 1.90250169e+00],
[-3.23544588e+00],
[-7.66921560e-01],
[-2.31809396e+00],
[ 2.55855052e+00],
[-2.24930267e+00],
[-2.71809264e+00],
[ 2.92720509e-01],
[-9.12947904e-02],
[-3.77359917e+00],
```

```
[ 2.28594736e+00],
[-2.59757427e+00],
[-4.33413253e+00],
[-2.87864390e+00],
[-2.70077725e+00],
[-2.88438786e+00],
[-1.05057678e+00],
[-4.34657870e+00],
[-2.49794333e+00],
[ 5.88486200e-01],
[-1.45711178e-01],
[-2.97219180e+00],
[-1.62866525e+00],
[-2.41038499e+00],
[-1.94362436e+00],
[-2.12161952e+00],
[-2.95155792e+00],
[-3.32287401e+00],
[ 3.17443180e-01],
[-4.02594923e+00],
[-3.27538206e+00],
[-1.90849143e+00],
[-7.62230736e-01],
[-2.50684831e+00],
[-3.22632733e+00],
[-1.34615263e+00],
[-3.53456081e+00],
[-2.79609327e+00],
[-3.15897145e+00],
[ 1.95073488e+00],
[-4.12072237e+00],
[ 1.24220899e+00],
[-2.75570346e+00],
[-3.50184933e+00],
[-3.54509958e+00],
[-7.41904925e-01],
[-1.76684075e+00],
[-1.40111854e+00],
[ 2.02407843e+00],
[ 2.06648229e+00],
[-4.81255350e+00],
[-9.09830685e-01],
[-1.79706333e+00],
[-5.75096449e-01],
[-2.62447762e+00],
[-3.72649000e+00],
[-1.55189623e+00],
```

```
[-2.46482633e+00],
[ 1.52546331e+00],
[ 8.37090141e-01],
[-3.62577230e+00],
[-4.23361130e+00],
[-2.98217412e+00],
[ 4.65187286e-01],
[-3.31732471e+00],
[ 5.69632518e-01],
[-2.22637942e+00],
[ 1.72602066e+00],
[-8.31157471e-01],
[ 1.50840519e+00],
[-1.03421887e+00],
[-5.98505788e-01],
[-3.27708115e+00],
[-3.52059450e+00],
[-2.62047418e+00],
[ 1.78309630e+00],
[ 2.16429506e+00],
[-3.76534145e+00],
[-2.26212574e+00],
[ 1.25010222e+00],
[ 3.32968113e-01],
[-8.92611141e-01],
[-2.87885941e+00],
[-1.66100668e+00],
[-1.49770161e+00],
[-1.05046578e+00],
[-2.72680051e+00],
[-5.83696053e-01],
[-1.83796789e+00],
[ 1.38797399e+00],
[-1.60433765e+00],
[ 1.45605001e+00],
[-2.37779968e+00],
[-1.75871668e+00],
[ 5.76030146e-01],
[ 2.00509279e+00],
[-4.32356529e+00],
[-2.35180492e+00],
[-1.81037754e+00],
[-3.82931079e+00],
[-3.44671276e+00],
[-1.48165278e+00],
[ 2.13719320e+00],
[ 9.42749355e-01],
```

```
[-2.87857863e+00],
[ 1.13463569e+00],
[-2.07271551e+00],
[-3.29387795e+00],
[-3.25151047e+00],
[-1.58388504e+00],
[ 3.29631957e+00],
[-3.49130595e+00],
[ 1.99548133e+00],
[-1.95487467e+00],
[ 3.74980115e-01],
[-2.81701102e+00],
[ 9.25780317e-01],
[-3.01359918e+00],
[-9.88332943e-01],
[-2.87791586e+00],
[-2.45430558e+00],
[ 9.60813069e-01],
[ 3.09765743e-01],
[-1.25314323e+00],
[-3.11797316e+00],
[ 1.10429279e+00],
[-1.96264176e-01],
[-3.73228178e+00],
[-2.90922264e+00],
[-2.51601795e+00],
[ 2.27588558e+00],
[-1.71748323e+00],
[ 1.91079231e+00],
[-3.30776179e+00],
[-3.26961483e+00],
[-4.13603622e+00],
[-3.22791687e+00],
[-2.24699050e+00],
[ 1.67390751e+00],
[-2.25427320e+00],
[-2.24739909e+00],
[-1.74667648e+00],
[-1.07094985e+00],
[-1.28392364e+00],
[-3.72065359e+00],
[ 8.78165689e-01],
[-2.85301137e+00],
[ 1.86951186e+00],
[-2.04781354e+00],
[-2.35799611e+00],
[-1.21562336e-01],
```

```
[ 1.86950968e+00],
[ 1.15621565e-01],
[-1.80696583e+00],
[ 1.26732873e+00],
[ 1.58383428e+00],
[-2.62990556e+00],
[-3.66340991e+00],
[-2.87260016e+00],
[-3.40709702e+00],
[-1.39001187e+00],
[ 2.26025578e+00],
[-2.76035594e+00],
[-3.09395563e+00],
[ 2.06301396e+00],
[-4.03022489e+00],
[ 1.75858586e+00],
[-3.08777127e+00],
[-1.73318812e+00],
[-2.08823955e+00],
[-3.44379921e+00],
[-2.52402028e+00],
[-4.00462593e+00],
[-1.66304005e+00],
[-1.90310794e+00],
[-3.25547038e+00],
[-4.36504620e+00],
[-3.83926402e+00],
[ 1.76874560e+00],
[ 2.45249831e+00],
[ 1.71110003e+00],
[-3.23924444e+00],
[-1.70937393e+00],
[-3.21031900e+00],
[-2.72877098e+00],
[ 1.36419437e+00],
[ 1.50910106e+00],
[-2.96159909e+00],
[-6.14811086e-01],
[-3.41291545e+00],
[-1.65950046e+00],
[ 2.81881212e+00],
[-2.98534699e+00],
[-5.51152993e-01],
[-1.40790138e+00],
[-3.88229571e+00],
[-3.37390345e+00],
[-3.91087717e+00],
```

```
[-1.69649985e+00],
[ 2.22788056e-01],
[-5.18033791e-01],
[ 1.38431519e+00],
[-6.23899075e-01],
[-7.61457464e-01],
[-1.94382448e+00],
[ 1.11266209e+00],
[-3.27635249e+00],
[ 1.92894146e+00],
[-2.58220693e+00],
[-1.17574690e-01],
[-4.50490474e+00],
[-4.26941900e+00],
[-3.30442253e+00],
[ 1.61812055e+00],
[-3.11886244e+00],
[-3.49898252e+00],
[ 2.49331586e+00],
[ 7.83083574e-01],
[ 1.82466681e+00],
[-1.11463799e+00],
[-2.56331424e+00],
[ 2.07391326e+00],
[ 1.97003689e+00],
[-3.14151796e+00],
[-4.25027416e+00],
[-1.48778796e+00],
[-3.77867074e+00],
[ 1.72494418e+00],
[ 1.78239776e+00],
[ 6.60259658e-01],
[-3.43294471e+00],
[-6.22108993e-01],
[ 2.06460460e+00],
[-2.49068080e+00],
[-5.71942491e+00],
[-4.93304389e-01],
[-2.77917640e+00],
[ 2.52818575e+00],
[-2.84500630e+00],
[-3.10148434e+00],
[-2.44387029e+00],
[-2.66587842e+00],
[ 1.21237454e+00],
[ 3.06587028e-01],
[-1.56879446e+00],
```

```
[-8.80276133e-01],
[-1.86699169e+00],
[ 3.31831427e-01],
[ 3.32810025e+00],
[ 2.26067462e+00],
[-2.55468774e+00],
[-1.32220448e+00],
[-2.82911947e+00],
[-5.58665052e+00],
[-2.25128293e+00],
[-5.45868015e-02],
[-1.46282369e-01],
[-3.47523769e-01],
[ 5.92412034e-01],
[-2.00283694e+00],
[ 1.50961972e+00],
[ 6.23287554e-01],
[ 3.03065953e-01],
[-4.01880385e+00],
[ 9.71173778e-01],
[ 7.80965571e-01],
[ 1.41212566e+00],
[-3.76180765e+00],
[-7.92093640e-01],
[-9.55524818e-01],
[-2.92449473e+00],
[ 2.23219553e-01],
[-3.00037619e+00],
[-2.53650330e+00],
[-7.18030330e-01],
[-2.51333140e+00],
[-2.47610407e+00],
[ 7.94732298e-01],
[-2.77578655e+00],
[ 1.62438164e+00],
[ 2.14111017e+00],
[-1.84195765e+00],
[-1.34580470e+00],
[ 2.47954256e+00],
[-3.61087416e+00],
[-2.47340157e+00],
[-2.23137554e+00],
[-2.23612655e+00],
[-2.77294778e+00],
[-2.41240682e+00],
[-2.46619052e+00],
[-2.35999650e+00],
```

```
[-3.49644834e+00],
[-1.76942580e+00],
[-2.02312637e-01],
[-2.63667942e+00],
[-2.24478765e+00],
[ 3.49047893e+00],
[-3.45427527e-01],
[-2.54881409e+00],
[-9.88265890e-01],
[ 2.35538159e-01],
[ 8.78922836e-01],
[-4.00955723e+00],
[ 1.46511205e+00],
[ 1.44464022e+00],
[ 1.63247194e+00],
[-3.22078167e+00],
[-1.96600385e+00],
[-1.68885706e+00],
[-3.52991985e+00],
[ 1.68474678e-01],
[ 1.59295997e+00],
[-1.75939943e+00],
[-4.28237542e+00],
[-3.44903029e+00],
[-9.51081485e-01],
[ 2.26669799e+00],
[-2.31916557e+00],
[-1.66974702e+00],
[ 6.22946317e-02],
[ 6.82835518e-01],
[-2.63264178e+00],
[ 4.61552180e+00],
[ 1.50521132e+00],
[-1.89803563e+00],
[-4.03561745e+00],
[ 5.96237483e-01],
[-1.95976233e+00],
[-3.18302882e+00],
[ 4.61986265e-01],
[ 2.57242195e+00],
[-3.73260501e+00],
[-2.42238745e+00],
[ 1.97647751e+00],
[-1.04808137e+00],
[-7.76115269e-01],
[ 1.34120667e+00],
[ 1.61832102e+00],
```

```
[ 1.19286393e-01],
[-3.01652499e+00],
[ 1.56393754e+00],
[-2.21188435e-01],
[-2.98927198e+00],
[-1.07763805e+00],
[-8.18978755e-01],
[ 1.73710522e+00],
[-2.02439874e+00],
[-1.42262980e+00],
[-1.63467756e+00],
[-1.39647537e+00],
[-3.96730998e+00],
[-2.30981349e+00],
[-1.53561954e+00],
[-2.77630341e+00],
[-3.89867505e+00],
[-2.52086219e+00],
[-1.97056897e+00],
[-3.02047660e+00],
[-2.62226193e+00],
[ 2.65715285e+00],
[-2.07225348e+00],
[-4.02636381e+00],
[-3.21866793e+00],
[ 1.36520814e+00],
[-2.93103438e+00],
[ 2.64429468e+00],
[-2.50135275e+00],
[-2.86708448e+00],
[-1.73434635e+00],
[-2.78638809e+00],
[-3.75156264e+00],
[-2.56567475e+00],
[-2.68789061e+00],
[ 2.35116098e+00],
[ 1.33061937e+00],
[-2.66164266e+00],
[ 4.39143431e-01],
[-1.39679321e+00],
[-5.77600075e-01],
[ 1.34187934e+00],
[ 2.07852114e+00],
[-2.89548287e+00],
[ 8.29326026e-01],
[-2.05289835e+00],
[-3.68460788e+00],
```

```
       [-3.17677951e+00],
       [ 1.03033907e+00],
       [-3.70377054e-01],
       [-2.97777380e+00],
       [-2.52739488e+00],
       [-3.02557119e+00],
       [ 1.93911584e+00],
       [-1.11761524e+00],
       [ 2.28641415e+00],
       [-1.72377667e+00],
       [ 1.10558275e+00],
       [ 7.95620851e-01],
       [ 1.04349712e+00],
       [-2.76351166e+00],
       [-2.91668271e+00],
       [-2.32992768e+00],
       [-6.07245423e-01],
       [-3.15210705e+00],
       [-2.56672852e+00],
       [ 1.46467941e+00],
       [ 1.40838789e+00],
       [ 1.92119437e+00],
       [-1.31508604e+00],
       [-3.37476850e+00],
       [ 1.05220703e+00],
       [ 1.51450592e+00],
       [-2.06843561e+00],
       [-2.97422815e+00],
       [ 1.57728789e+00],
       [-1.80549392e+00],
       [ 1.57877507e+00],
       [ 9.49862334e-01],
       [-6.13059147e-01],
       [-1.69803101e+00],
       [-3.40760948e+00],
       [-2.69548165e+00],
       [-4.57239569e+00],
       [-3.22910004e+00],
       [-2.05032475e+00],
       [-2.10972561e+00],
       [-3.31982891e+00],
       [-3.29239897e-02]])
```

[11]:
```
y_pred = np.where(f_cv < 0 ,0,1)
```

[12]:
```
y_pred2 = clf.predict(x_cv)
```

[13]:
```
conf1 = confusion_matrix(y_cv, y_pred)
conf2 = confusion_matrix(y_cv, y_pred2)
print('Confusion Matrix via Self Decision Function')
print(conf1)
print('Confusion Matrix via Prediction Method of SVM Classfifier')
print(conf2)
```

```
Confusion Matrix via Self Decision Function
[[526  32]
 [ 32 210]]
Confusion Matrix via Prediction Method of SVM Classfifier
[[526  32]
 [ 32 210]]
```

8F: Implementing Platt Scaling to find P(Y==1|X)

Check this PDF

0.5 TASK F

4. Apply SGD algorithm with (f_{cv}, y_{cv}) and find the weight W intercept b `Note: here our data is of one dimensional so we will have a one dimensional weight vector i.e W.shape (1,)`

Note1: Don't forget to change the values of y_{cv} as mentioned in the above image. you will calculate y+, y- based on data points in train data

Note2: the Sklearn's SGD algorithm doesn't support the real valued outputs, you need to use the code that was done in the `'Logistic Regression with SGD and L2'` Assignment after modifying loss function, and use same parameters that used in that assignment. if Y[i] is 1, it will be replaced with y+ value else it will replaced with y- value

5. For a given data point from X_{test}, $P(Y=1|X) = \frac{1}{1+exp(-(W*f_{test}+b))}$ where f_{test} `= decision_function(`X_{test}`)`, W and b will be learned as metioned in the above step

0.5.1 Creating y+ and y- datasets

[44]:
```
n_pos = y_train[y_train == 1].shape[0]

n_neg = y_train[y_train == 0].shape[0]

y_pos = ((n_pos + 1)/(n_pos + 2))

y_neg = ((1)/(n_neg + 2))

print('New Postive and Negative Values y_cv')
print(y_pos, y_neg)
```

```
y_cv_new = np.array([])

for ele in y_cv:
    if ele == 0:
        y_cv_new = np.append(y_cv_new, y_neg)
    else:
        y_cv_new = np.append(y_cv_new, y_pos)

print(y_cv.shape)
print(y_cv_new.shape)
```

```
New Postive and Negative Values y_cv
0.9989701338825953 0.0004478280340349306
(800,)
(800,)
```

0.5.2 Defining Functions for Platt's Scaling

[45]:
```
def sigmoid(z):
    return 1/(1 + np.exp(-z))

def gradient_dw(x, y, w, b, alpha, N):
    return x * (y - sigmoid(np.dot(w,x) + b)) + 2 * (alpha/N) * w

def gradient_db(x, y, w, b):
    return y - sigmoid(np.dot(w,x) + b)

def weights_init(x_train):
    b = 0
    w = np.zeros(x_train.shape[0])
    return w, b

def log_loss(y_true, y_pred):

    n = y_true.shape[0]
    log_loss = (-1/n) * np.sum(y_true * np.log10(y_pred) + (1-y_true) * np.
 ↪log10(1-y_pred))
    return log_loss

def train(x_train, y_train, alpha, eta, epochs, verbose = 0):

    w, b = weights_init(x_train[0])

    old_loss = 0

    N = x_train.shape[0]
```

```
    for epoch in range(1,epochs+1):

        for x , y in zip(x_train, y_train):
            w = w + eta * gradient_dw(x, y, w, b, alpha, N)
            b = b + eta * gradient_db(x, y, w, b)

        y_pred = np.array([])
        for x in x_train:
            y_pred = np.append(y_pred, sigmoid(np.dot(w,x) + b))


        epoch_loss = log_loss(y_train , y_pred)
        if verbose > 0:
            print('Epoch {} Loss {}'.format(epoch, epoch_loss))

        if np.absolute(epoch_loss - old_loss) < 10 ** -5:
            break

        old_loss = epoch_loss

    return w, b
```

0.5.3 Hyperparameter Tuning for Platt's Scaling

[47]:
```
alpha = [10**x for x in range(-5, 2)]
eta   = [10**x for x in range(-5, 2)]

loss_dict = {}

for a_i in alpha:
    for e_i in eta:
        print('Training for alpha = {} eta = {} parameters'.format(a_i, e_i))
        w, b = train(f_cv.reshape(-1,1), y_cv_new, a_i, e_i, 1000)
        y_pred = np.array([])
        for ele in f_cv:
            y_pred = np.append(y_pred, sigmoid(np.dot(w,ele)+ b))
        loss_dict['{}-{}'.format(a_i,e_i)]  = log_loss(y_cv, y_pred)
```

```
Training for alpha = 1e-05 eta = 1e-05 parameters
Training for alpha = 1e-05 eta = 0.0001 parameters
Training for alpha = 1e-05 eta = 0.001 parameters
Training for alpha = 1e-05 eta = 0.01 parameters
Training for alpha = 1e-05 eta = 0.1 parameters
Training for alpha = 1e-05 eta = 1 parameters
Training for alpha = 1e-05 eta = 10 parameters
```

```
/home/parth/AppliedAI/appliedai/lib/python3.7/site-
```

```
packages/ipykernel_launcher.py:18: RuntimeWarning: divide by zero encountered in
log10
/home/parth/AppliedAI/appliedai/lib/python3.7/site-
packages/ipykernel_launcher.py:44: RuntimeWarning: invalid value encountered in
double_scalars
/home/parth/AppliedAI/appliedai/lib/python3.7/site-
packages/ipykernel_launcher.py:18: RuntimeWarning: invalid value encountered in
multiply
```

```
Training for alpha = 0.0001 eta = 1e-05 parameters
Training for alpha = 0.0001 eta = 0.0001 parameters
Training for alpha = 0.0001 eta = 0.001 parameters
Training for alpha = 0.0001 eta = 0.01 parameters
Training for alpha = 0.0001 eta = 0.1 parameters
Training for alpha = 0.0001 eta = 1 parameters
Training for alpha = 0.0001 eta = 10 parameters
Training for alpha = 0.001 eta = 1e-05 parameters
Training for alpha = 0.001 eta = 0.0001 parameters
Training for alpha = 0.001 eta = 0.001 parameters
Training for alpha = 0.001 eta = 0.01 parameters
Training for alpha = 0.001 eta = 0.1 parameters
Training for alpha = 0.001 eta = 1 parameters
Training for alpha = 0.001 eta = 10 parameters
Training for alpha = 0.01 eta = 1e-05 parameters
Training for alpha = 0.01 eta = 0.0001 parameters
Training for alpha = 0.01 eta = 0.001 parameters
Training for alpha = 0.01 eta = 0.01 parameters
Training for alpha = 0.01 eta = 0.1 parameters
Training for alpha = 0.01 eta = 1 parameters
Training for alpha = 0.01 eta = 10 parameters
Training for alpha = 0.1 eta = 1e-05 parameters
Training for alpha = 0.1 eta = 0.0001 parameters
Training for alpha = 0.1 eta = 0.001 parameters
Training for alpha = 0.1 eta = 0.01 parameters
Training for alpha = 0.1 eta = 0.1 parameters
Training for alpha = 0.1 eta = 1 parameters
Training for alpha = 0.1 eta = 10 parameters
Training for alpha = 1 eta = 1e-05 parameters
Training for alpha = 1 eta = 0.0001 parameters
Training for alpha = 1 eta = 0.001 parameters
Training for alpha = 1 eta = 0.01 parameters
Training for alpha = 1 eta = 0.1 parameters
Training for alpha = 1 eta = 1 parameters
Training for alpha = 1 eta = 10 parameters
```

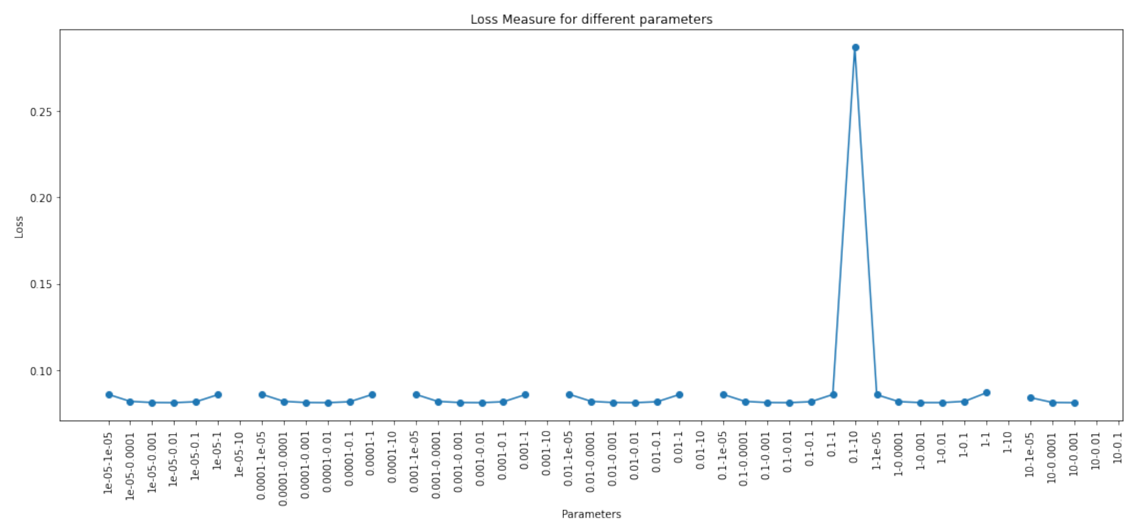
```
/home/parth/AppliedAI/appliedai/lib/python3.7/site-
packages/ipykernel_launcher.py:2: RuntimeWarning: overflow encountered in exp

/home/parth/AppliedAI/appliedai/lib/python3.7/site-
```

```
packages/ipykernel_launcher.py:32: RuntimeWarning: overflow encountered in add

Training for alpha = 10 eta = 1e-05 parameters
Training for alpha = 10 eta = 0.0001 parameters
Training for alpha = 10 eta = 0.001 parameters
Training for alpha = 10 eta = 0.01 parameters
Training for alpha = 10 eta = 0.1 parameters
Training for alpha = 10 eta = 1 parameters
Training for alpha = 10 eta = 10 parameters
```

[48]:
```
_, ax = plt.subplots(1,1,figsize=(15,7))
ax.plot(list(loss_dict.keys()), list(loss_dict.values()))
ax.scatter(list(loss_dict.keys()), list(loss_dict.values()))
for tick in ax.get_xticklabels():
    tick.set_rotation(90)
ax.set_title('Loss Measure for different parameters')
ax.set_ylabel('Loss')
ax.set_xlabel('Parameters')
plt.tight_layout()
plt.show()
```



[49]:
```
loss_dict = sorted(loss_dict.items(), key = lambda x: x[1] )
```

[50]:
```
print('Best Param ' ,','.join(loss_dict[0][0].split('-')))
```

```
Best Param  0.1,0.01
```

0.5.4 Training of Best Parameters

[51]:
```
w, b = train(f_cv.reshape(-1,1), y_cv_new, 1, 0.01, 1000)
```

[52]:
```
f_test = decision_function(x_test, clf)

y_pred = np.array([])
for ele in f_test:
    y_pred = np.append(y_pred, sigmoid(np.dot(w,ele)+ b))
```

[53]:
```
log_loss(y_test, y_pred)
```

[53]:
```
0.0854393007167031
```

[54]:
```
y_pred = y_pred *100
```

[55]:
```
print('Checkingtop 20 elements,  probabilities with actual values of y_test')
for ind,ele in enumerate(y_pred[:20]):
    print(ele, y_test[ind], sep=' => ')
```

```
Checkingtop 20 elements,  probabilities with actual values of y_test
0.7498162514295279 => 0
0.3664806515291726 => 0
99.06732116666606 => 1
90.05880425046945 => 1
6.552091901153191 => 0
0.8941745543653796 => 0
40.5085837179611 => 1
20.800904590261553 => 1
0.15918599060540412 => 0
95.19375767505129 => 1
0.4521516569767118 => 0
97.81665217773586 => 1
99.88635252115084 => 1
0.21248461768622784 => 0
1.0042084014886024 => 0
4.465097732352097 => 0
96.16297103305209 => 1
0.03952765487819778 => 0
4.2759718166763845 => 0
17.577004972012666 => 1
```

Note: in the above algorithm, the steps 2, 4 might need hyper parameter tuning, To reduce the complexity of the assignment we are excluding the hyerparameter tuning part, but intrested students can try that

If any one wants to try other calibration algorithm istonic regression also please check these tutorials

1. http://fa.bianp.net/blog/tag/scikit-learn.html#fn:1

2. https://drive.google.com/open?id=1MzmA7QaP58RDzocB0RBmRiWfl7Co_VJ7
3. https://drive.google.com/open?id=133odBinMOIVb_rh_GQxxsyMRyW-Zts7a
4. https://stat.fandom.com/wiki/Isotonic_regression#Pool_Adjacent_Violators_Algorithm