08 '22 June

Week 24 Day (159-206) | Wednesday

JUNE 2022

MON	TUE	WED	THU	FRI	SAT	SUN	MON	TUE	WED	THU	FRI	SAT	SUN
		1	2	3	4	5	6	7	8	9	10	11	12
13	14	15	16	17	18	19	20	21	22	23	24	25	26
27	28	29	30										

09.00 SUPPORT VECTOR MACHINE (SVM)

- FOR CLASSIFICATION, WE HAVE SEEN LOGISTIC REGRESSION

10.00 LDA, QDA and DECISION TREES.

- SUPPORT VECTOR MACHINE (SVM) IS ANOTHER ALGORITHM

11.00 FOR CLASSIFICATION

- MAIN ADVANTAGE - THEY CAN ACCOMODATE NON-LINEAR

12.00 BOUNDARIES.

13.00 — MAXIMUM MARGIN CLASSIFIER —

- MAXIMAL MARGINAL CLASSIFIER IS BASIC ALGORITHM FROM

14.00 WHICH SVM EXTENDS

- IT RELIES ON SEPERATING CLASSES USING HYPERPLANE

15.00

HYPERPLANE —

16.00 . IN A p-DIMENSIONAL SPACE, A HYPERPLANE IS A FLAT AFFINE SUBSPACE OF DIMENSION p-1

17.00 . IN A 2D SPACE, THE HYPERPLANE WILLBE A LINE

. IN A 3D SPACE, THE HYPERPLANE WILL BE FLAT PLANE

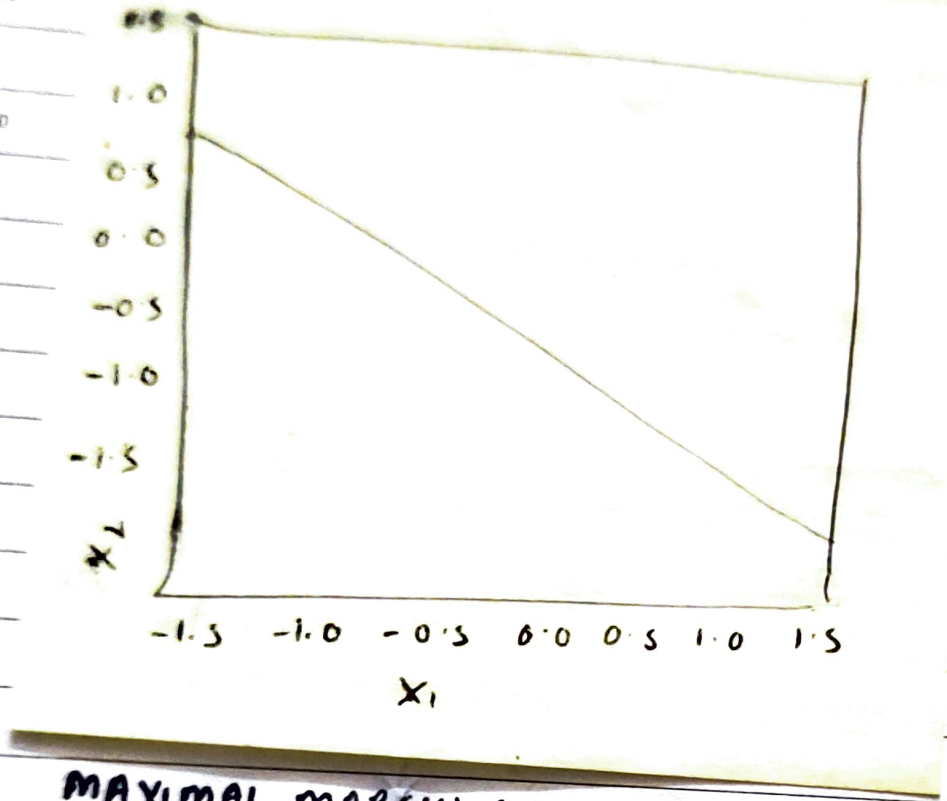
18.00

HYPERPLANE ——

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \_\_ + \beta_p X_p = 0$$

June '22 Thursday 09

09.00 — HYPERPLANE IN 2D —



10.00
11.00
12.00
13.00
14.00
15.00
16.00

MAXIMAL MARGIN CLASSIFIER

- IF DATA CAN BE SEPERATED PERFECTLY, THERE IS AN INFINITE NUMBER OF HYPERPLANES

17.00

- USE MARGINAL HYPERPLANE OR OPTIMAL SEPERATING HYPERPLANE.

18.00

→ TO DO SO : WE CALLULATE THE PERPENDICULAR DISTANCE BETWEEN EALH TRAINING OBSERVATION AND THE HYPERPLANE : MARGIN

- THE OPTIMAL SEPERATING HYPERPLANE IS THE ONE WITH LARGEST MARGIN

A good answer is one that knows when to stop.

10 '22 June

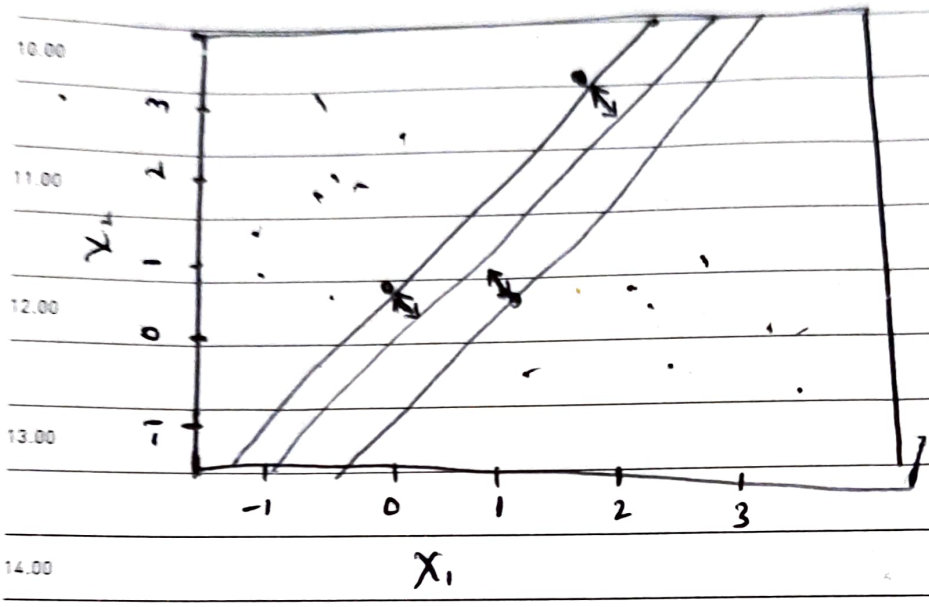
Week 24 Day 162-204

Friday

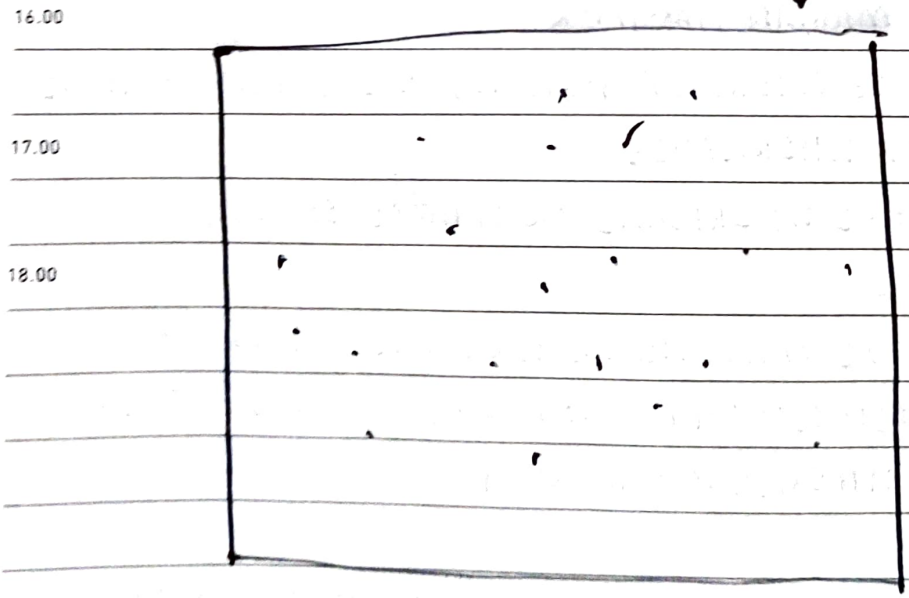
JUNE 2022

MON	TUE	WED	THU	FRI	SAT	SUN	MON	TUE	WED	THU	FRI	SAT	SUN
		1	2	3	4	5	6	7	8	9	10	11	12
13	14	15	16	17	18	19	20	21	22	23	24	25	26
27	28	29	30										

09.00 - MAXIMAL MARGIN CLASSIFIER



15.00 BUT WHAT IF THERE IS NO CLEAR SEPERATION BETWEEN THE CLASSES (POINTS)



THATS WHEN WE REQUIRE

→ SVM

JULY 2022

MON	TUE	WED	THU	FRI	SAT	SUN
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

June '22

Saturday 11

Week 24 Day (162-203)

09.00 SUPPORT VECTOR MACHINE (SVM)

- SVM IS EXTENSION OF MAXIMAL MARGIN CLASSIFER.

10.00 - IT USES KERNELS TO ENLARGE THE FEATURE SPACE AND ACCOMODATES FOR NON-LINEAR BOUNDARIES BETWEEN

11.00 CLASSESS.

12.00 KERNEL

- A KERNEL IS FUNCTION THAT QUALIFIES THE SIMILARITY OF

13.00 TWO OBSERVATIONS

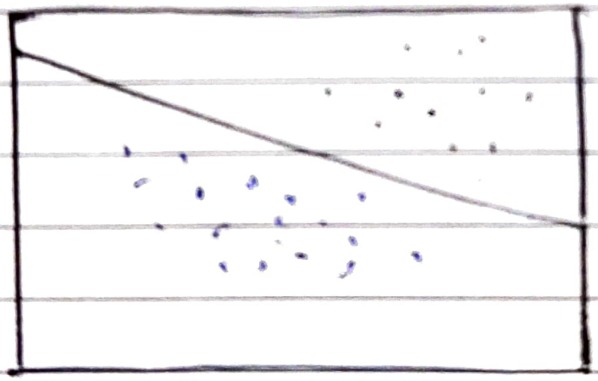
- THE KERNEL CAN BE OF ANY DEGREE

14.00 - A DEGREE GREATER THAN 1 ADDS MORE FLEXIBLITY TO THE BOUNDRY

15.00

SVM

16.00



17.00

Sunday 12

18.00

13 '22 June

Monday

Week 25 Day (164-201)

JUNE 2022

MON	TUE	WED	THU	FRI	SAT	SUN	MON	TUE	WED	THU	FRI	SAT	SUN
		1	2	3	4	5	6	7	8	9	10	11	12
13	14	15	16	17	18	19	20	21	22	23	24	25	26
27	28	29	30										

09.00 HANDS ON EXPERIENCE IN PYTHON (SVM)

10.00 ## SUPPORT VECTOR MACHINE (SVM) # CODEDANIEL

11.0

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
% matplotlib inline
import matplotlib.cm as cm
from scipy.io import loadmat
```

12.0

13.0

14.0

15.

16.

17.00

importing libraries

18.00

scipy.io loadmat - a function in the scipy library for reading MATLAB files (with extension .mat) and converting them into Python data structures.

```
def plot_data (X, y, xlabel, ylabel,
pos_label, neg_label, x_min, x_max,
y_min, y_max, axes = None):
plt.rcParams['figure.figsize'] = (20, 14)
pos = y[:, 0] == 1
neg = y[:, 0] == 0
if axes == None:
    axes = plt.gca()
axes.scatter(X[pos][:,0], X[pos][:,1],
marker='o', c='#003f5c', s=50,
linewidth=2, label=pos_label)
```

```
axes.scatter(X[neg][:,0], X[neg][:,1],
marker='o', c='#ffa600', s=50,
linewidth=2, label=neg_label)
axes.set_xlim([x_min, x_max])
axes.set_ylim([y_min, y_max])
axes.set_xlabel(xlabel, fontsize=12)
axes.set_ylabel(ylabel, fontsize=12)
axes.legend(bbox_to_anchor=(1,1),
    fancybox=True)
```

15 June 22
Week 25 Day (166, 199) Wednesday

09.00 X - a two dimensional numpy of shape (m, 2) containing the
10.00 coordinates of the data points to be plotted.
y - a one dimensional numpy array of shape (m)
11.00 containing the labels of data point (either 0 or 1)
xlabel - string specifying the label for the x-axis of
12.00 the plot.
ylabel - string specifying the label for y-axis of plot.
13.00 pos label - string specifying the label for the positive class (y=1)
neg label - string specifying the label for the negative class (y=0)
14.00 xmin - float specifying the minimum value of X-axis
x max - float specifying the maximum value of x-axis
15.00 ymin & ymax - same as X
axes - optimal parameter specifying matplotlib axes about-
16.00 object which create the plot.

17.00 The function first sets the size of plot using plt.rcParams
['figure.figsize']. It then split the data into array
18.00 into two array based on their labels (pos and neg).

It creates scatter plot for each class using plt.scatter with different colors and markers. The c parameter specifies the color of marker, s specifies the size and linewidth specifies width of marker edge. The function then sets the limit and labels of axis using axes.set_xlim, axes.set_ylim, axes.set_xlabel & axes.
set_ylabel - add legend, bbox - specifies location, fancybox - add rounded box

09.00 SVM With Small Regularization —

```
data1 = loadmat(datapath_1)
X = data1['X']
y = data1['y']
plot_data = (X, y, 'X', 'y',
'positive', 'negative', 0, 4.2, 0, 5)
```

datapath_1 is file path of the MATLAB file. code loads using scipy.io loadmat and stores them in numpy array X

10.00

11.00

12.00

13.00 plot_data function creates scatter plot of data points in X with their corresponding labels in y.

14.00

```
From sklearn import svm
# use C = 1 regularization parameter
clf = svm.SVC(kernel = 'linear', C=1.0,
decision_function_shape = 'ovr')
clf.fit(X, y, ravel())
# plot the data as well as boundry
plot_data(X, y, 'X', 'y', 'positive', 'negative',
0, 4.2, 0, 5)
# plot hyperplane
X_1, X_2 = np.meshgrid(np.arrange
(0.0, 5.0, 0.01)), np.arrange(0.0, 5.0, 0.01))
z = clf.predict(np.c_[X_1.ravel(), X_2.
ravel()])
z = z.reshape(X_1.shape)
plt.contour(X_1, X_2, z, [0.5], colors='b')
```

15.00

16.00

17.00

18.0

17 22 June

Friday

JUNE 2022

SVM with large Regularization

```
# Use C=100 regularization parameter
clf = svm.SVC(kernel='linear', C=100,
decision_function_shape = 'ovr')
# Plot Data as well as boundry
plot_data(X, y, 'X', 'y', 'positive', 'negative',
  0, 4.2, 0, 5)
# Plot hyperplane
X_1, X_2 = np.meshgrid(np.arrange(0.0, 5.0,
0.01), np.arrange(0.0, 5.0, 0.01))
z = clf.predict(np.c_[X_1.ravel(), X_2.ravel()
                ])
z = z.reshape(X_1.shape)
plt.contour(X_1, X_2, z, [0.5], colors='b')
```

LINEAR svm using scikit learn library svm.SVC class with the 'linear' kernel and C=1.0 regularization power parameter. After training classifier, you are plotting data using plot_data function with X and y as inputs along with other parameter to specify the plot details.

To plot the decision boundry or hyperplane, create meshgrid X_1 and X_2 coordinate using np meshgrid. The predicting class labels for each point in the meshgrid using trained classifier clf.predict(np.c_[X_1.ravel(), X_2.ravel()]). predicted Labels are reshaped to the same shape as meshgrid and then plot using plt.contour with contour level set to 0.5 (i.e decision boundry). Color is b (blue).

Hope is the pillar that holds

JULY 2022

MON	TUE	WED	THU	FRI	SAT	SUN
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

June '22

Saturday 18

Week 25 Day (169-196)

09.00 # Svm With Non Linear Boundry

```
data2 = loadmat(datapath_2)
X_2 = data2['X']
y_2 = data2['y']
plot_data(X_2, y_2, 'X',
'y', 'positive', 'negative', 0,
1.0, 0.38, 1)
```

10.00

11.00

12.00

13.00 It seems like you are loading data from a .mat using Scipy library loadmat function and storing the data in

14.00 X_2 and y_2 variables.

After loading, call plot_data function with X_2 and y_2

15.00 as input along with other parameters to specify the plot details such as axis labels and color scheme.

16.00

```
sigma = 0.1
gamma = 1/(2 * sigma**2)
clf = svm.SVC(kernel='rbf', gamma=gamma,
C=1.0, decision_function_shape='ovr')
clf.fit(X_2, y_2.ravel())
plot_data(X_2, y_2, 'X', 'y', 'positive',
'negative', 0, 1.0, 0.38, 1)
X_1, X_2 = np.meshgrid(np.arange(0.0,
1.0, 0.003), np.arange(0.38, 1.0, 0.003))
z = clf.predict(np.c_[(X_1.ravel(), X_2.
                       ravel()])
z = z.reshape(X_1.shape)
plt.contour(X_1, X_2, z, colors='b')
```

17.00

18.00

Sunday 19

20 '22 June

Week 25 Day (171-194) Monday

JUNE 2022

09.00 Svm With Cross Validation

10.00

```
data3 = loadmat(datapth_3)
X_3 = data3['X']
y_3 = data3['y']
plot_data(X_3, y_3, 'X',
'Y', 'positive', 'negative',
-0.55, 0.35, -0.8, 0.6)
```

11.00

12.00

13.00

```
Sigma = [0.01, 0.03, 0.1, 0.3, 1, 3, 10, 30]
C = [0.01, 0.03, 0.1, 0.3, 1, 3, 10, 30]
errors = list()
sigma_c = list()
for each in sigma:
    for each_c in C:
        clf = svm.SVC(kernel='rbf',
        gamma = 1/(2*each**2)), C=each_c,
    decision_function_shape = 'ovr')
    clf.fit(X_3, y_3.ravel())
errors.append(clf.score(data3['Xval'],
data3['yval'].ravel()))
sigma_c.append((each, each_c))
```

```
index = np.argmax(errors)
Sigma_max, C_max = sigma_c
                   [index]
print(f"The optimal value is
      {sigma_max}")
print(f"The optimal value of C is:
      {C_max}")
```

Output —

The optimal value of sigma is : 0.1

The optimal value of C is : 1

Next Code —

```
sigma = 0.1
gamma = 1/(2 * sigma ** 2)

optimal_clf = svm.SVC(kernel='rbf', gamma=gamma, C=1.0,
  decision_function_shape='ovr')
optimal_clf.fit(X_3, y_3.ravel())

plot_data(X_3, y_3, 'X', 'y', 'positive', 'negative', -0.55, 0.35,
  -0.8, 0.6)

X_1, X_2 = np.meshgrid(np.arrange(-0.6, 0.4, 0.004), np.predict
(-0.8, 0.6, 0.004))
Z = optimal_clf.predict(np.c_[X_1.ravel(), X_2.ravel()])
Z = Z.reshape(X_1.shape)
plt.contour(X_1, X_2, Z, [0.5], colors='b')
```

22 June

Week 26 Day (173-192) | Wednesday

MON	TUE	WED	THU	FRI	SAT	SUN
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30			

09.00 SPAM CLASSIFICATION

10.00
```
spam_train = loadmat (datapath_train)
spam_test = loadmat (datapath_test)
```
11.00
```
C = 0.1
X_train = spam_train ['X']
```
12.00
```
y_train = spam_train ['y']
```

13.00
```
X_test = spam_test ['Xtest']
y_test = spam_test ['ytest']
```
14.00
```
clf_spam = svm.SVC (kernel = 'linear', C = C, decision_function_
```
15.00
```
shape = 'ovr')
clf_spam.fit (X_trails, y_train.ravel())
```
16.00
```
train_acc = clf_spam.score (spam_train ['X'], spam_train ['y'].
```
17.00
```
ravel())
test_acc = clf_spam.score (X_test, y_test.ravel())
```
18.00
```
print (f" Training accuracy = {train_acc * 100}")
print (f" Test accuracy = {test_acc * 100}")
```