Assignment 4

Support Vector Machine

Date : 29th March,2019

Nanavati Tilak D.
(201811001)

Support Vector Machine :

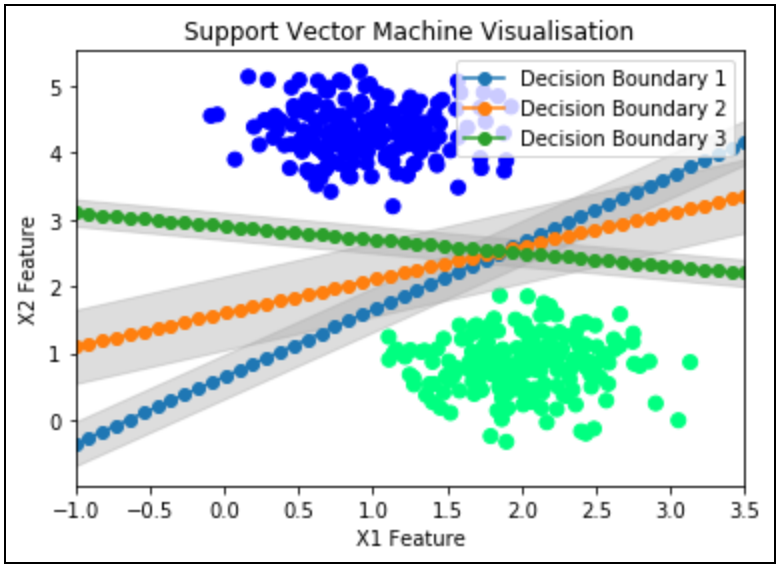
Support Vector Machine (**SVM**) is a supervised binary classification algorithm. Given a set of points of two types in **N** dimensional place **SVM** generates a **(N−1)** dimensional hyperplane to separate those points into two groups.

1. Code

```
# importing scikit learn with make_blobs
from sklearn.datasets.samples_generator import make_blobs
import matplotlib.pyplot as plt
import numpy as np

# creating datasets X containing n_samples
# Y containing two classes
X, Y = make_blobs(n_samples=400,centers=2,
                  random_state=0, cluster_std=0.40)

xfit = np.linspace(-1, 3.5)

# plotting scatters
plt.scatter(X[:, 0], X[:, 1], c=Y, s=50, cmap='winter');

cords = [(1, 0.65, 0.33), (0.5, 1.6, 0.55), (-0.2, 2.9, 0.2)]
"""checking for all the values individually to get the idea of the
Desicion Boundary which is optimum
"""
m, b, d = 1, 0.65, 0.33

yfit = m * xfit + b

plt.plot(xfit, yfit, '-o',label = 'Decision Boundary 1')
plt.fill_between(xfit, yfit - d, yfit + d, edgecolor='none',
color='#AAAAAA', alpha=0.4)

m, b, d = 0.5, 1.6, 0.55

yfit = m * xfit + b

plt.plot(xfit, yfit, '-o',label = 'Decision Boundary 2')
plt.fill_between(xfit, yfit - d, yfit + d, edgecolor='none',
```

```
color='#AAAAAA', alpha=0.4)

m, b, d = -0.2, 2.9, 0.2

yfit = m * xfit + b

plt.plot(xfit, yfit, '-o',label = 'Decision Boundary 3')
plt.fill_between(xfit, yfit - d, yfit + d, edgecolor='none',
color='#AAAAAA', alpha=0.4)

plt.legend(loc = 'upper right')
plt.xlim(-1, 3.5)
plt.xlabel('X1 Feature')
plt.ylabel('X2 Feature')
plt.title('Support Vector Machine Visualisation')
plt.show()
```

Output :



Observation :

The Support Vector Machine tends to increase the Marginal Distance between decision boundary and nearest data points. AS can ve seen from the figure there are infinite decision boundaries are possible but Support Vector machine helps find out optimal decision boundary.

Code :

```
import math as m
```

```
min_dist_1 = 10
```

```
min_dist_0 = 10
for i in range(400):
    x1 = X[i,0]
    y1 = X[i,1]
    y_class = Y[i]

    for j in range(50):

        dist = m.sqrt((xfit[j]-x1)**2+(yfit[j]-y1)**2)
        ##print(y_class)

        if(min_dist_1>dist and Y[i] == 1):
            min_dist_1 = dist
            temp1 = x1
            temp2 = y1

        if(min_dist_0>dist and Y[i] == 0):
            min_dist_0 = dist
            temp3 = x1
            temp4 = y1

print("Co-ordinates of the Support Vector From Negative Side:")
print("X1 Feature :",temp3,"X2 Feature :",temp4)
print("Support Vector Distance :",min_dist_0)
print("*****************************************************")
print()
print("*****************************************************")
print("Co-ordinates of the Support Vector From Positive Side:")
print("X1 Feature :",temp1,"X2 Feature :",temp2)
print("Support Vector Distance :",min_dist_1)
```

Output :

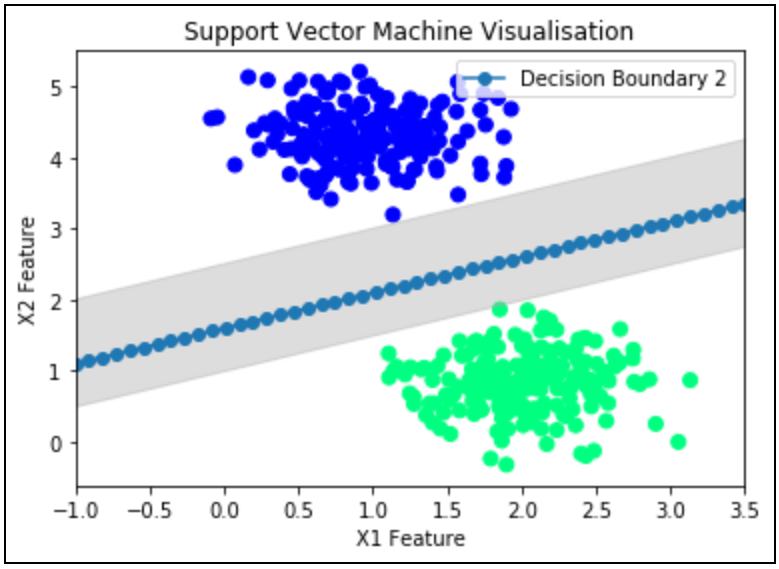
```
Co-ordinates of the Support Vector From Negative Side:
X1 Feature : 1.1358886168025473 X2 Feature : 3.1947502248777297
Support Vector Distance : 0.9188221824345093
**************************************************

**************************************************
Co-ordinates of the Support Vector From Positive Side:
X1 Feature : 1.8541408411758626 X2 Feature : 1.8626451317554364
Support Vector Distance : 0.594581842380754
```

Code :

```
#now as can be seen we are picking the optimum line as the
Decison Boundary
#that is Decision Boundary 2.

# importing scikit learn with make_blobs
from sklearn.datasets.samples_generator import make_blobs
import matplotlib.pyplot as plt
import numpy as np

# creating datasets X containing n_samples
# Y containing two classes
X, Y = make_blobs(n_samples=400,centers=2,
                  random_state=0, cluster_std=0.40)

xfit = np.linspace(-1, 3.5)

# plotting scatters
plt.scatter(X[:, 0], X[:, 1], c=Y, s=50, cmap='winter');

cords = [(1, 0.65, 0.33), (0.5, 1.6, 0.55), (-0.2, 2.9, 0.2)]
```

```
"""checking for all the values individually to get the idea of
the
Desicion Boundary which is optimum
"""
m, b, d = 0.5, 1.6, 0.59

yfit = m * xfit + b
yfit_new = m * xfit + 2.0

plt.plot(xfit, yfit,'-o',label = 'Decision Boundary 2')
#plt.plot(xfit, yfit_new,'-o',label = 'Decision Boundary New')
plt.fill_between(xfit, yfit - min_dist_1, yfit + min_dist_0,
edgecolor='none',
color='#AAAAAA', alpha=0.4)

plt.legend(loc = 'upper right')
plt.xlim(-1, 3.5)
plt.xlabel('X1 Feature')
plt.ylabel('X2 Feature')
plt.title('Support Vector Machine Visualisation')
plt.show()
```

Output :

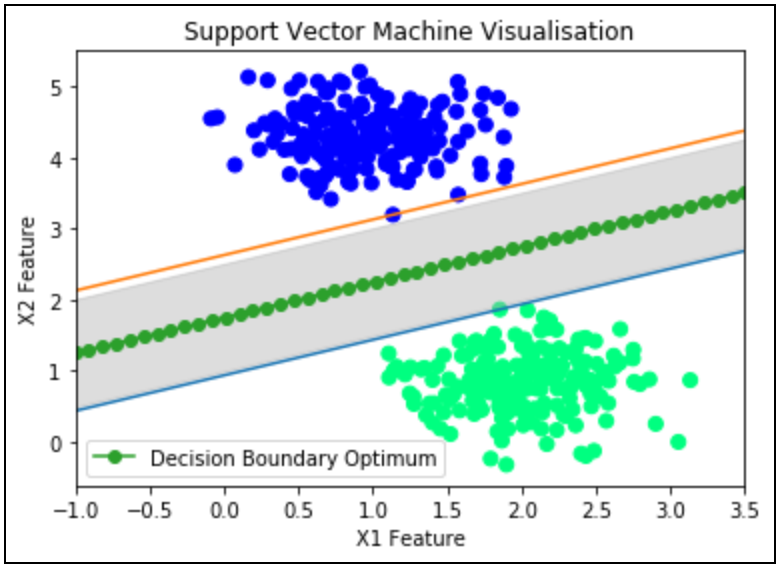


Code :

```
#now as can be seen we are picking the optimum line as the
Decison Boundary
#that is Decision Boundary 2.

# importing scikit learn with make_blobs
from sklearn.datasets.samples_generator import make_blobs
import matplotlib.pyplot as plt
import numpy as np

# creating datasets X containing n_samples
# Y containing two classes
X, Y = make_blobs(n_samples=400,centers=2,
                  random_state=0, cluster_std=0.40)

xfit = np.linspace(-1, 3.5)

# plotting scatters
plt.scatter(X[:, 0], X[:, 1], c=Y, s=50, cmap='winter');

cords = [(1, 0.65, 0.33), (0.5, 1.6, 0.55), (-0.2, 2.9, 0.2)]
"""checking for all the values individually to get the idea of
the
```

```
Desicion Boundary which is optimum
"""
theta = 0.5

#Plotting Support Vector Margin Lines
offset_1 = temp2-0.5 * temp1
plt.plot(xfit,0.5*xfit+offset_1)

offset_0 = temp4 - 0.5 * temp3
plt.plot(xfit,0.5*xfit+offset_0)

#Plotting the NEW DECISION BOUNDARY
yfit_new = m * xfit + (offset_0+offset_1)/2

plt.plot(xfit, yfit,'-o',label = 'Decision Boundary Optimum')

plt.fill_between(xfit,yfit-(min_dist_1+min_dist_0)/2,yfit+(min_d
ist_1+min_dist_0)/2,edgecolor='none',color='#AAAAAA',alpha=0.4)

plt.legend(loc = 'lower left')
plt.xlim(-1, 3.5)
plt.xlabel('X1 Feature')
plt.ylabel('X2 Feature')
plt.title('Support Vector Machine Visualisation')
plt.show()
```

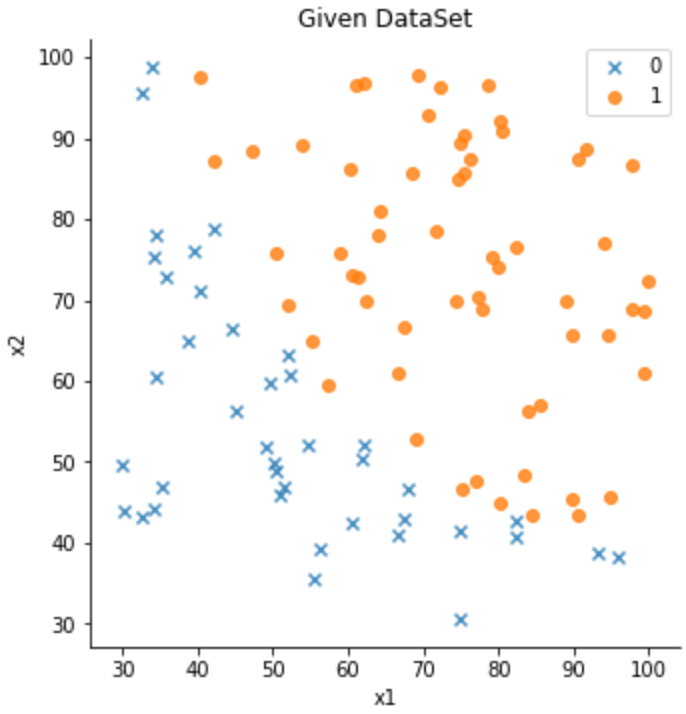
Output :



Code :

```
# Implementing SVM on the DataSet Provided:
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import math as m

dataset = pd.read_csv("ex2data1-logistic.csv")

sns.lmplot(x= "x1" , y= "x2" ,hue="y",data=dataset, fit_reg=
False ,legend= False,markers=["x","o"])
plt.legend()
plt.title( "Given DataSet" )
plt.show()
```

Output :

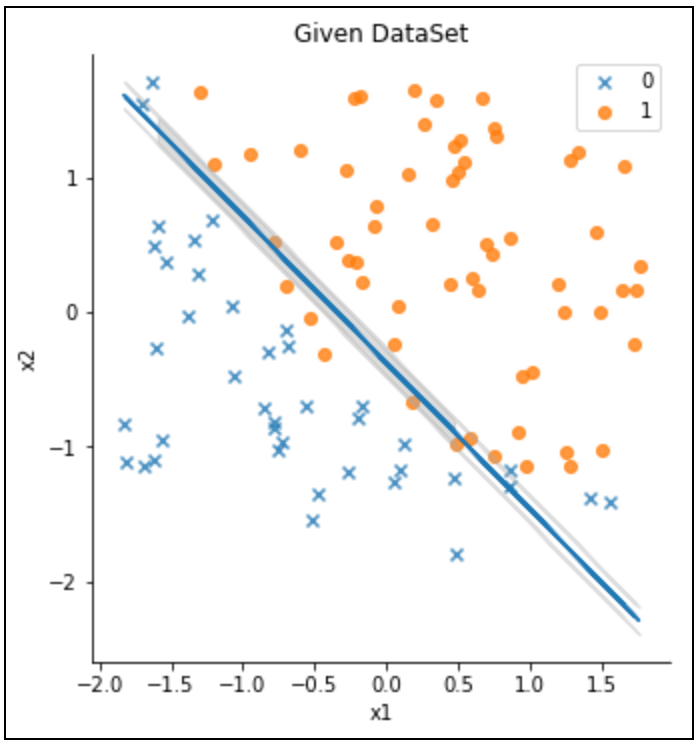


Code :

```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import math as m

dataset = pd.read_csv("ex2data1-logistic.csv")
y = dataset["y"]

mn = [dataset["x1"].mean(),dataset["x1"].mean()]
std = [dataset["x1"].std(),dataset["x1"].std()]

total = len(dataset["x1"])

for i in range(total):
    dataset.iloc[i,0] = (dataset.iloc[i,0]-mn[0])/std[0]
    dataset.iloc[i,1] = (dataset.iloc[i,1]-mn[1])/std[1]
```

```
ln = []
#print(dataset)
for i in range(total):
    c = theta[0]
    c = (c+theta[1]*dataset.iloc[i,0])/theta[2]
    ln.append(-c)
print(len(ln))
#print(ln)
sns.lmplot(x= "x1" , y= "x2" ,hue="y",data=dataset, fit_reg= False
,legend= False,markers=["x","o"])
plt.legend()
plt.title("Given DataSet")
plt.plot(dataset["x1"],ln)
d = 0.1

lnmd = []
lnpd = []

for l in ln:
    lnpd.append(l+d)
    lnmd.append(l-d)

plt.fill_between(dataset["x1"],lnmd,lnpd,
edgecolor='none',color='#AAAAAA', alpha=0.4)
plt.show()
```

Output :



Code :

```
import pandas as pd
import math as m
import numpy as np

def gard_desc(m1,x,y,theta):

    for j in range(1200):
        alpha = 0.01
        A = []
        for i in range(m1):
            A.append(np.dot(x[i],theta))

        c = []
        for i in range(m1):
            c.append(1/(1+m.exp(-A[i])))
```

```
        T = []
        for i in range(m1):
            #print(c[i])
            if(c[i]!=1):
                T.append(y[i]*m.log(c[i]) +
(1-y[i])*m.log(1-c[i]))
            else:
                T.append(0)

        B = [0,0,0,0,0]

        for i in range(m1):
            mul = (c[i]-y[i])
            x_m = pd.Series(x[i])
            B = (pd.Series(B) + (mul*x_m).tolist()).tolist()

        for i in range(len(theta)):
            theta[i] = theta[i] - (alpha*B[i])/m1

    return theta

dataset = pd.read_csv("ex2data1-logistic.csv")
theta = [0,0,0,0,0]

m1 = len(dataset["x1"])
x =[]
mn = [dataset["x1"].mean(),dataset["x1"].mean()]
std = [dataset["x1"].std(),dataset["x1"].std()]
for i in range(m1):
    l=[1]

    x1 = (dataset.iloc[i,0]-mn[0])/std[0]
    x2 = (dataset.iloc[i,1]-mn[1])/std[1]

    l.append(x1)
```

```
    l.append(x2)
    l.append(x1**2)
    l.append(x2**2)
    x.append(l)

#print(x)
y = list(dataset.iloc[:,2])
#print(y)
#print(np.dot(x[1],[1,1,1]))
theta_new = gard_desc(m1,x,y,theta)
print(theta_new)
```

Output :

```
[0.6244230254113816, 1.378501023539279, 1.2882509750416904,
-0.25371944638524385, -0.0570993709230383]
```

Code :

```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import math as m

dataset = pd.read_csv("ex2data1-logistic.csv")
y = dataset["y"]

mn = [dataset["x1"].mean(),dataset["x1"].mean()]
std = [dataset["x1"].std(),dataset["x1"].std()]

total = len(dataset["x1"])
```

```
for i in range(total):
    dataset.iloc[i,0] = (dataset.iloc[i,0]-mn[0])/std[0]
    dataset.iloc[i,1] = (dataset.iloc[i,1]-mn[1])/std[1]

ln = []
#print(dataset)
for i in range(total):
    c = theta[0]
    c = ((c+theta[3]*(dataset.iloc[i,0]**2))/theta[4])
    if c>0:
        ln.append(m.sqrt(c)+0.95)
    else:
        ln.append(-m.sqrt(-c)+0.95)
print(len(ln))
#print(ln)
sns.lmplot(x= "x1" , y= "x2" ,hue="y",data=dataset, fit_reg=
False ,legend= False,markers=["x","o"])
plt.legend()
plt.title("Given DataSet")
xd1 = np.arange(-2,2,0.04)
ln.sort()
plt.plot(-xd1+0.65,ln)
d = 0.2
lnmd = []
lnpd = []

for l in ln:
    lnpd.append(l+d)
    lnmd.append(l-d)

plt.fill_between(-xd1+0.70,lnmd,lnpd,edgecolor='none',color='#AA
AAAA', alpha=0.4)
plt.show()
```

Output :