ML 28 - Support Vector Machine Kernel By Virat Tiwari

December 28, 2023

1 Support Vector Machine Kernels By Virat Tiwari

[5]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
```

[6]:

```
x = np.linspace(-5.0, 5.0, 100)
y = np.sqrt(10**2 - x**2)
y=np.hstack([y,-y])
x=np.hstack([x,-x])
```

[7]:

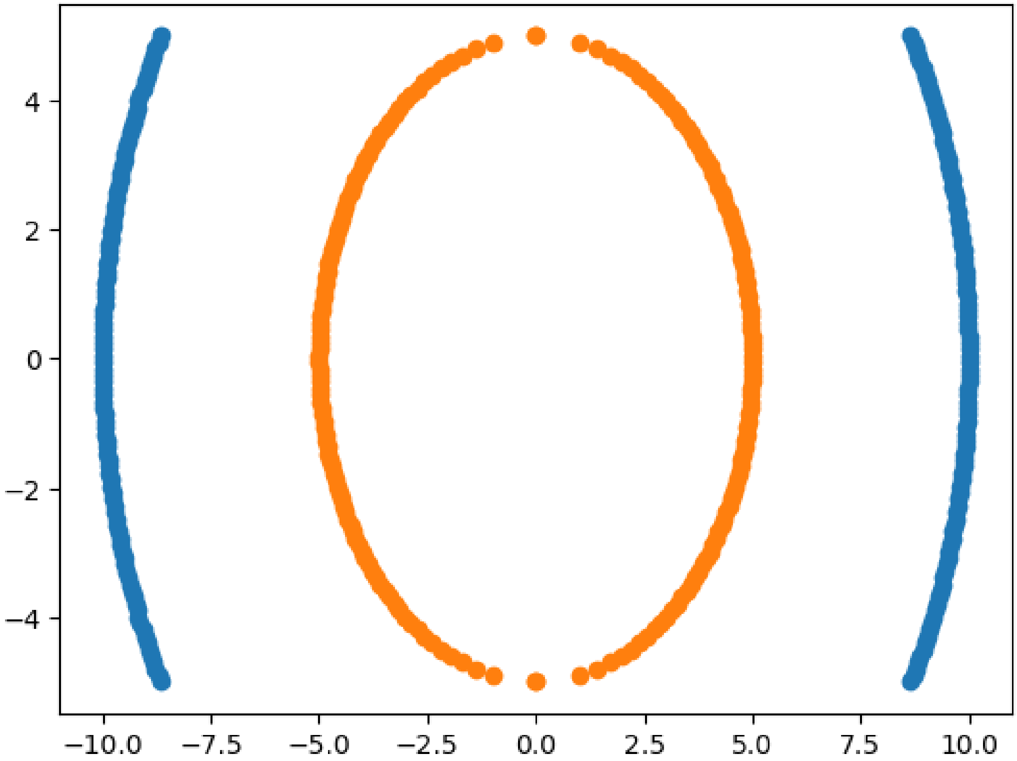
```
x1 = np.linspace(-5.0, 5.0, 100)
y1 = np.sqrt(5**2 - x1**2)
y1=np.hstack([y1,-y1])
x1=np.hstack([x1,-x1])
```

[8]:

```
plt.scatter(y,x)
plt.scatter(y1,x1)
```

[8]:

```
<matplotlib.collections.PathCollection at 0x7fabd7fa98d0>
```



[9]:
```
import pandas as pd
df1 =pd.DataFrame(np.vstack([y,x]).T,columns=['X1','X2'])
df1['Y']=0
df2 =pd.DataFrame(np.vstack([y1,x1]).T,columns=['X1','X2'])
df2['Y']=1
df = df1.append(df2)
df.head(5)
```

```
/tmp/ipykernel_110/1241201207.py:6: FutureWarning: The frame.append method is
deprecated and will be removed from pandas in a future version. Use
pandas.concat instead.
  df = df1.append(df2)
```

[9]:
```
         X1       X2  Y
0  8.660254 -5.00000  0
1  8.717792 -4.89899  0
2  8.773790 -4.79798  0
3  8.828277 -4.69697  0
4  8.881281 -4.59596  0
```

[10]:
```
df.tail()
```

```
[10]:             X1       X2  Y
      195 -1.969049 -4.59596  1
      196 -1.714198 -4.69697  1
      197 -1.406908 -4.79798  1
      198 -0.999949 -4.89899  1
      199 -0.000000 -5.00000  1
```

```
[11]: # Polynomial kernel
```

```
[12]: df['X1_Square']=df['X1']**2
      df['X2_Square']=df['X2']**2
      df['X1*X2']=df['X1']*df['X2']
      df.head()
```

```
[12]:         X1       X2  Y  X1_Square  X2_Square      X1*X2
      0  8.660254 -5.00000  0  75.000000  25.000000 -43.301270
      1  8.717792 -4.89899  0  75.999898  24.000102 -42.708375
      2  8.773790 -4.79798  0  76.979390  23.020610 -42.096467
      3  8.828277 -4.69697  0  77.938476  22.061524 -41.466150
      4  8.881281 -4.59596  0  78.877155  21.122845 -40.818009
```

```
[13]: # Independent and Dependent features

      X = df[['X1_Square','X2_Square','X1*X2']]
      y = df['Y']
```

```
[14]: from sklearn.model_selection import train_test_split
      X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                          test_size = 0.25,
                                                          random_state = 0)
```

```
[17]: df.head()
```

```
[17]:         X1       X2  Y  X1_Square  X2_Square      X1*X2
      0  8.660254 -5.00000  0  75.000000  25.000000 -43.301270
      1  8.717792 -4.89899  0  75.999898  24.000102 -42.708375
      2  8.773790 -4.79798  0  76.979390  23.020610 -42.096467
      3  8.828277 -4.69697  0  77.938476  22.061524 -41.466150
      4  8.881281 -4.59596  0  78.877155  21.122845 -40.818009
```

```
[19]: from sklearn.metrics import accuracy_score
      from sklearn.svm import SVC
      classifier = SVC(kernel="linear")
      classifier.fit(X_train, y_train)
      y_pred = classifier.predict(X_test)
      accuracy_score(y_test, y_pred)
```

```
[19]: 1.0
```

```
[20]: # Radial Basis Function Kerne
```

```
[21]: df.head()
```

```
[21]:          X1       X2  Y  X1_Square  X2_Square      X1*X2
      0  8.660254 -5.00000  0  75.000000  25.000000 -43.301270
      1  8.717792 -4.89899  0  75.999898  24.000102 -42.708375
      2  8.773790 -4.79798  0  76.979390  23.020610 -42.096467
      3  8.828277 -4.69697  0  77.938476  22.061524 -41.466150
      4  8.881281 -4.59596  0  78.877155  21.122845 -40.818009
```

```
[22]: # Indpeendent Features

      X=df.iloc[:,0:2]
      y=df.Y
```

```
[23]: X.head()
```

```
[23]:          X1       X2
      0  8.660254 -5.00000
      1  8.717792 -4.89899
      2  8.773790 -4.79798
      3  8.828277 -4.69697
      4  8.881281 -4.59596
```

```
[24]: y
```

```
[24]: 0      0
      1      0
      2      0
      3      0
      4      0
            ..
      195    1
      196    1
      197    1
      198    1
      199    1
      Name: Y, Length: 400, dtype: int64
```

```
[25]: from sklearn.model_selection import train_test_split
      X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                          test_size = 0.25,
                                                          random_state = 0)
```

[26]:

```
classifier = SVC(kernel="rbf")
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
accuracy_score(y_test, y_pred)
```

[26]:

```
1.0
```

[27]:

```
classifier = SVC(kernel="poly")
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
accuracy_score(y_test, y_pred)
```

[27]:

```
0.59
```

[28]:

```
# Sigmoid Kernel
```

[29]:

```
classifier = SVC(kernel="sigmoid")
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
accuracy_score(y_test, y_pred)
```

[29]:

```
0.51
```

THANK YOU SO MUCH !!

YOURS VIRAT TIWARI :)