Machine Learning Lab: Support Vector Machine Classifier
Nikhil Saraogi [21MCA1080]

Objective:-

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to **create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future**. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine.

Methodology:-

The main objective is to segregate the given dataset in the best possible way. The distance between the either nearest points is known as the margin. The objective is to select a hyperplane with the maximum possible margin between support vectors in the given dataset. SVM searches for the maximum marginal hyperplane in the following steps:

Generate hyperplanes which segregates the classes in the best way. Left-hand side figure showing three hyperplanes black, blue and orange. Here, the blue and orange have higher classification error, but the black is separating the two classes correctly.

Select the right hyperplane with the maximum segregation from the either nearest data points as shown in the right-hand side figure.

SVM Kernels

The SVM algorithm is implemented in practice using a kernel. A kernel transforms an input data space into the required form. SVM uses a technique called the kernel trick. Here, the kernel takes a low-dimensional input space and transforms it into a higher dimensional space. In other words, you can say that it converts nonseparable problem to separable problems by adding more dimension to it. It is most useful in non-linear separation problem. Kernel trick helps you to build a more accurate classifier.

1. **Linear Kernel-** A linear kernel can be used as normal dot product any two given observations. The product between two vectors is the sum of the multiplication of each pair of input values.

   K(x, xi) = sum(x * xi)

2. **Polynomial Kernel-** A polynomial kernel is a more generalized form of the linear kernel. The polynomial kernel can distinguish curved or nonlinear input space.

   K(x,xi) = 1 + sum(x * xi)^d

   Where d is the degree of the polynomial. d=1 is similar to the linear transformation. The degree needs to be manually specified in the learning algorithm.

3. **Radial Basis Function Kernel-** The Radial basis function kernel is a popular kernel function commonly used in support vector machine classification. RBF can map an input space in infinite dimensional space.

   K(x,xi) = exp(-gamma * sum((x – xi^2))

   Here gamma is a parameter, which ranges from 0 to 1. A higher value of gamma will perfectly fit the training dataset, which causes over-fitting. Gamma=0.1 is considered to be a good default value. The value of gamma needs to be manually specified in the learning algorithm.

Advantages

SVM Classifiers offer good accuracy and perform faster prediction compared to Naïve Bayes algorithm. They also use less memory because they use a subset of training points in the decision phase. SVM works well with a clear margin of separation and with high dimensional space.

Dataset Description: - <u>Voice Gender</u>

Gender Recognition by Voice and Speech Analysis

This database was created to identify a voice as male or female, based upon acoustic properties of the voice and speech. The dataset consists of 3,168 recorded voice samples, collected from male and female speakers.

The following acoustic properties of each voice are measured and included within the CSV:

- meanfreq: mean frequency (in kHz)

- sd: standard deviation of frequency
- median: median frequency (in kHz)
- Q25: first quantile (in kHz)
- Q75: third quantile (in kHz)
- IQR: interquantile range (in kHz)
- skew: skewness (see note in specprop description)
- kurt: kurtosis (see note in specprop description)
- sp.ent: spectral entropy
- sfm: spectral flatness
- mode: mode frequency
- centroid: frequency centroid (see specprop)
- peakf: peak frequency (frequency with highest energy)
- meanfun: average of fundamental frequency measured across acoustic signal
- minfun: minimum fundamental frequency measured across acoustic signal
- maxfun: maximum fundamental frequency measured across acoustic signal
- meandom: average of dominant frequency measured across acoustic signal
- mindom: minimum of dominant frequency measured across acoustic signal
- maxdom: maximum of dominant frequency measured across acoustic signal
- dfrange: range of dominant frequency measured across acoustic signal
- modindx: modulation index. Calculated as the accumulated absolute difference between adjacent measurements of fundamental frequencies divided by the frequency range
- label: male or female

Dataset Link - https://www.kaggle.com/datasets/primaryobjects/voicegender

Experiment with Result:-

Reading the data

```
[2]: df = pd.read_csv('voice.csv')
     df.head()
```

[2]:

	meanfreq	sd	median	Q25	Q75	IQR	skew	kurt	sp.ent	sfm	...	centroid	meanfun	minfun	maxfun	meandc
0	0.059781	0.064241	0.032027	0.015071	0.090193	0.075122	12.863462	274.402906	0.893369	0.491918	...	0.059781	0.084279	0.015702	0.275862	0.0078
1	0.066009	0.067310	0.040229	0.019414	0.092666	0.073252	22.423285	634.613855	0.892193	0.513724	...	0.066009	0.107937	0.015826	0.250000	0.0090
2	0.077316	0.083829	0.036718	0.008701	0.131908	0.123207	30.757155	1024.927705	0.846389	0.478905	...	0.077316	0.098706	0.015656	0.271186	0.0079
3	0.151228	0.072111	0.158011	0.096582	0.207955	0.111374	1.232831	4.177296	0.963322	0.727232	...	0.151228	0.088965	0.017798	0.250000	0.2014
4	0.135120	0.079146	0.124656	0.078720	0.206045	0.127325	1.101174	4.333713	0.971955	0.783568	...	0.135120	0.106398	0.016931	0.266667	0.7128

5 rows × 21 columns

```
[4]: df.shape
```

```
[4]: (3168, 21)
```

Defining X and Y values

```
X = df.loc[:,df.columns != 'label']
X.head()
```

	meanfreq	sd	median	Q25	Q75	IQR	skew	kurt	sp.ent	sfm	mode	meanfun	minfun	maxfun	meandom
0	0.059781	0.064241	0.032027	0.015071	0.090193	0.075122	12.863462	274.402906	0.893369	0.491918	0.000000	0.084279	0.015702	0.275862	0.007812
1	0.066009	0.067310	0.040229	0.019414	0.092666	0.073252	22.423285	634.613855	0.892193	0.513724	0.000000	0.107937	0.015826	0.250000	0.009014
2	0.077316	0.083829	0.036718	0.008701	0.131908	0.123207	30.757155	1024.927705	0.846389	0.478905	0.000000	0.098706	0.015656	0.271186	0.007990
3	0.151228	0.072111	0.158011	0.096582	0.207955	0.111374	1.232831	4.177296	0.963322	0.727232	0.083878	0.088965	0.017798	0.250000	0.201497
4	0.135120	0.079146	0.124656	0.078720	0.206045	0.127325	1.101174	4.333713	0.971955	0.783568	0.104261	0.106398	0.016931	0.266667	0.712812

In [20]:
```
Y = df.loc[:,df.columns == 'label']
Y.head()
```

Out[20]:

	label
0	male
1	male
2	male
3	male
4	male

Splitting dataset into training set and testing set for better generalisation

```
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size = 0.2 , random_state = 1)
```

```
X_train
```

```
array([[ 1.86207010e+00, -1.71596009e-01,  1.76797458e+00, ...,
        -7.08404312e-01,  2.10497090e-03,  2.38645483e-01],
       [-1.66398788e-01,  6.06764772e-01, -2.58218587e-01, ...,
        -7.08404312e-01,  2.70613827e-01,  1.11204756e-01],
       [ 8.75265120e-01, -1.02179942e+00,  7.67376592e-01, ...,
         2.13073231e+00, -1.37697161e-01,  4.19888377e-01],
       ...,
       [ 4.25197419e-01,  9.12515119e-02,  7.55436212e-01, ...,
        -4.61522866e-01,  2.37327605e-01, -1.99812508e-01],
       [-1.30836832e+00,  1.16448382e+00, -1.81602035e+00, ...,
        -7.08404312e-01, -1.38925910e+00,  1.99713759e+00],
       [ 4.56609368e-01,  3.54136977e-02,  8.80430493e-01, ...,
        -4.61522866e-01,  9.76281728e-01, -3.77184091e-01]])
```

Default Linear kernel

In [39]:
```
from sklearn.svm import SVC
classifier = SVC(kernel='linear')
classifier.fit(X_train,Y_train)
Y_pred=classifier.predict(X_test)
```

In [38]:
```
from sklearn.metrics import confusion_matrix
cm= confusion_matrix(Y_test,Y_pred)
accuracy = float(cm.diagonal().sum())/len(Y_test)
print("\nAccuracy : ", accuracy)
```

```
Accuracy :  0.9779179810725552
```

In [40]:
```
cm
```

Out[40]:
```
array([[305,   6],
       [  8, 315]], dtype=int64)
```

Default RBF kernel

In [45]:
```
svc=SVC(kernel='rbf')
svc.fit(X_train,Y_train)
Y_pred=svc.predict(X_test)
con= confusion_matrix(Y_test,Y_pred)
accuracy = float(con.diagonal().sum())/len(Y_test)
print('Accuracy Score:',accuracy )
```

```
Accuracy Score: 0.9779179810725552
```

In [46]:
```
con
```

Out[46]:
```
array([[305,   6],
       [  8, 315]], dtype=int64)
```

Default Polynomial kerne

In [47]:
```
svc=SVC(kernel='poly')
svc.fit(X_train,Y_train)
Y_pred=svc.predict(X_test)
c= confusion_matrix(Y_test,Y_pred)
accuracy = float(c.diagonal().sum())/len(Y_test)
print('Accuracy Score:',accuracy )
```

```
Accuracy Score: 0.9652996845425867
```

In [48]:
```
c
```

Out[48]:
```
array([[294,  17],
       [  5, 318]], dtype=int64)
```

Polynomial kernel is performing poorly. The reason behind this maybe it is over-fitting the training dataset.