Support Vector Machine

Prasanta K. Jana,IEEE Senior Member

Department of Computer Science and Engineering
Indian Institute of Technology (ISM), Dhanbad
E-mail: prasantajana@iitism.ac.in

Introduction

- What is a classification problem?
- How can it be thought as a prediction problem?
- Support Vector Machine (SVM) as classification technique,
  - Received considerable attention
  - SVM has its roots in Statistical learning theory,
  - Shown promising results in many practical applications.
  - For examples:
    - Handwritten digit recognition,
    - Text categorization,

Introduction

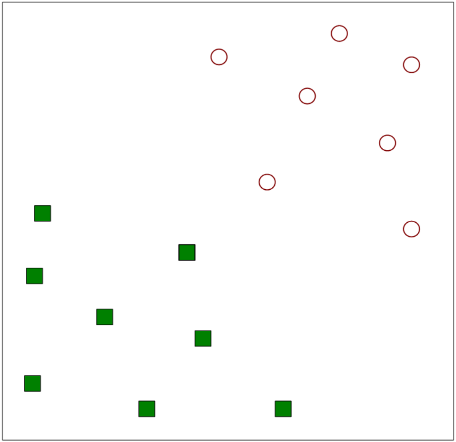
- SVM works very well with
  - High-dimensional data,
  - Avoids the curse of dimensionality problem.
- Another unique aspect of this approach is that
  - it represents the decision boundary using a subset of the training examples,known as the **support vectors**.

Goal of the SVM

- To find the optimal separating hyperplane
  - which **maximizes the margin** of training data.

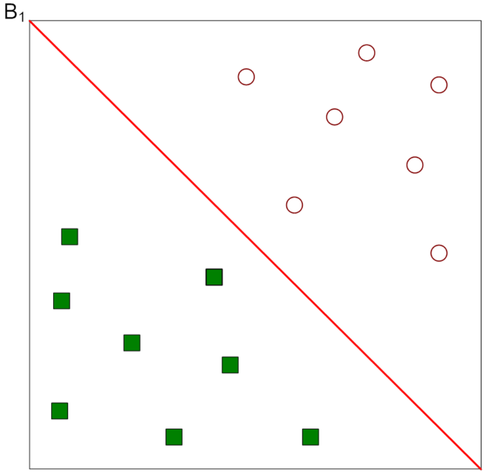
Introduction

- Find a linear hyperplane (decision boundary) that will separate the data,

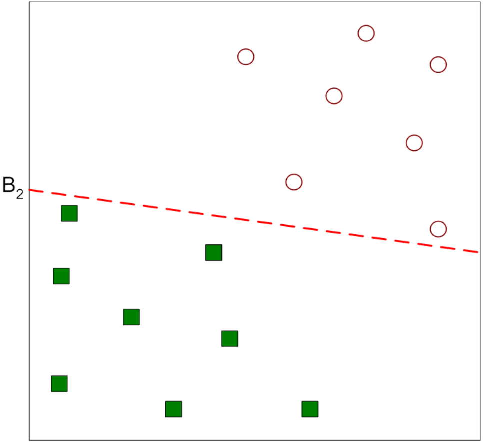


Introduction

- Identify the right hyper-plane (Scenario-1):

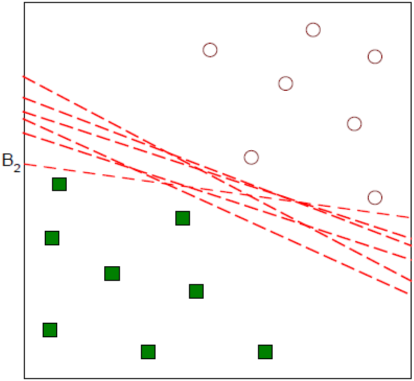


- Identify the right hyper-plane (Scenario-2):



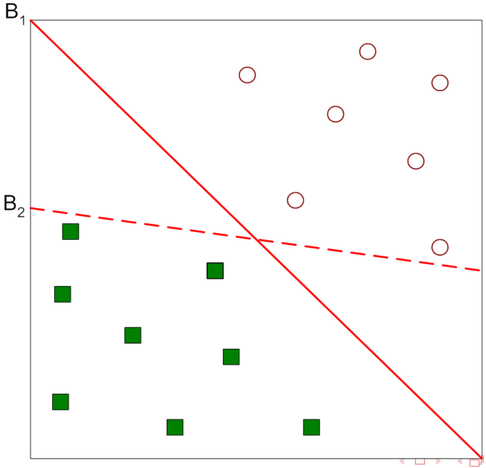
Introduction

- Identify the right hyper-plane (Scenario-3):



Introduction

- Which one is better? B1 or B2?
- How do you define better?

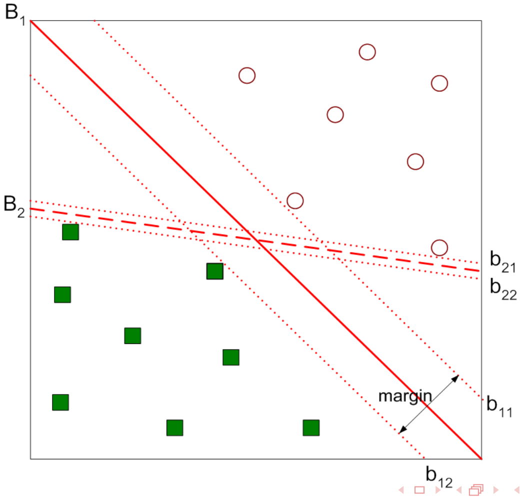


Maximum Margin Hyperplanes

- Decision boundary *B1 and B2*;
  - Associated with a pair of hyperplanes,denoted as *(b11, b12) and (b21, b22)*, respectively.
  - *(b11, b21)* obtained by moving a parallel hyperplane away from the decision boundary until it touches the closest square(s),
  - *(b21, b22)* obtained by moving the hyperplane until it touches the closest circle(s).
- The distance between these two hyperplanes is known as the **margin** of the classifier.
- In this example, *B1* turns out to be the **maximum margin hyperplane** of the training instances.

Maximum Margin Hyperplanes

- Find hyperplane maximizes the margin => B1 is better than B2



Rationale for Maximum Margin

- Decision boundaries with large margins,
  - Tend to have better generalization errors than those with small margins.
- Intuitively, if the margin is small,
  - Then any slight perturbations to the decision boundary can have quite a significant impact on its classification.
- Classifiers that produce decision boundaries with small margins are therefore more susceptible to model **overfitting** and tend to generalize poorly on previously unseen examples.

Linear SVM: Separable Case

- Consider a binary classification problem consisting of N training examples.
- Each example is denoted by (X_i, y_i),
- $X_i = (x_{i1}, x_{i2}, x_{i3}, ..., x_{id})T$ is the attribute set of the ith example.
- Let y_i=$\{$1, -1$\}$ denote its class label

Linear SVM: Separable Case

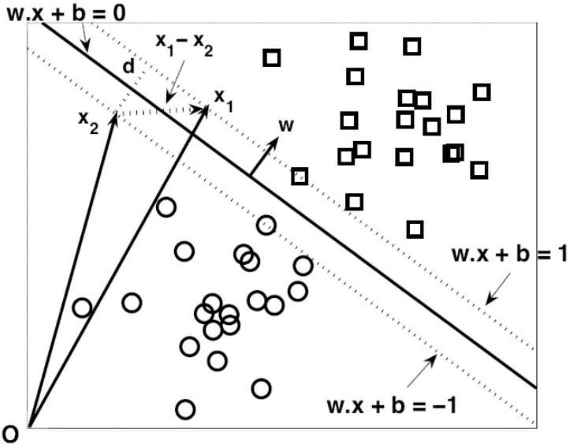
- A linear SVM is a classifier that searches for a hyperplane with the largest margin, **(maximal margin classifier)**

Linear Decision Boundary

- The decision boundary of a linear classifier can be written in the following form:

$$\mathrm{w.x} + \mathrm{b} = 0$$

Linear SVM: Separable Case



Decision boundary and margin of SVM.

Margin of a Linear Classifier

$$\text{Margin} = \frac{2}{\| w \|}$$

Learning a Linear SVM Model

- Note that the decision boundary of a linear classifier is:

$$\mathrm{w.x + b = 0}$$

- Therefore, the training phase of SVM involves estimating the parameters w and b from the training data.
- The parameters must be chosen such that

$$w.x_i + b >= 1 \text{ if } y_i = 1$$
$$w.x_i + b <= -1 \text{ if } y_i = -1$$

  or equivallently

$$y_i(w.x_i + b) >= 1 \text{ for } i = 1, 2, 3, \ldots, N$$

- SVM imposes additional condition that the margin must be maximal.
- However, maximizing the margin is equivalent to minimizing the following objective function *L(w)*:

$$L(w) = \frac{\| w \|^2}{2}$$

Learning a Linear SVM Model

- The learning task in SVM can be formalized as the following constrained optimization problem:

$$\min_{\mathbf{w}} \frac{\|\mathbf{w}\|^2}{2}$$

$$\text{subject to} \qquad y_i(\mathbf{w} \cdot \mathbf{x_i} + b) \geq 1, \quad i = 1, 2, \ldots, N.$$

- This is a convex optimization problem which can be solved by Lagrange multiplier method.

- The new objective function is known as the Lagrangian for the optimization problem:

$$L_P = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N} \lambda_i \Big(y_i(\mathbf{w} \cdot \mathbf{x_i} + b) - 1 \Big)$$

Learning a Linear SVM Model

- To minimize the Lagrangian, we must take the derivative of *Lp* with repect to **w** and **b** and set them zero:

$$\frac{\partial L_p}{\partial \mathbf{w}} = 0 \Longrightarrow \mathbf{w} = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i,$$

$$\frac{\partial L_p}{\partial b} = 0 \Longrightarrow \sum_{i=1}^{N} \lambda_i y_i = 0.$$

- The Karush-Kuhn-Tucker (KKT) condition:

$$\lambda_i \geq 0,$$
$$\lambda_i \left[y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \right] = 0$$

Learning a Linear SVM Model

- Dual formulation of the optimization problem:

$$L_D = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j \mathbf{x_i} \cdot \mathbf{x_j}$$

- The decision boundary can be expressed as follows:

$$\left(\sum_{i=1}^{N} \lambda_i y_i \mathbf{x_i} \cdot \mathbf{x} \right) + b = 0.$$

Learning a Linear SVM Model: Example

- Consider the two-dimensional data set which contains eight training instances.
- Let **w** = *(w1,w2)* and *b* denote the parameters of the decision boundary. We can solve for *w1* and *w2* in the following way:

$$
\begin{aligned}
w_1 &= \sum_i \lambda_i y_i x_{i1} = 65.5621 \times 1 \times 0.3858 + 65.5621 \times -1 \times 0.4871 = -6.64. \\
w_2 &= \sum_i \lambda_i y_i x_{i2} = 65.5621 \times 1 \times 0.4687 + 65.5621 \times -1 \times 0.611 = -9.32.
\end{aligned}
$$

- The bias term *b* can be computed for each support vector:

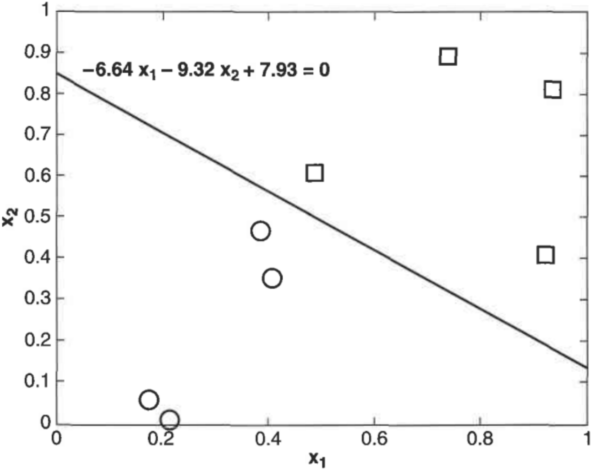
$$
\begin{aligned}
b^{(1)} &= 1 - \mathbf{w} \cdot \mathbf{x}_1 = 1 - (-6.64)(0.3858) - (-9.32)(0.4687) = 7.9300. \\
b^{(2)} &= -1 - \mathbf{w} \cdot \mathbf{x}_2 = -1 - (-6.64)(0.4871) - (-9.32)(0.611) = 7.9289.
\end{aligned}
$$

- Averaging these values, we obtain **b** = 7.93.

Learning a Linear SVM Model: Example

x_1	x_2	y	Lagrange Multiplier
0.3858	0.4687	1	65.5261
0.4871	0.611	−1	65.5261
0.9218	0.4103	−1	0
0.7382	0.8936	−1	0
0.1763	0.0579	1	0
0.4057	0.3529	1	0
0.9355	0.8132	−1	0
0.2146	0.0099	1	0

Learning a Linear SVM Model: Example



Example of a linearly separable data set.

Learning a Linear SVM Model: Example

- Once the parameters of the decision boundary are found, a test instance **z** is classified as follows:

$$f(\mathbf{z}) = sign(\mathbf{w} \cdot \mathbf{z} + b) = sign\left(\sum_{i=1}^{N} \lambda_i y_i \mathbf{x_i} \cdot \mathbf{z} + b\right)$$

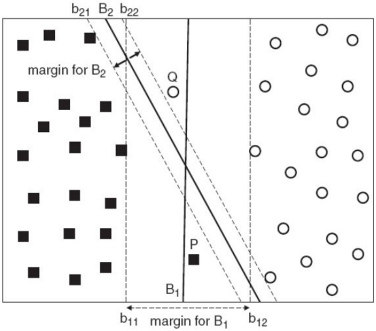
- It *f(z)* $= 1$, then the test instance is classified as a positive class; otherwise, it is classified as a negative class.

Linear SVM: Nonseparable Case

- When it is not possible to separate the training data linearly.
- SVM to construct a linear decision boundary even in situations,
    - Where the classes are not linearly separable.

Linear SVM: Nonseparable Case

- Similar example except two new ponits P and Q
- B1 misclssifies the examples while B2 classifies them correctly
- It does not mean that B2 is better dicision boundary than B1
- Because new examples may correspond to noise
- B1 may be preferred over B2 as it has wider margin and so is less susceptible to overfitting



Decision boundary of SVM for the nonseparable case.

Linear SVM: Nonseparable Case

- The previous SVM method (seperable case) constructs decision boundary which is mistake-free
- so we should examine how the formulation can be modified to learn a decision boundary that is tolerable to small training errors
- This introduces a SVM method known as soft margin approach
- To do this the learning method must consider the trade-off between the width of the margin and the no of traning errors

Linear SVM: Nonseparable Case

- The method introduces positive valued slack variables into the constraints of the linaer SVM seperable optimization problem, i.e.,

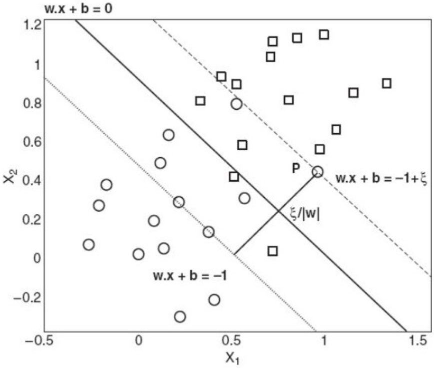
$$\mathbf{w} \cdot \mathbf{x_i} + b \geq 1 - \xi_i \quad \text{if } y_i = 1,$$
$$\mathbf{w} \cdot \mathbf{x_i} + b \leq -1 + \xi_i \quad \text{if } y_i = -1,$$

where $\forall i : \ \xi_i > 0.$

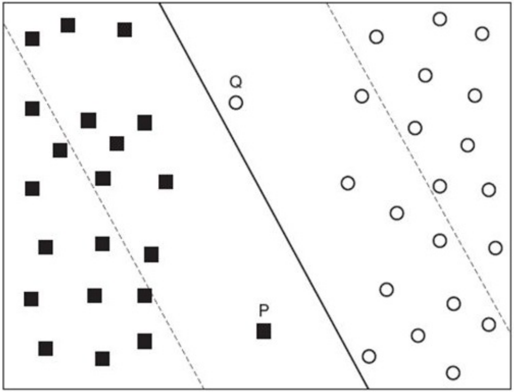
Linear SVM: Nonseparable Case

To understand let us consider the follwing figure

- P is one of the instances that violets the the constraints of the linear SVM seperable optimization problem
- $w.x + b = -1$ is hyperplane
- So, jai provides an estimate of the error of the decision boudary on the training example P



Linear SVM: Nonseparable Case



A decision boundary that has a wide margin but large training error.

Linear SVM: Nonseparable Case

- The modified objective function is given by the following equation:

$$f(\mathbf{w}) = \frac{\|\mathbf{w}\|^2}{2} + C(\sum_{i=1}^{N} \xi_i)^k,$$

- The Lagrangian for this constrained optimization problem can be written as follows:

$$L_P = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N} \xi_i - \sum_{i=1}^{N} \lambda_i\{y_i(\mathbf{w} \cdot \mathbf{x_i} + b) - 1 + \xi_i\} - \sum_{i=1}^{N} \mu_i \xi_i,$$

Linear SVM: Nonseparable Case

- The inequality constraints can be transformed into equality constraints using the following KKT conditions::

$$
\begin{aligned}
&\xi_i \geq 0, \quad \lambda_i \geq 0, \quad \mu_i \geq 0, \\
&\lambda_i\{y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 + \xi_i\} = 0, \\
&\mu_i \xi_i = 0.
\end{aligned}
$$

- Setting the first-order derivative of **L** with respect to w, b, and (ξ) to zero would result in the following equations:

$$
\begin{aligned}
\frac{\partial L}{\partial w_j} &= w_j - \sum_{i=1}^{N} \lambda_i y_i x_{ij} = 0 \implies w_j = \sum_{i=1}^{N} \lambda_i y_i x_{ij}. \\
\frac{\partial L}{\partial b} &= -\sum_{i=1}^{N} \lambda_i y_i = 0 \implies \sum_{i=1}^{N} \lambda_i y_i = 0. \\
\frac{\partial L}{\partial \xi_i} &= C - \lambda_i - \mu_i = 0 \implies \lambda_i + \mu_i = C.
\end{aligned}
$$

Linear SVM: Nonseparable Case

- The dual Lagrangian:

$$\begin{aligned}
L_D &= \frac{1}{2}\sum_{i,j}\lambda_i\lambda_j y_i y_j \mathbf{x}_i\cdot\mathbf{x}_j + C\sum_i \xi_i \\
&\quad -\sum_i \lambda_i\{y_i(\sum_j \lambda_j y_j \mathbf{x}_i\cdot\mathbf{x}_j + b) - 1 + \xi_i\} \\
&\quad -\sum_i (C-\lambda_i)\xi_i \\
&= \sum_{i=1}^{N}\lambda_i - \frac{1}{2}\sum_{i,j}\lambda_i\lambda_j y_i y_j \mathbf{x_i}\cdot\mathbf{x_j},
\end{aligned}$$

Nonlinear SVM

- Applying SVM to data sets that have nonlinear decision boundaries.
- The trick here is to transform the data from its original coordinate space in x into a new space O(x),
    - A linear decision boundary can be used to separate the instances in the transformed space.
- After doing the transformation,
    - We can apply a linear decision boundary in the transformed space.
- The **kernel trick** is a method for computing similarity in the transformed space using the original attribute set.

Learning a Nonlinear SVM Model

- The learning task for a nonlinear SVM can be formalized as the following optimization problem:

$$\min_{\mathbf{w}} \frac{\|\mathbf{w}\|^2}{2}$$

$$\text{subject to} \quad y_i(\mathbf{w} \cdot \Phi(\mathbf{x}_i) + b) \geq 1, \quad i = 1, 2, \ldots, N.$$

- Dual Lagrangian for the constrained optimization problem:

$$L_D = \sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$$

Learning a Nonlinear SVM Model

- Once the λ_i's are found using quadratic programming techniques, the parameters w and b can be derived using the following equations:

$$\mathbf{w} = \sum_i \lambda_i y_i \Phi(\mathbf{x}_i)$$

$$\lambda_i \{ y_i (\sum_j \lambda_j y_j \Phi(\mathbf{x}_j) \cdot \Phi(\mathbf{x}_i) + b) - 1 \} = 0,$$

- A test instance $\mathbf{z}$ can be classified using the following equation:

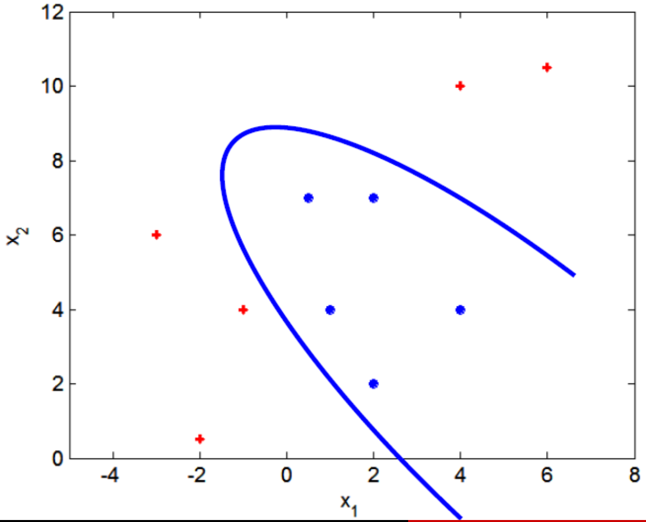
$$f(\mathbf{z}) = sign\big(\mathbf{w} \cdot \Phi(\mathbf{z}) + b\big) = sign\left(\sum_{i=1}^{n} \lambda_i y_i \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{z}) + b \right)$$

Nonlinear SVM

- Applying SVM to data sets that have nonlinear decision boundaries.
- The trick here is to transform the data from its original coordinate space in x into a new space O(x),
  - A linear decision boundary can be used to separate the instances in the transformed space.
- After doing the transformation,
  - We can apply a linear decision boundary in the transformed space.
- The **kernel trick** is a method for computing similarity in the transformed space using the original attribute set.

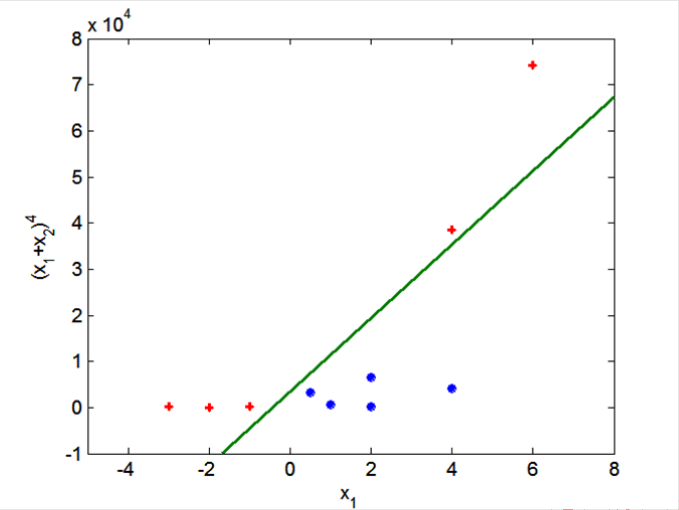
Nonlinear Support Vector Machines

- What if decision boundary is not linear?



Nonlinear Support Vector Machines

- Transform data into higher dimensional space



Characteristics of SVM

SVM has many desirable qualities that make it one of the most widely used classification algorithms. Following is a summary of the general characteristics of SVM:

- The SVM learning problem can be formulated as a **convex optimization problem**,
  - In which efficient algorithms are available to find the **global minimum** of the objective function.
  - Other classification methods, such as **rule-based classifiers and artificial neural networks**,
    - Employ a **greedy based strategy** to search the hypothesis space.
  - Such methods tend to find only **locally optimum solutions**.

Characteristics of SVM

- SVM performs **capacity control**,
  - By maximizing the margin of the decision boundary.
  - Nevertheless,the user must still provide other parameters such as the type of kernel function to use and the cost function C for introducing each slack variable.
- SVM can be applied to **categorical data**,
  - By introducing **dummy variables** for each categorical attribute value present in the data.
  - For example, if Marital Status has three values (Single, Married, Divorced), we can introduce a binary variable for each of the attribute values.

SVM : Sample Code

Sample Python Code to implement SVM for the following Data and Class Label.

x1 = [1,5,1.5,8,1,9] x2 = [2,8,1.8,8,0.6,11]

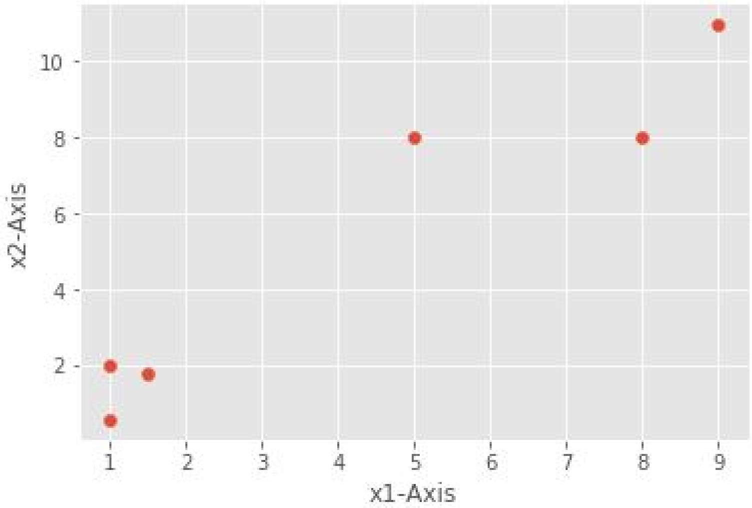
y = [0,1,0,1,0,1]

Comment is Shown by :

```
:Importing svm Python Libraries
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm

:Plotting of training data
x1 = [1,5,1.5,8,1,9]
x2 = [2,8,1.8,8,0.6,11]
plt.xlabel("x1-Axis")
plt.ylabel("x2-Axis")
plt.scatter(x1,x2)
plt.show()
```

SVM : Sample Code



SVM : Sample Code

```
:Data Initialization
:X = Data, Y= Label
X = np.array ([[1,2],[5,8],[1.5,1.8],[8,8],[1,0.6],[9,11]])
Y = [0,1,0,1,0,1]

:importing LinearSVC
clf=svm.LinearSVC()

:Fitting the model
clf.fit(X,Y)

:Obtaining Coefficient Vector
w=clf.coef_[0]

:Obtaining Learning Rate
a = -w[0]/w[1]
```

SVM : Sample Code

```
:X-scale for Hyperplane
XX=np.linspace(0,10)

YY = a * XX - clf.intercept_[0]/w[1]

:Plotting training data
plt.scatter(X[:,0],X[:,1],c=Y)
plt.show()

:Plotting Hyperplane
plt.xlabel("x1-Axis")
plt.ylabel("x2-Axis")
plt.plot(XX,YY)
plt.show()

:Test Data
X_pred=np.array([[3,4],[6,9],[0.43,0.89]])
```

SVM : Sample Code

```
: Declaring an Array to store predicted class label
: If Predicted class Belongs to 0, Denote by Color = Red &
    Marker = +
: If Predicted class Belongs to 1, Denote by Color = blue &
    Marker = ^

for i in range(len(X_pred)):
   temp=clf.predict(X_pred)
   if temp[i]==0:
        plt.scatter(X_pred[i,0],X_pred[i,1], color='r',
             marker='P')
   else :
      plt.scatter(X_pred[i,0],X_pred[i,1], color='b', marker='^')
 plt.show()
```

SVM : Sample Code