



Machine Learning for Data Analysis

MSc in Data Analytics

CCT College Dublin

Gaussian Naïve Bayes and Support Vector Machine

Week 5

Lecturer: Dr. Muhammad Iqbal*

Email: miqbal@cct.ie

- Introduction
- Find Fraudulent Orders using ML
- Conditional Probabilities
- Concept of Bayes Theorem
- Problem: Use of Bayes Theorem
- What is Naive Bayes Classifier?
- How Naive Bayes classifier works?
- Concept of Support Vector Machine
- Support Vector Machine
- Linear and Non-Linear SVM
- Comparison of Accuracy for Models

Introduction



- Hotmail was one of the world's first webmail services. Hotmail was launched in 1996 by Sabeer Bhatia and Jack Smith in California. In 1997 the company was acquired by Microsoft for approximately \$400m and relaunched as MSN Hotmail.
- Recall your inbox being full of spam messages ranging from various parts of the world, for example, you received an email from a dummy company that you have won the lottery of 1 million dollars. It became such a major issue that we spent most of our time filtering spam.
- Thanks to Gmail and tools like **SpamAssassin** that can detect the spam emails and we see our inbox all relevant emails.
- Using a method called a **Naive Bayesian Classifier (NBC)**, such tools have been able to mitigate the influx of spam to our inboxes. **NBC** is based on
 - **Bayes' theorem**
- A **Naive Bayes Classifier** is a supervised and probabilistic learning method. It does well with data in which the inputs are independent from one another. It also prefers problems where the probability of any attribute is greater than zero.



- In several machine learning applications, the relationship between the **attribute set** (independent variables) and the **class variable** (dependent) is non-deterministic. This means that the class label of a test record cannot be predicted with certainty even though its attribute set is identical to some of the training data sets.
- This kind of problem has emerged due to the presence of noisy data or the presence of certain confounding factors that affect classification.
- **For example**, consider the task of predicting whether a person is at risk for heart disease based on the person's diet and workout frequency.
- Although most people who eat healthily and exercise regularly have less chance of developing heart disease, they may still do so because of other factors such as heredity, excessive smoking, and alcohol abuse.
- Determining whether a person's diet is healthy or the workout frequency is sufficient, it is also subject to interpretation, which may introduce **uncertainties into the learning problem**. We introduce **Bayes theorem** to determine the modelling probabilistic relationships between the attribute set and the class variable.

Concept of Bayes Theorem

- **Naïve Bayes** is a family of model classes with a model tailored to different statistical distributions. Bayes' theorem is the premier method for understanding the probability of some event, $P(A | B)$, given some new information, $P(B | A)$, and a prior belief in the probability of the event, $P(A)$:

$$P(A | B) = \frac{P(B | A) P(A)}{P(B)}$$

- The **Bayesian method's** popularity became popular in the last decade, more and more rivaling traditional frequentist applications in academia, government, and business.
- In machine learning, one application of Bayes' theorem to classification comes in the form of the Naive Bayes classifier. Naive Bayes classifiers combine a number of desirable qualities in practical machine learning into a single classifier. These include

1. An intuitive approach
2. The ability to work with small data
3. Low computation costs for training and prediction
4. Often solid results in a variety of settings

Conditional probability

$$P(A | B) = \frac{P(A \cap B)}{P(B)}$$

	Yes	No	Total
Males	15	5	20
Females	17	3	20
Total	32	8	40

$$P(\text{yes} | \text{male}) = \frac{P(\text{yes and male})}{P(\text{male})} = \frac{15}{20}$$

Using Bayes Theorem for Classification

To illustrate this approach, consider the task of predicting whether a loan borrower will default on their payments. Figure shows a training set with the following attributes: **Home Owner, Marital Status, and Annual Income**. Loan borrowers who defaulted on their payments are classified as Yes, while those who repaid their loans are classified as No.

- Suppose we are given a test record with the following attribute set
 $X = (\text{Home Owner} = \text{No}, \text{Marital Status} = \text{Married}, \text{Annual Income} = \$120\text{K})$
- To classify the record, we need to compute the posterior probabilities $P(\text{Yes} | X)$ and $P(\text{No} | X)$ based on information available in the training data.
- If $P(\text{Yes} | X) > P(\text{No} | X)$, then the record is classified as Yes; otherwise, it is classified as No.

	binary	categorical	continuous	class
Tid	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Training set for predicting the loan default problem.

Gaussian Distribution Function

	binary		categorical	continuous	class
Tid	Home Owner	Marital Status	Annual Income	Defaulted Borrower	
1	Yes	Single	125K	No	
2	No	Married	100K	No	
3	No	Single	70K	No	
4	Yes	Married	120K	No	
5	No	Divorced	95K	Yes	
6	No	Married	60K	No	
7	Yes	Divorced	220K	No	
8	No	Single	85K	Yes	
9	No	Married	75K	No	
10	No	Single	90K	Yes	

Training set for predicting the loan default problem.

- We can assume a certain form of probability distribution for the continuous variable and estimate the parameters of the distribution using the training data.
- A **Gaussian distribution** is chosen to represent the class-conditional probability for continuous attributes.
- The distribution is characterized by two parameters, its mean (μ) and variance (σ^2). For each class (y_j), the class-conditional probability for attribute (X_i) is

$$P(X_i = x_i | Y = y_j) = \frac{1}{\sqrt{2\pi}\sigma_{ij}} \exp^{-\frac{(x_i - \mu_{ij})^2}{2\sigma_{ij}^2}}$$

- The parameter μ_{ij} can be estimated based on the sample mean of $X_i(\mathbf{x})$ for all training records that belong to the class y_j . Similarly, σ_{ij}^2 can be estimated from the sample variance (s^2) of such training records.

$$\bar{x} = \frac{125 + 100 + 70 + \dots + 75}{7} = 110$$

$$s^2 = \frac{(125 - 110)^2 + (100 - 110)^2 + \dots + (75 - 110)^2}{7(6)} = 2975$$

$$s = \sqrt{2975} = 54.54.$$

$$P(\text{Income}=120|\text{No}) = \frac{1}{\sqrt{2\pi}(54.54)} \exp^{-\frac{(120-110)^2}{2 \times 2975}} = 0.0072.$$

- For example, consider the annual income attribute as shown in Figure. The sample mean and variance for this attribute with respect to the class **No** are \rightarrow

Naïve Bayes Classifier

Example

Conditional probability

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

- Consider the data set as shown in Figure (a). We can compute the class conditional probability for each categorical attribute, along with the **sample mean and variance** for the continuous attribute. These probabilities are summarized as shown in Figure (b).

- To predict the class label of a test record

X = (Home Owner = No, Marital Status = Married, Income = \$120K),

- We need to compute the posterior probabilities $P(\text{No}|X)$ and $P(\text{Yes}|X)$.

- The **prior probabilities** of each class can be estimated by calculating the fraction of training records that belong to each class.

- Since there are three records that belong to the class **Yes** and seven records that belong to the class **No**, **$P(\text{Yes}) = 0.3$** and **$P(\text{No}) = 0.7$** .

- Using the information provided in Figure (b), the class-conditional probabilities can be computed as shown.

Tid	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

(a)

$P(\text{Home Owner}=\text{Yes}|\text{No}) = 3/7$
 $P(\text{Home Owner}=\text{No}|\text{No}) = 4/7$
 $P(\text{Home Owner}=\text{Yes}|\text{Yes}) = 0$
 $P(\text{Home Owner}=\text{No}|\text{Yes}) = 1$
 $P(\text{Marital Status}=\text{Single}|\text{No}) = 2/7$
 $P(\text{Marital Status}=\text{Divorced}|\text{No}) = 1/7$
 $P(\text{Marital Status}=\text{Married}|\text{No}) = 4/7$
 $P(\text{Marital Status}=\text{Single}|\text{Yes}) = 2/3$
 $P(\text{Marital Status}=\text{Divorced}|\text{Yes}) = 1/3$
 $P(\text{Marital Status}=\text{Married}|\text{Yes}) = 0$

For Annual Income:
If class=No: sample mean=110
sample variance=2975
If class=Yes: sample mean=90
sample variance=25

(b)

The naïve Bayes classifier for the loan classification problem.

$$\begin{aligned} P(X|\text{No}) &= P(\text{Home Owner} = \text{No}|\text{No}) \times P(\text{Status} = \text{Married}|\text{No}) \\ &\quad \times P(\text{Annual Income} = \$120\text{K}|\text{No}) \\ &= 4/7 \times 4/7 \times 0.0072 = 0.0024. \end{aligned}$$

$$\begin{aligned} P(X|\text{Yes}) &= P(\text{Home Owner} = \text{No}|\text{Yes}) \times P(\text{Status} = \text{Married}|\text{Yes}) \\ &\quad \times P(\text{Annual Income} = \$120\text{K}|\text{Yes}) \\ &= 1 \times 0 \times 1.2 \times 10^{-9} = 0. \end{aligned}$$

Naïve Bayes Classifier

Example

Conditional probability

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

Tid	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

(a)

The naïve Bayes classifier for the loan classification problem.

$P(\text{Home Owner}=\text{Yes}|\text{No}) = 3/7$
 $P(\text{Home Owner}=\text{No}|\text{No}) = 4/7$
 $P(\text{Home Owner}=\text{Yes}|\text{Yes}) = 0$
 $P(\text{Home Owner}=\text{No}|\text{Yes}) = 1$
 $P(\text{Marital Status}=\text{Single}|\text{No}) = 2/7$
 $P(\text{Marital Status}=\text{Divorced}|\text{No}) = 1/7$
 $P(\text{Marital Status}=\text{Married}|\text{No}) = 4/7$
 $P(\text{Marital Status}=\text{Single}|\text{Yes}) = 2/3$
 $P(\text{Marital Status}=\text{Divorced}|\text{Yes}) = 1/3$
 $P(\text{Marital Status}=\text{Married}|\text{Yes}) = 0$

For Annual Income:
 If class=No: sample mean=110
 sample variance=2975
 If class=Yes: sample mean=90
 sample variance=25

(b)

$$\begin{aligned}
 P(\mathbf{X}|\text{No}) &= P(\text{Home Owner} = \text{No}|\text{No}) \times P(\text{Status} = \text{Married}|\text{No}) \\
 &\quad \times P(\text{Annual Income} = \$120\text{K}|\text{No}) \\
 &= 4/7 \times 4/7 \times 0.0072 = 0.0024.
 \end{aligned}$$

$$\begin{aligned}
 P(\mathbf{X}|\text{Yes}) &= P(\text{Home Owner} = \text{No}|\text{Yes}) \times P(\text{Status} = \text{Married}|\text{Yes}) \\
 &\quad \times P(\text{Annual Income} = \$120\text{K}|\text{Yes}) \\
 &= 1 \times 0 \times 1.2 \times 10^{-9} = 0.
 \end{aligned}$$

$$P(\text{No}|\mathbf{X}) = P(\mathbf{X}|\text{No}) * P(\text{No}) / P(\mathbf{X})$$

Where $P(\mathbf{X})$ is a constant term and is shown by alpha (α) and $\alpha = 1/P(\mathbf{X})$ is a constant term

- Putting them together, the posterior probability for class **No** is
- $P(\text{No}|\mathbf{X}) = \alpha \times 7/10 \times 0.0024 = 0.0016\alpha$ and
- Using a similar approach, we can show that the posterior probability for class **Yes** is zero.
- $P(\text{Yes}|\mathbf{X}) = \alpha \times 3/10 \times 0 = 0$, because its class-conditional probability is zero. So

$P(\text{No}|\mathbf{X}) > P(\text{Yes}|\mathbf{X})$, the record is classified as No.

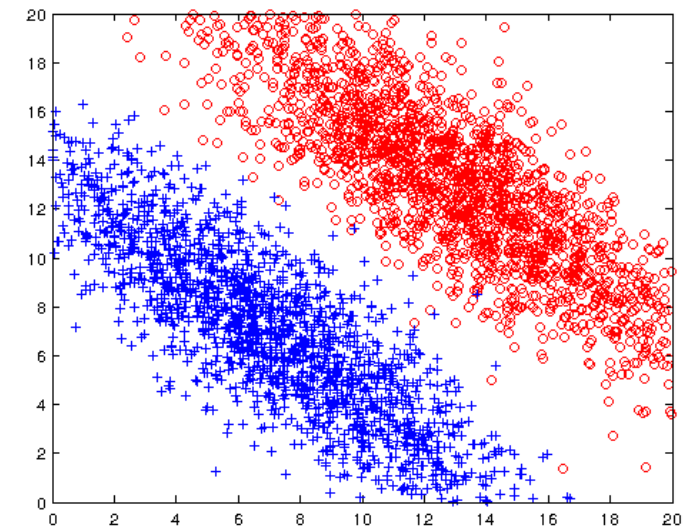
Bayes Theorem for Classification

- **Approach:**

- Compute posterior probability $P(\mathbf{Y} \mid X_1, X_2, \dots, X_d)$ using the Bayes theorem

$$P(Y \mid X_1 X_2 \dots X_n) = \frac{P(X_1 X_2 \dots X_d \mid Y) P(Y)}{P(X_1 X_2 \dots X_d)}$$

- **Maximum a-posteriori:** Choose \mathbf{Y} that maximizes $P(\mathbf{Y} \mid X_1, X_2, \dots, X_d)$
- Equivalent to choosing value of \mathbf{Y} that maximizes $P(X_1, X_2, \dots, X_d \mid \mathbf{Y}) P(\mathbf{Y})$
- How to estimate $P(X_1, X_2, \dots, X_d \mid \mathbf{Y})$?



Naïve Bayes can construct oblique decision boundaries

Naïve Bayes Classifier

Example

Name	Give Birth	Can Fly	Live in Water	Have Legs	Class
human	yes	no	no	yes	mammals
python	no	no	no	no	non-mammals
salmon	no	no	yes	no	non-mammals
whale	yes	no	yes	no	mammals
frog	no	no	sometimes	yes	non-mammals
komodo	no	no	no	yes	non-mammals
bat	yes	yes	no	yes	mammals
pigeon	no	yes	no	yes	non-mammals
cat	yes	no	no	yes	mammals
leopard shark	yes	no	yes	no	non-mammals
turtle	no	no	sometimes	yes	non-mammals
penguin	no	no	sometimes	yes	non-mammals
porcupine	yes	no	no	yes	mammals
eel	no	no	yes	no	non-mammals
salamander	no	no	sometimes	yes	non-mammals
gila monster	no	no	no	yes	non-mammals
platypus	no	no	no	yes	mammals
owl	no	yes	no	yes	non-mammals
dolphin	yes	no	yes	no	mammals
eagle	no	yes	no	yes	non-mammals

Give Birth	Can Fly	Live in Water	Have Legs	Class
yes	no	yes	no	?

A: attributes

M: mammals

N: non-mammals

$$P(A | M) = \frac{6}{7} \times \frac{6}{7} \times \frac{2}{7} \times \frac{2}{7} = 0.06$$

$$P(A | N) = \frac{1}{13} \times \frac{10}{13} \times \frac{3}{13} \times \frac{4}{13} = 0.0042$$

$$P(A | M)P(M) = 0.06 \times \frac{7}{20} = 0.021$$

$$P(A | N)P(N) = 0.004 \times \frac{13}{20} = 0.0027$$

$$P(A|M)P(M) > P(A|N)P(N)$$

=> Mammals

Train and Test a Classifier

$$P(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

- **Problem:** You have only **continuous features** and you want to train a Naive Bayes classifier.
- **Solution:** Use a **Gaussian Naive Bayes classifier** in scikit-learn.
- The most common type of **Naive Bayes classifier** is the **Gaussian Naive Bayes**. In Gaussian Naive Bayes, we assume that the likelihood of the feature values, \mathbf{x} , given an observation is of class y , follows a normal distribution

$$p(x_j | y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} e^{-\frac{(x_j - \mu_y)^2}{2\sigma_y^2}}$$

- where σ_y^2 and μ_y are the **variance** and **mean values** of feature x_j for class y . Because of the assumption of the normal distribution, Gaussian Naive Bayes is best used in the cases when all our features are continuous.

```
# Importing the dataset
dataset = pd.read_csv('C:/Users/munaw/Downloads/iris.csv')
```

```
# Looking at the dataset
dataset.head()
```

```
# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 82)
X_train.shape, X_test.shape
```

```
((120, 4), (30, 4))
```

```
# Feature Scaling to bring the variable in a single scale
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
# Fitting Naive Bayes Classification to the Training set with linear kernel
from sklearn.naive_bayes import GaussianNB
nvclassifier = GaussianNB()
nvclassifier.fit(X_train, y_train)
```

```
GaussianNB()
```

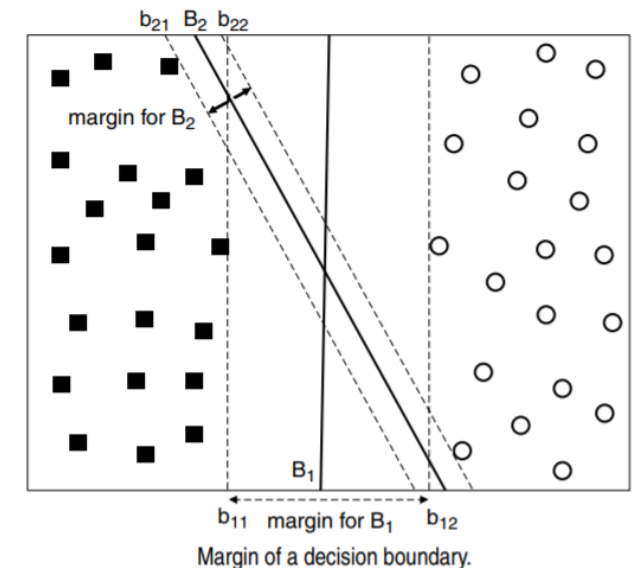
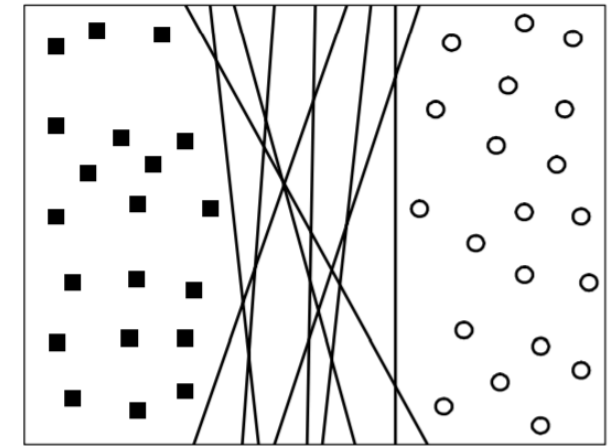
```
# Predicting the Test set results
y_pred = nvclassifier.predict(X_test)
print(y_pred)
```

```
['virginica' 'virginica' 'setosa' 'setosa' 'setosa' 'virginica'
'versicolor' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
'virginica' 'setosa' 'setosa' 'setosa' 'setosa' 'virginica' 'versicolor'
'setosa' 'versicolor' 'setosa' 'virginica' 'setosa' 'virginica'
'virginica' 'versicolor' 'virginica' 'setosa' 'virginica' 'versicolor']
```

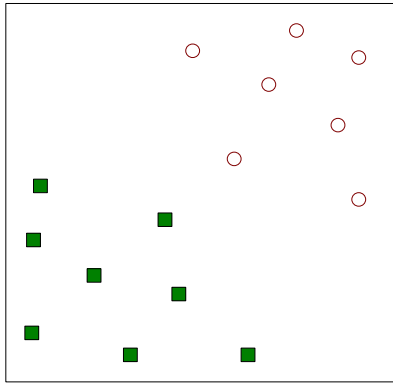
To avoid an impact of zero probability, we assume a smallest number using a Gaussian distribution.

Support Vector Machine (SVM)

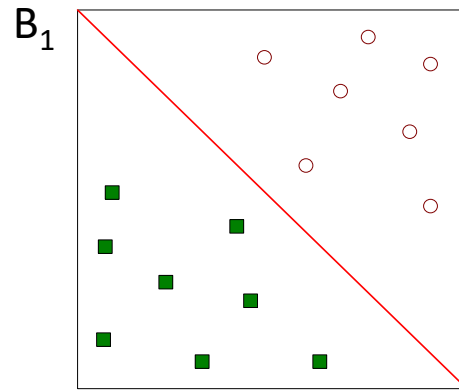
- A classification technique that has received considerable attention is **Support Vector Machine (SVM)**.
- This technique has its roots in statistical learning theory and has shown promising empirical results in many practical applications, from handwritten digit recognition to text categorization.
- The unique aspect of this approach is that it represents the decision boundary using a subset of the training examples, known as the **support vectors**.
- To illustrate the basic idea behind **SVM**, we first introduce the concept of a maximal margin hyperplane (B_1 and B_2) and explain the rationale of choosing such a **hyperplane**.
- We need to understand how a linear **SVM** can be trained to explicitly look for this type of hyperplane in linearly separable data.



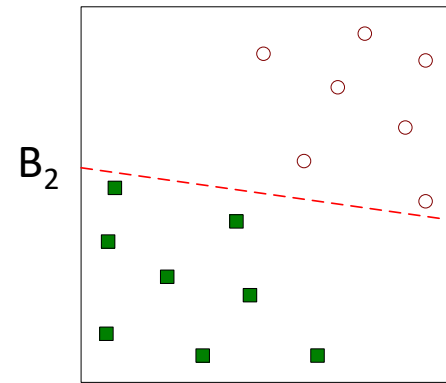
Support Vector Machines



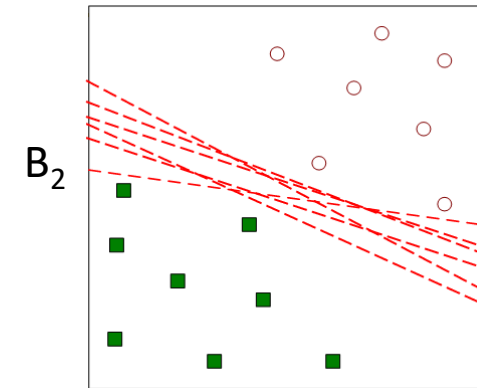
- Find a linear hyperplane (decision boundary) that will separate the data



- One Possible Solution

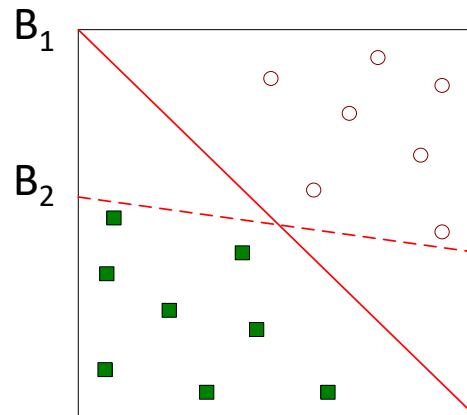


- Another possible solution

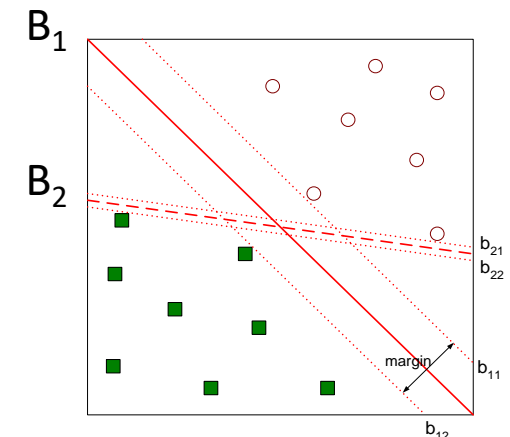


- Other possible solutions

- Which one is better? B_1 or B_2 ?
- How do you define better?



- Find hyperplane that **maximizes** the margin $\Rightarrow B_1$ is better than B_2



Separating Hyperplanes

The points closest to the **hyperplane** are called as the **support vector points** and the distance of the **vectors** from the **hyperplane** are called the **margins**.

- The hyperplane is a function which is used to differentiate between features. In 2-D, the function used to classify between features is a line. The function used to classify the features in a 3-D is called as a plane similarly the function which classifies the point in higher dimension is called as a hyperplane. Now we know about the hyperplane lets move back to SVM. Let's say there are “**m**” dimensions:
- The equation of the hyperplane in the ‘**m**’ dimensions can be given as

$$\begin{aligned} y &= w_0 + w_1x_1 + w_2x_2 + w_3x_3 \dots \\ &= w_0 + \sum_{i=1}^m w_ix_i \\ &= w_0 + w^T X \\ &= b + w^T X \end{aligned}$$

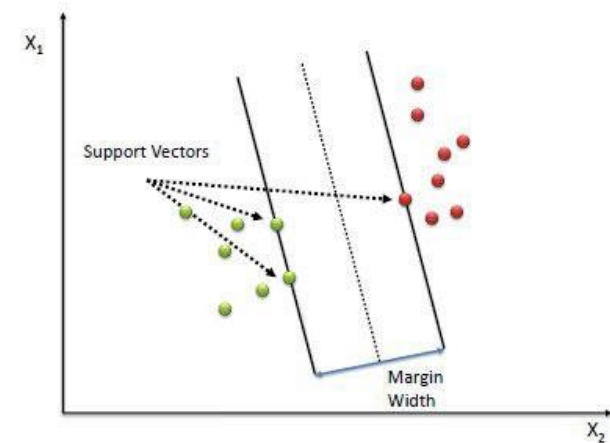
where,

w_i = weight vectors ($w_0, w_1, w_2, w_3, \dots, w_m$)

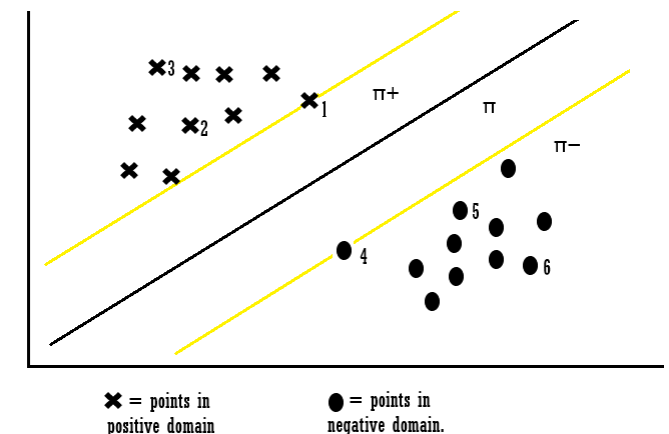
b = biased term (w_0)

x = variables

$$\begin{aligned} \pi &= b + w^T X = 0 \\ \pi^+ &= b + w^T X = 1 \\ \pi^- &= b + w^T X = -1 \end{aligned}$$



3 hyperplanes namely (π, π^+, π^-)



Separating Hyperplanes

Decision Surfaces

- Linear model:

$$f(\vec{x}) = \begin{cases} +1 & \text{if } \vec{w} \bullet \vec{x} + b \geq +1 \\ -1 & \text{if } \vec{w} \bullet \vec{x} + b \leq -1 \end{cases}$$

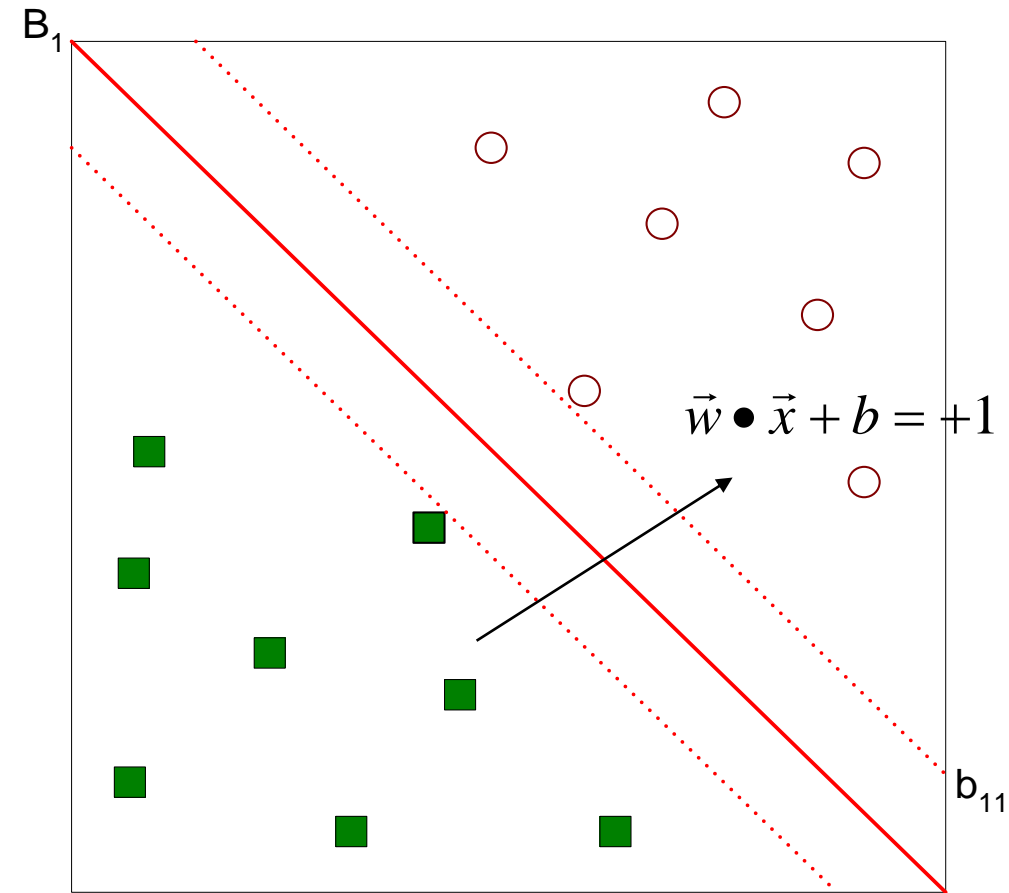
$$\vec{w} \bullet \vec{x} + b = 0$$

- Learning the model is equivalent to determining the values of \vec{w} and b

$$\vec{w} \bullet \vec{x} + b = -1$$

- How to find \vec{w} and b from training data?

$$f(\vec{x}) = \begin{cases} 1 & \text{if } \vec{w} \bullet \vec{x} + b \geq 1 \\ -1 & \text{if } \vec{w} \bullet \vec{x} + b \leq -1 \end{cases}$$



$$\text{Margin} = \frac{b_{12} - b_{11}}{\|\vec{w}\|}$$

Separating Hyperplanes

Decision Surfaces

$$y_i = f(\vec{x}_i) = \begin{cases} 1 & \text{if } \vec{w} \cdot \vec{x}_i + b \geq 1 \\ -1 & \text{if } \vec{w} \cdot \vec{x}_i + b \leq -1 \end{cases}$$

For the point **x1**:

$$\begin{aligned} y_1 &= 1 \\ y_1(w^T x_1 + b) &= 1 \end{aligned}$$

Explanation: For the point **x1**, we can say that point lies **on the hyperplane** and the equation determines that the product of our actual output and the hyperplane equation **is 1** which means the point is correctly classified in the positive domain.

For the point **x3**:

$$\begin{aligned} y_1 &= 1 \\ y_1(w^T x_1 + b) &> 1 \end{aligned}$$

Explanation: For the point **x3**, we can say that point lies **away from the hyperplane** and the equation determines that the product of our actual output and the hyperplane equation **is greater than 1** which means the point is correctly classified in the positive domain.

For the point **x4**:

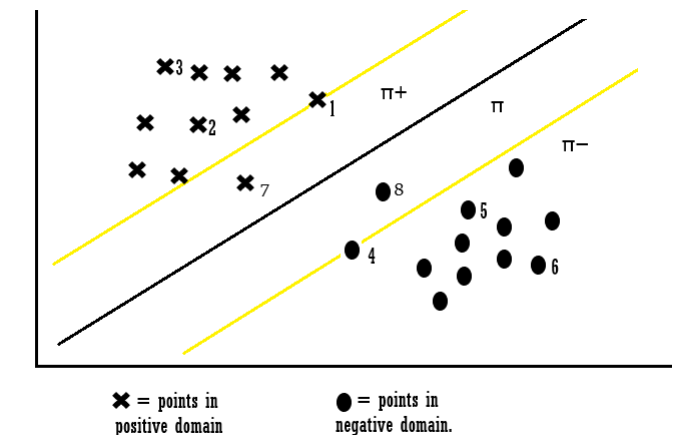
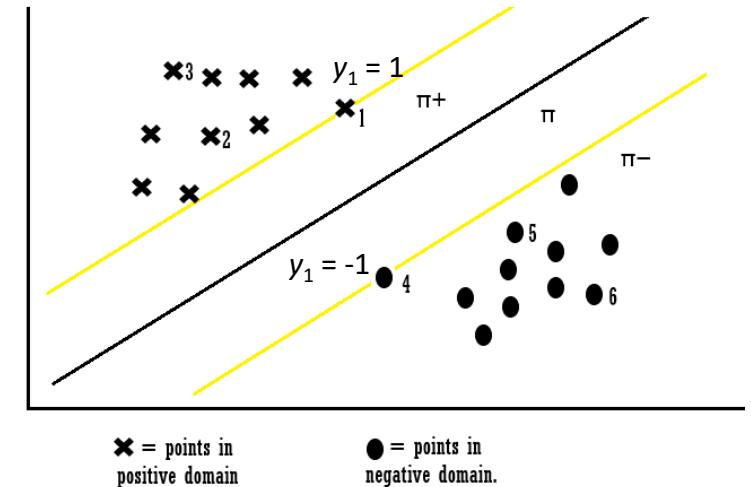
$$\begin{aligned} y_1 &= -1 \\ y_1(w^T x_1 + b) &= 1 \end{aligned}$$

Explanation: For the point **x4**, we can say that point lies **on the hyperplane in the negative region** and the equation determines that the product of our actual output and the hyperplane equation **is equal to 1** which means the point is correctly classified in the negative domain.

For the point **x6**:

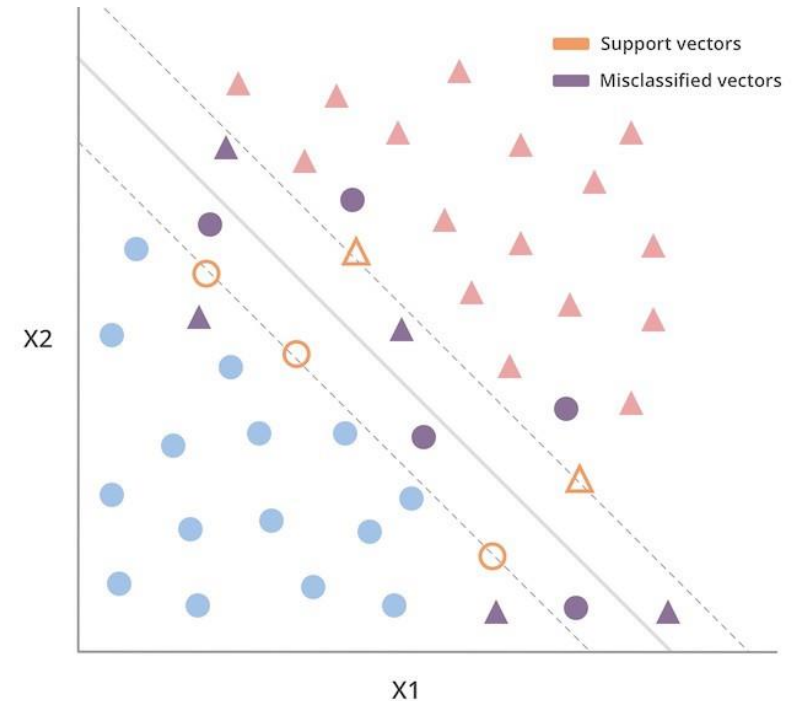
$$\begin{aligned} y_1 &= -1 \\ y_1(w^T x_1 + b) &> 1 \end{aligned}$$

Explanation: For the point **x6**, we can say that point lies **away from the hyperplane in the negative region** and the equation determines that the product of our actual output and the hyperplane equation **is greater than 1** which means the point is correctly classified in the negative domain.



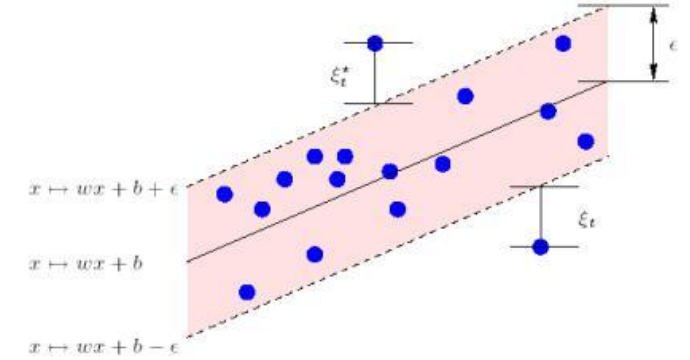
Some Room for Misclassification

- The addition of a margin improves the quality of the predictions in the testing set, but it assumes the classes are completely separable.
- In most real-world problems, data is messy and it's not completely separable.
- **SVM** shares an important characteristic with the algorithm that came before it, support vector classifiers. It allows the algorithm to make mistakes, and assign the wrong class to some vectors.
- So, instead of trying to completely separate the vectors into two classes, **SMV** makes a trade-off.
- It allows for some vectors to fall inside the margin and on the wrong side of the decision boundary.

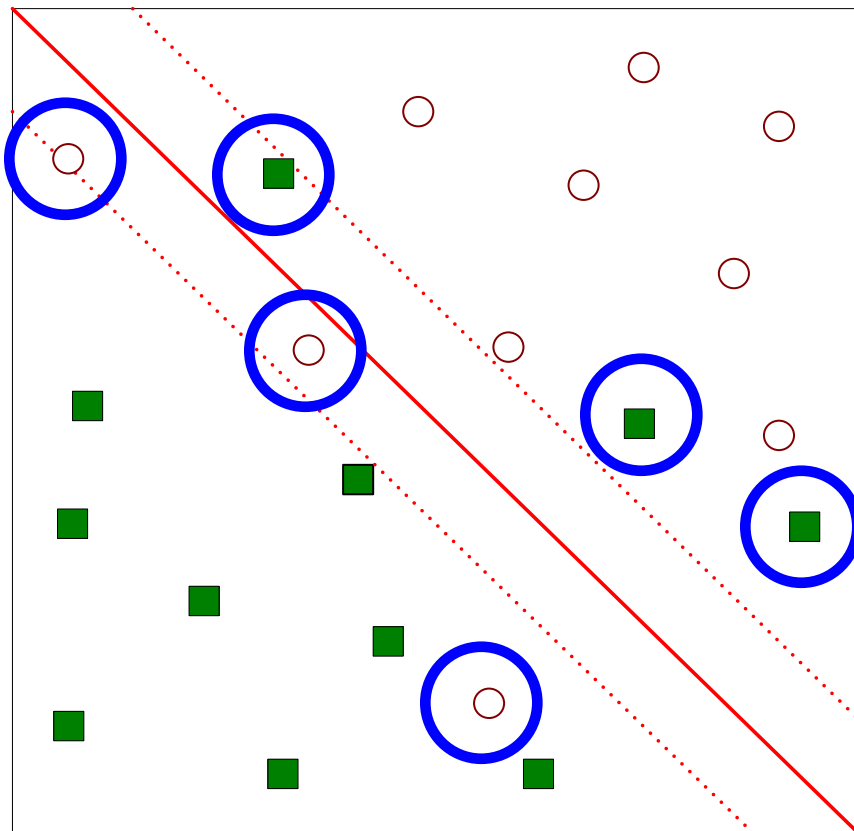


- Decision boundary and margin for support vector classifiers, along with the corresponding support vectors.

Support Vector Machines



- What if the problem is not linearly separable?



- Introduce slack variables

- Need to minimize:

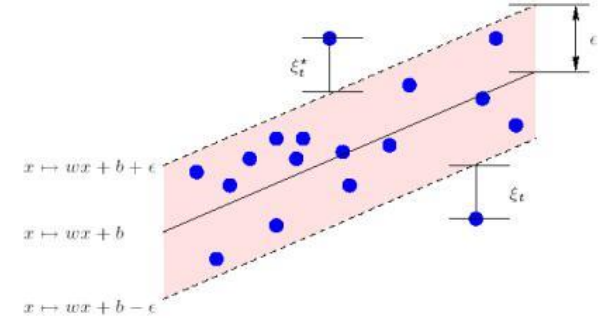
$$L(w) = \frac{\|\vec{w}\|^2}{2} + C \left(\sum_{i=1}^N \xi_i^k \right)$$

- Subject to:

$$y_i = \begin{cases} 1 & \text{if } \vec{w} \cdot \vec{x}_i + b \geq 1 - \xi_i \\ -1 & \text{if } \vec{w} \cdot \vec{x}_i + b \leq -1 + \xi_i \end{cases}$$

- If k is 1 or 2, this leads to same objective function as linear SVM but with different constraints.
- Slack variables are introduced to allow certain constraints to be violated. We want the number of points within the margin to be as small as possible, and of course we want their penetration of the margin to be as small as possible.

Support Vector Machine



- **Support Vector Machines** allow some misclassification during the learning process. So, they can do a better job at classifying most vectors in the testing set.
- Besides the margin, our model now includes **slack variables**, which tell us two things
 - if a test observation misclassified
 - where the observation is relative to the decision boundary and the margin
- **Slack variables** can have three possible values
$$\epsilon_i = \begin{cases} 0, & \text{correctly classified} \\ > 0, & \text{wrong side of the margin} \\ > 1, & \text{wrong side of the hyperplane} \end{cases}$$
- And number of misclassified vectors is bound by a parameter **C**.

$$y_i(b + \langle w \cdot X_i \rangle) \geq M(1 - \epsilon_i) \quad \sum_{i=1}^n \epsilon_i \leq C$$

- Where the classification based on and the vectors fall relative to the margin, including slack variables.

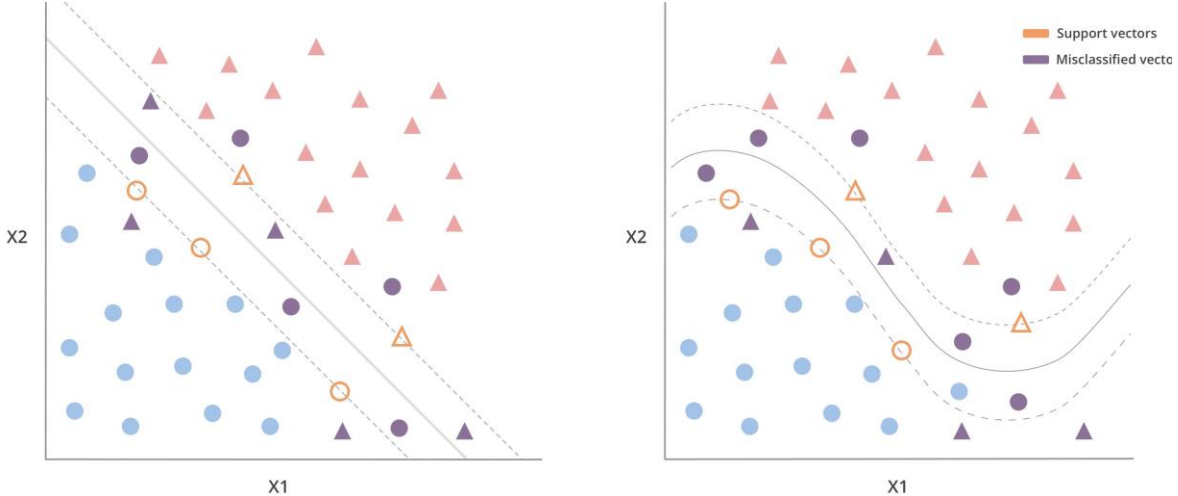
- As can be seen, the model catches a lot more subtlety. But it's still built on top of maximum margin classifiers. For instance, if we set parameter **C** to zero, meaning it allows zero slack variables, it falls back to a maximum margin classifier. So we have a linear decision boundary, a margin that is as large as possible and no vectors allowed inside it.
- The higher the number of slack variables, the higher the number of misclassified vectors allowed. This impacts the width of the margin, because picking different support vectors. It also controls the Bias-Variance trade-off of the model.

Large number of slack variables	Small number of slack variables
<ul style="list-style-type: none">• Wide margin• Low variance• High bias• More support vectors	<ul style="list-style-type: none">• Narrow margin• High variance• Low bias• Fewer support vectors

- How the number of slack variables control the bias-variance trade-off?

- Having some room for misclassification makes SVMs more flexibility, but it only applies to a limited set of problems.
- In most real-world problems, it's hard to separate data into two classes with a linear decision boundary. Even with some room for error.

Support Vector Machine

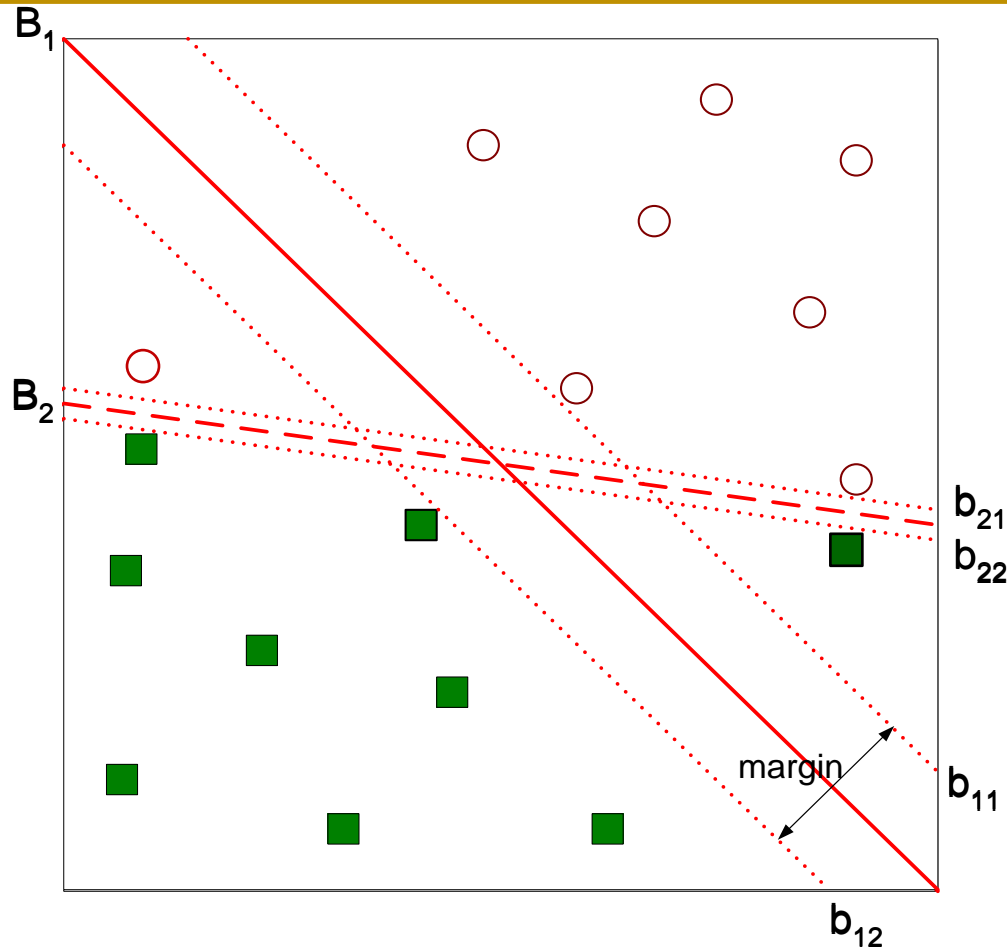
- The functions that define these transformations of features are called **kernels**.
 - They work as similarity functions between observations in the training and testing sets. Whenever we have a model that is represented with inner products, we can plug in a kernel function.
 - For instance, using a linear kernel is equivalent to transforming feature space linearly. In this instance, the decision boundary is linear, making it the same as a support vector classifier.
- 
- Decision boundary and margin for SVM, along with the corresponding support vectors, using a linear kernel (right) and a polynomial kernel (left).
- We are projecting the initial feature space into a polynomial feature space using polynomial kernels. Consequently, a higher order polynomial is used to establish the decision boundary that divides the classes.
 - Support vector classifiers differ from support vector machines in that they employ kernels. Additionally, they make it possible to take on more challenging issues. However, expanding the feature space might need more computing. Because fitting a model can be costly in terms of both time and resources when the feature space is large enough.

- **SVMs** share the characteristics of the **margin classifiers** that came before it. What is unique about them? How can they define both linear and non-linear decision boundaries?
- To support non-linear decision boundaries, SVMs use functions to transform the original feature space into a new space that can represent those non-linear relationships.
- For instance, if we augment the original feature space with the square its features. In this case, we applied a quadratic function to the original feature set to create the square of those features. Now we have the **original feature** and their **quadratic version**, in this augmented space. And so, implicitly, there's a function that maps these two feature spaces.

$$X_1, X_2, X_3 \rightarrow X_1, X_1^2, X_2, X_2^2, X_3, X_3^2$$

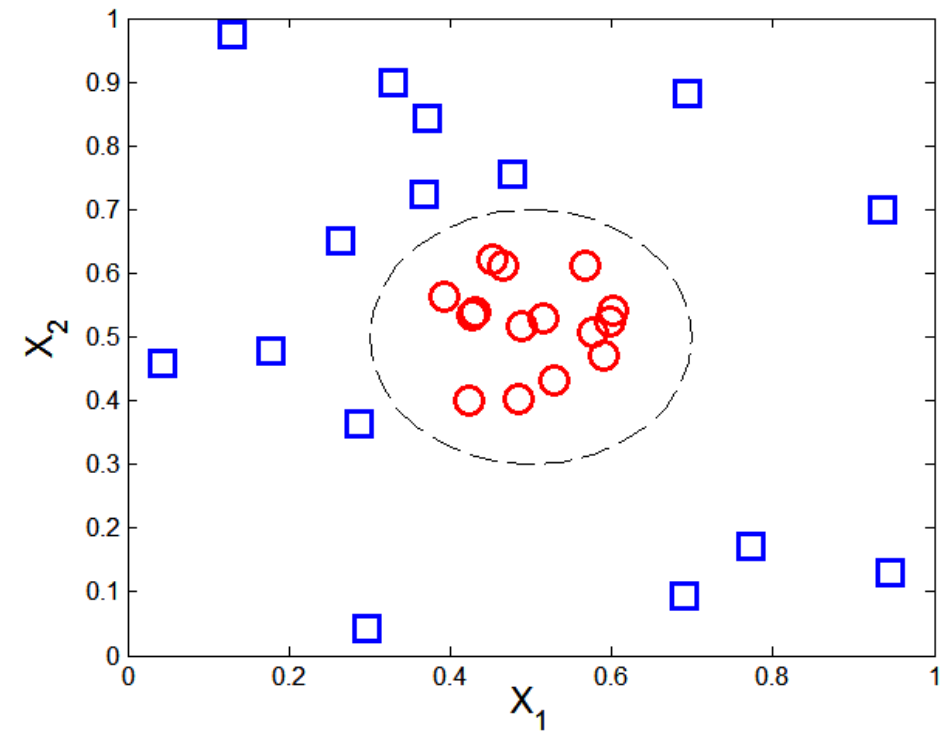
- If we try to draw the decision boundary in the original feature space, it has a quadratic shape. But if we train our model in the augmented feature space, we find a linear decision boundary that separates the two classes. Because it is a transformation, the quadratic boundary in original feature space corresponds to a linear one in the augmented feature space.

Support Vector Machines



- Find the hyperplane that optimizes both factors

- What if decision boundary is not linear?



$$y(x_1, x_2) = \begin{cases} 1 & \text{if } \sqrt{(x_1 - 0.5)^2 + (x_2 - 0.5)^2} > 0.2 \\ -1 & \text{otherwise} \end{cases}$$

Linear and Non-linear Kernels

- **Polynomial Kernel:** It is a rather generalized form of the linear kernel. It can distinguish curved or nonlinear input space.
- It is non-uniform in nature due to the polynomial combination of the base features.
- But the issue with the polynomial kernel is, the number of higher dimension features increases exponentially. As a result, this is computationally more expensive than RBF or linear kernel.

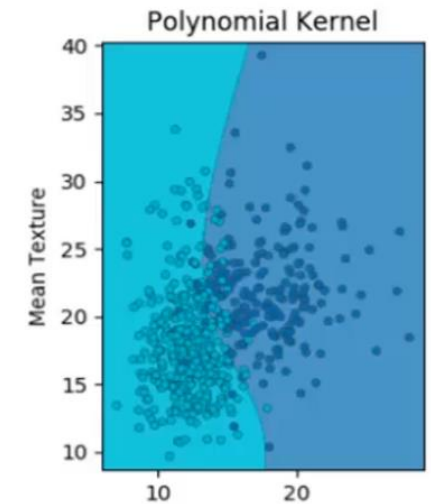
$$K(X_1, X_2) = (a + X_1^T X_2)^b$$

b = degree of kernel & a = constant term.

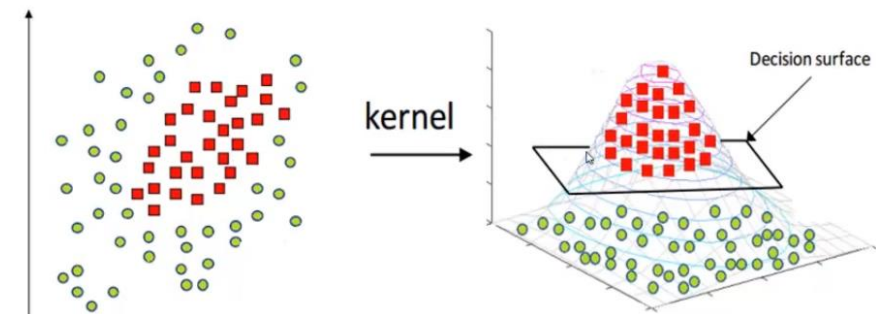
- **Radial Basis Function Kernel:** The radial basis function kernel is commonly used in SVM classification, it can map the space in infinite dimensions. Following is the RBF kernel equation.

$$K(X_1, X_2) = \text{exponent}(-\gamma \|X_1 - X_2\|^2)$$

$\|X_1 - X_2\|$ = Euclidean distance between X_1 & X_2



Gaussian / RBF kernel



Learning Linear SVM

Supporting Material

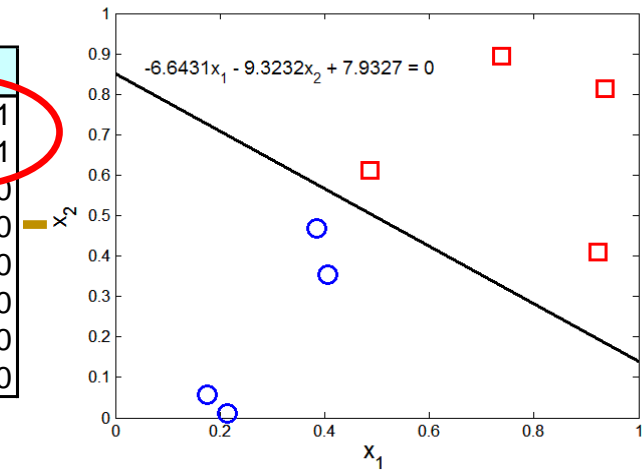
- Decision boundary depends only on support vectors
- If you have data set with same support vectors, decision boundary will not change
- How to classify using **SVM** once **w** and **b** are found? Given a test record, **x_i**

$$y_i = f(\vec{x}_i) = \begin{cases} 1 & \text{if } \vec{w} \cdot \vec{x}_i + b \geq 1 \\ -1 & \text{if } \vec{w} \cdot \vec{x}_i + b \leq -1 \end{cases}$$

Support vectors

x1	x2	y	λ
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

Example of Linear SVM



Example 5.5. Consider the two-dimensional data set shown in Figure 5.24, which contains eight training instances. Using quadratic programming, we can solve the optimization problem stated in Equation 5.43 to obtain the Lagrange multiplier λ_i for each training instance. The Lagrange multipliers are depicted in the last column of the table. Notice that only the first two instances have non-zero Lagrange multipliers. These instances correspond to the support vectors for this data set.

Let $\mathbf{w} = (w_1, w_2)$ and b denote the parameters of the decision boundary. Using Equation 5.39, we can solve for w_1 and w_2 in the following way:

$$w_1 = \sum_i \lambda_i y_i x_{i1} = 65.5261 \times 1 \times 0.3858 + 65.5261 \times -1 \times 0.4871 = -6.64.$$

$$w_2 = \sum_i \lambda_i y_i x_{i2} = 65.5261 \times 1 \times 0.4687 + 65.5261 \times -1 \times 0.611 = -9.32.$$

The bias term b can be computed using Equation 5.42 for each support vector:

$$b^{(1)} = 1 - \mathbf{w} \cdot \mathbf{x}_1 = 1 - (-6.64)(0.3858) - (-9.32)(0.4687) = 7.9300.$$

$$b^{(2)} = -1 - \mathbf{w} \cdot \mathbf{x}_2 = -1 - (-6.64)(0.4871) - (-9.32)(0.611) = 7.9289.$$

Averaging these values, we obtain $b = 7.93$. The decision boundary corresponding to these parameters is shown in Figure 5.24.

Summary

- Table compares the accuracies of three different classifiers, **decision tree classifiers, Naïve Bayes classifiers, and support vector machines**, on various data sets.
- Since the learning problem is formulated as a convex optimization problem, efficient algorithms are available to find the global minima of the objective function (many of the other methods use greedy approaches and find locally optimal solutions)
- Overfitting is addressed by maximizing the margin of the decision boundary, but the user still needs to provide the type of kernel function and cost function
- Difficult to handle missing values
- Robust to noise
- High computational complexity for building the model

Comparing the accuracy of various classification methods.

Data Set	Size (N)	Decision Tree (%)	naïve Bayes (%)	Support vector machine (%)
Anneal	898	92.09	79.62	87.19
Australia	690	85.51	76.81	84.78
Auto	205	81.95	58.05	70.73
Breast	699	95.14	95.99	96.42
Cleve	303	76.24	83.50	84.49
Credit	690	85.80	77.54	85.07
Diabetes	768	72.40	75.91	76.82
German	1000	70.90	74.70	74.40
Glass	214	67.29	48.59	59.81
Heart	270	80.00	84.07	83.70
Hepatitis	155	81.94	83.23	87.10
Horse	368	85.33	78.80	82.61
Ionosphere	351	89.17	82.34	88.89
Iris	150	94.67	95.33	96.00
Labor	57	78.95	94.74	92.98
Led7	3200	73.34	73.16	73.56
Lymphography	148	77.03	83.11	86.49
Pima	768	74.35	76.04	76.95
Sonar	208	78.85	69.71	76.92
Tic-tac-toe	958	83.72	70.04	98.33
Vehicle	846	71.04	45.04	74.94
Wine	178	94.38	96.63	98.88
Zoo	101	93.07	93.07	96.04

- Introduction to Data Mining, 2nd Edition, Pang-Ning Tan, Michael Steinbach, Anuj Karpatne, Vipin Kumar, 2019, Pearson.
- Introduction to Machine Learning with Python A Guide for Data Scientists, Andreas C. Müller and Sarah Guido, Copyright © 2017, O'Reilly.
- Machine Learning with Python Cookbook: Practical Solutions from Preprocessing to Deep Learning Paperback – 23 Mar. 2018. by. Chris Albon
- Thoughtful Machine Learning by Matthew Kirk Published by O'Reilly Media, Inc., 2014
- <https://towardsdatascience.com/support-vector-machines-svm-c9ef22815589>
- <https://github.com/ApoorvRusia/Naive-Bayes-classification-on-Iris-dataset>
- Some images are used from google search repository to enhance the level of learning.
- Sample datasets for Regression: <https://www.kaggle.com/tags/regression>

Copyright Notice

The following material has been communicated to you by or on behalf of CCT College Dublin in accordance with the Copyright and Related Rights Act 2000 (the Act).

The material may be subject to copyright under the Act and any further reproduction, communication or distribution of this material must be in accordance with the Act.

Do not remove this notice