Tenth Session

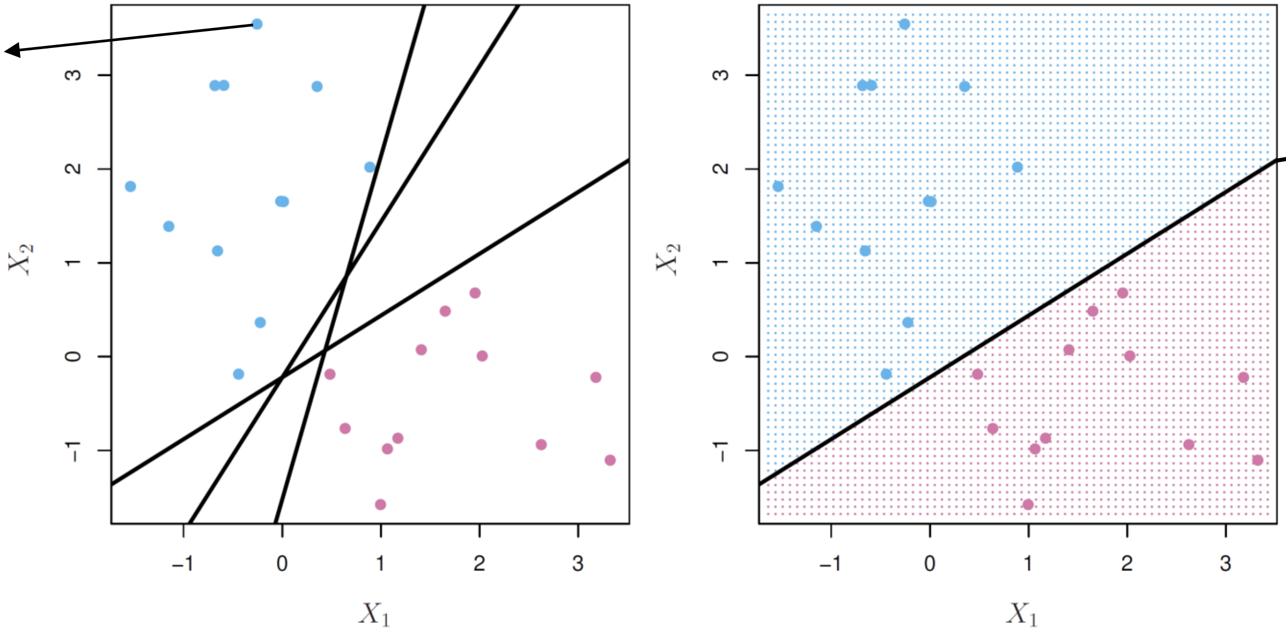
Alireza Moradi

Support vector machine

Support vector machine (SVM)

- **Support vector machine (SVM)** is a supervised machine learning algorithm that classifies data by finding an **optimal line** or hyperplane that maximizes the distance between each class in an N-dimensional space.

Train or test data?

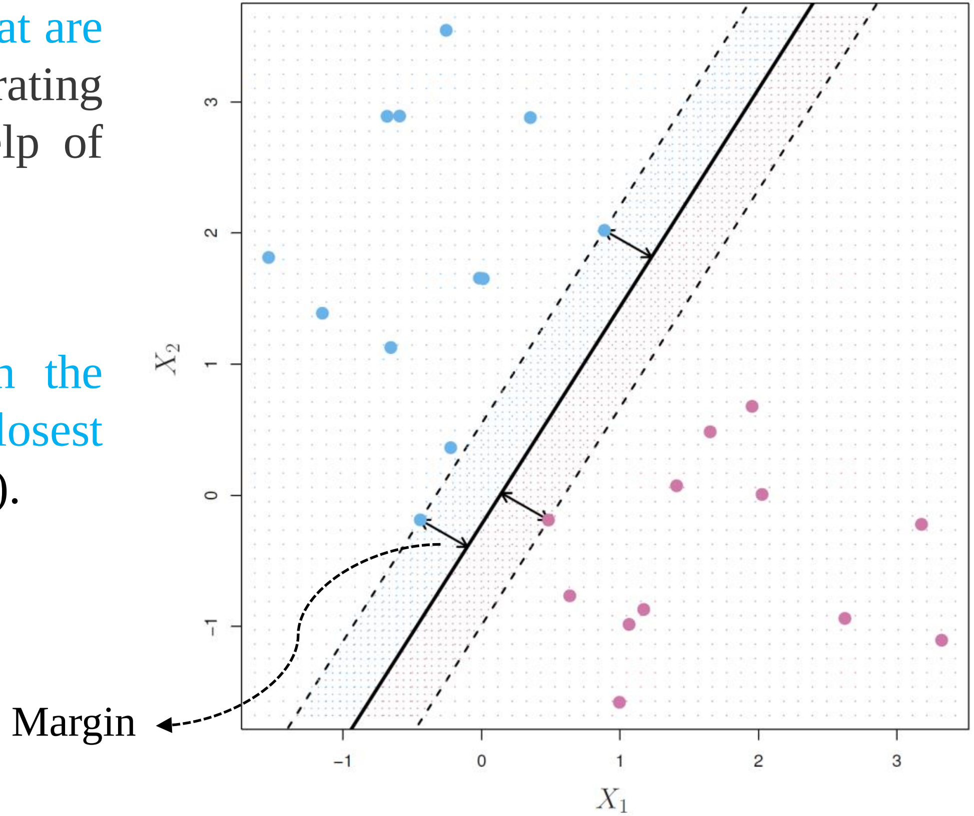


Decision boundary

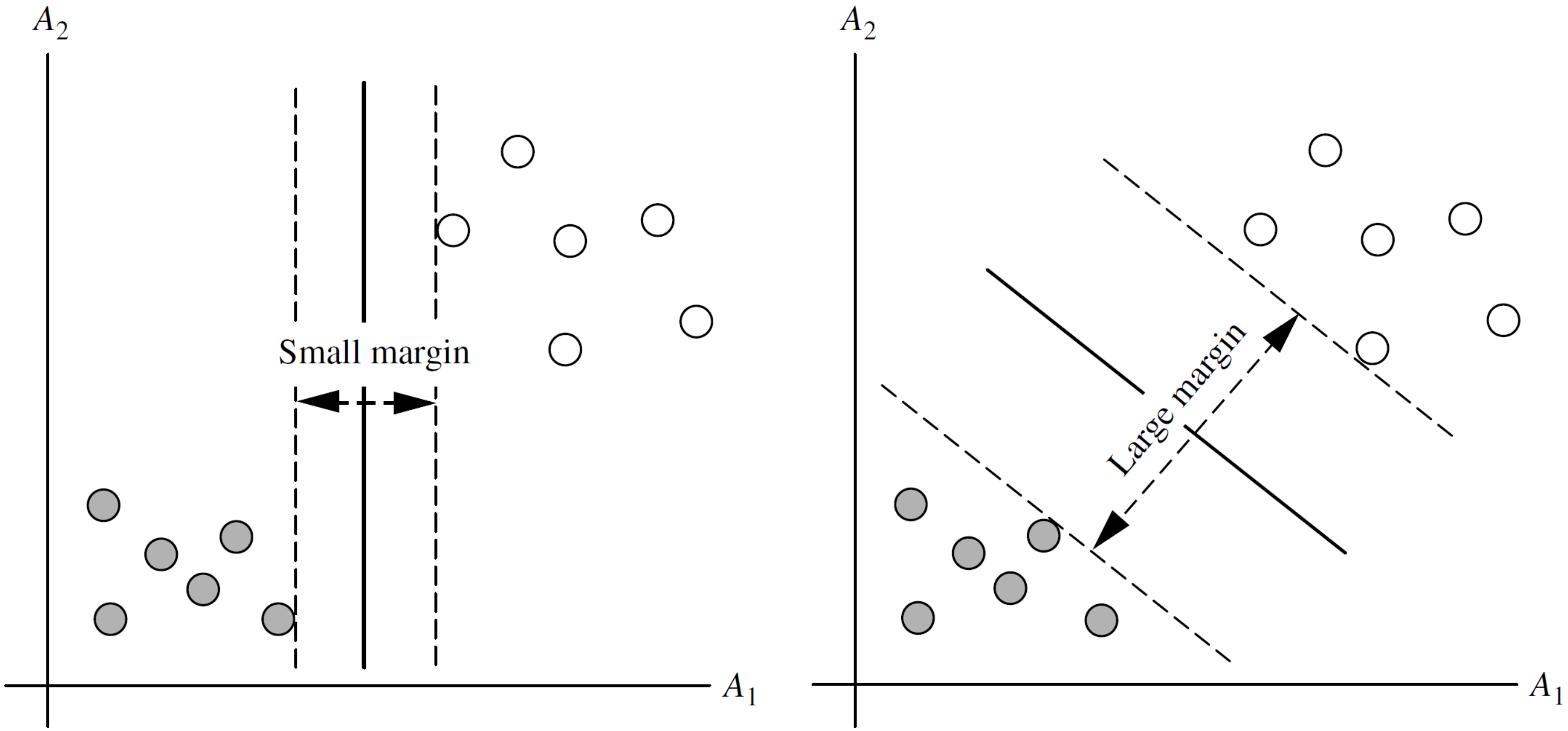
SVM terminology

- **Support Vectors** are the points that are closest to the hyperplane. A separating line will be defined with the help of these data points.

- **Margin** is the distance between the hyperplane and the observations closest to the hyperplane (support vectors).



SVM terminology



SVM; optimal hyperplane

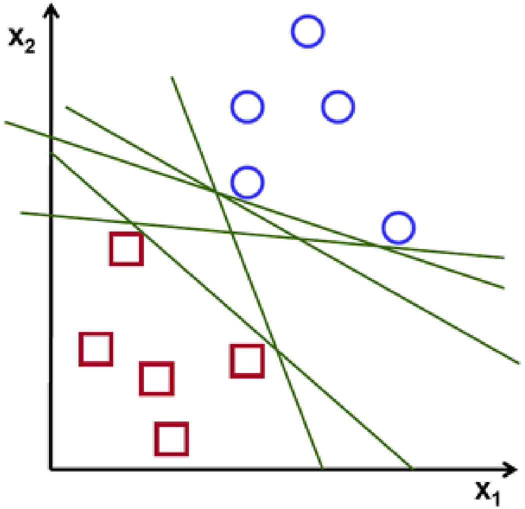
- The remaining question is how to identify the optimal hyperplane.

- **Among all separating hyperplanes**, we try to find the one that makes the biggest margin.

- A hyperplane in p dimensions is a flat subspace of dimension p-1.

- In general, the equation for a hyperplane has the form:

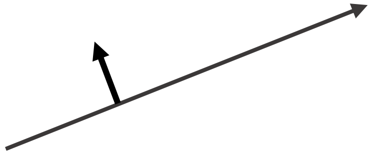
$$\beta_0 + \beta_1 . X_1 + \beta_2 . X_2 + \ \ldots + \beta_p . X_p = 0$$



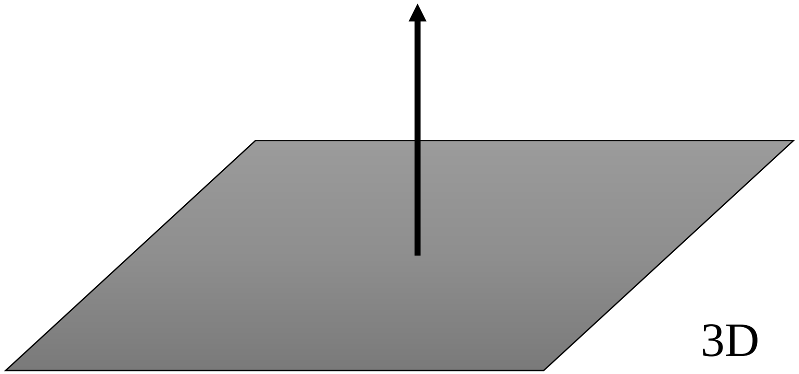
SVM; optimal hyperplane

- The vector $\beta = (\beta_1, \beta_2, \ldots, \beta_p)$ is called the **normal vector of hyperplane**. (It is also denoted by W)

- The normal vector points in a direction orthogonal to the surface of a hyperplane.



2D



3D

SVM; optimal hyperplane

- We have a Constrained optimization problem:

$$\underset{\beta_0,\beta_1,\dots,\beta_p}{maximise}\ M$$

subject to:

$$\sum_{j=1}^{p} \beta_j^2 = 1$$

$$y_i \cdot \left(\beta_0 + \beta_1.X_{i1} + \beta_2.X_{i2} + \ \dots + \beta_p.X_{ip}\right) \geq M \quad \forall i = 1,2,\cdots,n$$

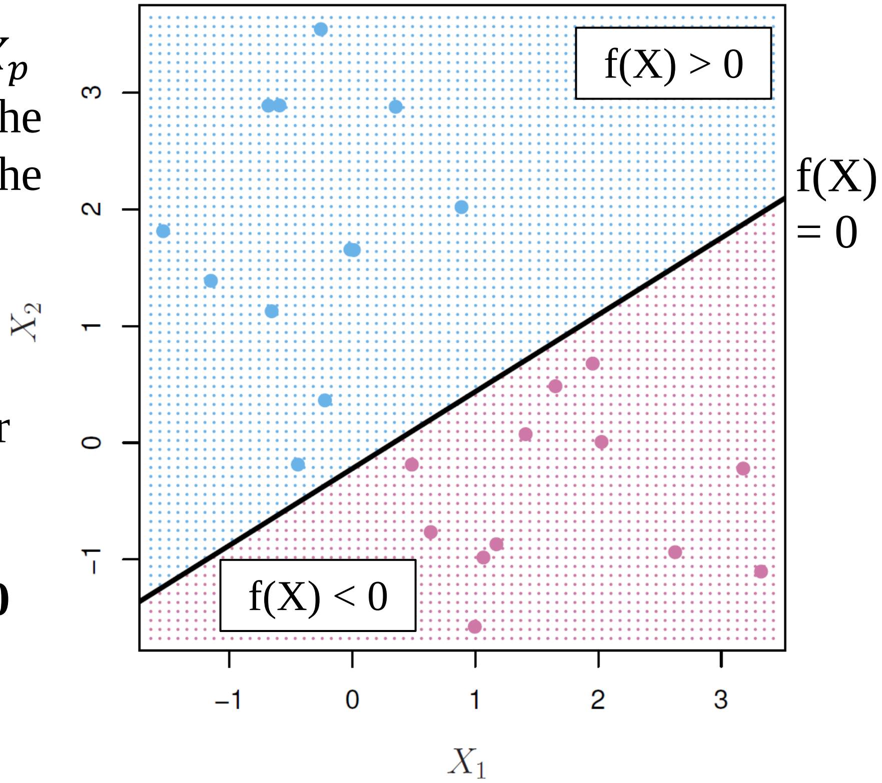
✓ This can be rephrased as a **convex quadratic program** and solved efficiently.

SVM; optimal hyperplane

- If f(X) $= \beta_0 + \beta_1 . X_1 + \beta_2 . X_2 + \ \dots + \beta_p . X_p$ then f(X) > 0 for points on one side of the hyperplane, and f(X) < 0 for points on the other.

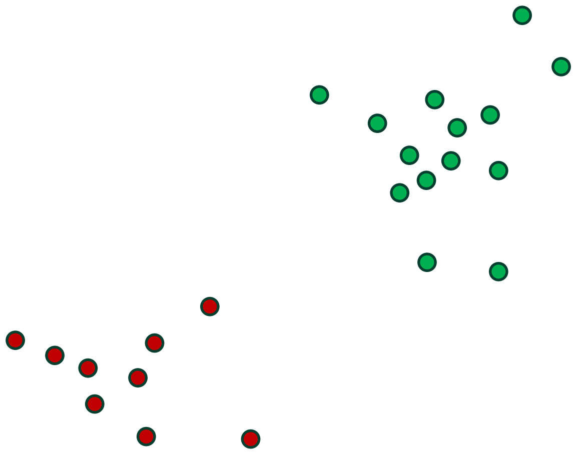
- If we code the colored points as $y_i = \ +1$ for blue, and $y_i = -1$ for red:

  then if y_i . f(X) ≥ 0 for all i, **f(X) = 0** denotes a separating hyperplane.

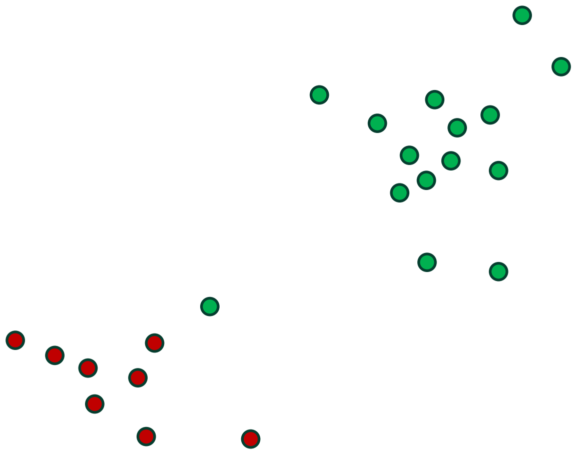


SVM; Examples

- Example 1:



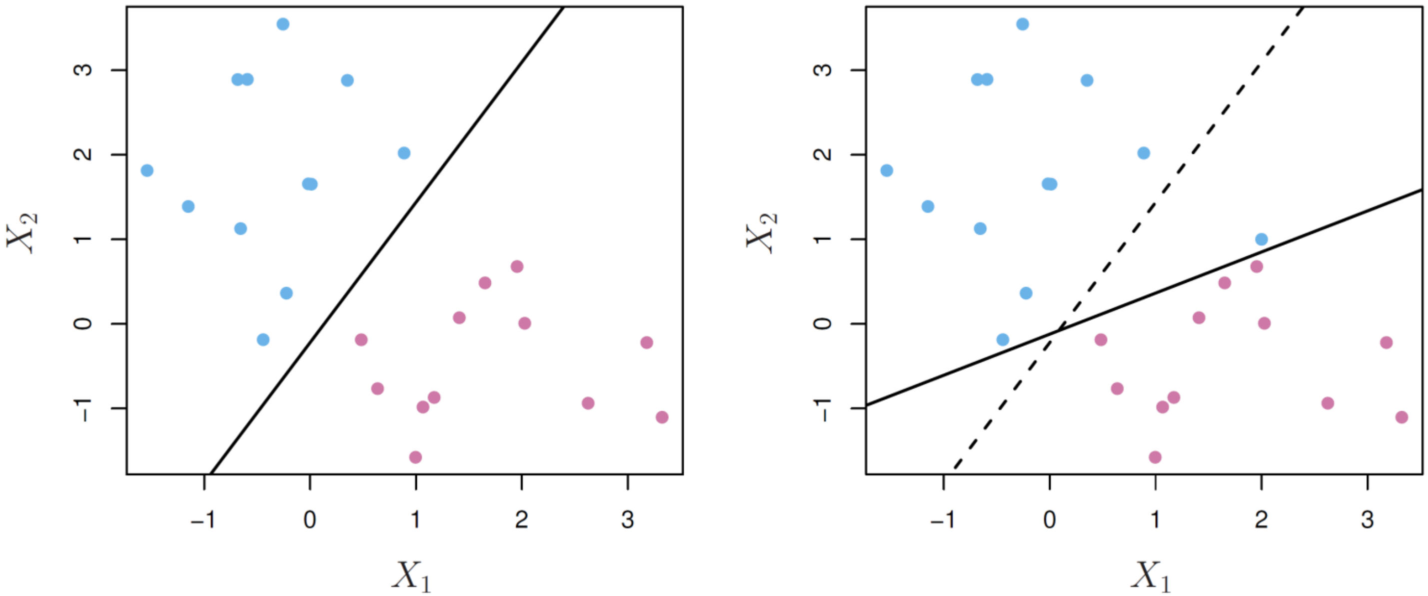
✓ Example 2:



✓ Do SVMs directly provide probability estimates?

SVM; Examples

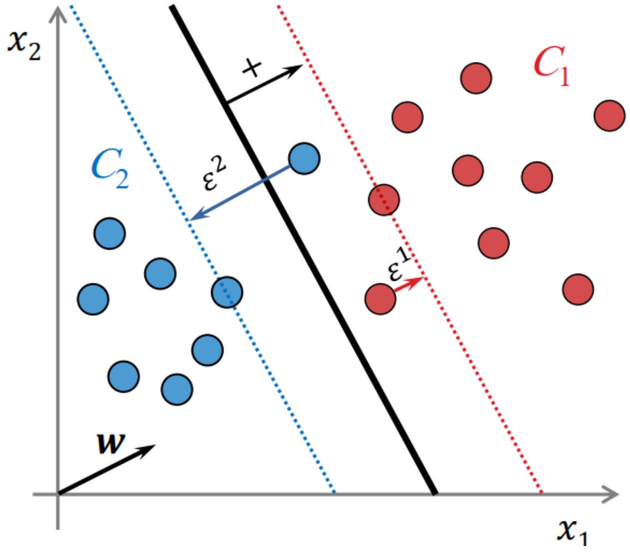
- Noisy Data can lead to a poor solution for the maximal-margin classifier.



✓ What solution can you propose?

Soft margin SVM

- Solution: The support vector classier maximizes a soft margin.



$$\underset{\beta_0,\beta_1,\cdots,\beta_p}{maximise}\ M$$

subject to:

$$\sum_{j=1}^{p} \beta_j^2 = 1$$

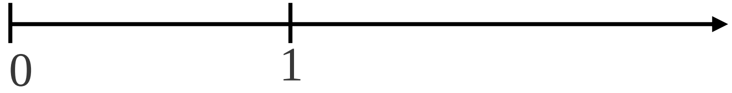
$$y_i \cdot \left(\beta_0 + \beta_1 . X_{i1} + \beta_2 . X_{i2} + \ \ldots + \beta_p . X_{ip}\right) \geq M(1-\varepsilon_i) \qquad \forall i = 1,2,\cdots,n$$

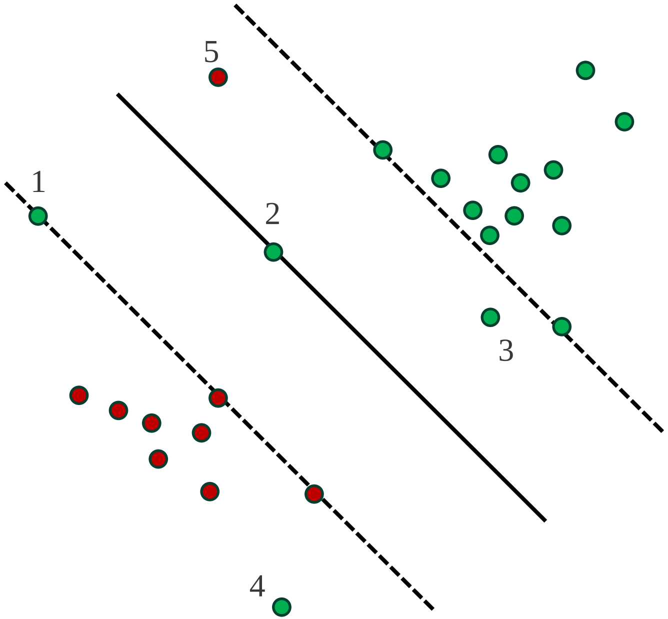
$$\varepsilon_i \geq 0 \qquad \forall i = 1,2,\cdots,n$$

$$\sum_{i=1}^{n} \varepsilon_i \leq C$$

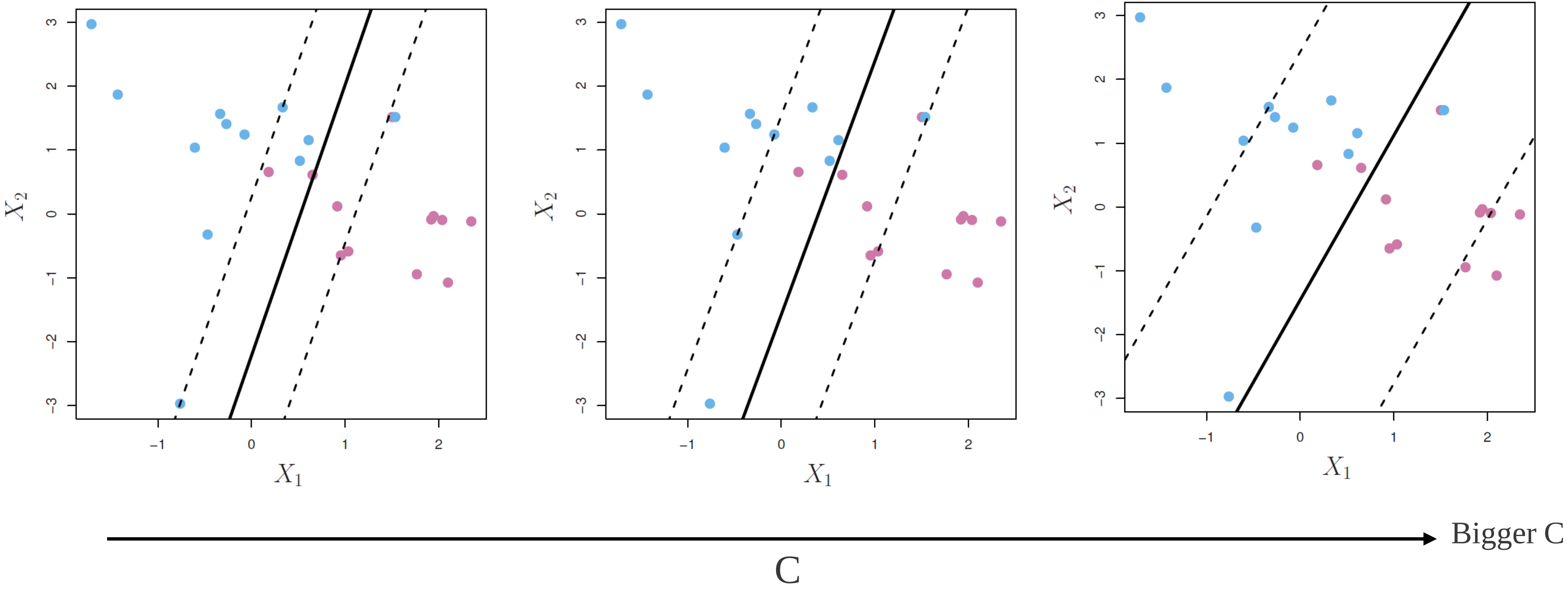
Soft margin SVM

- Different values of ε_i:





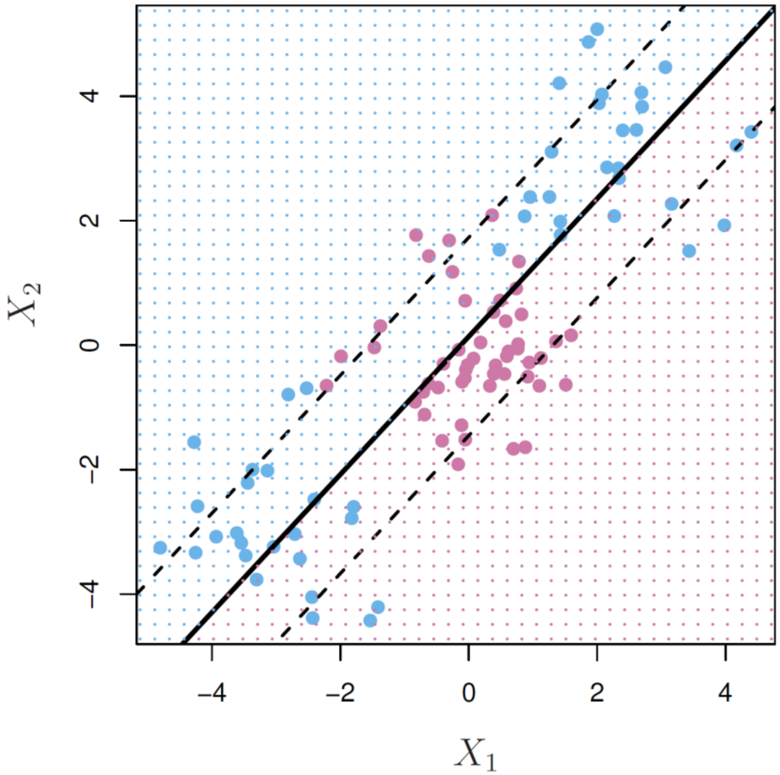
Soft margin SVM



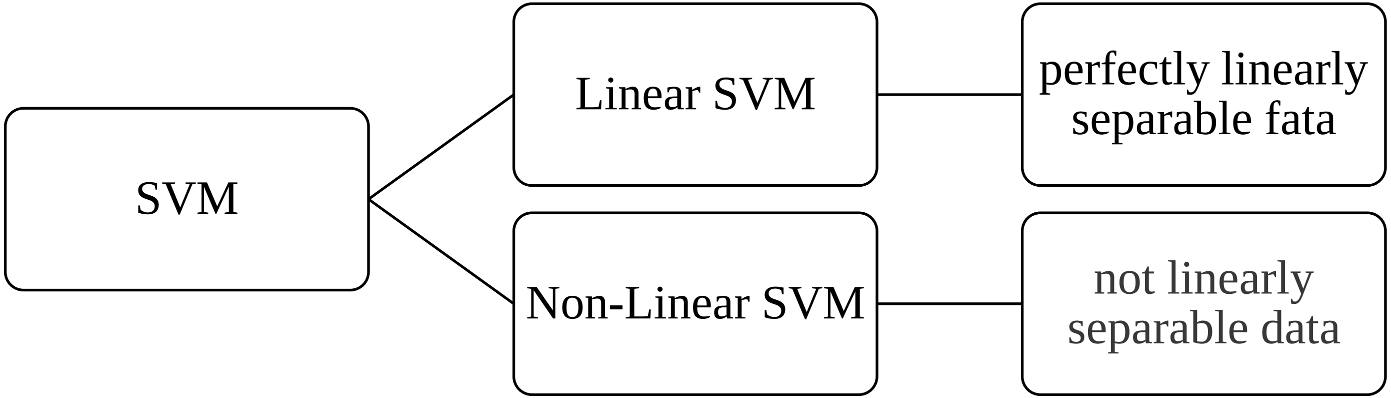
Support vector machine (SVM)

- Data may not be linearly sparable in its feature space.
- No matter what value of C in soft margin, a linear boundary won't work.

✓ What is your solution?



Support vector machine (SVM)



- When the data is not linearly separable then we can use **Non-Linear SVM**, which means when the data points cannot be separated into 2 classes by using a hyperplane. Then we use some advanced techniques like **kernel** tricks to classify them.

- In most real-world applications we do not find linearly separable datapoints.

Non-Linear SVM

- While feature expansion can be used to create non-linear decision boundaries, a more common approach is to utilize kernel functions.

✓ Feature expansion: (How?)

$$\beta_0 + \beta_1 . x_1 + \beta_2 . x_2 = 0$$

$$\Downarrow$$

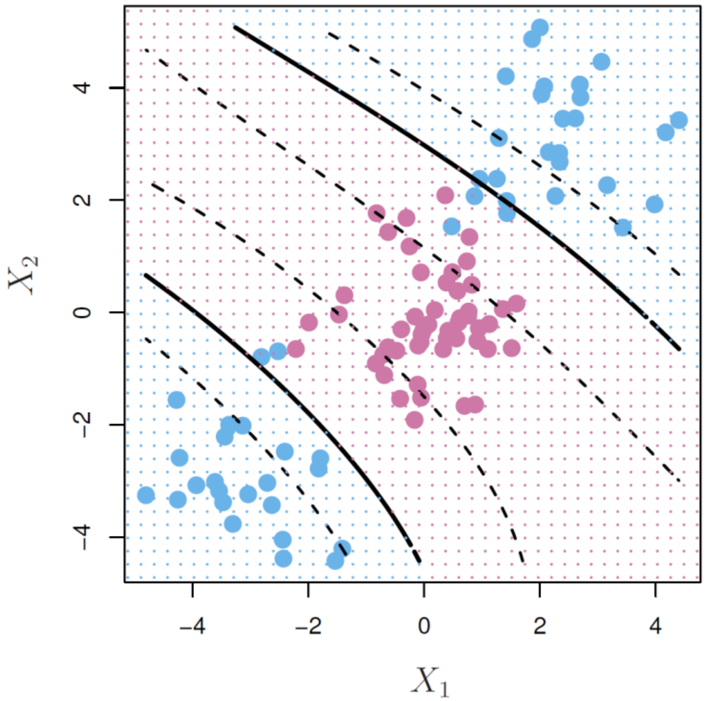
$$\beta_0 + \beta_1 . x_1 + \beta_2 . x_2 + \beta_3 . x_1^2 + \beta_4 . x_2^2 = 0$$

- This leads to nonlinear decision boundaries in the original space.

Non-Linear SVM

- Here we use a basis expansion of cubic polynomials.

- The support-vector classifier in the enlarged space solves the problem in the lower-dimensional space.



✓ From 2 variables to 9

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1^2 + \beta_4 X_1 X_2 + \beta_5 X_2^2 + \beta_6 X_1^3 + \beta_7 X_1^2 X_2 + \beta_8 X_1 X_2^2 + \beta_9 X_2^3 = 0$$

Non-Linear SVM

- While feature expansion can create non-linear decision boundaries, it requires **manually choosing the feature mapping** and can lead to **high computational costs**.

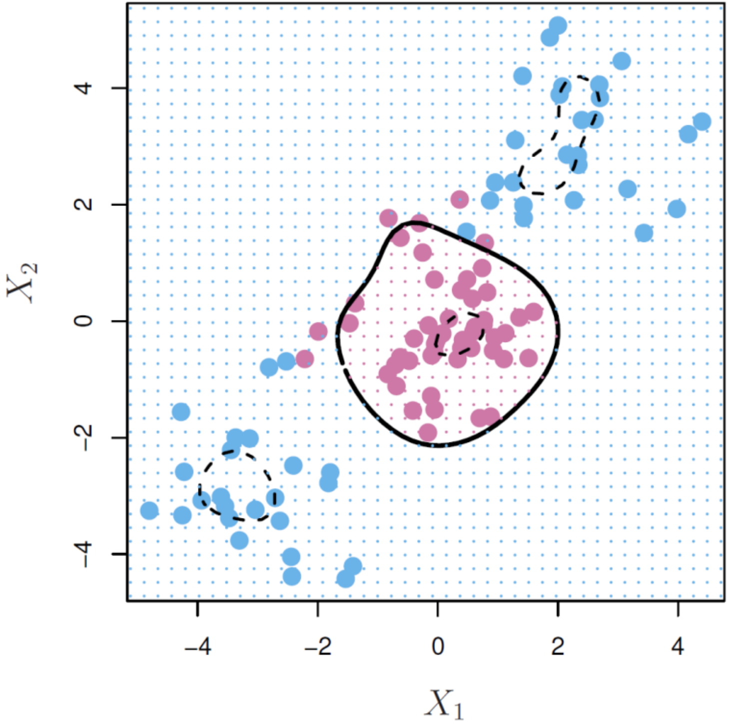
- Kernel functions offer a **more automated** and **efficient** alternative. However, a detailed explanation of their inner workings goes beyond the scope of this course.

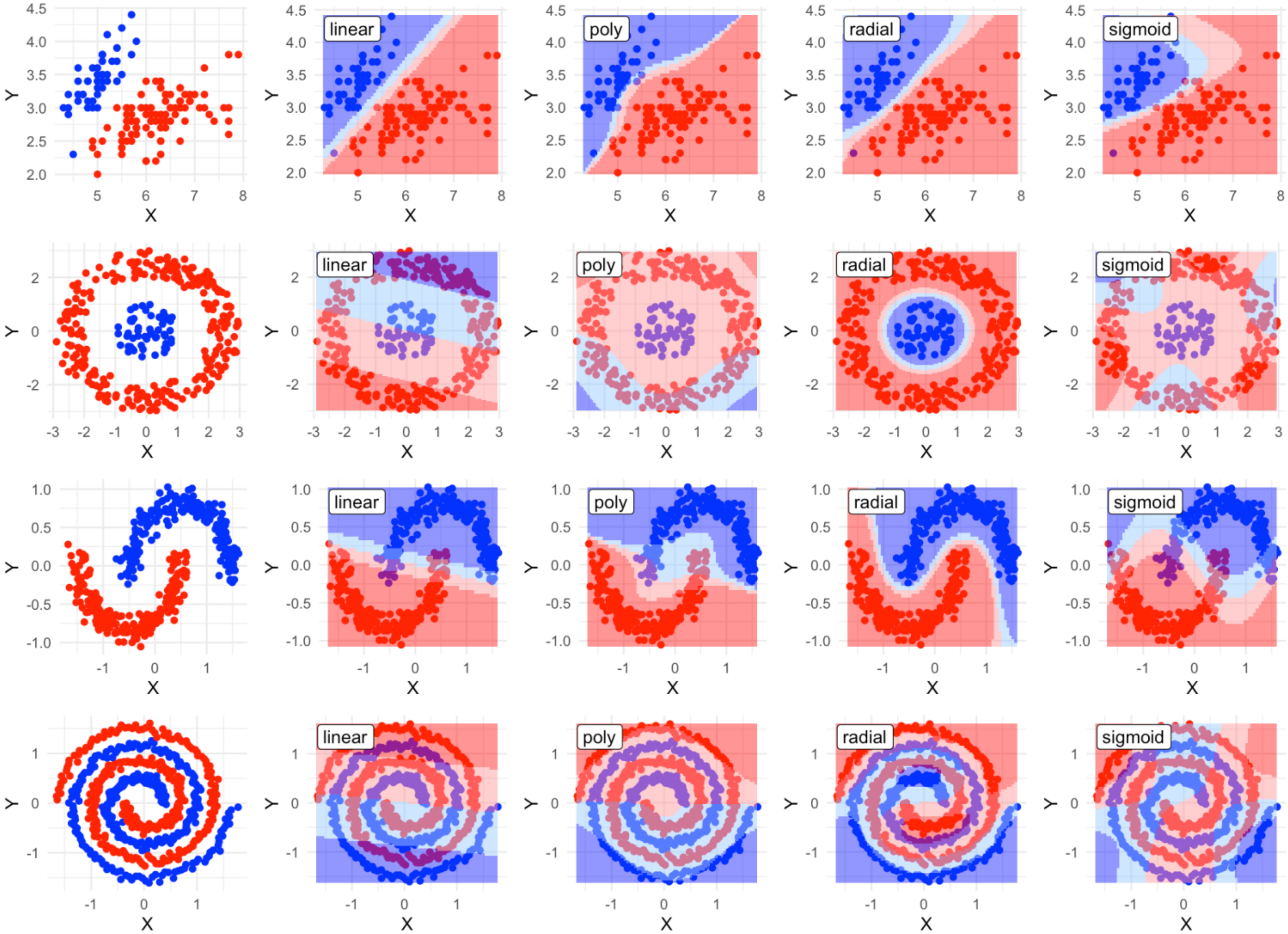
- Some kernel functions which you can use in SVM are given below:
  - ✓ Linear
  - ✓ Polynomial
  - ✓ Gaussian Radial Basis Function (RBF)
  - ✓ Sigmoid
  - ✓ Tanh

- Common kernels are provided in software packages, but it is also possible to specify custom kernels.

Non-Linear SVM

- For example, we used radial(RBF) kernel here:





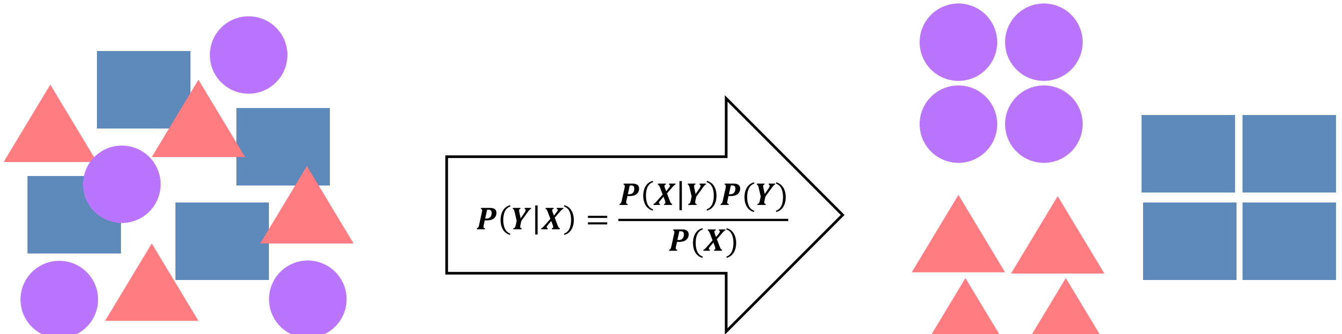
linear
poly
radial
sigmoid
X
Y

Naïve Bayes

Naïve Bayes

- **Naive Bayes** is a statistical classification technique based on Bayes Theorem. It is one of the simplest supervised learning algorithms.

- The algorithm calculates the probability of each class given the input features and selects the class with the highest probability as the predicted class.



Naïve Bayes Classifier

Naïve Bayes Formula

Likelihood Probability

Class Prior Probability

Posterior Probability

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

Predictor Prior Probability
or evidence

Naïve Bayes Formula

- In **Class Prior Probability,** the term "prior" refers to the fact that this probability is determined prior to considering any features or evidence.

- **Likelihood Probability** or **conditional probability** represents the probability of observing a particular feature value given a specific class.

- **Posterior Probability** is the probability of a specific class given the observed evidence or features. The term "posterior" refers to the fact that it is calculated after considering the evidence.

- **Predictor Prior Probability** or **evidence** refers to the probability of observing a particular set of evidence or features in the dataset.

Naïve Bayes Formula

- Probability of each class when we have one feature:

  Probability of class k :

$$P(C_k|X) = \frac{P(X|C_k)P(C_k)}{P(X)}$$

Naïve Bayes Formula

- When we have multiple features, Naive Bayes classifier assumes that the effect of a particular feature in a class is independent of other features.

- For example, a loan applicant is desirable or not depending on his/her income, previous loan and transaction history, age, and location.

- Even if these features are interdependent, these features are still considered independently.

- This assumption simplifies computation, and that's why it is considered as **naive**. This assumption is called **class conditional independence**.

Naïve Bayes Formula

- Probability of each class when we have **multiple features**:

$$P\left(C_k \middle| x_1, x_2, \dots, x_p\right) = \frac{P(x_1, x_2, \dots, x_p | C_k) \times P(C_k)}{P(x_1, x_2, \dots, x_p)}$$

- So, with class conditional independence assumption the probability of class n given multiple features, changes to:

$$P\left(C_k \middle| x_1, x_2, \dots, x_p\right) = \frac{P(x_1 | C_k) \times P(x_2 | C_k) \times \cdots \times P\left(x_p \middle| C_k\right) \times P(C_k)}{P(x_1) \times P(x_2) \times \cdots \times P(x_p)}$$

Naïve Bayes Formula

- In Naive Bayes classification, **we can omit the denominator $\boldsymbol{P(X)}$** because it acts as a constant scaling factor across all the classes.

- The reason for this is that $\boldsymbol{P(X)}$, also known as the **evidence**, represents the probability of observing the input features x, regardless of the class.

- When it comes to classification, we can write the formula as follows:

$$P'\left(C_k \middle| x_1, x_2, \dots, x_p\right) = P(x_1|C_k) \times P(x_2|C_k) \times \cdots \times P\left(x_p \middle| C_k\right) \times P(C_k)$$

Naïve Bayes; Example

- In the next few slides, we will explain this algorithm using a simple example.

- Let’s say we have a table that decided if we should play tennis under certain circumstances. These could be the outlook of the weather, the temperature, the humidity, and the strength of the wind.

- You can see the table in the next slide.

Day	Outlook	Temperature	humidity	wind	Play?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Naïve Bayes; Example

- In Naive Bayes classification, one of the key steps to solve the classification problem is to **calculate the likelihood of each feature given each class**. This step involves estimating the conditional probabilities P(feature | class) for each feature and class pair.

- Likelihood probabilities for our example:

Outlook	**Play=Yes**	**Play=No**
Sunny	2/9	3/5
Overcast	4/9	0/5
Rain	3/9	2/5

Naïve Bayes; Example

- Likelihood probabilities:

Temperature	Play=Yes	Play=No
Hot	2/9	2/5
Mild	4/9	2/5
Cool	3/9	1/5

Humidity	Play=Yes	Play=No
High	3/9	4/5
Normal	6/9	1/5

Wind	Play=Yes	Play=No
Strong	3/9	3/5
Weak	6/9	2/5

Naïve Bayes; Example

- Now we should compute the probability of each class also known as the class prior probability:

$$P(Play = Yes) = \frac{9}{14}$$

$$P(Play = No) = \frac{5}{14}$$

✓ How did we compute these?

Naïve Bayes; Example

- Say we were given a new instance, and we want to know if we can play a game or not. This new instance is:

X = (Outlook=Sunny, Temperature=Cool, Humidity=High, Wind=Strong)

Naïve Bayes; Example

- Say we were given a new instance, and we want to know if we can play a game or not. This new instance is:

X = (Outlook=Sunny, Temperature=Cool, Humidity=High, Wind=Strong)

- To classify the new instance using Naive Bayes, we should **calculate the posterior probability for each class** based on the input features.

- By computing the posterior probabilities for each class, one can **determine the most probable class** or the class with the highest probability and classify the new instance accordingly.

- We need to lookup the results from the tables before.

Naïve Bayes; Example

- We can calculate the Posterior probability* using the tables we saw before:

$$\boldsymbol{P'(Play = Yes|X)} = P(X|Play = Yes) \times P(Play = Yes) =$$

$$P(Outlook = Sunny, Temperature = Cool, Humidity = High, Wind = Strong|Play = Yes) \times P(Play = Yes) =$$

$$\frac{2}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{9}{14} = 0.0053$$

$$\boldsymbol{P'(Play = No|X)} = P(X|Play = No) \times P(Play = No) =$$

$$P(Outlook = Sunny, Temperature = Cool, Humidity = High, Wind = Strong|Play = No) \times P(Play = No) =$$

$$\frac{3}{5} \times \frac{1}{5} \times \frac{4}{5} \times \frac{3}{5} \times \frac{5}{14} = 0.0206$$

- Since 0.0206 is greater than 0.0053 then the prediction is ‘no’, we cannot play a game of tennis today.