The Concept Learning Task

Concept learning involves identifying a general rule from specific examples. A computer learns by analyzing labeled data—where examples are marked as "yes" (belongs to the concept) or "no" (does not belong).

- The goal is to identify patterns that define a category.
- It helps machines make predictions about new, unseen data.

For example, a system can be trained to classify emails as spam or not by learning from past labeled emails.

General-to-Specific Ordering of Hypotheses

A hypothesis is a rule the computer uses to classify data. Hypotheses range from very general to very specific:

- **General Hypothesis:** Covers many cases, sometimes including incorrect ones. Example: "All emails are spam."
- **Specific Hypothesis:** Focuses on fewer cases with more accuracy. Example: "Emails with 'Free money' in the subject are spam."

Components of Concept Learning

1. **Instance (X):** The input example with specific features or attributes.
2. **Target Concept (C):** The rule or pattern the system aims to learn.
3. **Hypothesis (H):** A possible rule the machine generates to classify instances.

Example: Identifying Cats

Animal	Fur	Whiskers	Meows?	Cat?
🐈	Yes	Yes	Yes	Yes
🐕	Yes	Yes	No	No
🦁	Yes	Yes	No	No
🐱	Yes	Yes	Yes	Yes

Instance (X): The input example with specific features, such as fur, whiskers, and sound.

Target Concept (C): The rule the model needs to learn, like identifying a cat.

Hypothesis (H): A possible rule the model forms, such as "Fur + Whiskers + Meows = Cat."

Bayes' Theorem

Understanding Bayes' Theorem

Bayes' Theorem is a mathematical formula used to update probabilities based on new evidence. It helps in determining the probability of an event occurring given that another event has already occurred.

The formula for Bayes' Theorem is:

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

Where:

- **P(A | B)** = Probability of event **A** occurring given that **B** has happened (posterior probability).
- **P(B | A)** = Probability of event **B** occurring given that **A** has happened (likelihood).
- **P(A)** = Prior probability of event **A** (before seeing evidence).
- **P(B)** = Total probability of event **B** (evidence).

Example: Diagnosing a Disease

Let's say a doctor wants to determine the probability that a patient has a rare disease given that they tested positive.

Given Data:

- **P(Disease) = 0.01** (Only 1% of people have the disease).
- **P(No Disease) = 0.99** (99% of people don't have the disease).
- **P(Positive Test | Disease) = 0.95** (If a person has the disease, the test correctly detects it 95% of the time).
- **P(Positive Test | No Disease) = 0.05** (If a person doesn't have the disease, the test still falsely shows positive 5% of the time).

Step 1: Calculate P(Positive Test)

We use the law of total probability:

$$P(PositiveTest) = P(PositiveTest|Disease) \cdot P(Disease) + P(PositiveTest|NoDisease) \cdot P(NoDisease)$$

$$P(PositiveTest) = (0.95 \times 0.01) + (0.05 \times 0.99)$$

$$P(PositiveTest) = 0.0095 + 0.0495 = 0.059$$

Step 2: Apply Bayes' Theorem

$$P(Disease|PositiveTest) = \frac{P(PositiveTest|Disease) \cdot P(Disease)}{P(PositiveTest)}$$

$$P(Disease|PositiveTest) = \frac{0.95 \times 0.01}{0.059}$$

$$P(Disease|PositiveTest) = \frac{0.0095}{0.059} = 0.161$$

.

Applications of Bayes' Theorem

- **Medical Diagnosis** – Estimating disease probability given a test result.
- **Spam Filtering** – Determining whether an email is spam based on word frequency.
- **Machine Learning** – Used in Naïve Bayes classifiers for text classification and sentiment analysis.
- **Fraud Detection** – Identifying fraudulent transactions based on past fraud patterns.

Bayes Optimal Classifier

The Bayes Optimal Classifier (BOC) is the most accurate possible classifier for a given problem. It makes predictions by considering all possible hypotheses and their probabilities to minimize classification error. It is an ideal benchmark for machine learning models but is often impractical due to its high computational cost.

How It Works

1. It evaluates all possible hypotheses rather than selecting just one.
2. It weighs each hypothesis based on its probability.
3. It combines these probabilities to make the most accurate prediction.
4. It achieves the lowest possible error rate, making it the best theoretical classifier.

Example: Imagine deciding whether to carry an umbrella based on multiple factors such as the weather forecast, season, cloud cover, and humidity. BOC would analyze all

possible weather data and past patterns to make the most accurate prediction, considering every factor and its probability.

Comparison with Naïve Bayes

Aspect	**Bayes Optimal Classifier**	**Naïve Bayes**
Assumptions	Considers all possible hypotheses	Assumes feature independence
Complexity	Computationally expensive	Simple and efficient
Accuracy	Theoretically the best	Works well in many cases but may not always be optimal
Usage	Theoretical benchmark	Commonly used in real-world applications

Limitations

- Computationally expensive and impractical for large datasets.
- Used mainly as a benchmark rather than a real-world classifier.

Naïve Bayes Classifier

Naïve Bayes is a classification algorithm that predicts whether a data point belongs to class YES or NO based on multiple features. It assumes that all features are independent, which simplifies probability calculations.

We want to calculate:

$$P(YES|X_1, X_2, ..., X_n) \quad \text{and} \quad P(NO|X_1, X_2, ..., X_n)$$

where **X1, X2, ..., Xn** are the features (e.g., words in an email, symptoms of a disease, etc.).

Using Bayes' formula:

$$P(YES|X_1, X_2, ..., X_n) = \frac{P(X_1, X_2, ..., X_n|YES) \cdot P(YES)}{P(X_1, X_2, ..., X_n)}$$

$$P(NO|X_1, X_2, ..., X_n) = \frac{P(X_1, X_2, ..., X_n|NO) \cdot P(NO)}{P(X_1, X_2, ..., X_n)}$$

Since **P(X1, X2, ..., Xn)** is the same for both classes, we only compare:

$$P(X_1, X_2, ..., X_n|YES) \cdot P(YES) \quad \text{vs} \quad P(X_1, X_2, ..., X_n|NO) \cdot P(NO)$$

Whichever is **higher**, we assign that class.

EXAMPLE: Solve the Play Tennis dataset problem using Naïve Bayes, calculating the probability of playing tennis (YES) or not (NO) given that the outlook is Sunny and the temperature is Hot.

Outlook	**Temperature**	**Humidity**	**Wind**	**Play Tennis**
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

Step 1: Given Data

We want to compute:

P(Yes | Outlook=Sunny,Temperature=Hot)

P(No | Outlook=Sunny,Temperature=Hot)

Total instances: 14
Total Play Tennis = Yes: 9
Total Play Tennis = No: 5

Step 2: Prior Probabilities

Total samples = 14
P(Yes) = 9/14
P(No) = 5/14

Step 3: Probability Table for OUTLOOK

Outlook	P(Yes)	P(No)
Sunny	2/9	3/5
Overcast	4/9	0/5
Rain	3/9	2/5

Step 4: Probability Table for TEMPERATURE

Temperature	P(Yes)	P(No)
Hot	2/9	2/5
Mild	4/9	2/5
Cool	3/9	1/5

Step 5: Compute Naïve Bayes Probabilities

For Yes:

$$P(\text{Yes} \mid \text{Sunny,Hot}) \propto P(\text{Sunny} \mid \text{Yes}) \times P(\text{Hot} \mid \text{Yes}) \times P(\text{Yes})$$

$$\frac{2}{9} \times \frac{2}{9} \times \frac{9}{14}$$

$$= 0.2222 \times 0.2222 \times 0.6428$$

$$= 0.0317$$

For **No**:

$$P(\text{No} \mid \text{Sunny,Hot}) \propto P(\text{Sunny} \mid \text{No}) \times P(\text{Hot} \mid \text{No}) \times P(\text{No})$$

$$\frac{3}{5} \times \frac{2}{5} \times \frac{5}{14}$$

$$= 0.6 \times 0.4 \times 0.3571$$

$$= 0.0857$$

Since **0.0857 > 0.0317**, the final prediction is No (Tennis will NOT be played).

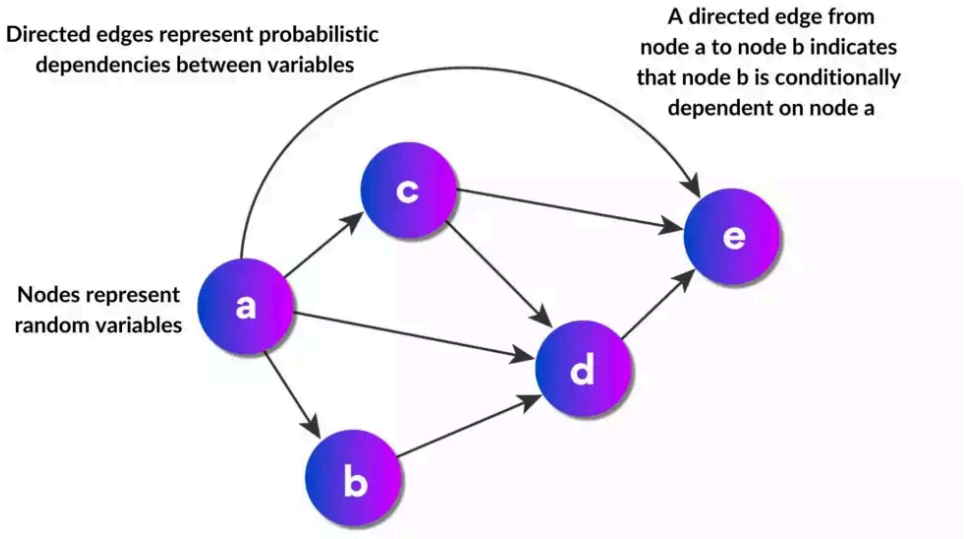
Bayesian Belief Network

A Bayesian Belief Network (BBN), also known as a Bayesian Network (BN), is a probabilistic graphical model that represents a set of variables and their conditional dependencies using a Directed Acyclic Graph (DAG). It is widely used in probabilistic reasoning, decision-making, and machine learning.

Key Components of a Bayesian Belief Network

1. Graphical Structure: The network's structure is represented by a directed acyclic graph (DAG) where nodes represent random variables, and directed edges represent probabilistic dependencies between variables. A directed edge from node A to node B indicates that node B is conditionally dependent on node A.

2. Conditional Probability Tables (CPTs): Associated with each node in the graph are conditional probability tables, which specify the probability distribution of a node given its parent nodes in the graph. These tables quantify how the probability of a node's value changes based on the values of its parent nodes.



Expectation-Maximization (EM) Algorithm - Notes

The Expectation-Maximization (EM) Algorithm is an iterative method used to estimate parameters in statistical models when some data is missing or hidden (latent variables). It is commonly used in clustering, density estimation, and probabilistic models like Gaussian Mixture Models (GMMs).

The EM algorithm is useful for estimating probabilities in models with hidden data, but it can be computationally expensive and may require multiple runs with different initial values to find a good solution.

How It Works

The EM algorithm consists of two main steps, repeated until convergence:

1. **Expectation Step (E-Step)**
    - Estimates the missing/hidden data using current parameter values.
    - Computes the expected likelihood of the observed data.
2. **Maximization Step (M-Step)**
    - Updates the model parameters to maximize the expected likelihood found in the E-step.
    - Ensures parameters improve in each iteration.

How EM Works in Fraud Detection:

1. **E-Step:** Based on current estimates, calculate the probability of each transaction being fraudulent or legitimate. This is done using available transaction features like amount, location, frequency, and user behavior.
2. **M-Step:** Update the model parameters (e.g., fraud likelihood thresholds, transaction risk scores) to maximize the probability of correctly classifying fraudulent transactions. Adjust based on patterns detected in known fraud cases.
3. **Iterations:** Repeat the E-step and M-step until fraud detection stabilizes and the classification is refined.

Applications

- Clustering (e.g., Gaussian Mixture Models)
- Missing Data Imputation
- Natural Language Processing (NLP)
- Medical Diagnosis

Advantages

- Handles missing/incomplete data well.
- Works for complex probabilistic models.

Limitations

- Slow convergence in some cases.
- Requires careful initialization for good results.

Support Vector Machine

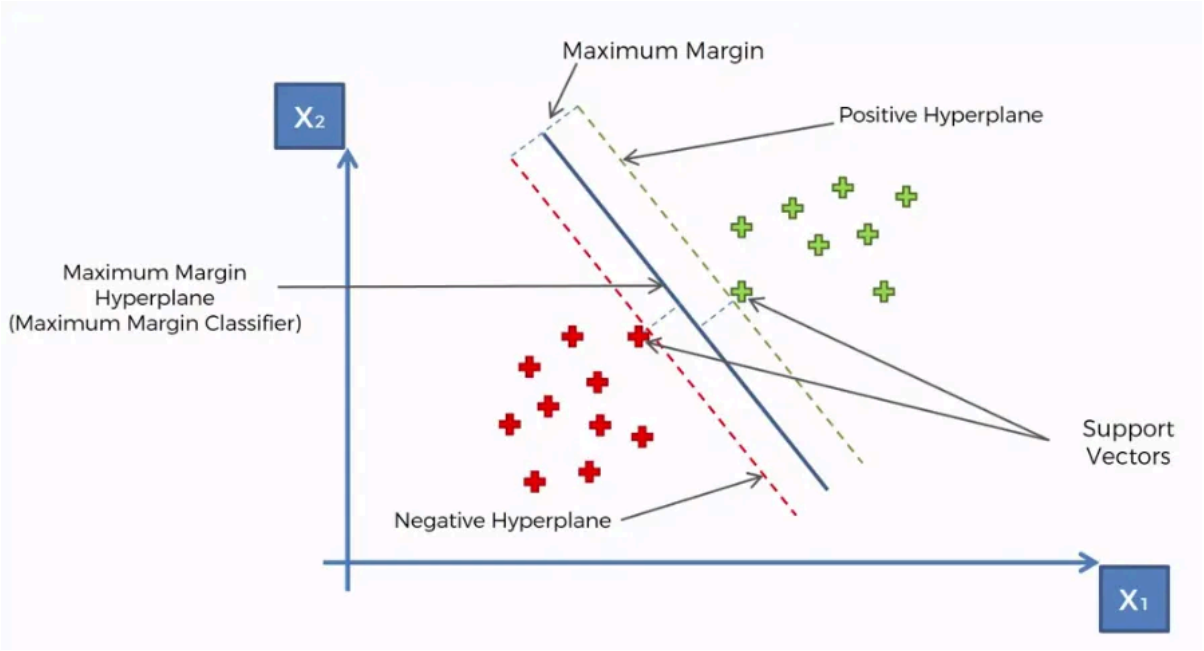
- SVM is a supervised machine learning algorithm used for both classification and regression tasks.
- It became popular in the late 1990s and is widely used for classification problems.
- In classification, SVM finds an optimal hyper-plane that separates data into different classes.
- In regression, it fits a line or curve to data points.

SVM for Classification

- It is a discriminative classifier that finds the best separating hyper-plane.
- The hyper-plane divides the space into two parts where each class belongs to one side.
- If data is linearly separable, SVM finds a straight line (in 2D) or a plane (in higher dimensions) to classify the data.
- If data is not linearly separable, SVM uses kernel tricks to find a suitable decision boundary.

Geometric Intuition of SVM

- **Hyperplane**: A decision boundary separating different classes in feature space, represented by the equation **wx + b = 0** in linear classification. The optimal hyper-plane is the plane that maximizes the distance between the closest data points of different classes.
- **Support Vectors**: The closest data points to the hyperplane, crucial for determining the hyperplane and margin in SVM.
- **Margin**: The distance between the hyperplane and the support vectors. SVM aims to maximize this margin for better classification performance.



- **Positive Hyper-Plane**: The hyper-plane touching the positive class support vectors.
- **Negative Hyper-Plane**: The hyper-plane touching the negative class support vectors.
- **Kernel**: A function that maps data to a higher-dimensional space, enabling SVM to handle non-linearly separable data.
- **Hard Margin**: A maximum-margin hyperplane that perfectly separates the data without misclassifications.
- **Soft Margin**: Allows some misclassifications by introducing slack variables, balancing margin maximization and misclassification penalties when data is not perfectly separable.

Classification Condition in SVM

For a given dataset with feature vectors xi and corresponding class labels $y_i \in \{-1,1\}$, the decision function of an SVM is:

$f(x)=w \cdot x+b$

where:

- w is the weight vector (normal to the hyper-plane),
- x is the input feature vector,
- b is the bias term.

This condition ensures that:

- If $y_i=1$ (positive class), then $w \cdot x_i+b \geq 1$
- If $y_i=-1$ (negative class), then $w \cdot x_i+b \leq -1$

A data point x_i is correctly classified if it satisfies: $y_i(w \cdot x_i+b) \geq 1$

Kernels in Support Vector Machine (SVM)

In Support Vector Machines (SVMs), the goal is to find a hyperplane that best separates different classes of data. However, when data is not linearly separable, we use kernels to transform it into a higher-dimensional space where it becomes separable.

A kernel function is a mathematical function that transforms input data into a higher-dimensional space without explicitly computing the transformation.

Types of Kernels in SVM

1. Linear Kernel: Used when data is linearly separable. It computes the dot product between feature vectors.

- **Formula:** $K(x,y)=x \cdot y$
- When data is already linearly separable.

If K(x,y) is **large**, the vectors are similar.

If K(x,y) is **small**, the vectors are dissimilar.

2. Polynomial Kernel: Used when data has a nonlinear relationship. It maps input data into a higher polynomial feature space.

- **Formula:**

$$K(x, y) = (x \cdot y + c)^d$$

  Where:
  - c is a constant (controls bias).
  - d is the polynomial degree (controls complexity).
- Use it when the data follows a polynomial relationship or when linear separation is insufficient, requiring an additional feature space.

The large value means the kernel has magnified the similarity, allowing better separation in high-dimensional space.

3. Gaussian Kernel : The most commonly used kernel for non-linear problems. It maps data into infinite-dimensional space, making it very powerful.

- **Formula:**

$$K(x, y) = \exp\left(-\frac{\|x - y\|^2}{2\sigma^2}\right)$$

Where:

- // x-y // 2 is the squared Euclidean distance between points.
- σ (sigma) controls the spread of the decision boundary.
- If x and y are close, K(x,y) is large.
- If x and y are far, K(x,y) is small.
- Use it when the data is highly complex and not linearly separable or when flexible decision boundaries are required.

Since the value is very small, it means that x and y are not similar.

How to Choose the Right Kernel

Kernel	Best for
Linear Kernel	Linearly separable data, text classification
Polynomial Kernel	Polynomially separable data
Gaussian Kernel	Complex, non-linear data

Advantages of SVM

- Works well for both linear and non-linear classification.
- Effective in high-dimensional spaces.
- Robust to overfitting, especially in high-dimensional data.

Disadvantages of SVM

- Computationally expensive for large datasets.
- Choosing the right kernel function is crucial.
- Does not perform well when there is a lot of noise in data.

Applications of SVM

- **Text classification** (e.g., spam filtering, sentiment analysis).
- **Image classification** (e.g., handwriting recognition, face detection).
- **Medical diagnosis** (e.g., cancer detection).
- **Financial analysis** (e.g., stock price prediction).