CSE 403: Machine Learning

Chapter 4: Classifications

Md Atikuzzaman
Lecturer
Department of Computer Science & Engineering
atik@cse.green.edu.bd

GREEN UNIVERSITY OF BANGLADESH

Outline

1. Motivation
2. Classification
3. K-Nearest Neighbors
4. Support Vector Machines (SVM)
5. Kernel Functions
6. Naive Bayes Classifier
7. References

The Power of Classification: Real-World Scenarios

"We make categorical decisions every second. How do machines do it?"

1. The Morning Inbox

Spam vs. Ham

You wake up to 50 emails. Your "Inbox" has work updates, while the "Junk" folder trapped a phishing scam.

The Question:
How did the server distinguish a scam from a salary slip?

2. The Diagnosis

Benign vs. Malignant

A radiologist uploads an MRI scan. An AI system instantly flags a small region as "High Risk" for a tumor.

The Question:
How can pixels on a screen predict a life-threatening disease?

3. The Commute

Stop vs. Go

A self-driving car sees a shape in the fog. It must decide in milliseconds: is it a **Person** or a **Plastic Bag**?

The Question:
How do we teach a car to value safety over speed?

$\rightarrow$ **This is Supervised Classification.**

Introduction to Classification

Definition

Classification is a supervised learning task where the objective is to approximate a mapping function $f : \mathbf{x} \rightarrow y$ from input variables $\mathbf{x}$ to discrete output variables y.

Mathematical Notation:

- **Input Features:** $\mathbf{x} \in \mathbb{R}^n$ (Feature vector, e.g., pixel values, financial metrics).
- **Target Label:** y (Categorical class label).
- **Dataset:** $\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}), \ldots, (\mathbf{x}^{(m)}, y^{(m)})\}$.

Binary Classification

$y \in \{0, 1\}$ or $\{-1, +1\}$

- Spam vs. Non-Spam
- Fraud vs. Legitimate Transaction

Multi-Class Classification

$y \in \{0, 1, \ldots, K\}$

- Handwritten Digits (0-9)
- Medical Diagnosis (Benign, Stage I, Stage II)

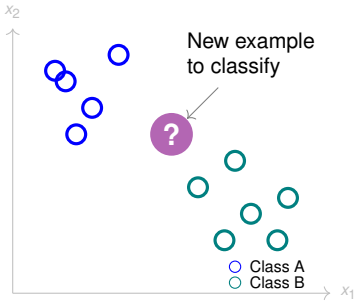
K-Nearest Neighbors

The Concept

KNN is a **non-parametric, lazy learning** algorithm, which uses proximity to make classifications or predictions about the grouping of an individual data point.

The Scenario:

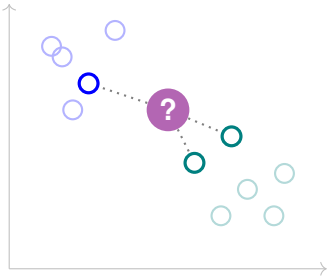
- We have existing data clusters (Class A and Class B).
- A **new, unknown example** arrives (the purple point).
- *Goal:* To which group does it belong?



K-Nearest Neighbors

Step 1: Choose K
We select the number of neighbors, e.g., $K = 3$.

Step 2: Calculate Distances
We compute the distance between the query point **x** and *every* other point in the dataset using **Distance Metric**. The algorithm then identifies the K points with the smallest distance values.



Distance Metrics in K-Nearest Neighbors

Choosing the Right Measure of Similarity

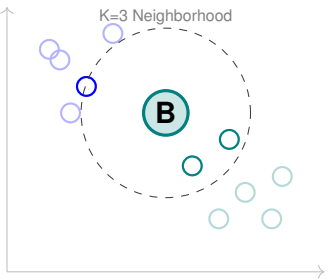
Metric Name	Equation	Best For...
Euclidean (L_2)	$d = \sqrt{\sum(x_i - y_i)^2}$	Standard, continuous numerical data (e.g., physical distances).
Manhattan (L_1)	$d = \sum \lvert x_i - y_i \rvert$	High-dimensional data or grid-based paths (e.g., city blocks).
Minkowski (L_p)	$d = (\sum \lvert x_i - y_i \rvert^p)^{1/p}$	Generalization where p is tunable ($p = 1 \rightarrow L_1$, $p = 2 \rightarrow L_2$).
Hamming	$d = \sum \mathbb{I}(x_i \neq y_i)$	Categorical variables, binary strings (e.g., genetic sequences).
Cosine Distance	$d = 1 - \frac{\mathbf{x} \cdot \mathbf{y}}{\lvert\lvert\mathbf{x}\rvert\rvert \lvert\lvert\mathbf{y}\rvert\rvert}$	Text data, NLP, Recommender Systems (measures angle, not magnitude).

K-Nearest Neighbors

Step 3: Majority Vote
We count the classes of the K nearest neighbors.

- **2 Neighbors** are Class B
- **1 Neighbor** is Class A

Result: The new point is classified as **Class B**.



Key Hyperparameters:

- **Value of K:** Odd numbers (1, 3, 5) are preferred to avoid ties in binary classification.
- **Scaling:** Must normalize features so one dimension doesn't dominate the distance metric.

Pros and Cons of K-Nearest Neighbors

Advantages (Pros)

- **Simple & Intuitive:** easy to understand and implement; few hyperparameters (just K and distance metric).
- **No Training Phase:** It is a **lazy learner** training is instant because it just stores the data.
- **Non-Parametric:** Makes no assumptions about the underlying data distribution (good for non-linear boundaries).
- **Adapts Easily:** Instantly incorporates new data without retraining.

Disadvantages (Cons)

- **Computationally Expensive:** Prediction is slow because it must calculate distance to *every* training point.
- **Memory Intensive:** Must store the entire dataset in RAM.
- **Sensitive to Scale:** Features with large ranges dominate distance calculations (requires normalization).
- **Curse of Dimensionality:** Performance degrades rapidly as the number of features increases.

Best for small datasets with few features; struggles with "Big Data."

K-Nearest Neighbors Summary

- **Type:** Supervised learning method used for both regression and classification
- **Model Family:** Non-parametric approach based on consensus among the K nearest training instances
- **Objective Function:** Distance-based metrics such as Euclidean, Minkowski, and Hamming distance
- **Optimizer:** None at training. Prediction via nearest-neighbor search using efficient data structures (e.g., KD-trees, hashing)
- **Probabilistic View:** Local neighborhood estimation of the conditional distribution $P_{\text{data}}(y \mid x)$

Linear SVM: Formal Definitions

The Problem Setup

Given a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$ where $y_i \in \{-1, +1\}$.
We seek a separating hyperplane $\mathcal{H}$ defined by:

$$w^T x + b = 0$$

Geometric Margin (r_i):
The perpendicular Euclidean distance from a point x_i to the hyperplane:

$$r_i = \frac{y_i(w^T x_i + b)}{||w||}$$

Functional Margin Constraint:
We enforce that for all points:

$$y_i(w^T x_i + b) \geq 1$$

This implies the closest points (Support Vectors) satisfy equality.

Linear SVM: Geometric Visualization

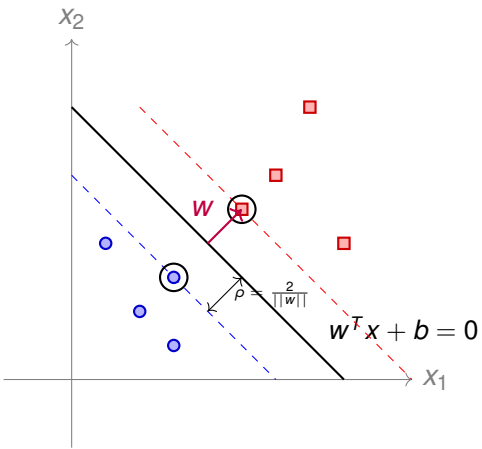
Margin Optimization:

- The total margin width is $\rho = \frac{2}{||w||}$.
- **Goal:** Maximize $\rho \implies$ Minimize $||w||$.

Canonical Optimization

$$\min_{w} \frac{1}{2}||w||^2$$

(Convex Quadratic Programming)



Soft Margin SVM (Primal Formulation)

Real-world data is rarely linearly separable. We relax the constraints using **Slack Variables** $\xi_i \geq 0$ to allow misclassifications.

Primal Objective with Hinge Loss

$$\min_{w,b,\xi} \underbrace{\frac{1}{2}||w||^2}_{\text{Regularization (L2)}} + \underbrace{C\sum_{i=1}^{N}\xi_i}_{\text{Empirical Risk}}$$

Constraints (KKT Conditions):

$$y_i(w^T x_i + b) \geq 1 - \xi_i, \quad \forall i$$
$$\xi_i \geq 0, \quad \forall i$$

This is equivalent to minimizing Hinge Loss: $\max(0, 1 - y_i f(x_i))$.

Hyperparameter C Analysis:

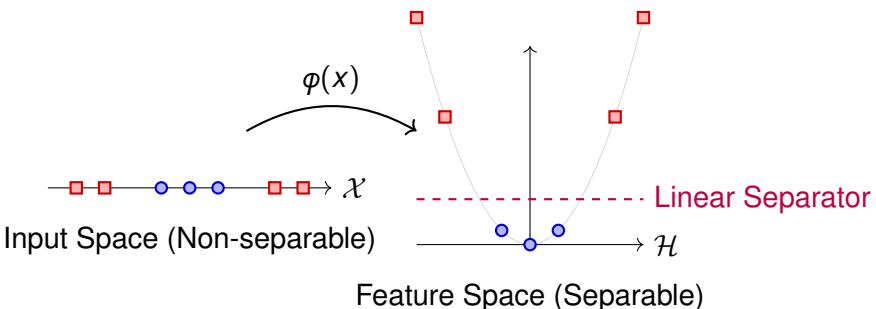
- $C \to \infty$**:** Hard Margin. High variance, potential overfitting.
- $C \to 0$**:** Large Soft Margin. High bias, potential underfitting.

The Kernel Trick & Mercer's Theorem

Motivation: To classify non-linear data, we project input space $\mathcal{X}$ to a high-dimensional feature space $\mathcal{H}$ via $\varphi : \mathcal{X} \to \mathcal{H}$.

The Kernel Trick: If a function K satisfies *Mercer's Condition* (positive semi-definite), we can compute the dot product in $\mathcal{H}$ without explicit mapping:

$$K(x_i, x_j) = \langle \varphi(x_i), \varphi(x_j) \rangle_{\mathcal{H}}$$



Common Kernel Functions

The choice of Kernel defines the notion of similarity in the high-dimensional space.

Kernel	Mathematical Form	Properties & Use Cases
Linear	$x_i^T x_j$	Text classification (high dim features). Fast, less prone to overfitting.
Polynomial	$(\gamma x_i^T x_j + r)^d$	Computer Vision. Interactions between features (degree d).
RBF (Gaussian)	$\exp(-\gamma \|x_i - x_j\|^2)$	**Universal Approximator**. Maps to infinite-dimensional space. Default choice.
Sigmoid	$\tanh(\gamma x_i^T x_j + r)$	Proxy for Neural Networks (Multilayer Perceptron).

1. Linear Kernel

Concept:

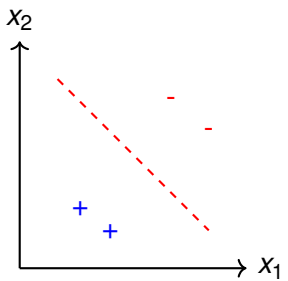
- The simplest kernel function.
- Does **not** map data to a higher dimensional space (effectively $K(x, y) = \Phi(x)^T \Phi(y)$ where $\Phi(x) = x$).
- Creates a linear decision boundary (a straight line or hyperplane).

Best Used When:

- The number of features is very large (e.g., Text Classification, Bag-of-Words).
- Speed is critical (very fast training/inference).
- Data is already linearly separable.

Mathematical Form

$$K(x_i, x_j) = x_i^T x_j + c$$



2. Polynomial Kernel

Concept:

- Maps data into a space of feature interactions.
- Allows for curved decision boundaries.
- Considers combinations of features (e.g., x_1^2, $x_1 x_2$).

Hyperparameters:

- **Degree (d):** Controls complexity. High d = complex curves (risk of overfitting).
- **Coef0 (r):** Controls how much the model is influenced by high-degree vs. low-degree polynomials.

Mathematical Form

$$K(x_i, x_j) = (\gamma x_i^T x_j + r)^d$$

Example ($d = 2$):
Maps 2D inputs (x_1, x_2) to:
$(x_1^2, x_2^2, \sqrt{2}x_1x_2, \sqrt{2}x_1, \dots)$

3. Radial Basis Function (RBF) Kernel

Concept:

- Also known as the **Gaussian Kernel**.
- Implicitly maps data to an **infinite-dimensional** space (via Taylor Series expansion).
- Similarity decays with Euclidean distance: if $\|x_i - x_j\|$ is large, $K \to 0$.

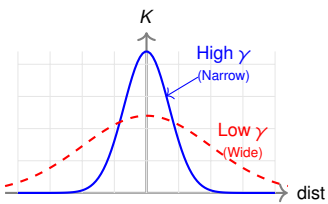
The Gamma (γ) Parameter:

- **High γ:** Narrow peak. Model is sensitive to local noise.
  Risk: Overfitting.
- **Low γ:** Wide peak. Model is smoother.
  Risk: Underfitting.

Mathematical Form

$$K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2)$$

Visualizing γ Influence



4. Sigmoid Kernel

Concept:

- Derived from Neural Networks.
- Mimics a two-layer Perceptron (MLP) using the kernel trick.
- The activation function is the Hyperbolic Tangent (tanh).

Caveats:

- It is **not** positive semi-definite (PSD) for all parameters, which technically violates Mercer's condition (though often works in practice).
- Generally used less often than RBF or Linear today.

Mathematical Form

$$K(x_i, x_j) = \tanh(\gamma x_i^T x_j + r)$$

Interpretation:
"Support Vector Neural Network"

Naive Bayes Classifier

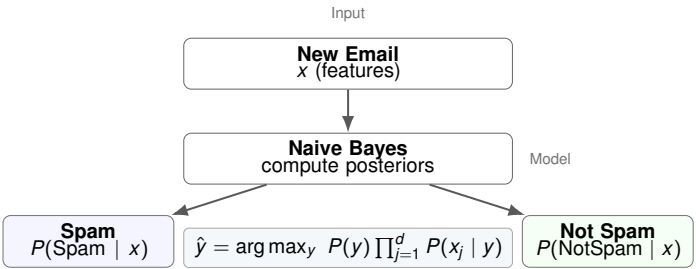
The Concept

Naive Bayes is a **probabilistic, generative** classifier using Bayes theorem, assuming features are conditionally independent given the class.

Scenario: Spam Detection

- Features: words in an email
- Predict: Spam vs Not Spam
- Compute: $P(\text{Spam} \mid x)$ and $P(\text{NotSpam} \mid x)$

Very fast in training and real-time prediction. Strong baseline for text.



Motivation: Why Naive Bayes?

The Need for Speed & Baselines

- Real-world problems often require **fast**, interpretable, and **strong** baselines.
- **Key Applications:**
  - Text classification (Spam, Sentiment Analysis)
  - Medical triage (Probabilistic diagnosis)
  - High-dimensional sparse data (TF-IDF)

Ideal Model Properties

- Learns from **small data**.
- Provides **posterior probabilities** $P(y|\mathbf{x})$.
- $\mathcal{O}(N)$ training/prediction.

Core Idea: Combine *Bayes' Rule* with a simplifying *Independence Assumption*.

Naive Bayes: Bayes Rule + Independence

Bayes theorem

$$P(y \mid \mathbf{x}) = \frac{P(\mathbf{x} \mid y)\, P(y)}{P(\mathbf{x})} \quad \Rightarrow \quad \boxed{P(y \mid \mathbf{x}) \propto P(\mathbf{x} \mid y)\, P(y)}$$

Naive conditional independence

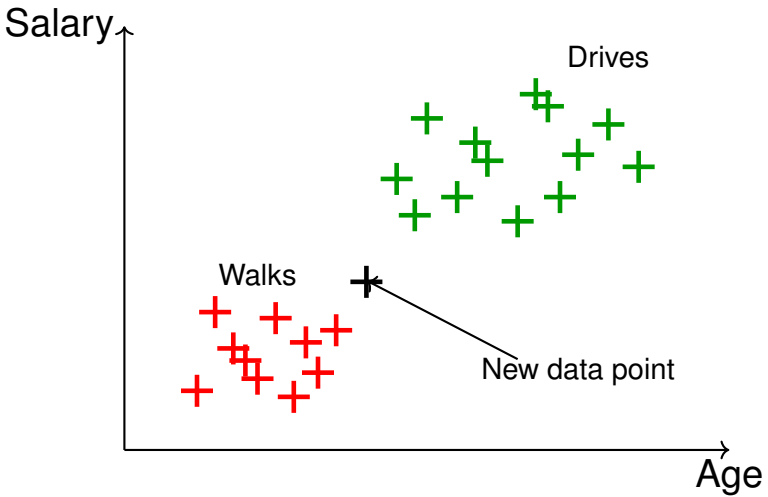
For $x = [x_1, \ldots, x_d]$,

$$P(x \mid y) = \prod_{j=1}^{d} P(x_j \mid y)$$

- $P(y)$: prior, $P(x \mid y)$: likelihood, $P(y \mid x)$: posterior

We compare unnormalized scores; $P(x)$ cancels for all classes.

Naive Bayes Classification Example



Naive Bayes Classification Example

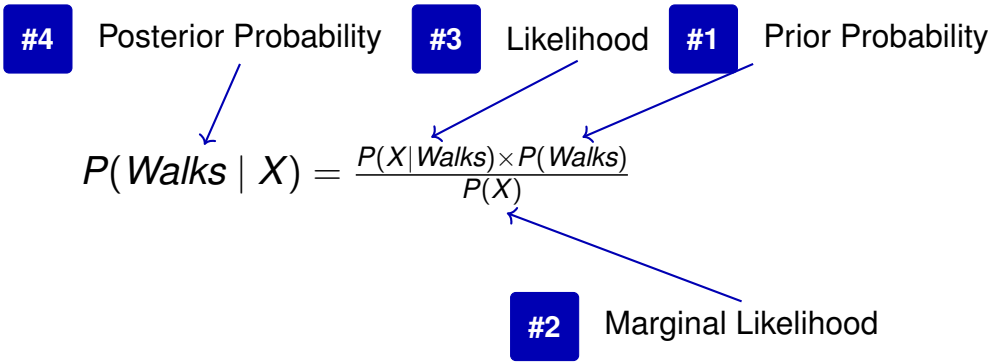
Probability walks given Age & Salary

$$P(\text{Walks} \mid X) = \frac{P(X \mid \text{Walks}) \times P(\text{Walks})}{P(X)}$$

Where:

- X represents the features.
- In this example, $X =$ **(Age, Salary)**.

Naive Bayes Classification Example



References

1. C. M. Bishop, *Pattern Recognition and Machine Learning*, 1st Ed., Springer, 2006.
2. R. O. Duda, P. E. Hart, D. G. Stork, *Pattern Classification*, 2nd Ed., Wiley-Interscience, 2000.
3. I. Goodfellow, Y. Bengio, A. Courville, *Deep Learning*, 1st Ed., The MIT Press, 2016.
4. A. Zhang, Z. C. Lipton, M. Li, A. J. Smola, *Dive into Deep Learning*, 1st Ed., Cambridge University Press, 2023.