Machine Learning Algorithm: Naive Bayes Classifier

Bayes' theorem, independence assumption, concrete examples

K. Kadri

Outline

Problem: predicting a class

- We want to predict a label $y \in \{0, 1\}$ or multiple classes.
- Example:
  - Email: **spam / not spam**
  - Medical test: **disease / no disease**
- We need a model that uses available features to estimate:

$$p(y \mid \mathrm{x})$$

Key idea

Naive Bayes predicts the class with the highest **posterior probability** based on Bayes' theorem.

Bayes' Theorem

Definition

$$p(y \mid \mathrm{x}) = \frac{p(\mathrm{x} \mid y)\, p(y)}{p(\mathrm{x})}$$

- $p(y)$: prior probability of the class
- $p(\mathrm{x} \mid y)$: likelihood
- $p(y \mid \mathrm{x})$: posterior probability

Goal

Choose the class with the highest posterior:

$$\hat{y} = \arg\max_{y}\ p(y \mid \mathrm{x})$$

Naive assumption

- Exact computation of $p(\mathrm{x} \mid y)$ is hard:

$$\mathrm{x} = (x_1, x_2, \ldots, x_n)$$

- Naive Bayes assumes:

$$p(\mathrm{x} \mid y) = \prod_{j=1}^{n} p(x_j \mid y)$$

Meaning

Features are assumed **conditionally independent** given the class.

Resulting classifier

$$p(y \mid \mathrm{x}) \propto p(y) \prod_{j=1}^{n} p(x_j \mid y)$$

Example: spam vs. not spam

Suppose we observe the words:

$$\text{“free”, “money”}$$

- We estimate from training emails:
  - $p(\text{spam}) = 0.3$
  - $p(\text{not spam}) = 0.7$
  - $p(\text{“free”} \mid \text{spam}) = 0.8$
  - $p(\text{“money”} \mid \text{spam}) = 0.6$
  - $p(\text{“free”} \mid \text{not spam}) = 0.1$
  - $p(\text{“money”} \mid \text{not spam}) = 0.05$

$$p(\text{spam} \mid \text{x}) \propto 0.3 \times 0.8 \times 0.6 = 0.144$$

$$p(\text{not spam} \mid \text{x}) \propto 0.7 \times 0.1 \times 0.05 = 0.0035$$

Prediction

Email classified as **spam**.

Training Naive Bayes

Training step

- Estimate class prior:

$$p(y) = \frac{\text{count}(y)}{\text{total samples}}$$

- Estimate likelihoods:

$$p(x_j \mid y) = \frac{\text{count}(x_j, y)}{\text{count}(y)}$$

Efficiency

Training is extremely fast:

- no gradient descent
- no matrix inversion
- scalable to large datasets

Prediction Algorithm

For a new sample x

Compute:

$$p(y)\prod_{j=1}^{n} p(x_j \mid y)$$

$$\hat{y} = \arg\max_{y} p(y)\prod_{j=1}^{n} p(x_j \mid y)$$

Decision rule

Choose the class with the highest posterior probability.

Problem: zero probability

- If a word never appears in spam,

$$p(\text{word} \mid \text{spam}) = 0$$

- Then:

$$p(\text{spam} \mid \text{x}) = 0$$

- One word can ruin classification.

Solution: Laplace smoothing

$$p(x_j \mid y) = \frac{\text{count}(x_j, y) + 1}{\text{count}(y) + K}$$

where K = number of possible values.

Advantages

- Extremely fast training
- Works well with text data
- Probabilistic output
- Robust with few samples

Typical use cases

- Spam detection
- Sentiment analysis
- Medical diagnosis
- Document classification

Limits and Best Practices

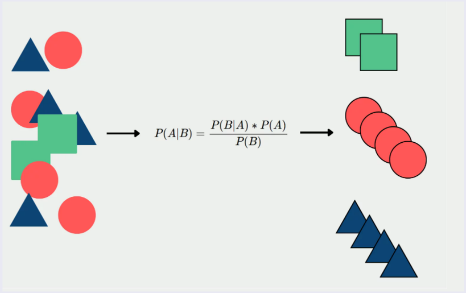
- Independence assumption often unrealistic
- Performs poorly with correlated features
- Continuous features require distributions (Gaussian NB)

Best practices

- Use TF-IDF for text
- Apply Laplace smoothing
- Compare with logistic regression / SVM

From Features → Bayes Theorem → Prediction

Flow of the Naive Bayes Classifier



Interpretation

The classifier transforms raw features into class probabilities using Bayes' theorem, then selects the most likely class.

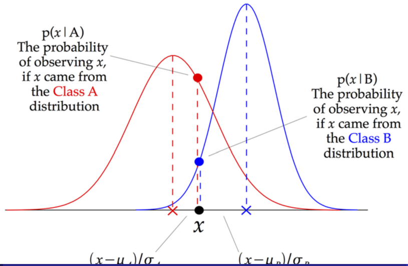
Gaussian Naive Bayes: Intuition

Idea

For continuous features, we assume:

$$x_j \mid y \sim \mathcal{N}(\mu_{y,j}, \sigma^2_{y,j})$$

- Each class models features as bell-shaped Gaussian curves
- Classes differ by mean and variance
- Probability estimated from the curve height


p(x | A)
The probability of observing x, if x came from the Class A distribution
p(x | B)
The probability of observing x, if x came from the Class B distribution
x
(x−μ_A)/σ_A
(x−μ_B)/σ_B

Naive Bayes vs Logistic Regression

Naive Bayes

- Generative model
- Learns: $p(x \mid y)$ and $p(y)$
- Fast, works well with text
- Strong independence assumption

Logistic Regression

- Discriminative model
- Learns: $p(y \mid x)$ directly
- More flexible
- Needs optimization and feature scaling

Naive Bayes vs Logistic Regression

Rule of thumb

Naive Bayes performs surprisingly well with sparse text features, while Logistic Regression usually wins when independence is violated.