Lecture 4.1: Classification: linear decision boundary

谢丹
清华大学数学系

October 12, 2025

Classification Problem

- **Goal**: Assign input data $\mathbf{x}$ to one of K classes
- **Input**: Feature vector $\mathbf{x} \in \mathbb{R}^D$
- **Output**: Class label $y \in \{1, 2, \ldots, K\}$
- **Approaches**:
  - Discriminant functions
  - Generative models
  - Discriminative models (e.g., Logistic Regression)

Discriminant Functions

Definition

A function $f_k(\mathbf{x})$ for each class k that directly maps input $\mathbf{x}$ to class assignments:

$$y = \arg\max_k f_k(\mathbf{x})$$

Linear Discriminant:

$$f_k(\mathbf{x}) = \mathbf{w}_k^T \mathbf{x} + b_k$$

Non-linear Discriminant:

$$f_k(\mathbf{x}) = \phi(\mathbf{w}_k^T \mathbf{x} + b_k)$$

Key Property

Directly models decision boundaries without estimating probability distributions

Linear Discriminant Function Formulation I

Basic Form

For a linear discriminant function:

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$$

where:

- $\mathbf{w}$: weight vector
- b: bias term
- $\mathbf{x}$: input feature vector

Classification Rule

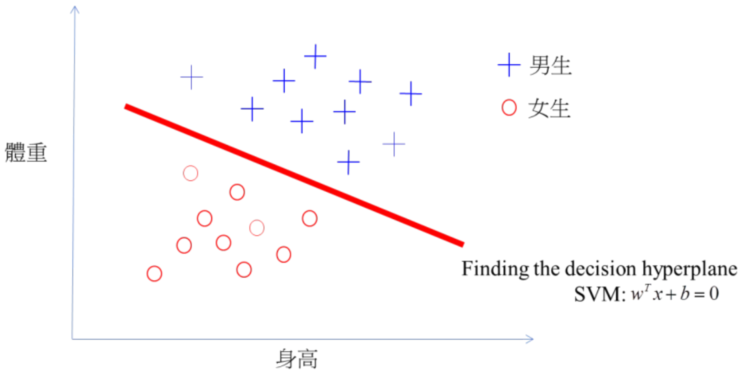
$$y = \begin{cases} +1 & \text{if } f(\mathbf{x}) \geq 0 \\ -1 & \text{if } f(\mathbf{x}) < 0 \end{cases}$$

Linear Discriminant Function Formulation II

Multi-class Extension

$$f_k(\mathbf{x}) = \mathbf{w}_k^T \mathbf{x} + b_k, \quad y = \arg\max_k f_k(\mathbf{x})$$

Namely, the assignment for the class is given by the function with maximal value.



男生
女生
體重
身高
Finding the decision hyperplane
SVM: $w^T x + b = 0$

Method 1: Least Squares Approach I

Objective Function

Minimize sum of squared errors:

$$\boxed{J(\mathbf{w}) = \sum_{i=1}^{N} (y_i - (\mathbf{w}^T \mathbf{x}_i + b))^2}$$

Matrix Formulation

$$J(\mathbf{w}) = (\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w})$$

Closed-form Solution

Method 1: Least Squares Approach II

$$\mathbf{w} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

- Requires $\mathbf{X}^T\mathbf{X}$ to be invertible
- Sensitive to outliers
- Computationally efficient for small datasets

One can also consider error function by adding regularization terms $\lambda \sum w_i^2$. (Ridge classification)

Method 2: Perceptron Loss Function I

The driving force behind perceptron learning

Definition

The perceptron uses a **hinge loss** function defined as:

$$\text{Perceptron loss :} \quad \boxed{L(\mathbf{w}) = \sum_{i \in M} -y_i(\mathbf{w} \cdot \mathbf{x}_i + b)}$$

where:

- M: set of misclassified samples (prediction is not the same as the true observed)
- $y_i \in \{-1, +1\}$: true label
- $\mathbf{w}$: weight vector
- b: bias term
- $\mathbf{x}_i$: input features

Method 2: Perceptron Loss Function II

The driving force behind perceptron learning

Key Properties

- **Convex**: Guarantees convergence
- **Piecewise linear**: Simple gradients
- **Zero for correct classifications**
- **Positive for misclassifications**

Gradient

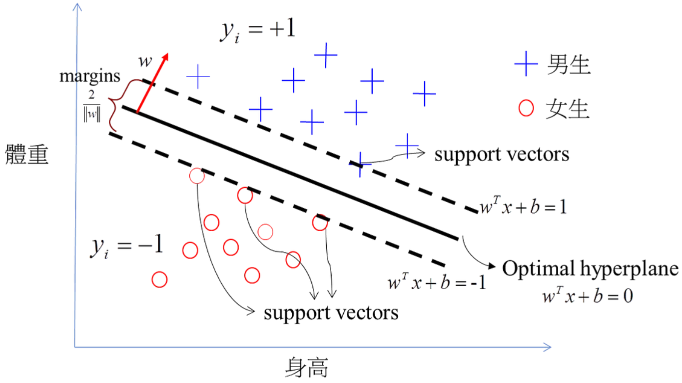
$$\nabla L(\mathbf{w}) = \sum_{i \in M} -y_i \mathbf{x}_i$$

$$\frac{\partial L}{\partial b} = \sum_{i \in M} -y_i$$

Leads to the update rule:

$$\mathbf{w} \leftarrow \mathbf{w} + \eta y_i \mathbf{x}_i$$

Method 3: Support Vector Machines (SVM)



Transforming SVM into a Programming Problem I

From geometric intuition to optimization formulation

Original Geometric Problem

Maximize the margin: $\max \frac{2}{\|\mathbf{w}\|}$

Subject to: $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1$ for all i (linear separable)

Step 1: Equivalent Reformulation

Instead of maximizing $\frac{2}{\|\mathbf{w}\|}$, minimize $\|\mathbf{w}\|$:

$$\min \|\mathbf{w}\| \quad \text{subject to} \quad y_i(\mathbf{w} \cdot \mathbf{w}_i + b) \geq 1$$

Step 2: Convex Optimization Form

Transforming SVM into a Programming Problem II

From geometric intuition to optimization formulation

For computational convenience, use squared norm:

$$\min \frac{1}{2}\|\mathbf{w}\|^2 \quad \text{subject to} \quad y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1$$

- Convex objective function
- Linear constraints
- Quadratic Programming (QP) problem

Step 3: Primal QP Formulation

$$\min_{\mathbf{w},b} \frac{1}{2}\mathbf{w}^T\mathbf{w}$$

subject to:

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \quad i = 1, \ldots, n$$

Properties

- Convex objective
- Linear constraints
- Global optimum guaranteed

Transforming SVM into a Programming Problem III

From geometric intuition to optimization formulation

Step 4: Practical Implementation

- Use QP solvers (CVXOPT, MOSEK)
- Or specialized SVM libraries (LIBSVM, scikit-learn)
- Handle large datasets with optimization techniques

Soft-Margin SVM I

Soft-Margin SVM: Handling Noise and Overlap

Real data is rarely perfectly separable. We introduce slack variables ξ_i to allow misclassifications.

$$\min_{\mathbf{w},b,\boldsymbol{\xi}} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i$$

$$\text{subject to} \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0 \quad \forall i$$

Parameter C controls the trade-off between a large margin and classifying points correctly.

Hard-Margin SVM: Perfect Separation

The Ideal Case

Assume linearly separable data with labels $y_i \in \{-1, +1\}$

- Decision boundary:
  $w^T x + b = 0$
- Margin boundaries:
  $w^T x + b = \pm 1$
- Constraint:
  $y_i(w^T x_i + b) \geq 1$

Soft-Margin SVM: Handling Reality

The Problem with Hard-Margin

Real data is rarely perfectly separable!

Solution: Introduce slack variables $\xi_i \geq 0$

Relaxed Constraints

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

Slack Interpretation

- $\xi_i = 0$: Correct classification beyond margin
- $0 < \xi_i \leq 1$: Inside margin but correct side
- $\xi_i > 1$: Misclassified

Soft-Margin SVM Optimization

Primal Problem

$$\min_{w,b,\xi} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi_i$$

subject to:

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

- $\frac{1}{2}\|w\|^2$: Maximizes margin
- $\sum \xi_i$: Minimizes classification errors
- $C > 0$: Trade-off parameter

The Critical Observation

Constraints Tell Us Something

From the constraints:

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

and

$$\xi_i \geq 0$$

Optimization Insight

Since we're minimizing $\sum \xi_i$, at optimum:

$$\xi_i = \max\left(0, 1 - y_i(w^T x_i + b)\right)$$

The optimal slack is determined by the point's margin violation!

Substitution and Hinge Loss Emergence I

Substitute Optimal Slack

Replace ξ_i in the objective function:

$$\min_{w,b} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \max\left(0, 1 - y_i(w^T x_i + b)\right)$$

Define Hinge Loss

Let $f(x_i) = w^T x_i + b$, then:

$$L_{\text{hinge}}(y_i, f(x_i)) = \max(0, 1 - y_i f(x_i))$$

Final Form

Substitution and Hinge Loss Emergence II

$$\text{SVM loss : } \boxed{\min_{w,b} \lambda \|w\|^2 + \frac{1}{n}\sum_{i=1}^{n} \max(0, 1 - y_i f(x_i))}$$

where $\lambda = \frac{1}{2nC}$

So the soft SVM could be thought of as using Hinge loss plus a regularization term, and therefore a probability interpretation.

Multi-class Training Strategies

One-vs-Rest (OvR)

- Train K binary classifiers $f_k(x)$.
- Each separates one class from all others (Given one class and regard the other classes as another class)
- Final: $\arg\max_k f_k(\mathbf{x})$

One-vs-One (OvO)

- Train $\frac{K(K-1)}{2}$ classifiers $f_{ij}(x)$.
- Each separates one pair of classes
- Final: majority voting

Generative Models

Bayesian Approach

Model the joint distribution $p(\mathbf{x}, y)$ using:

$$p(y|\mathbf{x}) = \frac{p(\mathbf{x}|y)p(y)}{p(\mathbf{x})}$$

- **Class prior**: $p(y)$ - probability of each class
- **Class-conditional density**: $p(\mathbf{x}|y)$ - distribution of features given class
- **Posterior**: $p(y|\mathbf{x})$ - probability of class given features

Examples

- Linear Discriminant Analysis (LDA)
- Quadratic Discriminant Analysis (QDA)
- Naive Bayes classifiers

Linear Discriminant Analysis (LDA)

- Classes: $k = 1, 2, \ldots, K$
- Data: $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$ where $\mathbf{x}_i \in \mathbb{R}^p$
- Class labels: $y_i \in \{1, 2, \ldots, K\}$
- Goal: Estimate parameters of the LDA model using MLE

LDA Model Assumptions

Key Assumptions

1. **Class-conditional distributions are Gaussian:**

$$P(\mathbf{x}|y = k) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma})$$

2. **Shared covariance matrix:** All classes have same $\boldsymbol{\Sigma}$
3. **Class priors:** $P(y = k) = \pi_k$, with $\sum_{k=1}^{K} \pi_k = 1$

Parameters to Estimate

- Class means: $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$
- Common covariance: $\boldsymbol{\Sigma}$
- Class priors: $\pi_1, \ldots, \pi_K$

Complete Data Likelihood

Joint Probability

$$P(\mathbf{x}, y) = P(\mathbf{x}|y)P(y)$$

Complete Data Likelihood

$$\mathcal{L}(\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) = \prod_{i=1}^{N} P(\mathbf{x}_i|y_i)P(y_i)$$

Let $C_k = \{i : y_i = k\}$ and $N_k = |C_k|$, then:

$$\mathcal{L} = \prod_{k=1}^{K} \prod_{i \in C_k} \pi_k \cdot \mathcal{N}(\mathbf{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma})$$

Log-Likelihood Function

Taking Logarithms

$$\ell = \log \mathcal{L} = \sum_{k=1}^{K} \sum_{i \in C_k} \left[\log \pi_k + \log \mathcal{N}(\mathbf{x}_i | \boldsymbol{\mu}_k, \boldsymbol{\Sigma})\right]$$

Gaussian Log-Density

$$\log \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = -\frac{p}{2}\log(2\pi) - \frac{1}{2}\log|\boldsymbol{\Sigma}| - \frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})$$

Complete Log-Likelihood

$$\ell = \sum_{k=1}^{K} N_k \log \pi_k - \frac{Np}{2}\log(2\pi) - \frac{N}{2}\log|\boldsymbol{\Sigma}| - \frac{1}{2}\sum_{k=1}^{K}\sum_{i \in C_k}(\mathbf{x}_i - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_k)$$

MLE for Class Priors π_k I

Constrained Optimization

Maximize $\sum_{k=1}^{K} N_k \log \pi_k$ subject to $\sum_{k=1}^{K} \pi_k = 1$
Lagrangian:

$$\mathcal{L}_\pi = \sum_{k=1}^{K} N_k \log \pi_k + \lambda \left(1 - \sum_{k=1}^{K} \pi_k\right)$$

First Order Conditions

$$\frac{\partial \mathcal{L}_\pi}{\partial \pi_k} = \frac{N_k}{\pi_k} - \lambda = 0 \quad \Rightarrow \quad \pi_k = \frac{N_k}{\lambda}$$

$$\sum_{k=1}^{K} \pi_k = \sum_{k=1}^{K} \frac{N_k}{\lambda} = \frac{N}{\lambda} = 1 \quad \Rightarrow \quad \lambda = N$$

MLE for Class Priors π_k II

MLE Solution

$$\hat{\pi}_k = \frac{N_k}{N}$$

MLE for Class Means $\boldsymbol{\mu}_k$ I

Relevant Part of Log-Likelihood

$$\ell_{\boldsymbol{\mu}} = -\frac{1}{2} \sum_{k=1}^{K} \sum_{i \in C_k} (\mathbf{x}_i - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_k)$$

Taking Derivative

$$\frac{\partial \ell_{\boldsymbol{\mu}}}{\partial \boldsymbol{\mu}_k} = \sum_{i \in C_k} \boldsymbol{\Sigma}^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_k) = 0$$

Since $\boldsymbol{\Sigma}^{-1}$ is invertible:

$$\sum_{i \in C_k} (\mathbf{x}_i - \boldsymbol{\mu}_k) = 0 \quad \Rightarrow \quad N_k \boldsymbol{\mu}_k = \sum_{i \in C_k} \mathbf{x}_i$$

MLE for Class Means $\boldsymbol{\mu}_k$ II

MLE Solution

$$\hat{\boldsymbol{\mu}}_k = \frac{1}{N_k} \sum_{i \in C_k} \mathbf{x}_i$$

MLE for Common Covariance $\boldsymbol{\Sigma}$

Relevant Part of Log-Likelihood

$$\ell_{\boldsymbol{\Sigma}} = -\frac{N}{2}\log|\boldsymbol{\Sigma}| - \frac{1}{2}\sum_{k=1}^{K}\sum_{i\in C_k}(\mathbf{x}_i - \boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_k)$$

Matrix Trace Identity

Using $\mathbf{a}^T\mathbf{B}\mathbf{a} = \mathrm{tr}(\mathbf{B}\mathbf{a}\mathbf{a}^T)$:

$$\sum_{k=1}^{K}\sum_{i\in C_k}(\mathbf{x}_i - \boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_k) = \mathrm{tr}\left(\boldsymbol{\Sigma}^{-1}\mathbf{S}\right)$$

where $\mathbf{S} = \sum_{k=1}^{K}\sum_{i\in C_k}(\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^T$

Solving for Σ I

Simplified Objective

$$\ell_{\boldsymbol{\Sigma}} = -\frac{N}{2}\log|\boldsymbol{\Sigma}| - \frac{1}{2}\mathrm{tr}(\boldsymbol{\Sigma}^{-1}\mathbf{S})$$

Matrix Derivatives

$$\frac{\partial \log|\boldsymbol{\Sigma}|}{\partial \boldsymbol{\Sigma}} = \boldsymbol{\Sigma}^{-1}$$

$$\frac{\partial \mathrm{tr}(\boldsymbol{\Sigma}^{-1}\mathbf{S})}{\partial \boldsymbol{\Sigma}} = -\boldsymbol{\Sigma}^{-1}\mathbf{S}\boldsymbol{\Sigma}^{-1}$$

First Order Condition

Solving for $\boldsymbol{\Sigma}$ II

$$\frac{\partial \ell_{\boldsymbol{\Sigma}}}{\partial \boldsymbol{\Sigma}} = -\frac{N}{2}\boldsymbol{\Sigma}^{-1} + \frac{1}{2}\boldsymbol{\Sigma}^{-1}\mathbf{S}\boldsymbol{\Sigma}^{-1} = 0$$

Multiply by $2\boldsymbol{\Sigma}$ on left and right:

$$-N\boldsymbol{\Sigma} + \mathbf{S} = 0 \quad \Rightarrow \quad \boldsymbol{\Sigma} = \frac{1}{N}\mathbf{S}$$

Final LDA MLE Estimators

Complete Set of MLE Estimators

1. **Class priors:** $\hat{\pi}_k = \frac{N_k}{N}$
2. **Class means:** $\hat{\boldsymbol{\mu}}_k = \frac{1}{N_k} \sum_{i \in C_k} \mathbf{x}_i$
3. **Common covariance:**

$$\hat{\boldsymbol{\Sigma}} = \frac{1}{N} \sum_{k=1}^{K} \sum_{i \in C_k} (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)^T$$

Unbiased Version (Common Practice)

$$\hat{\boldsymbol{\Sigma}}_{\text{unbiased}} = \frac{1}{N-K} \sum_{k=1}^{K} \sum_{i \in C_k} (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)^T$$

LDA Classification Rule

Bayes Classifier with Estimated Parameters

$$\hat{y} = \arg\max_k \hat{\pi}_k \cdot \mathcal{N}(\mathbf{x}|\hat{\boldsymbol{\mu}}_k, \hat{\boldsymbol{\Sigma}})$$

Why "Linear"?

The discriminant functions $\delta_k(\mathbf{x})$ are linear in $\mathbf{x}$ due to shared $\boldsymbol{\Sigma}$:

$$\delta_k(\mathbf{x}) = \mathbf{x}^T \hat{\boldsymbol{\Sigma}}^{-1} \hat{\boldsymbol{\mu}}_k - \frac{1}{2} \hat{\boldsymbol{\mu}}_k^T \hat{\boldsymbol{\Sigma}}^{-1} \hat{\boldsymbol{\mu}}_k + \log \hat{\pi}_k$$

Quadratic Discriminant Analysis (QDA)

Assumptions

- Gaussian class-conditional densities
- Class-specific covariance matrices
- $p(\mathbf{x}|y = k) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$

$$\log p(y = k|\mathbf{x}) \propto -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1}(\mathbf{x} - \boldsymbol{\mu}_k) - \frac{1}{2}\log|\boldsymbol{\Sigma}_k| + \log \pi_k$$

Decision Boundary

Quadratic in $\mathbf{x}$ due to different covariance matrices:

$$(\mathbf{x}-\boldsymbol{\mu}_1)^T \boldsymbol{\Sigma}_1^{-1}(\mathbf{x}-\boldsymbol{\mu}_1)+\log|\boldsymbol{\Sigma}_1| = (\mathbf{x}-\boldsymbol{\mu}_2)^T \boldsymbol{\Sigma}_2^{-1}(\mathbf{x}-\boldsymbol{\mu}_2)+\log|\boldsymbol{\Sigma}_2|$$

What is Naive Bayes?

Definition

Naive Bayes is a probabilistic classification algorithm based on Bayes' theorem with a strong (naive) independence assumption between features.

- **Simple** yet powerful
- **Fast** training and prediction
- **Probabilistic** outputs
- Works well with high-dimensional data

Why "Naive"?

The Naive Assumption

Features are conditionally independent given the class label:

$$P(X_1, X_2, \ldots, X_d|Y) = P(X_1|Y) \cdot P(X_2|Y) \cdots P(X_d|Y)$$

Real-world Example

- Classifying emails as spam/ham
- Words like "free" and "money" may be correlated
- Naive Bayes assumes they're independent given "spam"
- Surprisingly, this often works well in practice!

Naive Bayes Probability Model

Complete Formula

For features $X_1, X_2, \dots, X_d$ and class Y:

$$P(Y|X_1, \dots, X_d) = \frac{P(Y)\prod_{j=1}^{d} P(X_j|Y)}{P(X_1, \dots, X_d)}$$

Classification Rule

We predict the class with highest probability:

$$\hat{y} = \arg\max_{y} P(y) \prod_{j=1}^{d} P(x_j|y)$$

- $P(X_1, \dots, X_d)$ is constant for all classes
- We can ignore it for comparison

Gaussian Naive Bayes

For Continuous Features

Assumes features follow normal distribution:

$$P(X_j|Y = y_k) = \frac{1}{\sqrt{2\pi\sigma_{jk}^2}} \exp\left(-\frac{(x_j - \mu_{jk})^2}{2\sigma_{jk}^2}\right)$$

Parameter Estimation

- $\mu_{jk} = \frac{1}{n_k} \sum_{i:y_i=y_k} x_j^{(i)}$
- $\sigma_{jk}^2 = \frac{1}{n_k} \sum_{i:y_i=y_k} (x_j^{(i)} - \mu_{jk})^2$
- n_k: number of samples in class y_k

Multinomial Naive Bayes

For Discrete Counts

Commonly used for text classification:

$$P(X_j|Y = y_k) = \frac{\text{count}(X_j, Y = y_k) + \alpha}{\sum_{l=1}^{d} \text{count}(X_l, Y = y_k) + \alpha d}$$

- α: Smoothing parameter
- Prevents zero probabilities
- Laplace smoothing when $\alpha = 1$

Example

Word counts in documents:

Word	Spam Count
free	150
money	120
...	...

Bernoulli Naive Bayes

For Binary Features

Models presence/absence of features:

$$P(X_j|Y = y_k) = P(j|y_k)^{x_j}(1 - P(j|y_k))^{1-x_j}$$

Application

- $x_j = 1$ if feature j is present
- $x_j = 0$ if feature j is absent
- Useful for document classification with binary word presence

Training Algorithm

Step 1: Estimate Priors

$$P(Y = y_k) = \frac{\text{number of samples in class } y_k}{\text{total samples}}$$

Step 2: Estimate Likelihoods

- Gaussian: Compute mean and variance for each feature per class
- Multinomial: Compute frequency counts for each feature per class
- Bernoulli: Compute probability of feature presence per class

Discriminate model: Logistic Regression

Discriminative Approach

Directly model posterior probability $p(y|\mathbf{x})$ without modeling $p(\mathbf{x}|y)$

Binary case:

$$p(y=1|\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x} + b)$$

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

Multiclass case (Softmax):

$$p(y=k|\mathbf{x}) = \frac{\exp(\mathbf{w}_k^T\mathbf{x} + b_k)}{\sum_{j=1}^{K}\exp(\mathbf{w}_j^T\mathbf{x} + b_j)}$$

Advantage

Makes fewer assumptions about data distribution compared to generative models

Maximum Likelihood Estimation

Maximize log-likelihood of training data:

$$\mathcal{L}(\mathbf{w}) = \sum_{i=1}^{N} [y_i \log p(y_i|\mathbf{x}_i, \mathbf{w}) + (1 - y_i) \log(1 - p(y_i|\mathbf{x}_i, \mathbf{w}))]$$

Gradient for Binary Case

$$\nabla_{\mathbf{w}} \mathcal{L} = \sum_{i=1}^{N} (y_i - p(y_i = 1|\mathbf{x}_i, \mathbf{w}))\mathbf{x}_i$$

Comparison of Approaches

Method	Type	Boundary	Assumptions	Pros
Discriminant Functions	Non-probabilistic	Flexible	None	Simple, fast
LDA	Generative	Linear	Gaussian, shared covariance	Robust to small data
QDA	Generative	Quadratic	Gaussian, different covariance	Flexible boundaries
Logistic Regression	Discriminative	Linear	Linear decision boundary	Optimal for classification

Table: Comparison of classification methods

- **Generative**: Better with small datasets, can generate samples, handles missing data
- **Discriminative**: Often better performance with large datasets, focuses on decision boundary

When to Use Each Method I

Discriminant Functions

- When probabilistic interpretation is not needed
- When computational efficiency is critical
- For simple, interpretable models

Generative Models (LDA/QDA)

- When dataset is small
- When you want to generate new samples
- When features follow approximately Gaussian distribution
- When you need to handle missing data

Logistic Regression

When to Use Each Method II

- For large datasets
- When you want well-calibrated probabilities
- When Gaussian assumptions are violated
- As a baseline for more complex models

Non-parametric methods

KNearest-Neighbor, Decision tree, and random forest.