Northeastern University
Khoury College of
Computer Sciences

Linear Discriminant Analysis & Naive Bayes

DS 4400 | Machine Learning and Data Mining I

Zohair Shafi

Spring 2026

Wednesday | February 11, 2026

Updates

- **Wanrou**

  1:30-3:00 PM Tuesday (17th) - In person - Richards Hall 243

  5:00-6:30 PM Wednesday (18th) - Virtual

  1. Linear algebra

     1a. Vectors

     1b. Matrices

     1c. Vector and Matrix operations

  2. Probabilities

     2a. Bayes' rule and conditional probability

     2b. Distributions

     2c. CDFs and PDFs

- **Zaiba**

  1:00-2:30PM Wednesday (18th) - In person - EL 311

  2:00-3:00 PM Thursday (19th) - Virtual

  1. Probabilities

     1a. Bayes' rule and conditional probability

     1b. Distributions

     1c. CDFs and PDFs

  2. Derivatives

     2a. Gradients

     2b. Derivatives of some common functions

     2c. Chain Rule, Product Rule, Quotient Rule

Today’s Outline

- **Linear Discriminant Analysis**
- Naive Bayes

Linear Discriminant Analysis (LDA)

- LDA is a **generative** classifier

Linear Discriminant Analysis (LDA)

- LDA is a **generative** classifier
- Instead of directly predicting $\mathbb{P}(Y \mid X)$ like logistic regression, it models the joint distribution $\mathbb{P}(X, Y)$ by modeling:
  - $\mathbb{P}(X \mid Y)$: How features are distributed **within each class**
  - $\mathbb{P}(Y)$: Prior probability of each class
- Then it uses Bayes' theorem to compute $\mathbb{P}(Y \mid X)$ for classification.

Linear Discriminant Analysis (LDA)

- LDA is a **generative** classifier
- **Key idea**:
  - Assume each class generates data from a Gaussian distribution.
  - Find the decision boundary that optimally separates these Gaussian clouds.

Linear Discriminant Analysis (LDA)

Assumptions

- Assumption 1:
  - Class conditional probabilities are Gaussian (normal distribution)
  - $\mathbb{P}(X \mid Y = k) = \mathcal{N}(X \mid \mu_k, \Sigma)$

Linear Discriminant Analysis (LDA)

Assumptions

- Assumption 1:
  - Class conditional probabilities are Gaussian (normal distribution)
  - $\mathbb{P}(X \mid Y = k) = \mathcal{N}(X \mid \mu_k, \Sigma)$
- Assumption 2:
  - All classes share the same co-variance matrix Σ (homoscedasticity)

Linear Discriminant Analysis (LDA)

Assumptions

- Assumption 1:
  - Class conditional probabilities are Gaussian (normal distribution)
  - $\mathbb{P}(X \mid Y = k) = \mathcal{N}(X \mid \mu_k, \Sigma)$
- Assumption 2:
  - All classes share the same co-variance matrix Σ (homoscedasticity)
- Assumption 3:
  - Classes differ only in their means μ_k
- These assumptions lead to linear decision boundaries - hence **Linear** Discriminant Analysis.

Linear Discriminant Analysis (LDA)

Assumptions

- Assumption 1:
  - Class conditional probabilities are Gaussian (normal distribution)
  - $\mathbb{P}(X \mid Y = k) = \mathcal{N}(X \mid \mu_k, \Sigma)$
- Assumption 2:
  - All classes share the same co-variance matrix Σ (homoscedasticity)
- Assumption 3:
  - Classes differ only in their means μ_k
- These assumptions lead to linear decision boundaries - hence **Linear** Discriminant Analysis.

Linear Discriminant Analysis (LDA)

Gaussian Conditional Probability

- Assumption:
  - Class conditional probabilities are Gaussian (normal distribution)
  - $\mathbb{P}(X \mid Y = k) = \mathcal{N}(X \mid \mu_k, \Sigma)$

Linear Discriminant Analysis (LDA)

Gaussian Conditional Probability

- Assumption:
  - Class conditional probabilities are Gaussian (normal distribution)
  - $\mathbb{P}(X \mid Y = k) = \mathcal{N}(X \mid \mu_k, \Sigma)$

$$\text{Gaussian PDF: } f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$$\mathbb{P}(x \mid Y = k) = \frac{1}{(2\pi)^{\frac{d}{2}} \mid \Sigma \mid^{\frac{1}{2}}} exp(-\frac{1}{2}(x - \mu_k)^T \Sigma^{-1}(x - \mu_k))$$

Linear Discriminant Analysis (LDA)

Computing Predictions

Have:

$$\mathbb{P}(x \mid Y = k) = \frac{1}{(2\pi)^{\frac{d}{2}} \mid \Sigma \mid^{\frac{1}{2}}} exp(-\frac{1}{2}(x - \mu_k)^T \Sigma^{-1}(x - \mu_k))$$

Linear Discriminant Analysis (LDA)

Computing Predictions

Have:

$$\mathbb{P}(x \mid Y = k) = \frac{1}{(2\pi)^{\frac{d}{2}} \mid \Sigma \mid^{\frac{1}{2}}} exp(-\frac{1}{2}(x - \mu_k)^T \Sigma^{-1}(x - \mu_k))$$

Want:

$$\mathbb{P}(Y \mid X)$$

Linear Discriminant Analysis (LDA)

Computing Predictions

Have:

$$\mathbb{P}(x \mid Y = k) = \frac{1}{(2\pi)^{\frac{d}{2}} \mid \Sigma \mid^{\frac{1}{2}}} exp(-\frac{1}{2}(x - \mu_k)^T \Sigma^{-1}(x - \mu_k))$$

Want:

$$\mathbb{P}(Y \mid X)$$

Use Bayes' Theorem:

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)} = \frac{P(B \mid A) \cdot P(A)}{P(B)}$$

Linear Discriminant Analysis (LDA)

Computing Predictions

$$\mathbb{P}(Y = k \,|\, X = x) = \frac{\mathbb{P}(X = x \,|\, Y = k) \cdot \mathbb{P}(Y = k)}{\mathbb{P}(X = x)}$$

Linear Discriminant Analysis (LDA)

Computing Predictions

$$\mathbb{P}(Y = k \,|\, X = x) = \frac{\mathbb{P}(X = x \,|\, Y = k) \cdot \mathbb{P}(Y = k)}{\mathbb{P}(X = x)}$$

$$\mathbb{P}(Y = k \,|\, X = x) = \frac{\mathbb{P}(X = x \,|\, Y = k) \cdot \mathbb{P}(Y = k)}{\sum_{j=1}^{k} \mathbb{P}(X = x \,|\, Y = j) \cdot \mathbb{P}(Y = j)}$$

Linear Discriminant Analysis (LDA)

Computing Predictions

$$\mathbb{P}(Y = k \,|\, X = x) = \frac{\mathbb{P}(X = x \,|\, Y = k) \cdot \mathbb{P}(Y = k)}{\sum_{j=1}^{k} \mathbb{P}(X = x \,|\, Y = j) \cdot \mathbb{P}(Y = j)}$$

We classify to the class with highest posterior probability:

$$\hat{y} = argmax\ _k \mathbb{P}(Y = k \,|\, X = x)$$

Linear Discriminant Analysis (LDA)

Computing Predictions

$$\mathbb{P}(Y = k \,|\, X = x) = \frac{\mathbb{P}(X = x \,|\, Y = k) \cdot \mathbb{P}(Y = k)}{\sum_{j=1}^{k} \mathbb{P}(X = x \,|\, Y = j) \cdot \mathbb{P}(Y = j)}$$

We classify to the class with highest posterior probability:

$$\hat{y} = argmax\ _k \mathbb{P}(Y = k \,|\, X = x)$$

Note for all k, the **denominator is going to be the same**

Linear Discriminant Analysis (LDA)

Computing Predictions

$$\mathbb{P}(Y = k \,|\, X = x) = \frac{\mathbb{P}(X = x \,|\, Y = k) \cdot \mathbb{P}(Y = k)}{\sum_{j=1}^{k} \mathbb{P}(X = x \,|\, Y = j) \cdot \mathbb{P}(Y = j)}$$

$$\hat{y} = argmax_{\,k}\ \mathbb{P}(X = x \,|\, Y = k) \cdot \mathbb{P}(Y = k)$$

Taking log on RHS (monotonic transform)

$$\hat{y} = argmax_{\,k}\ log(\mathbb{P}(X = x \,|\, Y = k)) + log(\mathbb{P}(Y = k))$$

Linear Discriminant Analysis (LDA)

Computing Predictions

$$\hat{y} = \arg\max_k \left[\log P(X = x \mid Y = k) + \log P(Y = k) \right]$$

Linear Discriminant Analysis (LDA)

Computing Predictions

$$\hat{y} = \arg\max_k \left[\log P(X = x \mid Y = k) + \log P(Y = k)\right]$$

Expanding the log of the Gaussian:

$$\log P(X = x \mid Y = k) = \log \left[\frac{1}{(2\pi)^{d/2} \mid \Sigma \mid^{1/2}} \exp\left(-\frac{1}{2}(x - \mu_k)^T \Sigma^{-1} (x - \mu_k)\right)\right]$$

Linear Discriminant Analysis (LDA)

Computing Predictions

$$\hat{y} = \arg\max_k \left[\log P(X = x \mid Y = k) + \log P(Y = k)\right]$$

Expanding the log of the Gaussian:

$$\log P(X = x \mid Y = k) = \log \left[\frac{1}{(2\pi)^{d/2} \mid \Sigma \mid^{1/2}} \exp\left(-\frac{1}{2}(x - \mu_k)^T \Sigma^{-1} (x - \mu_k)\right)\right]$$

Using $\log(ab) = \log(a) + \log(b)$:

$$\log\left(\frac{1}{(2\pi)^{d/2}}\right) + \log\left(\frac{1}{\mid \Sigma \mid^{1/2}}\right) + \log\left(\exp\left(-\frac{1}{2}(x - \mu_k)^T \Sigma^{-1} (x - \mu_k)\right)\right)$$

Linear Discriminant Analysis (LDA)

Computing Predictions

$$\log\left(\frac{1}{(2\pi)^{d/2}}\right) + \log\left(\frac{1}{|\Sigma|^{1/2}}\right) + \log\left(\exp\left(-\frac{1}{2}(x-\mu_k)^T\Sigma^{-1}(x-\mu_k)\right)\right)$$

Simplifying each term:

Linear Discriminant Analysis (LDA)

Computing Predictions

$$\log\left(\frac{1}{(2\pi)^{d/2}}\right) + \log\left(\frac{1}{|\Sigma|^{1/2}}\right) + \log\left(\exp\left(-\frac{1}{2}(x-\mu_k)^T\Sigma^{-1}(x-\mu_k)\right)\right)$$

Simplifying each term:

$$= -\frac{d}{2}\log(2\pi) - \frac{1}{2}\log|\Sigma| - \frac{1}{2}(x-\mu_k)^T\Sigma^{-1}(x-\mu_k)$$

Linear Discriminant Analysis (LDA)

Computing Predictions

The full expression to maximize is:

$$\hat{y} = arg\max_{k} -\frac{d}{2}\log(2\pi) - \frac{1}{2}\log|\Sigma| - \frac{1}{2}(x-\mu_k)^T\Sigma^{-1}(x-\mu_k) + \log P(Y=k)$$

Linear Discriminant Analysis (LDA)

Computing Predictions

The full expression to maximize is:

$$\hat{y} = arg\max_k -\frac{d}{2}\log(2\pi) - \frac{1}{2}\log|\Sigma| - \frac{1}{2}(x-\mu_k)^T\Sigma^{-1}(x-\mu_k) + \log P(Y=k)$$

Terms that don't depend on k:

$-\frac{d}{2}\log(2\pi)$: depends only on dimensionality d

$-\frac{1}{2}\log|\Sigma|$: Σ is same for all classes (LDA assumption)

Linear Discriminant Analysis (LDA)

Computing Predictions

The full expression to maximize is:

$$\hat{y} = \arg\max_k \left[-\frac{1}{2}(x - \mu_k)^T \Sigma^{-1} (x - \mu_k) + \log P(Y = k) \right]$$

Linear Discriminant Analysis (LDA)

Computing Predictions

The full expression to maximize is:

$$\hat{y} = \arg\max_k \left[-\frac{1}{2}(x-\mu_k)^T\Sigma^{-1}(x-\mu_k) + \log P(Y=k) \right]$$

Expanding:

$$(x-\mu_k)^T\Sigma^{-1}(x-\mu_k) = x^T\Sigma^{-1}x - x^T\Sigma^{-1}\mu_k - \mu_k^T\Sigma^{-1}x + \mu_k^T\Sigma^{-1}\mu_k$$

Linear Discriminant Analysis (LDA)

Computing Predictions

The full expression to maximize is:

$$\hat{y} = \arg\max_k \left[-\frac{1}{2}(x-\mu_k)^T\Sigma^{-1}(x-\mu_k) + \log P(Y=k) \right]$$

Expanding:

$$(x-\mu_k)^T\Sigma^{-1}(x-\mu_k) = x^T\Sigma^{-1}x - x^T\Sigma^{-1}\mu_k - \mu_k^T\Sigma^{-1}x + \mu_k^T\Sigma^{-1}\mu_k$$

Note that $x^T\Sigma^{-1}\mu_k$ and $\mu_k^T\Sigma^{-1}x$ are both scalars, and since Σ^{-1} is **symmetric**:

$$\mu_k^T\Sigma^{-1}x = (x^T\Sigma^{-1}\mu_k)^T = x^T\Sigma^{-1}\mu_k$$

Linear Discriminant Analysis (LDA)

Computing Predictions

$$(x-\mu_k)^T\Sigma^{-1}(x-\mu_k) = x^T\Sigma^{-1}x - x^T\Sigma^{-1}\mu_k - \mu_k^T\Sigma^{-1}x + \mu_k^T\Sigma^{-1}\mu_k$$

Therefore

$$(x-\mu_k)^T\Sigma^{-1}(x-\mu_k) = x^T\Sigma^{-1}x - 2x^T\Sigma^{-1}\mu_k + \mu_k^T\Sigma^{-1}\mu_k$$

Lets put this back into the original equation

Linear Discriminant Analysis (LDA)

Computing Predictions

$$\hat{y} = -\frac{1}{2}\left[x^T\Sigma^{-1}x - 2x^T\Sigma^{-1}\mu_k + \mu_k^T\Sigma^{-1}\mu_k\right] + \log P(Y = k)$$

Linear Discriminant Analysis (LDA)

Computing Predictions

$$\hat{y} = -\frac{1}{2}\left[x^T\Sigma^{-1}x - 2x^T\Sigma^{-1}\mu_k + \mu_k^T\Sigma^{-1}\mu_k\right] + \log P(Y = k)$$

$$= -\frac{1}{2}x^T\Sigma^{-1}x + x^T\Sigma^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma^{-1}\mu_k + \log P(Y = k)$$

Linear Discriminant Analysis (LDA)

Computing Predictions

$$\hat{y} = -\frac{1}{2}\left[x^T\Sigma^{-1}x - 2x^T\Sigma^{-1}\mu_k + \mu_k^T\Sigma^{-1}\mu_k\right] + \log P(Y=k)$$

$$= -\frac{1}{2}x^T\Sigma^{-1}x + x^T\Sigma^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma^{-1}\mu_k + \log P(Y=k)$$

The term $-\frac{1}{2}x^T\Sigma^{-1}x$ depends only on x, not on k.

When comparing classes, this is the same for all k, so we drop it.

Linear Discriminant Analysis (LDA)

Computing Predictions

Finally:

$$\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log P(Y = k)$$

Linear Discriminant Analysis (LDA)

Computing Predictions

Finally:

$$\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log P(Y = k)$$

This is linear in x because:

$x^T \Sigma^{-1} \mu_k$ is linear in x (dot product with a constant vector)

The other terms are constants (don't depend on x)

Linear Discriminant Analysis (LDA)

Computing Predictions

Finally:

$$\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log P(Y = k)$$

We can also re-write as:

$$\delta_k(x) = \theta_{1_k}^T x + \theta_{0_k}$$

Where:

$$\theta_{1_k} = \Sigma^{-1} \mu_k$$

$$\theta_{0_k} = -\frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log P(Y = k)$$

Linear Discriminant Analysis (LDA)

Parameter Estimation

$$\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log P(Y = k)$$

What do we need?

Class Priors: $\mathbb{P}(Y = k)$

Class Means: μ_k

Covariance: Σ

Linear Discriminant Analysis (LDA)

Parameter Estimation

What do we need?

Class Priors: $\mathbb{P}(Y = k) = \frac{N_k}{N}$

Class Means: $\mu_k = \frac{1}{N_k} \sum_{i:y_i=k} x_i$

Covariance: $\Sigma = \frac{1}{N-K} \sum_{k=1}^{K} \sum_{i:y_i=k} (x_i - \mu_k)(x_i - \mu_k)^T$

Linear Discriminant Analysis (LDA)

Parameter Estimation

Suppose you have a dataset with

$d = 3$ features (height, weight, age)

$k = 2$ classes (male, female)

$$\begin{bmatrix} Index & Height & Weight & Age & Sex \\ 1 & 175 & 70 & 25 & F \\ 2 & 180 & 80 & 30 & F \\ 3 & 170 & 68 & 22 & F \\ 4 & 185 & 85 & 35 & F \\ 5 & 170 & 50 & 25 & M \\ 6 & 175 & 60 & 30 & M \\ 7 & 170 & 58 & 22 & M \\ 8 & 165 & 65 & 35 & M \end{bmatrix}$$

Linear Discriminant Analysis (LDA)

Parameter Estimation

$$\begin{bmatrix} Index & Height & Weight & Age & Sex \\ 1 & 175 & 70 & 25 & F \\ 2 & 180 & 80 & 30 & F \\ 3 & 170 & 68 & 22 & F \\ 4 & 185 & 85 & 35 & F \\ 5 & 170 & 50 & 25 & M \\ 6 & 175 & 60 & 30 & M \\ 7 & 170 & 58 & 22 & M \\ 8 & 165 & 65 & 35 & M \end{bmatrix}$$

Then:

$$\mu_F = \begin{bmatrix} \bar{x}_{\text{height}} \\ \bar{x}_{\text{weight}} \\ \bar{x}_{\text{age}} \end{bmatrix} = \begin{bmatrix} \frac{175+180+170+185}{4} \\ \frac{70+80+68+85}{4} \\ \frac{25+30+22+35}{4} \end{bmatrix} = \begin{bmatrix} 177.5 \\ 75.75 \\ 28 \end{bmatrix}$$

Linear Discriminant Analysis (LDA)

LDA For Dimensionality Reduction

- Beyond classification, LDA can project high-dimensional data onto a **lower-dimensional subspace** that maximizes class separation.

Linear Discriminant Analysis (LDA)

LDA For Dimensionality Reduction

- Beyond classification, LDA can project high-dimensional data onto a **lower-dimensional subspace** that maximizes class separation.
- This is sometimes called Fisher's Linear Discriminant Analysis.
- What do we want?

$$z = W^T x$$

Where $z \in \mathbb{R}^r$, $X \in \mathbb{R}^d$ and $W \in \mathbb{R}^{d \times r}$

We want the projected data to have **maximum separation between classes** and **minimum spread within classes**.

Linear Discriminant Analysis (LDA)

LDA For Dimensionality Reduction

- Scatter Matrix (1/3)
  - Within Scatter Matrix
  - $$S_W = \sum_{k=1}^{K} S_k = \sum_{k=1}^{K} \sum_{i:y_i=k} (x_i - \mu_k)(x_i - \mu_k)^T$$
  - where S_k is the scatter matrix for class k of shape $\mathbb{R}^{d \times d}$
  - **Intuition**: Small S_W means points are tightly clustered around their class means.

Linear Discriminant Analysis (LDA)

LDA For Dimensionality Reduction

- Scatter Matrix (2/3)
  - Between Scatter Matrix
  - $$S_B = \sum_{k=1}^{K} N_k(\mu_k - \mu)(\mu_k - \mu)^T$$
  - where $\mu = \frac{1}{N}\sum_{i}^{N} x_i$ is the global mean
  - **Intuition**: Large S_B means class centers are far apart from each other.

Linear Discriminant Analysis (LDA)

LDA For Dimensionality Reduction

- Scatter Matrix (3/3)
  - Total Scatter Matrix
  - $S_T = S_W + S_B$

Linear Discriminant Analysis (LDA)

LDA For Dimensionality Reduction

- Optimization Objective
- We want projections where:
  - Between-class scatter is large (classes far apart - Large S_B)
  - Within-class scatter is small (classes compact - Small S_W)

Linear Discriminant Analysis (LDA)

LDA For Dimensionality Reduction

- Optimization Objective
- We want projections where:
  - Between-class scatter is large (classes far apart - Large S_B)
  - Within-class scatter is small (classes compact - Small S_W)
- For a **single** projection direction w
  - Projected within-class scatter = $w^T S_W w$
  - Projected between-class scatter = $w^T S_B w$

Linear Discriminant Analysis (LDA)

LDA For Dimensionality Reduction

- Optimization Objective
- We want projections where:
  - Between-class scatter is large (classes far apart - Large S_B)
  - Within-class scatter is small (classes compact - Small S_W)
- For a **single** projection direction w
  - Projected within-class scatter = $w^T S_W w$
  - Projected between-class scatter = $w^T S_B w$

Linear Discriminant Analysis (LDA)

LDA For Dimensionality Reduction

- Optimization Objective
- We want projections where:
  - Between-class scatter is large (classes far apart - Large S_B)
  - Within-class scatter is small (classes compact - Small S_W)
- For a **single** projection direction w
  - Projected within-class scatter = $w^T S_W w$
  - Projected between-class scatter = $w^T S_B w$

$$\text{Maximize} : J(w) = \frac{w^T S_B w}{w^T S_W w}$$

Linear Discriminant Analysis (LDA)

LDA For Dimensionality Reduction

$$\text{Maximize} : J(w) = \frac{w^T S_B w}{w^T S_W w}$$

To maximize, compute derivative and set to zero.

$$S_B \cdot w = \lambda S_W \cdot w$$

$$S_W^{-1} S_B = \lambda w$$

This is the standard Eigen-decomposition

The optimal w^* is the Eigenvector corresponding to the largest Eigenvalue

Linear Discriminant Analysis (LDA)

LDA For Dimensionality Reduction

$$\text{Maximize} : J(w) = \frac{w^T S_B w}{w^T S_W w}$$

Linear Discriminant Analysis (LDA)

LDA For Dimensionality Reduction

$$\text{Maximize} : J(w) = \frac{w^T S_B w}{w^T S_W w}$$

To maximize, compute derivative and set to zero.

Linear Discriminant Analysis (LDA)

LDA For Dimensionality Reduction

$$\text{Maximize} : J(w) = \frac{w^T S_B w}{w^T S_W w}$$

To maximize, compute derivative and set to zero.

$$S_B \cdot w = \lambda S_W \cdot w$$

$$S_W^{-1} S_B = \lambda w$$

Linear Discriminant Analysis (LDA)

LDA For Dimensionality Reduction

$$\text{Maximize} : J(w) = \frac{w^T S_B w}{w^T S_W w}$$

To maximize, compute derivative and set to zero.

$$S_B \cdot w = \lambda S_W \cdot w$$

$$S_W^{-1} S_B \cdot w = \lambda w$$

This is the standard Eigen-decomposition

The optimal w^* is the Eigenvector corresponding to the largest Eigenvalue

Linear Discriminant Analysis (LDA)

LDA Summary

Pros

- Simple, fast, closed-form solution
- No hyperparameters to tune
- Works well when assumptions approximately hold
- Provides probabilistic outputs
- Built-in dimensionality reduction
- Stable with small datasets

Cons

- Assumes Gaussian distributions
- Assumes shared covariance (linear boundaries only)
- Sensitive to outliers (affect mean and covariance estimates)
- Cannot capture non-linear relationships
- Fails if features are highly non-Gaussian

Today’s Outline

- Linear Discriminant Analysis
- **Naive Bayes**

Naive Bayes Classifier

- Like LDA, Naive Bayes is a generative classifier using Bayes' theorem

$$\mathbb{P}(Y = k \,|\, X = x) = \frac{\mathbb{P}(X = x \,|\, Y = k) \cdot \mathbb{P}(Y = k)}{\mathbb{P}(X = x)}$$

Naive Bayes Classifier

- Like LDA, Naive Bayes is a generative classifier using Bayes' theorem

$$\mathbb{P}(Y = k \,|\, X = x) = \frac{\mathbb{P}(X = x \,|\, Y = k) \cdot \mathbb{P}(Y = k)}{\mathbb{P}(X = x)}$$

- The challenge:
  - Estimating $\mathbb{P}(X \,|\, Y = k)$ for high-dimensional X is difficult.

Naive Bayes Classifier

- Like LDA, Naive Bayes is a generative classifier using Bayes' theorem

$$\mathbb{P}(Y = k \,|\, X = x) = \frac{\mathbb{P}(X = x \,|\, Y = k) \cdot \mathbb{P}(Y = k)}{\mathbb{P}(X = x)}$$

- The challenge:
  - Estimating $\mathbb{P}(X \,|\, Y = k)$ for high-dimensional X is difficult.
  - With d features, each taking v possible values, we'd need to estimate v^d parameters per class - **exponential in dimensionality**.

Naive Bayes Classifier

Naive Bayes assumes features are conditionally independent given the class

- Like LDA, Naive Bayes is a generative classifier using Bayes' theorem

$$\mathbb{P}(Y = k \,|\, X = x) = \frac{\mathbb{P}(X = x \,|\, Y = k) \cdot \mathbb{P}(Y = k)}{\mathbb{P}(X = x)}$$

- The challenge:
  - Estimating $\mathbb{P}(X \,|\, Y = k)$ for high-dimensional X is difficult.
  - With d features, each taking v possible values, we'd need to estimate v^d parameters per class - **exponential in dimensionality**.

Naive Bayes Classifier

- Like LDA, Naive Bayes is a generative classifier using Bayes' theorem

$$\mathbb{P}(Y = k \,|\, X = x) = \frac{\mathbb{P}(X = x \,|\, Y = k) \cdot \mathbb{P}(Y = k)}{\mathbb{P}(X = x)}$$

- $\mathbb{P}(X = x \,|\, Y = k) = \mathbb{P}(x_1, x_2, x_3, \ldots, x_d \,|\, Y = k) = \Pi_{j=1}^{d} \mathbb{P}(x_j \,|\, Y = k)$
- This is almost always wrong in practice - features are usually correlated.
- But it reduces parameters from exponential to linear: d parameters per class instead of v^d

Naive Bayes Classifier

- Example:
  - For spam classification with words “free” and “money”
    - Reality: $\mathbb{P}$("free", "money" | spam) $\neq \mathbb{P}$("free"|spam) $\cdot \mathbb{P}$(“money" | spam)
    - These words are correlated in spam emails
    - Naive Bayes pretends they're **independent**

Naive Bayes Classifier

Want to predict:

$$\mathbb{P}(Y = k \,|\, X = x) = \frac{\mathbb{P}(X = x \,|\, Y = k) \cdot \mathbb{P}(Y = k)}{\mathbb{P}(X = x)}$$

Naive Assumption:

$$\mathbb{P}(X = x \,|\, Y = k) = \mathbb{P}(x_1, x_2, x_3, \ldots, x_d \,|\, Y = k) = \Pi_{j=1}^{d} \mathbb{P}(x_j \,|\, Y = k)$$

Naive Bayes Classifier

$$\hat{y} = arg\max_{k} \mathbb{P}(Y = k) \cdot \Pi_{j=1}^{d} \mathbb{P}(x_j \mid Y = k)$$

For numerical stability, we take logs

$$\hat{y} = arg\max_{k} log(\mathbb{P}(Y = k)) + \sum_{j=1}^{d} log(\mathbb{P}(x_j \mid Y = k))$$

Naive Bayes Classifier

$$\hat{y} = arg\max_{k} \mathbb{P}(Y = k) \cdot \Pi_{j=1}^{d} \mathbb{P}(x_j \mid Y = k)$$

For numerical stability, we take logs

$$\hat{y} = arg\max_{k} log(\mathbb{P}(Y = k)) + \sum_{j=1}^{d} log(\mathbb{P}(x_j \mid Y = k))$$

How do you find $\mathbb{P}(x_j \mid Y = k)$?

Naive Bayes Classifier

Gaussian Naive Bayes

For continuous features, assume each feature follows a Gaussian Distribution

$$\mathbb{P}(x_j \mid Y = k) = \frac{1}{\sqrt{2\pi\sigma_{kj}^2}} \cdot e^{-\frac{(x_j - \mu_{kj})^2}{2\sigma_{kj}^2}}$$

Each class-feature combination has its own mean μ_{kj} and variance σ_{kj}^2

Naive Bayes Classifier

$$\mu_{kj} = \frac{1}{N_k} \sum_{i:y_i=k} x_{ij}$$

$$\sigma_{kj}^2 = \frac{1}{N_k} \sum_{i:y_i=k} (x_{ij} - \mu_{kj})^2$$

Gaussian Naive Bayes

For continuous features, assume each feature follows a Gaussian Distribution

$$\mathbb{P}(x_j \mid Y = k) = \frac{1}{\sqrt{2\pi\sigma_{kj}^2}} \cdot e^{-\frac{(x_j - \mu_{kj})^2}{2\sigma_{kj}^2}}$$

Each class-feature combination has its own mean μ_{kj} and variance σ_{kj}^2

Naive Bayes Classifier

Multinomial Naive Bayes

For discrete features/count data like word frequencies

$$\mathbb{P}(x_j \mid Y = k) = \theta_{kj}^{x_j}$$

Where θ_{kj} is the probability of feature j in class k and x_j is the count of feature j

$$\theta_{kj} = \frac{\text{count of feature } j \text{ in class } k}{\text{total count of all features in class } k}$$

Naive Bayes Classifier

Bernoulli Naive Bayes

For binary data like word frequencies

$$\mathbb{P}(x_j \mid Y = k) = \theta_{kj}^{x_j} \cdot (1 - \theta_{kj})^{1 - x_j}$$

Where θ_{kj} is the probability of feature j in class k and x_j is the count of feature j

$$\theta_{kj} = \frac{\text{count of feature } j \text{ in class } k}{\text{total samples in class } k}$$

Naive Bayes Classifier

Example

Task

Classify emails as spam or not spam

Training Data

Document	x_1 = “free”	x_2 = “money”	x_3 = “meeting”	x_4 = “lunch”	Class
1	3	2	0	0	spam
2	2	1	0	0	spam
3	0	0	2	1	not spam
4	0	0	1	2	not spam

Naive Bayes Classifier

Example

$$\theta_{kj} = \frac{\text{count of feature } j \text{ in class } k}{\text{total count of all features in class } k}$$

Task

Classify emails as spam or not spam

$$\mathbb{P}(\textbf{spam}) = \frac{2}{4} = 0.5$$

$$\mathbb{P}(\textbf{not spam}) = \frac{2}{4} = 0.5$$

Training Data

Document	x_1 = “free”	x_2 = “money”	x_3 = “meeting”	x_4 = “lunch”	Class
1	3	2	0	0	spam
2	2	1	0	0	spam
3	0	0	2	1	not spam
4	0	0	1	2	not spam

Naive Bayes Classifier

Example

$$\theta_{kj} = \frac{\text{count of feature } j \text{ in class } k}{\text{total count of all features in class } k}$$

Task

Classify emails as spam or not spam

Training Data

Document	x_1 = “free”	x_2 = “money”	x_3 = “meeting”	x_4 = “lunch”	Class
1	3	2	0	0	spam
2	2	1	0	0	spam
3	0	0	2	1	not spam
4	0	0	1	2	not spam

$$\mathbb{P}(\textbf{spam}) = \frac{2}{4} = 0.5$$

$$\mathbb{P}(\textbf{not spam}) = \frac{2}{4} = 0.5$$

For **spam** class:

- $\mathbb{P}(\text{"free"}|\text{spam}) = \frac{(5+1)}{(8)} = \frac{6}{8} = 0.75$

Naive Bayes Classifier

Example

$$\theta_{kj} = \frac{\text{count of feature } j \text{ in class } k}{\text{total count of all features in class } k}$$

Task

Classify emails as spam or not spam

Training Data

Document	x_1 = “free”	x_2 = “money”	x_3 = “meeting”	x_4 = “lunch”	Class
1	3	2	0	0	spam
2	2	1	0	0	spam
3	0	0	2	1	not spam
4	0	0	1	2	not spam

$$\mathbb{P}(\textbf{spam}) = \frac{2}{4} = 0.5$$

$$\mathbb{P}(\textbf{not spam}) = \frac{2}{4} = 0.5$$

For **spam** class:

- $\mathbb{P}(\text{"free"}|\text{spam}) = \frac{(5+1)}{(8)} = \frac{6}{8} = 0.75$
- $\mathbb{P}(\text{"money"}|\text{spam}) = \frac{(3+1)}{(8)} = \frac{4}{8} = 0.5$
- $\mathbb{P}(\text{"meeting"}|\text{spam}) = \frac{(0+1)}{(8)} = \frac{1}{8} = 0.125$
- $\mathbb{P}(\text{"lunch"}|\text{spam}) = \frac{(0+1)}{(8)} = \frac{1}{8} = 0.125$

Naive Bayes Classifier

Example

$$\theta_{kj} = \frac{\text{count of feature } j \text{ in class } k}{\text{total count of all features in class } k}$$

Task

Classify emails as spam or not spam

Training Data

Document	x_1 = “free”	x_2 = “money”	x_3 = “meeting”	x_4 = “lunch”	Class
1	3	2	0	0	spam
2	2	1	0	0	spam
3	0	0	2	1	not spam
4	0	0	1	2	not spam

$$\mathbb{P}(\textbf{spam}) = \frac{2}{4} = 0.5$$

$$\mathbb{P}(\textbf{not spam}) = \frac{2}{4} = 0.5$$

For **not spam** class:

- $\mathbb{P}(\text{"free"}|\text{not spam}) = \frac{(0+1)}{(6)} = \frac{1}{6} = 0.166$
- $\mathbb{P}(\text{"money"}|\text{not spam}) = \frac{(0+1)}{(6)} = \frac{1}{6} = 0.166$
- $\mathbb{P}(\text{"meeting"}|\text{not spam}) = \frac{(3+1)}{(6)} = \frac{4}{6} = 0.66$
- $\mathbb{P}(\text{"lunch"}|\text{not spam}) = \frac{(3+1)}{(6)} = \frac{4}{6} = 0.66$

Naive Bayes Classifier

Example

$$\theta_{kj} = \frac{\text{count of feature } j \text{ in class } k}{\text{total count of all features in class } k}$$

$$\mathbb{P}(x_j \mid Y = k) = \theta_{kj}^{x_j}$$

For **spam** class:

- $\mathbb{P}(\text{"free"}|\text{spam}) = \frac{(5+1)}{(8)} = \frac{6}{8} = 0.75$
- $\mathbb{P}(\text{"money"}|\text{spam}) = \frac{(3+1)}{(8)} = \frac{4}{8} = 0.5$
- $\mathbb{P}(\text{"meeting"}|\text{spam}) = \frac{(0+1)}{(8)} = \frac{1}{8} = 0.125$
- $\mathbb{P}(\text{"lunch"}|\text{spam}) = \frac{(0+1)}{(8)} = \frac{1}{8} = 0.125$

For **not spam** class:

- $\mathbb{P}(\text{"free"}|\text{not spam}) = \frac{(0+1)}{(6)} = \frac{1}{6} = 0.166$
- $\mathbb{P}(\text{"money"}|\text{not spam}) = \frac{(0+1)}{(6)} = \frac{1}{6} = 0.166$
- $\mathbb{P}(\text{"meeting"}|\text{not spam}) = \frac{(3+1)}{(6)} = \frac{4}{6} = 0.66$
- $\mathbb{P}(\text{"lunch"}|\text{not spam}) = \frac{(3+1)}{(6)} = \frac{4}{6} = 0.66$

New Email: $x =$ “free money”

Naive Bayes Classifier

Example

$$\theta_{kj} = \frac{\text{count of feature } j \text{ in class } k}{\text{total count of all features in class } k}$$

$$\mathbb{P}(x_j \mid Y = k) = \theta_{kj}^{x_j}$$

For **spam** class:

- $\mathbb{P}(\text{"free"}|\text{spam}) = \frac{(5+1)}{(8)} = \frac{6}{8} = 0.75$
- $\mathbb{P}(\text{"money"}|\text{spam}) = \frac{(3+1)}{(8)} = \frac{4}{8} = 0.5$
- $\mathbb{P}(\text{"meeting"}|\text{spam}) = \frac{(0+1)}{(8)} = \frac{1}{8} = 0.125$
- $\mathbb{P}(\text{"lunch"}|\text{spam}) = \frac{(0+1)}{(8)} = \frac{1}{8} = 0.125$

For **not spam** class:

- $\mathbb{P}(\text{"free"}|\text{not spam}) = \frac{(0+1)}{(6)} = \frac{1}{6} = 0.166$
- $\mathbb{P}(\text{"money"}|\text{not spam}) = \frac{(0+1)}{(6)} = \frac{1}{6} = 0.166$
- $\mathbb{P}(\text{"meeting"}|\text{not spam}) = \frac{(3+1)}{(6)} = \frac{4}{6} = 0.66$
- $\mathbb{P}(\text{"lunch"}|\text{not spam}) = \frac{(3+1)}{(6)} = \frac{4}{6} = 0.66$

New Email: $x =$ “free money”

$$\mathbb{P}(spam \mid x) = \mathbb{P}(Y = spam) \cdot \Pi_{j=1}^{d} \mathbb{P}(x_j \mid Y = spam)$$

Naive Bayes Classifier

Example

$$\theta_{kj} = \frac{\text{count of feature } j \text{ in class } k}{\text{total count of all features in class } k}$$

$$\mathbb{P}(x_j \mid Y = k) = \theta_{kj}^{x_j}$$

For **spam** class:

- $\mathbb{P}(\text{"free"} \mid \text{spam}) = \frac{(5+1)}{(8)} = \frac{6}{8} = 0.75$
- $\mathbb{P}(\text{"money"} \mid \text{spam}) = \frac{(3+1)}{(8)} = \frac{4}{8} = 0.5$
- $\mathbb{P}(\text{"meeting"} \mid \text{spam}) = \frac{(0+1)}{(8)} = \frac{1}{8} = 0.125$
- $\mathbb{P}(\text{"lunch"} \mid \text{spam}) = \frac{(0+1)}{(8)} = \frac{1}{8} = 0.125$

For **not spam** class:

- $\mathbb{P}(\text{"free"} \mid \text{not spam}) = \frac{(0+1)}{(6)} = \frac{1}{6} = 0.166$
- $\mathbb{P}(\text{"money"} \mid \text{not spam}) = \frac{(0+1)}{(6)} = \frac{1}{6} = 0.166$
- $\mathbb{P}(\text{"meeting"} \mid \text{not spam}) = \frac{(3+1)}{(6)} = \frac{4}{6} = 0.66$
- $\mathbb{P}(\text{"lunch"} \mid \text{not spam}) = \frac{(3+1)}{(6)} = \frac{4}{6} = 0.66$

New Email: $x =$ “free money”

$$\mathbb{P}(spam \mid x) = \mathbb{P}(Y = spam) \cdot \Pi_{j=1}^{d} \mathbb{P}(x_j \mid Y = spam)$$

$$\mathbb{P}(spam \mid x) = 0.5 \cdot (0.75)^1 \cdot (0.5)^1 = 0.1875$$

Naive Bayes Classifier

Example

$$\theta_{kj} = \frac{\text{count of feature } j \text{ in class } k}{\text{total count of all features in class } k}$$

$$\mathbb{P}(x_j \mid Y = k) = \theta_{kj}^{x_j}$$

For **spam** class:

- $\mathbb{P}(\text{"free"}|\text{spam}) = \frac{(5+1)}{(8)} = \frac{6}{8} = 0.75$
- $\mathbb{P}(\text{"money"}|\text{spam}) = \frac{(3+1)}{(8)} = \frac{4}{8} = 0.5$
- $\mathbb{P}(\text{"meeting"}|\text{spam}) = \frac{(0+1)}{(8)} = \frac{1}{8} = 0.125$
- $\mathbb{P}(\text{"lunch"}|\text{spam}) = \frac{(0+1)}{(8)} = \frac{1}{8} = 0.125$

For **not spam** class:

- $\mathbb{P}(\text{"free"}|\text{not spam}) = \frac{(0+1)}{(6)} = \frac{1}{6} = 0.166$
- $\mathbb{P}(\text{"money"}|\text{not spam}) = \frac{(0+1)}{(6)} = \frac{1}{6} = 0.166$
- $\mathbb{P}(\text{"meeting"}|\text{not spam}) = \frac{(3+1)}{(6)} = \frac{4}{6} = 0.66$
- $\mathbb{P}(\text{"lunch"}|\text{not spam}) = \frac{(3+1)}{(6)} = \frac{4}{6} = 0.66$

New Email: $x =$ “free money”

$$\mathbb{P}(spam \mid x) = \mathbb{P}(Y = spam) \cdot \Pi_{j=1}^{d} \mathbb{P}(x_j \mid Y = spam)$$

$$\mathbb{P}(spam \mid x) = 0.5 \cdot (0.75)^1 \cdot (0.5)^1 = 0.1875$$

$$\mathbb{P}(not\ spam \mid x) = \mathbb{P}(Y = not\ spam) \cdot \Pi_{j=1}^{d} \mathbb{P}(x_j \mid Y = not\ spam)$$

$$\mathbb{P}(not\ spam \mid x) = 0.5 \cdot (0.166)^1 \cdot (0.166)^1 = 0.013$$

Naive Bayes Classifier

Why does it work?

- Despite violating independence assumption, Naive Bayes often performs surprisingly well.
- Why?
  - Classification only needs **correct ranking**: We don't need accurate probabilities - just need $\mathbb{P}(spam \mid X) > \mathbb{P}(not\ spam \mid X)$ when the email is actually spam.
  - The independence assumption can distort probabilities while preserving the ranking.
  - High bias, low variance tradeoff: The strong assumption **reduces model complexity**, preventing overfitting **especially with limited data**.
  - Conditional independence may approximately hold: Within a class, features are sometimes less correlated than across classes.

Naive Bayes Classifier

Pros

- Extremely fast training (just counting)
- Fast prediction
- Handles high-dimensional data well
- Works with **small training** sets
- Handles missing features naturally
- Easy to implement and interpret
- Often surprisingly accurate

Cons

- Independence assumption is usually wrong
- Probability estimates are unreliable
- Cannot learn feature interactions
- Continuous features require distributional assumptions
- Correlated features are “double-counted”

Classifiers

Comparison	Naive Bayes	Logistic Regression	LDA
Type	Generative	Discriminative	Generative
Assumption	Conditional Independence Between Features	Conditional Independence Between Rows of Data	Gaussian and shared covariance
Training	Closed Form / Counting	Gradient Descent	Closed Form
Data	Better with small data	Better with large data else risk overfitting	Works well across data sizes
Probabilities	Poorly calibrated, care more about correct ranking	Well calibrated	Well calibrated
Missing features	Handles naturally	Requires pre-processing	Requires pre-processing

Next Class

- Recap and if we have time, decision trees.