COMS4995W32
Applied Machine Learning

Dr. Spencer W. Luo

Columbia University | Fall 2025

Generative vs. Discriminative Approaches: Naive Bayes vs. Linear Regression

Agenda

- Motivation
- Naive Bayes (Generative)
- Linear Regression (Discriminative)
- Case Study

G vs. D: The Big Picture

- Generative approach (e.g., Naive Bayes)
  - Learns how the data is generated
  - Like: “learning the recipe 🍳”
- Discriminative approach (e.g., Linear Regression)
  - Directly learns the mapping
  - Like: “tasting and giving a score 🍽️”
- Same goal → **make predictions**
  - Yet 2 very different philosophies 📖

Generative Philosophy

Learns the story of how data is produced (joint distribution)

- Examples:
  - Naive Bayes
    - assumes words are independent
    - builds probability recipe for each class
  - Language models
    - predict next word by modeling sentence generation

Generative Philosophy

✅ Strengths of Naive Bayes

- 📉 Performs well on small datasets (low variance, simple structure)
- 🏗️ Strong baseline - often hard to beat with simple methods

❌ Limitations of Naive Bayes

- 🧩 Naive assumption: features are conditionally independent
- 🔗 Fails if features are correlated (e.g., “New” + “York”)

Discriminative Philosophy

Learns the boundary or function that best predicts outcomes

- Examples:
  - Linear Regression (fit a line directly to minimize error)
  - Logistic Regression, SVM, modern neural networks

Discriminative Philosophy

✅ Strengths of Discriminative Models

- 🎯 Directly optimize prediction accuracy - learn p(y|x)
- 📈 Often higher accuracy than generative models (fewer assumptions)

❌ Limitations of Discriminative Models

- 🧩 Cannot model data generation p(x,y)
- 📊 Need more data than generative models to perform well

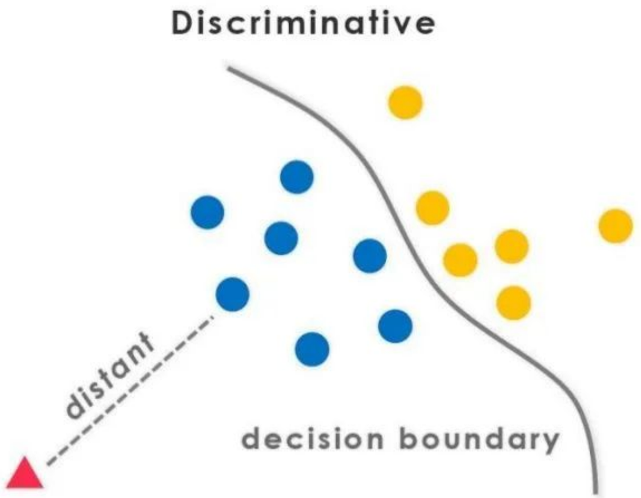
Same Goal, Different Roads.

Why This Matters?

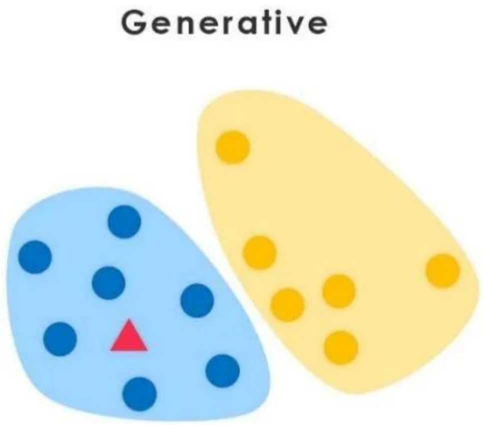
Different learning philosophies → different trade-offs

- Generative = Once we need to understand data generation
- Discriminative = When prediction accuracy is the only goal

Discriminative vs. Generative



- Only care about estimating the conditional probabilities
- Very good when underlying distribution of data is really complicated (e.g. texts, images, movies)



- Model observations (x,y) first, then infer p(y|x)
- Good for missing variables, better diagnostics
- Easy to add prior knowledge about data

Naive Bayes (Generative)

Bayes' Rule

- Prediction is based on updating belief/class after seeing evidence

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

- Prior p(y): how common each class is
- Likelihood p(x|y): how features look under each class
- Posterior p(y|x): updated belief after seeing

- 💡 belief updating → start with a guess p(y), refine with data p(x|y)

Naive Bayes Assumption

- Assuming features are conditionally independent given the class

$$p(x|y) = \prod_{j=1}^{d} p(x_j|y)$$

- Big simplification: turns complex joint distribution into a product
- Works surprisingly well for various applications
  - 🖼️ each pixel ≈ independent feature
  - 📖 word presence treated as independent

- Intuition: Given the class, every word casts its own vote 🗳️

Example

Suppose we want to classify emails as Spam ($y=1$) or Ham ($y=0$).

Vocabulary (features): {lottery, meeting, beef}

Thus each feature x_j stands for whether that word appears in the email (0/1)

👎 "Win a free lottery ticket" → $x = [1, 0, 0]$
👍 "We will have a meeting" → $x = [0, 1, 0]$
👍 "beef sometimes holds a meeting" → $x = [0, 1, 1]$

🤔 "You have won a lottery!"

Bernoulli Naive Bayes (binary features)

- Features: binary word presence/absence (0 or 1)
- Each word = yes/no vote 🗳️
  - If present → contributes evidence
  - If absent → neutral or negative evidence
- Common for short text (SMS, tweets, headlines)
- Formula

$$P(x_j \mid y) = \theta_{j|y}^{x_j}(1 - \theta_{j|y})^{1-x_j}$$

Learning Parameters (MLE)

- Class prior estimated by class frequencies:

$$\hat{P}(y) = \frac{N_y}{N}$$

- Word probabilities estimated by frequency inside each class:

$$\widehat{\theta}_{j|y} = P(x_j = 1|y) = \frac{N_{j,y}}{N_y} = \frac{\text{Count of word } x_j \text{ in class } y}{\text{Total word counts in class } y}$$

Bernoulli Naive Bayes (binary features)

$$P(x_j = 1 \mid y) \;=\; \hat{\theta}_{j|y} \qquad\qquad P(x_j = 0 \mid y) \;=\; 1 - \hat{\theta}_{j|y}$$

$$P(x_j \mid y) = \theta_{j|y}^{x_j}(1 - \theta_{j|y})^{1-x_j}$$

Smoothing (Laplace / Add-α)

Problem: unseen word → zero probability

Solution: add α to counts

$$\widehat{\theta}_{j|y} = \frac{N_{j,y} + \alpha}{N_y + \alpha d}$$

where d = vocabulary size

Effect: prevents zeros, makes model more robust

Numerical Stability

😷 Multiplying many small probabilities → underflow

- Example: 0.001^100 ≈ 0 on a computer

Solution: switch to log-space (turn products into sums)

$$\log P(y \mid x) \propto \log P(y) + \sum_{j=1}^{d} \log P(x_j \mid y)$$

Benefits:

- Avoids numerical underflow
- Computation becomes addition (faster, more stable)

PS: All modern NB implementations use log probabilities internally

Gaussian Naive Bayes (continuous features)

- Features: continuous values (e.g. numeric features)
- Assumption: each feature follows a Gaussian distribution within each class

$$P(x_j|y) = \frac{1}{\sqrt{2\pi\sigma_{j,y}^2}} \exp\left(-\frac{(x_j - \mu_{j,y})^2}{2\sigma_{j,y}^2}\right)$$

Practical Pipeline

🛠️ Step 1 - Preprocessing

- Tokenize, lowercase, remove stopwords
- Handle missing values if needed

🧮 Step 2 - Feature Extraction

- Bag-of-Words / binary counts → Bernoulli NB
- Word counts / TF–IDF → Multinomial NB
- Continuous features → Gaussian NB

Practical Pipeline

🤖 Step 3 - Classifier

- Choose model: BernoulliNB, MultinomialNB, GaussianNB
- Learn parameters by frequency counts (MLE + smoothing)

📊 Step 4 - Evaluation

- Metrics: Accuracy, Precision, Recall, Confusion Matrix
- Use cross_val_score for validation

When NB Works / Fails

✅ Works well:

- High-dimensional sparse features (text, word counts)
- Small datasets (few samples per class)

❌ Fails when:

- Strong feature correlations
- Complex dependencies between features

Linear Regression

From Generative to Discriminative

- Naive Bayes: model joint distribution P(x,y)
- Linear Regression: directly fit f(x) ≈ y

Do not learn the recipe, just draw the line that fits the taste ✏️

Goal: minimize classification error, not model data generation

Hypothesis Form

Linear model = weighted sum of features

$$\hat{y} = \theta^\top x = \sum_{j=1}^{d} \theta_j x_j + b$$

Interpretation

- Each feature contributes proportionally to the prediction
- The weight tells us how much that feature matters

Trick

- add x_0 = 1 → bias term included
- Ensures model can fit data that does **not** pass through the origin

Loss Function (MSE)

Residual sum of squares (RSS):

$$L(\theta) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

Mean Squared Error (MSE):

$$\mathrm{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \theta^\top x_i)^2$$

Intuition: penalizes more on large errors

Gradient Descent (Iterative Optimization)

Update rule

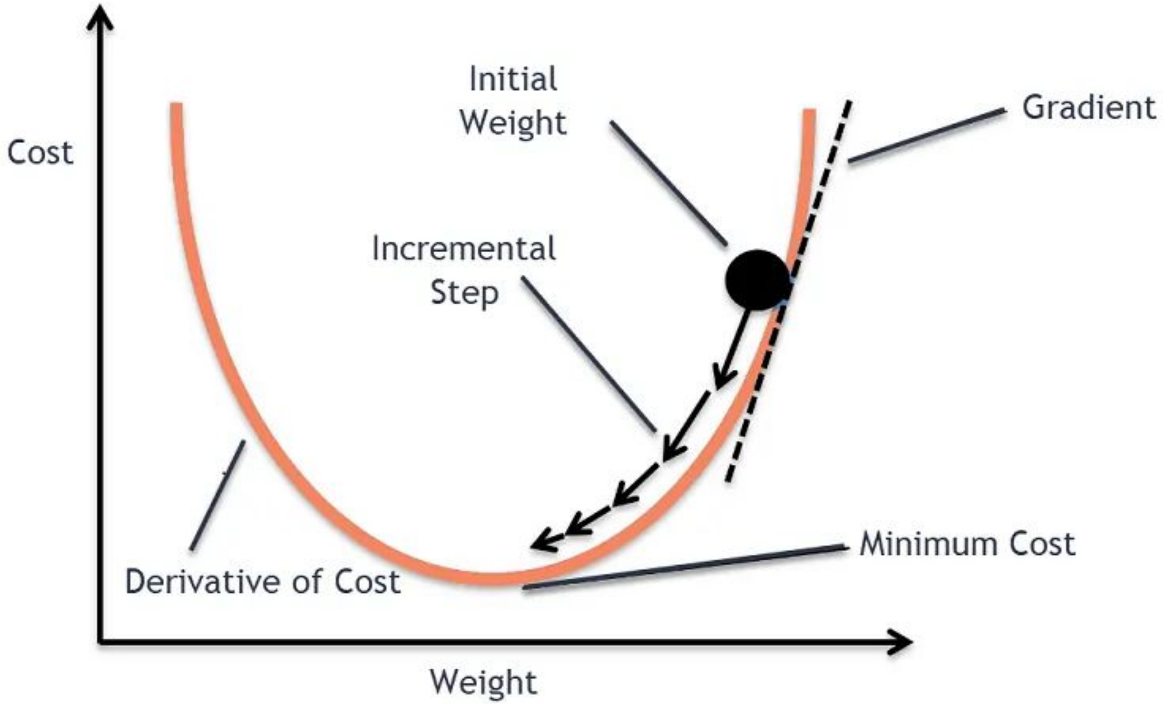
$$\theta \leftarrow \theta - \eta \nabla_\theta L(\theta)$$

where \eta = learning rate/step size

Works for large datasets & online learning

Cornerstone of DL 🤖

Gradient Descent Intuition



Ordinary Least Squares (Closed Form)

Analytic solution when $X^\top X$ invertible:

$$\theta^* = (X^\top X)^{-1} X^\top y$$

Analytic solution = one-shot computation
Fast for small/medium datasets

Limitation: unstable with collinearity or ill-conditioned

Regularization Motivation

Problems:

- Overfitting with too many features
- Multicollinearity → unstable coefficients

- Solution: add penalty to shrink coefficients
- Leads to Ridge & Lasso regression

Ridge Regression

Loss with L-2 penalty:

$$L(\theta) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \theta^\top x_i)^2 + \boxed{\alpha||\theta||_2^2}$$

- Always unique solution
- Coefficients shrink towards zero (but not exactly zero)
- Good when many small/medium effects are expected

Lasso Regression

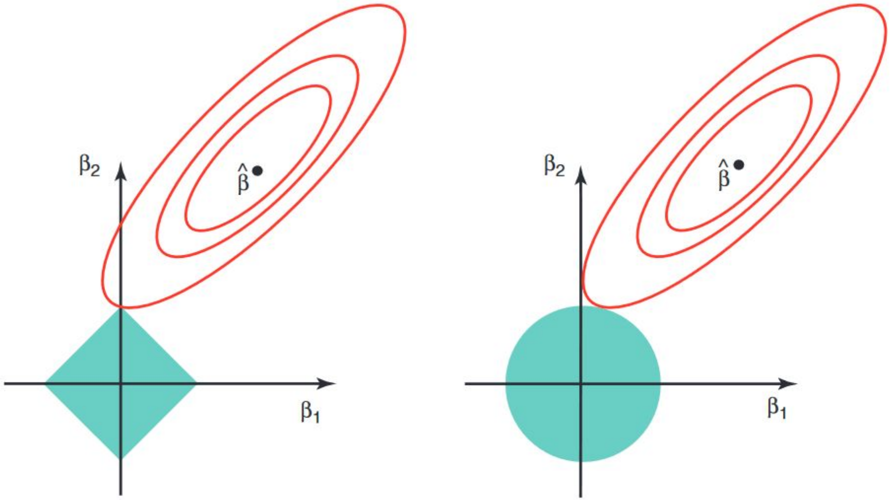
Loss with L-1 penalty:

$$L(\theta) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \theta^\top x_i)^2 + \boxed{\alpha\|\theta\|_1}$$

- Encourages sparsity → some coefficients exactly 0
- Automatic feature selection
- Harder optimization (non-differentiable at 0)

Ridge vs Lasso (Geometry)

Lasso	Ridge
diamond constraint → corners	circular constraint → smooth shrinkage



Summary of Linear Regression

- Strengths: simple, interpretable, fast baseline
- Limitations: sensitive to outliers, assumes linearity
- Extensions: logistic regression, neural networks

NB vs. LR

Comparison

Aspect	Naive Bayes (Generative)	Linear Regression (Discriminative)
Model	Learns joint P(x,y)	Learns conditional P(y∣x) directly
Philosophy	Recipe of data → simulate how data is generated	Boundary drawing → just separate outcomes
Features	Like word presence (independent assumption)	Fits weighted sum of features
Strengths	Simple, fast, good for text, small data	Interpretable, flexible, accurate with enough data
Limitations	Independence assumption unrealistic	Sensitive to outliers, needs more data

Case Study