Lecture 7: Naive Bayes & Probabilistic Models

Luu Minh Sao Khue

Bayesian reasoning, Gaussian and Multinomial Naive Bayes, the independence assumption, and comparison with logistic regression. Includes spam classification as a case study.

1. Learning Objectives

After this lecture, you will be able to:

- Recall and apply Bayes' theorem in classification.
- Explain prior, likelihood, and posterior probabilities.
- Understand the independence assumption in Naive Bayes.
- Differentiate Gaussian and Multinomial Naive Bayes models.
- Compare Naive Bayes with logistic regression conceptually and practically.
- Implement and evaluate a spam classifier using Naive Bayes.

2. Intuition

Naive Bayes (NB) is a **probabilistic classifier** based on **Bayes' theorem**. It predicts the most likely class C_k given a feature vector $x = (x_1, x_2, \ldots, x_n)$ by computing

$$P(C_k \mid x) = \frac{P(x \mid C_k)P(C_k)}{P(x)}.$$

Because $P(x)$ is the same for all classes, we only need the numerator:

$$P(C_k \mid x) \propto P(x \mid C_k)P(C_k).$$

NB assumes features are **conditionally independent given the class**, which makes estimation tractable even for high-dimensional data:

$$P(x \mid C_k) = \prod_{i=1}^{n} P(x_i \mid C_k).$$

Despite the strong independence assumption, NB performs surprisingly well in practice—especially for text classification tasks like spam detection, where word occurrences are almost independent given the topic.

3. Bayes' Theorem Refresher

Let A and B be events with $P(B) > 0$. Then:

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

- $P(A)$ — prior probability (belief before observing data),
- $P(B \mid A)$ — likelihood (data probability given the hypothesis),
- $P(A \mid B)$ — posterior probability (updated belief after seeing data),
- $P(B)$ — evidence (normalization term).

In NB, each class C_k is a hypothesis, and x is the observed data. We classify by:

$$\hat{C} = \arg\max_{C_k} P(C_k) \prod_{i=1}^{n} P(x_i \mid C_k).$$

4. Independence Assumption

NB assumes all features x_i are independent given the class:

$$P(x_1, \ldots, x_n \mid C_k) = \prod_{i=1}^{n} P(x_i \mid C_k).$$

Although rarely true in reality, this assumption simplifies computation and reduces the number of parameters drastically—from exponential to linear in n.

Example. For a spam classifier:

$$P(\text{spam} \mid \text{email}) \propto P(\text{spam}) \prod_{i} P(\text{word}_i \mid \text{spam}).$$

5. Gaussian Naive Bayes (GNB)

Used for continuous features (e.g., age, income).

5.1 Model

Assume each feature x_i follows a Gaussian distribution within each class C_k:

$$P(x_i \mid C_k) = \frac{1}{\sqrt{2\pi\sigma_{ik}^2}} \exp\left(-\frac{(x_i - \mu_{ik})^2}{2\sigma_{ik}^2}\right).$$

5.2 Classification Rule

Compute log-posterior (for numerical stability):

$$\log P(C_k \mid x) \propto \log P(C_k) - \frac{1}{2} \sum_{i=1}^{n} \log(2\pi\sigma_{ik}^2) - \frac{1}{2} \sum_{i=1}^{n} \frac{(x_i - \mu_{ik})^2}{\sigma_{ik}^2}.$$

Choose class with highest log-probability.

6. Multinomial Naive Bayes (MNB)

Used for discrete or count data such as word frequencies in text.

6.1 Model

Let x_i denote the count of word i in the document. For class C_k:

$$P(x \mid C_k) = \frac{(\sum_i x_i)!}{\prod_i x_i!} \prod_i P(w_i \mid C_k)^{x_i},$$

where $P(w_i \mid C_k)$ is the probability of word i in class C_k, estimated by:

$$\hat{P}(w_i \mid C_k) = \frac{N_{ik} + \alpha}{N_k + \alpha d}.$$

Here:

- N_{ik} = count of word i in class C_k,
- $N_k = \sum_i N_{ik}$ = total word count in class C_k,
- d = number of distinct words,
- α = smoothing parameter (Laplace smoothing).

6.2 Decision Rule

$$\hat{C} = \arg\max_{C_k} \left[\log P(C_k) + \sum_i x_i \log P(w_i \mid C_k) \right].$$

7. Probabilistic Reasoning: Prior and Posterior

NB explicitly models:

- **Prior** $P(C_k)$ — how frequent each class is,
- **Likelihood** $P(x_i \mid C_k)$ — how likely each feature value is,
- **Posterior** $P(C_k \mid x)$ — final belief after observing data.

The model is generative — it defines a full joint probability distribution $P(x, C_k)$ and can generate new samples. In contrast, discriminative models (like logistic regression) model only $P(C_k \mid x)$ directly.

8. Naive Bayes vs Logistic Regression

	Naive Bayes	Logistic Regression
Type	Generative	Discriminative
Model	$P(x \mid y)P(y)$	$P(y \mid x)$
Assumption	Feature independence	Linear decision boundary
Output	Posterior via Bayes rule	Sigmoid of linear score
Parameters	Estimated via counts or MLE	Estimated via optimization
Strengths	Fast, low data, robust to noise	Better calibrated, flexible
Weaknesses	Strong independence assumption	Needs more data, slower

Mathematical Connection. If features are binary and independent, log odds in NB are approximately linear in x:

$$\log \frac{P(y=1 \mid x)}{P(y=0 \mid x)} = \log \frac{P(y=1)}{P(y=0)} + \sum_i \log \frac{P(x_i \mid y=1)}{P(x_i \mid y=0)}.$$

This looks like logistic regression with fixed coefficients.

9. Python Example: Spam Classification

9.1 Dataset and Preprocessing

We will use a simple text dataset with spam/ham labels.

```
from sklearn.datasets import fetch_20newsgroups
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report

# Load binary dataset (spam-like: talk.politics.misc vs rec.sport.hockey)
categories = ['talk.politics.misc', 'rec.sport.hockey']
data = fetch_20newsgroups(subset='all', categories=categories)
X_train, X_test, y_train, y_test = train_test_split(
    data.data, data.target, test_size=0.3, random_state=42)

# Convert text to bag-of-words
vectorizer = CountVectorizer(stop_words='english', max_features=5000)
X_train_vec = vectorizer.fit_transform(X_train)
X_test_vec = vectorizer.transform(X_test)
```

9.2 Train Naive Bayes and Logistic Regression

```
nb = MultinomialNB(alpha=1.0)
lr = LogisticRegression(max_iter=1000)
```

```
nb.fit(X_train_vec, y_train)
lr.fit(X_train_vec, y_train)

print("=== Naive Bayes ===")
print(classification_report(y_test, nb.predict(X_test_vec)))

print("=== Logistic Regression ===")
print(classification_report(y_test, lr.predict(X_test_vec)))
```

9.3 Interpretation

- NB usually converges faster and works well with sparse data.
- Logistic regression can outperform NB when features are correlated.
- Both models output probabilities; threshold tuning improves precision/recall.

10. Practical Tips

- Apply Laplace smoothing ($\alpha > 0$) to avoid zero probabilities.
- Use log-probabilities to prevent underflow when multiplying many small values.
- Standardize features for Gaussian NB; use raw counts or TF–IDF for text NB.
- NB is a strong baseline for text, email, and document classification.
- Logistic regression is more flexible for correlated or continuous features.

11. Summary

- Naive Bayes applies Bayes' theorem with an independence assumption:

$$P(C_k \mid x) \propto P(C_k) \prod_i P(x_i \mid C_k).$$

- Gaussian NB models continuous features; Multinomial NB models counts.
- Probabilistic reasoning involves prior, likelihood, and posterior.
- NB is generative and simple but assumes independence.
- Logistic regression is discriminative and flexible.
- In practice: NB is fast, interpretable, and performs well on text data.

12. Exercises

1. Derive the Naive Bayes decision rule $\hat{C} = \arg\max_k P(C_k) \prod_i P(x_i \mid C_k)$.
2. Show that log-odds of NB are linear under binary features.
3. Implement Gaussian NB and compare with logistic regression on a numeric dataset.
4. Apply Laplace smoothing with different α values and evaluate accuracy.
5. Compare calibration curves of NB and logistic regression.
6. Explain why NB is a generative model while logistic regression is not.