Comprehensive Notes: Probabilistic Classification and Naïve Bayes

Table of Contents

---

1. Introduction to Bayesian Classification {#introduction}

Definition

A **probabilistic classifier** performs probabilistic prediction by predicting class membership probabilities rather than making hard classifications.

Key Characteristics

- **Foundation**: Based on Bayes' Theorem
- **Performance**: Comparable to decision trees and other classifiers
- **Incremental Learning**: Each training example can incrementally increase/decrease hypothesis probability
- **Baseline Standard**: Provides optimal decision-making baseline for comparison with other methods

Classification Problem Setup

- **Input**: Set of records with attributes $X_1, X_2, ..., X_n, Y$
- **Features**: $X_1, X_2, ..., X_n$ (predictor variables)
- **Class**: Y (target variable)
- **Task**: Predict class Y for new records where Y is unknown
- **Output**: Class prediction + probabilistic confidence score

---

2. Probability Theory Fundamentals {#probability-fundamentals}

Probability as Uncertainty Measure

- Assigns numerical degree of belief between 0 and 1 to each event
- Provides framework for characterizing uncertainty

Types of Random Variables

Boolean Random Variables

- Binary outcomes: True/False
- Example: cavity = {true, false}

Discrete Random Variables

- Finite set of possible values
- Example: weather = {sunny, rainy, cloudy, snow}
- Probabilities: P(weather = sunny), P(weather = rainy), etc.

Continuous Random Variables

- Infinite range of possible values
- Example: temperature (continuous values)
- Handled through:
  - **Discretization**: Binning into ranges (< 10, [10, 20], > 20)
  - **Probability Density Functions**: e.g., Normal distribution

Prior vs. Posterior Probabilities

Prior Probability

- Probability before evidence is obtained
- Notation: P(A)
- Example: P(rain) = 0.1

Posterior Probability

- Probability after evidence is obtained
- Notation: P(A|B) - "probability of A given B"
- Example: P(rain|cloudy) = 0.8

---

3. Bayes' Theorem {#bayes-theorem}

Simple Form

For two events A and C:

$$P(C|A) = P(A \cap C) / P(A) = P(A|C) \times P(C) / P(A)$$

Interpretation: Links prior probabilities of two events with their posterior probabilities

Medical Diagnosis Example

Problem: Probability of disease given positive test result

Given:

- P(C) = 0.01 (1% population has disease)
- P(¬C) = 0.99 (99% population healthy)
- P(A|C) = 0.95 (95% sensitivity - test positive when disease present)
- P(A|¬C) = 0.06 (6% false positive rate)

Solution:

$$P(C|A) = P(A|C) \times P(C) / P(A)$$
where $P(A) = P(A|C) \times P(C) + P(A|\neg C) \times P(\neg C)$
$P(A) = 0.95 \times 0.01 + 0.06 \times 0.99 = 0.0689$
$P(C|A) = (0.95 \times 0.01) / 0.0689 = 0.1379 = 13.79\%$

Key Insight: Even with 95% test accuracy, only 13.79% chance of having disease when testing positive!

General Form for Classification

$$P(Y|X_1, X_2, ..., X_m) = P(X_1, X_2, ..., X_m|Y) \times P(Y) / P(X_1, X_2, ..., X_m)$$

Where:

- Y = class variable
- $X_1, X_2, ..., X_m$ = feature attributes

---

4. Naïve Bayes Classifiers {#naive-bayes-classifiers}

The Combinatorial Problem

- With m attributes and k values each: k^m possible combinations
- Impractical to compute joint probabilities for all combinations

Simplification Strategy

Since we only need relative probabilities for comparison:

$$P(Y|X_1, ..., X_m) \propto P(X_1, ..., X_m|Y) \times P(Y)$$

(We can ignore the denominator $P(X_1, ..., X_m)$)

Conditional Independence Assumption

Key Assumption: Each attribute is conditionally independent of every other attribute given the class label

$$P(X_1, X_2, ..., X_m|Y) = P(X_1|Y) \times P(X_2|Y) \times ... \times P(X_m|Y)$$

Final Naïve Bayes Formula

$$P(Y|X_1, ..., X_m) \propto P(X_1|Y) \times P(X_2|Y) \times ... \times P(X_m|Y) \times P(Y)$$

Detailed Example: Computer Purchase Prediction

Dataset: Customer data with features (age, income, student, credit_rating) → buys_computer

Test Instance: X = (age=youth, income=medium, student=yes, credit_rating=fair)

Step 1: Calculate prior probabilities

- P(buys_computer = yes) = 9/14 = 0.643
- P(buys_computer = no) = 5/14 = 0.357

Step 2: Calculate conditional probabilities For buys_computer = yes:

- P(age=youth|yes) = 2/9 = 0.222
- P(income=medium|yes) = 4/9 = 0.444
- P(student=yes|yes) = 6/9 = 0.667
- P(credit_rating=fair|yes) = 6/9 = 0.667

For buys_computer = no:

- P(age=youth|no) = 3/5 = 0.6
- P(income=medium|no) = 2/5 = 0.4
- P(student=yes|no) = 1/5 = 0.2
- P(credit_rating=fair|no) = 2/5 = 0.4

Step 3: Calculate final probabilities

$$P(X|yes) \times P(yes) = 0.222 \times 0.444 \times 0.667 \times 0.667 \times 0.643 = 0.028$$
$$P(X|no) \times P(no) = 0.6 \times 0.4 \times 0.2 \times 0.4 \times 0.357 = 0.007$$

Result: Since 0.028 > 0.007, predict buys_computer = yes

---

5. Implementation Details {#implementation-details}

Numerical Underflow Problem

Issue: With many attributes, probability products become extremely small, causing underflow

Solution: Use logarithms

$$\log(p_1 \times p_2 \times \ldots \times p_m) = \log(p_1) + \log(p_2) + \ldots + \log(p_m)$$

Benefits:

- Converts multiplication to addition
- Preserves relative ratios
- Avoids numerical underflow

Zero Count Problem

Issue: If an attribute value never appears with a class, P(X|Y) = 0, making entire product zero

Example:

- 1000 training examples for buys_computer = yes
- 0 examples with income = low
- Result: P(income=low|yes) = 0

Solution: Laplace Smoothing (Add-one)

- Add 1 to all counts before calculating probabilities

- Formula: P(X|Y) = (count(X,Y) + 1) / (count(Y) + |V|)
- Where |V| is the number of possible values for attribute X

Example with Laplace Smoothing:

- Original: 0/1000 = 0, 990/1000 = 0.990, 10/1000 = 0.010
- With smoothing: 1/1003 = 0.001, 991/1003 = 0.988, 11/1003 = 0.011

Alternative: Add-ε Smoothing

For smaller datasets where adding 1 has significant impact, add small ε > 0 instead

---

6. Handling Continuous Features {#continuous-features}

Two Main Approaches

1. Discretization/Binning

- Divide continuous range into discrete intervals
- Example: income ranges (-∞, 30k], (30k, 60k], (60k, +∞)
- Treat each bin as categorical value

2. Gaussian Distribution Assumption

Assume continuous features follow normal distribution

Probability Density Function:

$$f(x, \mu, \sigma) = (1/\sqrt{(2\pi\sigma^2)}) \times e^{(-(x-\mu)^2/(2\sigma^2))}$$

Parameter Estimation:

$$\text{Mean: } \mu = (1/N) \times \Sigma_{i=1}^{N} x_i$$
$$\text{Variance: } \sigma^2 = (1/N) \times \Sigma_{i=1}^{N} (x_i - \mu)^2$$

Practical Example

Income data for buys_computer = yes: [30, 36, 47, 50, 56, 60, 63, 70, 110] (K dollars)

Calculations:

- Mean μ = 58K

- Variance $\sigma^2 = 481.56$
- Standard deviation $\sigma = 21.94$

Probability calculation:

```
P(income = 47|buys_computer = yes) = f(47, 58, 21.94) = 0.016
```

Technical Note on Continuous Probabilities

- Technically, P(X = exact_value) = 0 for continuous variables
- We actually compute $P(r \leq X \leq r + \varepsilon) \approx f(X, \mu, \sigma) \times \varepsilon$
- Since ε is constant across classes, it cancels out in comparisons

Python Implementation with SciPy

python

```python
import numpy as np
from scipy.stats import norm
import matplotlib.pyplot as plt

# Example data
data = range(9)
mu = np.mean(data)
sigma = np.std(data)

# Generate PDF
x = np.arange(-6, 14, 0.1)
y = norm.pdf(x, mu, sigma)
plt.plot(x, y)
plt.show()
```

---

7. Practical Considerations {#practical-considerations}

Scikit-learn Implementation

For Categorical Features

python

```python
from sklearn.naive_bayes import CategoricalNB
classifier = CategoricalNB()
```

For Continuous Features (Gaussian)

```python
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
```

When to Use Each Approach

- **CategoricalNB**: Discrete/categorical features
- **GaussianNB**: Continuous features that approximately follow normal distribution
- **MultinomialNB**: Count-based features (e.g., word frequencies)

Performance Optimization Tips

1. **Feature Selection**: Remove irrelevant features to reduce noise
2. **Feature Scaling**: For continuous features, consider normalization
3. **Cross-validation**: Use k-fold CV to estimate performance
4. **Hyperparameter Tuning**: Adjust smoothing parameters

---

8. Advantages and Disadvantages {#advantages-disadvantages}

Advantages ✅

1. Simplicity

- Easy to implement and understand
- Fast training and prediction
- Minimal computational requirements

2. Performance

- Good baseline performance in most cases
- Competitive with more complex algorithms
- Works well with small datasets

3. Theoretical Foundation

- Based on solid probability theory
- Provides probabilistic outputs (not just classifications)

- Handles missing values naturally

4. Scalability

- Incremental learning possible
- Handles large feature spaces well
- Memory efficient

Disadvantages ❌

1. Independence Assumption

- **Main limitation**: Assumes conditional independence between features
- **Reality**: Most real-world features are correlated
- **Impact**: Can lead to suboptimal performance

2. Dependency Examples

- **Medical diagnosis**: Symptoms like fever and cough are often correlated
- **Text classification**: Words in documents are contextually related
- **Image recognition**: Pixel values are spatially correlated

3. Categorical Data Issues

- Requires smoothing for unseen categories
- Can be sensitive to irrelevant features
- May not capture complex feature interactions

Addressing Limitations

1. Feature Engineering

- Create composite features to capture interactions
- Use domain knowledge to select relevant features
- Apply dimensionality reduction techniques

2. Advanced Bayesian Methods

- **Bayesian Belief Networks**: Model dependencies between variables
- **Tree-Augmented Naïve Bayes**: Allows limited dependencies
- **Semi-Naïve Bayes**: Relaxes independence for some features

3. Ensemble Methods

- Combine Naïve Bayes with other classifiers
- Use voting or stacking approaches
- Leverage complementary strengths

Summary

Naïve Bayes classifiers represent a powerful and elegant application of probability theory to machine learning. Despite their "naïve" assumption of conditional independence, they often perform surprisingly well in practice and serve as an excellent baseline for classification tasks.

Key Takeaways:

1. Based on solid mathematical foundation (Bayes' theorem)
2. Simple to implement and interpret
3. Handles both categorical and continuous features
4. Requires careful handling of numerical issues (underflow, zero counts)
5. Independence assumption is main limitation but often manageable
6. Excellent starting point for classification problems

When to Use:

- Text classification and spam filtering
- Medical diagnosis systems
- Real-time applications requiring fast predictions
- Baseline comparisons for other algorithms
- Problems with many features relative to training examples

When to Consider Alternatives:

- Strong feature dependencies exist
- Complex non-linear relationships present
- High accuracy requirements with sufficient training data
- Features have complex interactions that cannot be captured independently