Basic Probability and Its Applications in AI

Bui Van Tai

Probability helps us understand uncertainty and make predictions. In this document, we'll learn basic probability concepts and see how they are used in artificial intelligence.

1. Introduction

Experiment and Event

An **experiment** is any procedure that can be repeated multiple times under the same conditions and has a well-defined set of possible outcomes. The set of all possible outcomes is called the **sample space**, usually denoted by Ω or S.

An **event** is a subset of the sample space - it's a collection of possible outcomes that we're interested in. Events are typically denoted by capital letters like A, B, etc.

Example: Consider the experiment of rolling a six-sided dice. In this case:

- The **experiment** is the act of rolling the dice
- The **sample space** $\Omega = \{1, 2, 3, 4, 5, 6\}$ contains all possible outcomes
- Some examples of **events** could be:
  - Event A: Rolling an odd number $= \{1, 3, 5\}$
  - Event B: Rolling a even number $= \{2, 4, 6\}$
  - Event C: Rolling a prime number $= \{2, 3, 5\}$

Each time we perform this experiment, exactly one outcome from the sample space will occur. An event is said to occur if the actual outcome is one of the outcomes in that event's set.

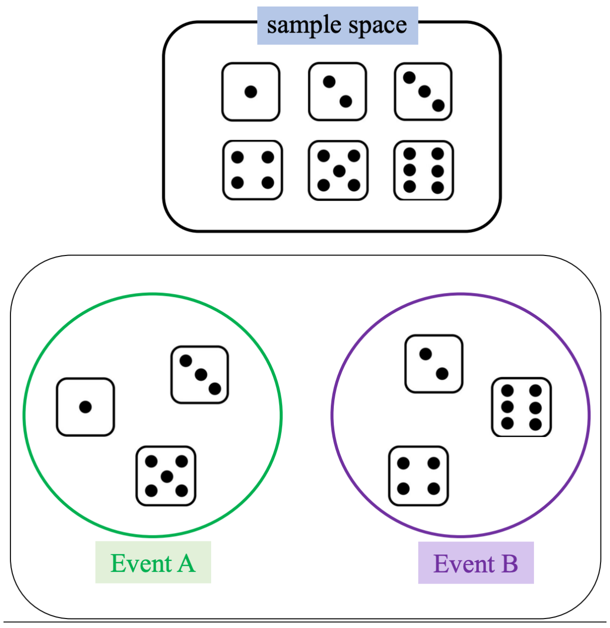

Figure 1: Example of Experiment and Event

Operations on Event

Union of Events:

The union of two events A and B, denoted $A \cup B$, is the event that occurs if either A or B or both occur.
Example:

$$\begin{aligned} A &= \{1,3,5\} \\ B &= \{2,4,6\} \\ A \cup B &= \{1,2,3,4,5,6\} \end{aligned}$$

Intersection of Events:

The intersection of two events A and B, denoted $A \cap B$, is the event that occurs if both A and B occur.
Example:

$$\begin{aligned} A &= \{1,3,5\} \\ B &= \{2,4,6\} \\ A \cap B &= \{2,4,6\} \end{aligned}$$

Complementary Event:

The complementary event of an event A, denoted A^c, is the event that occurs if A does not occur.
Example:

$$\begin{aligned} A &= \{1,3,5\} \\ A^c &= \{2,4,6\} \end{aligned}$$

Mutually Exclusive Events:

Two events A and B are mutually exclusive if they cannot occur at the same time, i.e., $A \cap B = \emptyset$.
Example:

$$\begin{aligned} A &= \{1,3,5\} \\ B &= \{2,4,6\} \\ A \cap B &= \emptyset \end{aligned}$$

Disjoint Events

Two events A and B are disjoint if they cannot occur at the same time, i.e., $A \cap B = \emptyset$.
Example:

$$\begin{aligned} A &= \{\text{getting heads}\} \\ B &= \{\text{getting tails}\} \\ A \cap B &= \emptyset \end{aligned}$$

2. Probability & Bayes' Rule

Classical Geometric definition

Empirical probability (experimental probability)

Rule of probability

Total probability

Bayes' Rule

3. Naive Bayes Classifier

3.1 Naive Bayes Classifier for Discrete Random Variables

Problem Example: Predicting Whether to Play Tennis

We are given a small dataset to predict whether a person will play tennis based on four **discrete features** (a "naive" hypothesis, yet effective): *Outlook*, *Temperature*, *Humidity* and *Wind.*

Dataset (5 Samples)

Outlook	Temp	Humidity	Wind	Play
Sunny	Hot	High	Weak	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	High	Strong	No
Sunny	Cool	High	Strong	???

Step 1: Bayes's Theorem Overview

We want to compute:

$$P(\text{Play=Yes} \mid X), \quad P(\text{Play=No} \mid X)$$

where

$$X = (\text{Outlook} = \text{Sunny}, \text{Temp} = \text{Cool}, \text{Humidity} = \text{High}, \text{Wind} = \text{Strong})$$

According to Bayes's Theorem:

$$P(\text{Class} \mid X) = \frac{P(X \mid \text{Class}) \cdot P(\text{Class})}{P(X)}$$

Since $P(X)$ is constant for both classes, we compare:

$$P(X \mid \text{Yes}) \cdot P(\text{Yes}) \quad \text{and} \quad P(X \mid \text{No}) \cdot P(\text{No})$$

Step 1: Compute Prior Probabilites

From the dataset:

$$P(\text{Yes}) = \frac{3}{5} = 0.6 \quad , \quad P(\text{No}) = \frac{2}{5} = 0.4$$

Step 2: Likelihoods (with Laplace Smoothing)

Apply Laplace smoothing (add-one) to avoid zero probabilities

$$P(x_i \mid \text{Class}) = \frac{\text{count}(x_i, \text{Class}) + 1}{\text{count}(\text{Class}) + N_i}$$

Where N_i is the number of possible values for feature i.
- Outlook: 3 values (Sunny, Overcast, Rain)

- Temperature: 3 values (Hot, Mild, Cool)
- Humidity: 2 values (High, Normal)
- Wind: 2 values (Weak, Strong)

Step 3: Compute Likelihoods

For Class = Yes (3 samples)

$$P(X \mid \text{Yes}) = P(\text{Sunny} \mid \text{Yes}) \cdot P(\text{Cool} \mid \text{Yes}) \cdot P(\text{High} \mid \text{Yes}) \cdot P(\text{Strong} \mid \text{Yes})$$

$$= \frac{1+1}{3+3} \cdot \frac{1+1}{3+3} \cdot \frac{2+1}{3+2} \cdot \frac{0+1}{3+2} = \frac{2}{6} \cdot \frac{2}{6} \cdot \frac{3}{5} \cdot \frac{1}{5} = \frac{12}{900}$$

$$P(X \mid \text{Yes}) \cdot P(\text{Yes}) = \frac{12}{900} \cdot 0.6 = \frac{7.2}{900} \approx 0.008$$

For Class = No (2 samples)

$$P(X \mid \text{No}) = P(\text{Sunny} \mid \text{No}) \cdot P(\text{Cool} \mid \text{No}) \cdot P(\text{High} \mid \text{No}) \cdot P(\text{Strong} \mid \text{No})$$

$$= \frac{1+1}{2+3} \cdot \frac{0+1}{2+3} \cdot \frac{2+1}{2+2} \cdot \frac{1+1}{2+2} = \frac{2}{5} \cdot \frac{1}{5} \cdot \frac{3}{4} \cdot \frac{2}{4} = \frac{12}{400}$$

$$P(X \mid \text{No}) \cdot P(\text{No}) = \frac{12}{400} \cdot 0.4 = \frac{4.8}{400} = 0.012$$

Conclusion

Since:

$$P(\text{No} \mid X) > P(\text{Yes} \mid X)$$

We conclude that the person will **not play tennis**.

Python Code

```python
# Training data (5 samples)
data = [
    ['Sunny', 'Hot', 'High', 'Weak', 'No'],
    ['Overcast', 'Hot', 'High', 'Weak', 'Yes'],
    ['Rain', 'Mild', 'High', 'Weak', 'Yes'],
    ['Sunny', 'Cool', 'Normal', 'Weak', 'Yes'],
    ['Rain', 'Mild', 'High', 'Strong', 'No']
]

# Input to predict
X = ['Sunny', 'Cool', 'High', 'Strong']

# Extract labels and features
labels = [row[-1] for row in data]
features = [row[:-1] for row in data]

# Calculate prior probability P(Class)
def prior_prob(class_label):
    return sum(1 for label in labels if label == class_label) / len(labels)

# Calculate conditional probability P(x_i | Class) with Laplace smoothing
def cond_prob(feature_idx, feature_val, class_label):
    count = 0
    total = 0
    unique_vals = set(row[feature_idx] for row in features)

    for i, row in enumerate(features):
        if labels[i] == class_label:
            total += 1
            if row[feature_idx] == feature_val:
                count += 1

    # Apply Laplace smoothing
    return (count + 1) / (total + len(unique_vals))

# Compute P(X | Yes) * P(Yes)
yes_prob = prior_prob('Yes')
for i in range(len(X)):
    yes_prob *= cond_prob(i, X[i], 'Yes')

# Compute P(X | No) * P(No)
no_prob = prior_prob('No')
for i in range(len(X)):
    no_prob *= cond_prob(i, X[i], 'No')

# Final prediction
print(f"P(X | Yes) * P(Yes) = {yes_prob}")
print(f"P(X | No) * P(No) = {no_prob}")
if yes_prob > no_prob:
    print("=> Prediction: Play")
else:
    print("=> Prediction: Do not play")
```

Listing 1: Naive Bayes Classifier

Output

```
P(X | Yes) * P(Yes) = 0.008
P(X | No) * P(No) = 0.012
=> Prediction: Do not play
```

3.2 Naive Bayes for Continuous Random Variable

Unlike discrete variables, which can take only a finite or countable set of distinct values, *continuous variables* can take infinite number of possible values (height, weight, temperature,...).

This example shows how to apply Navie Bayes Classifier to continuous data using Gaussian distribution.

We are given 5 BMI values and their corresponding class labels:

BMI (x)	Class (y)
18	0 (Healthy)
20	0
22	0
26	1 (Sick)
28	1
24	???

We want to classify a new sample with BMI=24.

Step 1: Compute mean and variance for each class

For class C_0 (Healthy):

General formulas:

$$\mu_0 = \frac{1}{n_0}\sum_{i=1}^{n_0} x_i \quad \textit{and} \quad \sigma_0^2 = \frac{1}{n_0}\sum_{i=1}^{n_0}(x_i - \mu_0)^2$$

Apply to data: [18, 20, 22]

$$\mu_0 = \frac{18+20+22}{3} = 20$$

$$\sigma_0^2 = \frac{(18-20)^2 + (20-20)^2 + (22-20)^2}{3} = \frac{8}{3} \approx 2.67$$

$$\sigma_0 = \sqrt{2.67} \approx 1.63$$

For Class C_1 (Sick):

Apply to data: [26, 28]

$$\mu_1 = \frac{26+28}{2} = 27 \quad , \quad \sigma_1^2 = \frac{(26-27)^2 + (28-27)^2}{2} = 1 \quad , \quad \sigma_1 = \sqrt{1} = 1$$

Step 2: Compute Pior Probabilities

$$P(C_0) = \frac{n_0}{n} = \frac{3}{5} = 0.6$$

$$P(C_1) = \frac{n_1}{n} = \frac{2}{5} = 0.4$$

Step 3: Use Gaussian Probability Density Function

General formula:

$$P(x|C_k) = \frac{1}{\sqrt{2\pi\sigma_k^2}} \cdot e^{-\frac{(x-\mu_k)^2}{2\sigma_k^2}}$$

For class 0:

$$P(x=24|C_0) = \frac{1}{\sqrt{2\pi \cdot 2.67}} \cdot e^{-\frac{(24-20)^2}{2 \cdot 2.67}} \approx 0.0122$$

$$P(C_0|x=24) \propto 0.0122 \cdot 0.6 = 0.0073$$

For class 1:

$$P(x=24|C_1) = \frac{1}{\sqrt{2\pi \cdot 1}} \cdot e^{-\frac{(24-27)^2}{2 \cdot 1}} \approx 0.0044$$

$$P(C_1|x=24) \propto 0.0044 \cdot 0.4 = 0.0018$$

Step 4: Final Prediction

$$P(C_0|x=24) \propto 0.0073 \quad P(C_1|x=24) \propto 0.0018 \Rightarrow \text{Predict Class 0 (Healthy)}$$

Python Code

We use scikit-learn library, which provides efficient and easy-to-use tools for implementing Naive Bayes Classifier models, making the process simpler.

```python

from sklearn.naive_bayes import GaussianNB
import numpy as np

# Input data
X = np.array([[18], [20], [22], [26], [28]])
y = np.array([0, 0, 0, 1, 1])

# Train model
model = GaussianNB()
model.fit(X, y)

# Predict for BMI = 24
x_test = np.array([[24]])
predicted_class = model.predict(x_test)
print("Predicted class:", predicted_class[0])
```

Listing 2: NBC For Continuous Random Variable

```
Predicted class: 0
```

Bayes'theorem demonstrates its wide-ranging and adaptable applicability in various problem domains and data types.

Discussion and Conclusion

The results of the experiments demonstrate that the Naive Bayes classifier, despite its simplicity, can make predictions that are intuitively logical. For example, in the tennis scenario, the model predicted that the person would not play tennis when the weather conditions were "Sunny, Cool, High humidity, and Strong wind". This aligns with common sense: if the weather is too extreme (e.g., strong wind or high humidity), it is less likely someone would want to play tennis.

However, while the model gives reasonable results, it also has clear limitations. The most important one lies in its core assumption: Naive Bayes treats all features as **conditionally independent given the class**. In reality, weather features such as temperature and humidity, or wind and outlook, are often correlated. This assumption prevents the model from capturing interactions between features — Ignoring such dependencies may oversimplify the model and reduce accuracy in more complex datasets.

Despite these limitations, Naive Bayes remains a valuable tool, especially for classification tasks where speed and simplicity are prioritized. Its interpretability makes it ideal for introductory use in artificial intelligence and for baseline comparisons in real-world applications.