Mathematical foundation

Introduction to Probability

1. Conceptual Overview

Probability is a measure of **how likely** an event is to happen.

- It's always a number between **0** (impossible) and **1** (certain).
- In machine learning, probability helps us **model uncertainty**, make predictions, and evaluate risks.
- Common uses in real life:
  - **Risk assessment**: Probability of project failure.
  - **Forecasting**: Probability of hitting sales targets.
  - **Quality control**: Probability that a product is defective.

Key idea: Probability connects uncertain real-world situations to mathematical models so that we can make informed decisions.

---

2. Mathematical Foundation

If all outcomes are equally likely:

$$P(E) = \frac{\text{Number of favorable outcomes}}{\text{Total number of outcomes}}$$

Where:

- P(E) = Probability of event E
- Values range: $0 \leq P(E) \leq 1$

Example: Tossing a fair coin:

- Sample space: S={H,T}
- Probability of getting heads:

$P(\text{Heads}) = \frac{1}{2}$

If outcomes are not equally likely, we use:

$P(E) = \sum_{x \in E} p(x)$

where p(x) is the probability assigned to outcome x.

3. Practical Example (Python)

```
import random

# Simulate probability of getting heads in 10,000 coin tosses
trials = 10000
heads_count = sum(1 for _ in range(trials) if random.choice(["H", "T"]) == "H")
prob_heads = heads_count / trials

print(f"Estimated Probability of Heads: {prob_heads:.4f}")
```

4. Key Takeaways

- Probability measures **likelihood** from 0 to 1.
- **0** means impossible, **1** means certain.
- Basic formula: $P(E) = \frac{\text{favorable}}{\text{total outcomes}}$ for equally likely events.
- In ML, probability is essential for **modeling uncertainty**, **Bayesian methods**, and **evaluation metrics**.

Experiments, Outcomes, Sample Space, and Events

1. Conceptual Overview

These are the **building blocks** of probability theory.

They help us describe and structure uncertain situations in a formal way.

- **Experiment**: A procedure or process with a set of possible outcomes, which can be repeated under identical conditions.
    - Example: Tossing a coin, rolling a die, picking a card.
- **Outcome**: A single possible result of the experiment.
    - Example: Getting "Heads" in a coin toss, rolling a "4" on a die.
- **Sample Space (S)**: The set of all possible outcomes of an experiment.
    - Example: For a coin toss, S={H,T}.
- **Event (E)**: A subset of the sample space that represents outcomes of interest.
    - Example: "Rolling an even number" when rolling a die → E={2,4,6}.

---

2. Mathematical Foundation

- **Notation**:
    - Experiment → $\mathcal{E}$
    - Sample space → $S = \{\omega_1, \omega_2, \ldots, \omega_n\}$
    - Event $E \subseteq S$
- **Probability of Event**:

$$P(E) = \frac{|E|}{|S|} \quad \text{(if equally likely outcomes)}$$

- **Important Relationships**:
    - P(S) = 1 (Probability of the whole sample space is always 1)
    - $P(\emptyset) = 0$ (Empty set has zero probability)
    - If $E_1 \cap E_2 = \emptyset$ (mutually exclusive events):
      $P(E_1 \cup E_2) = P(E_1) + P(E_2)$

---

3. Practical Example (Python)

```
# Rolling a fair six-sided die
import random

S = [1, 2, 3, 4, 5, 6]  # Sample space
E_even = {2, 4, 6}       # Event: rolling an even number

# Simulate
trials = 10000
count_even = sum(1 for _ in range(trials) if random.choice(S) in E_even)

prob_even = count_even / trials
print(f"Estimated Probability of rolling even: {prob_even:.4f}")
```

---

4. Key Takeaways

- **Experiment** → well-defined procedure
- **Outcome** → a single possible result
- **Sample space (S)** → set of all outcomes
- **Event (E)** → subset of S
- For equally likely outcomes: $P(E) = \frac{|E|}{|S|}$
- Probability rules:
  - P(S) = 1
  - $P(\emptyset) = 0$
  - Add probabilities for **mutually exclusive** events

Special Case: Equally Likely Outcomes

1. Conceptual Overview

In many simple probability problems, we assume **all outcomes in the sample space have the same chance of occurring**.

This is the **equally likely outcomes** case, which makes probability calculation straightforward.

- Examples:
  - Rolling a fair six-sided die → Each face has a probability $\frac{1}{6}$
  - Tossing a fair coin → Each side has a probability $\frac{1}{2}$
  - Drawing a card from a well-shuffled standard deck → Each card has a probability $\frac{1}{52}$

---

2. Mathematical Foundation

If **all outcomes are equally likely**:

$$P(E) = \frac{\text{Number of favorable outcomes}}{\text{Total number of outcomes}}$$

Where:

- E = event of interest
- "Favorable outcomes" = outcomes in E
- "Total outcomes" = number of elements in S

Example:

A batch has 100 mangoes, 5 of which are rotten.

- Total outcomes: $|S|$ = 100
- Favorable outcomes (rotten mangoes): $|E|$ = 5

$P(\text{rotten}) = \frac{5}{100} = 0.05$

---

3. Practical Example (Python)

```
# Probability of picking a rotten mango
total_mangoes = 100
```

```
rotten_mangoes = 5

prob_rotten = rotten_mangoes / total_mangoes
print(f"Probability of picking a rotten mango: {prob_rotten:.2f}")
```

4. Key Takeaways

- **Equally likely outcomes** mean each outcome has the same probability.
- Formula: $P(E) = \frac{\text{favorable}}{\text{total}}$
- Often applies to **games of chance** (coins, dice, cards).
- Real-life problems may require adjusting if outcomes are **not** equally likely.

Random Variables (Definition and Types)

1. Conceptual Overview

A **Random Variable (RV)** is a **function** that assigns a **numerical value** to each outcome in a sample space.

It transforms qualitative or categorical outcomes into numbers we can analyze mathematically.

- **Why they matter in ML & statistics**:
  - They allow us to work with numbers rather than abstract events.
  - Used to define **probability distributions**.
  - Foundation for concepts like **expectation**, **variance**, and **hypothesis testing**.

Two main types:

1. **Discrete Random Variables** – take on **finite or countable** values.

   Example: Number of heads in 3 coin tosses (0, 1, 2, 3).

2. **Continuous Random Variables** – take on **uncountable** values (often from an interval).

Example: Height of students in a class.

2. Mathematical Foundation

A **Random Variable** is a function:

$X : S \to \mathbb{R}$

Where:

- S = sample space
- $\mathbb{R}$ = real numbers

Discrete RVs are described by a **Probability Mass Function (PMF)**:

$$p_X(x) = P(X = x), \quad \sum_x p_X(x) = 1$$

Continuous RVs are described by a **Probability Density Function (PDF)**:

$$f_X(x) \geq 0, \quad \int_{-\infty}^{\infty} f_X(x)\,dx = 1$$

And:

$$P(a \leq X \leq b) = \int abfX(x)\,dx P(a \leq X \leq b) = \int_a^b f_X(x)\,dx$$

3. Practical Example (Python)

```
import random

# Example: Random variable = "Number of heads in 3 coin tosses"
def simulate_heads(num_tosses=3):
    return sum(1 for _ in range(num_tosses) if random.choice(["H", "T"]) ==
"H")

trials = 10000
```

```
results = [simulate_heads() for _ in range(trials)]

# Approximate PMF
pmf = {k: results.count(k)/trials for k in set(results)}
print("PMF:", pmf)
```

---

4. Key Takeaways

- A **Random Variable** assigns numbers to outcomes of an experiment.
- **Discrete** RV → countable outcomes, described by PMF.
- **Continuous** RV → uncountable outcomes, described by PDF.
- They are the bridge between **probability theory** and **statistical analysis**.

Discrete Random Variables & Probability Mass Function (PMF)

---

1. Conceptual Overview

A **Discrete Random Variable** takes on a **finite** or **countably infinite** set of possible values.

Examples:

- Number of heads in 10 coin tosses ($0, 1, \ldots, 10$)
- Number of defective items in a batch
- Number of customers arriving in an hour (if modeled as count data)

The **Probability Mass Function (PMF)** describes how probability is distributed over the possible values of a discrete RV.

- It tells us **the probability that the RV equals a specific value**.

---

2. Mathematical Foundation

Let X be a discrete RV.

Its PMF is:

$$p_X(x) = P(X = x)$$

Where:

- $p_X(x) \geq 0$ for all x
- $\sum_x p_X(x) = 1$ (probabilities must sum to 1)

Example:

Toss a fair coin twice:

- Sample space S={HH,HT,TH,TT}
- Let X = number of heads.
- Possible values: 0, 1, 2

  PMF:

$$p_X(0) = \frac{1}{4}, \quad p_X(1) = \frac{2}{4}, \quad p_X(2) = \frac{1}{4}$$

---

3. Practical Example (Python)

```
import random

# Discrete RV: number of heads in 2 coin tosses
def toss_twice():
    return sum(1 for _ in range(2) if random.choice(["H", "T"]) == "H")

trials = 10000
outcomes = [toss_twice() for _ in range(trials)]

# Compute PMF
pmf = {value: outcomes.count(value)/trials for value in sorted(set(outcomes))}
print("PMF:", pmf)
```

---

4. Key Takeaways

- **Discrete RV**: countable set of possible values.
- **PMF**: assigns probabilities to each possible value.
- Properties:
  - Non-negative probabilities ($p_X(x) \geq 0$)
  - Probabilities sum to 1
- PMFs are used in **Bernoulli**, **Binomial**, **Poisson**, **Geometric**, etc. distributions.

Examples of Discrete Random Variables: Bernoulli, Binomial, Geometric

---

1. Conceptual Overview

These are **common discrete probability distributions** used in statistics and machine learning:

1. **Bernoulli Random Variable**
   - Represents a single trial with two possible outcomes: **success (1)** or **failure (0)**.
   - Example: Flipping a coin once (Heads = 1, Tails = 0).
2. **Binomial Random Variable**
   - Represents the **number of successes** in a fixed number of **independent Bernoulli trials**.
   - Example: Number of heads in 10 coin tosses.
3. **Geometric Random Variable**
   - Represents the **number of trials until the first success**.
   - Example: Number of coin tosses until we get a head.

---

2. Mathematical Foundation

1. **Bernoulli Distribution**
    - PMF:

$$p_X(x) = p^x(1-p)^{1-x}, \quad x \in \{0, 1\}$$

    - Mean: $\mu = p$
    - Variance: $\sigma^2 = p(1-p)$
2. **Binomial Distribution**
    - PMF:

$$p_X(k) = \binom{n}{k} p^k (1-p)^{n-k}, \quad k = 0, 1, \ldots, n$$

    - Mean: $\mu = np$
    - Variance: $\sigma^2 = np(1-p)$
3. **Geometric Distribution** (first success on trial k)
    - PMF:

$$p_X(k) = (1-p)^{k-1} p, \quad k = 1, 2, \ldots$$

    - Mean: $\mu = \frac{1}{p}$
    - Variance: $\sigma^2 = \frac{1-p}{p^2}$

---

3. Practical Example (Python)

```
import random

# Simulate Bernoulli, Binomial, Geometric

# Bernoulli trial
def bernoulli(p=0.5):
    return 1 if random.random() < p else 0
```

```
# Binomial: number of successes in n trials
def binomial(n=10, p=0.5):
    return sum(bernoulli(p) for _ in range(n))

# Geometric: number of trials until first success
def geometric(p=0.5):
    count = 0
    while True:
        count += 1
        if bernoulli(p) == 1:
            return count

# Test simulation
print("Bernoulli trial:", bernoulli(0.7))
print("Binomial sample:", binomial(10, 0.5))
print("Geometric sample:", geometric(0.3))
```

4. Key Takeaways

- **Bernoulli** → one trial, success/failure.
- **Binomial** → multiple trials, count the number of successes.
- **Geometric** → trials until first success.
- All three are **discrete** and based on the idea of independent trials with constant probability p.
- Widely used in ML for modeling **binary classification**, **success/failure experiments**, and **waiting time problems**.

Continuous Random Variables & Probability Density Function (PDF)

1. Conceptual Overview

A **Continuous Random Variable (CRV)** can take on **uncountably infinite values**, typically within an interval of real numbers.

- Example: Height of people, time taken to finish a task, sensor readings.
- Unlike discrete variables, the probability of a CRV taking **any exact value** is **0**.
    - Instead, we measure probabilities over **intervals**.

The **Probability Density Function (PDF)** describes how probability is distributed across values of a continuous random variable.

---

2. Mathematical Foundation

Let X be a continuous random variable.

- The PDF is a function $f_X(x)$ such that:
    1. $f_X(x) \geq 0$ for all x.
    2. The total probability is 1:

$$\int_{-\infty}^{\infty} f_X(x)\, dx = 1$$

- The probability that X lies between two values a and b is:

$$P(a \leq X \leq b) = \int_a^b f_X(x)\, dx$$

Important:

- P(X = x) = 0 (since exact points have no probability mass).
- Only **intervals** matter for continuous distributions.

---

3. Practical Example (Python)

Let's simulate from a **Uniform(0,1)** distribution (a common continuous RV).

```
import numpy as np
```

```
# Generate 10000 samples from Uniform(0,1)
samples = np.random.uniform(0, 1, 10000)

# Estimate probability that X lies between 0.2 and 0.5
prob = np.mean((samples >= 0.2) & (samples <= 0.5))

print(f"Estimated P(0.2 <= X <= 0.5): {prob:.3f}")
```

---

4. Key Takeaways

- **Discrete RVs** → PMF, probabilities for exact values.
- **Continuous RVs** → PDF, probabilities over intervals.
- For continuous RV:
  - P(X = x) = 0
  - $P(a \leq X \leq b) = \int_a^b f_X(x)\,dx$
- PDFs must integrate to 1 across their domain.
- Common continuous distributions: **Gaussian, Uniform, Exponential**.

Examples of Continuous Random Variables: Gaussian, Uniform, Exponential

---

1. Conceptual Overview

Three of the most important **continuous random variables** are:

1. **Gaussian (Normal) Distribution**
   - Symmetrical “bell curve.”
   - Models natural phenomena like heights, exam scores, and sensor noise.
   - Most widely used in statistics and ML (e.g., linear regression assumptions).
2. **Uniform Distribution**

    - All values in an interval are **equally likely**.
    - Example: Random number generator between 0 and 1.
3. **Exponential Distribution**
    - Models **waiting times** between independent events in a Poisson process.
    - Example: Time between arrivals of buses, time until a machine breaks.

---

2. Mathematical Foundation

1. **Gaussian (Normal) Distribution**
    - PDF:

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

    - Mean: μ, Variance: σ^2.
2. **Uniform Distribution (a, b)**
    - PDF:

$$f(x) = \begin{cases} \frac{1}{b-a}, & a \leq x \leq b \\ 0, & \text{otherwise} \end{cases}$$

    - Mean: $\mu = \frac{a+b}{2}$, Variance: $\sigma^2 = \frac{(b-a)^2}{12}$.
3. **Exponential Distribution**
    - PDF:

$$f(x) = \lambda e^{-\lambda x}, \quad x \geq 0$$

    - Mean: $\mu = \frac{1}{\lambda}$, Variance: $\sigma^2 = \frac{1}{\lambda^2}$.

---

3. Practical Example (Python)

```
import numpy as np
```

```
# Gaussian: mean=0, std=1
gaussian_samples = np.random.normal(0, 1, 10000)

# Uniform: range [0,1]
uniform_samples = np.random.uniform(0, 1, 10000)

# Exponential: lambda=1
exponential_samples = np.random.exponential(1, 10000)

print(f"Gaussian mean ~ {np.mean(gaussian_samples):.2f}, variance ~ {np.var(gaussian_samples):.2f}")
print(f"Uniform mean ~ {np.mean(uniform_samples):.2f}, variance ~ {np.var(uniform_samples):.2f}")
print(f"Exponential mean ~ {np.mean(exponential_samples):.2f}, variance ~ {np.var(exponential_samples):.2f}")
```

4. Key Takeaways

- **Gaussian**: Symmetric bell curve, defined by mean μ and variance σ^2.
- **Uniform**: Equal probability in the interval [a, b].
- **Exponential**: Models waiting times, parameterized by rate λ.
- These distributions are **fundamental building blocks** in probability, statistics, and ML models.

Expectation of Random Variables

1. Conceptual Overview

The **Expectation** (or **Expected Value**, **Mean**) of a random variable represents its **average value** if an experiment were repeated many times.

- Think of it as the **center of mass** of the probability distribution.
- In machine learning:

- Used in **loss functions** (e.g., expected error).
- Forms the basis of statistical estimators.
- Helps summarize distributions with a single number.

---

2. Mathematical Foundation

For a random variable X:

- **Discrete Random Variable (PMF $p_X(x)$)**:

$$E[X] = \sum_x x \, p_X(x)$$

- **Continuous Random Variable (PDF $f_X(x)$)**:

$$E[X] = \int_{-\infty}^{\infty} x \, f_X(x) \, dx$$

Properties of Expectation:

1. Linearity:

   E[aX + bY] = aE[X] + bE[Y]

2. For a constant c:

   E[c] = c

3. Expectation of indicator function:

   $E[\mathbf{1}_A] = P(A)$

Example (Dice Roll):

Let X = outcome of rolling a fair die.

- S={1,2,3,4,5,6}, $p(x) = \frac{1}{6}$.

$$E[X] = \sum_{x=1}^{6} x \cdot \frac{1}{6} = \frac{1+2+3+4+5+6}{6} = 3.5$$

---

3. Practical Example (Python)

```
import numpy as np

# Example: Expected value of rolling a fair die
die_faces = np.array([1, 2, 3, 4, 5, 6])
probabilities = np.repeat(1/6, 6)

# Theoretical expectation
expected_value = np.sum(die_faces * probabilities)
print("Theoretical Expectation:", expected_value)

# Simulation
samples = np.random.choice(die_faces, size=100000, p=probabilities)
print("Simulated Expectation:", np.mean(samples))
```

---

4. Key Takeaways

- **Expectation** = long-run average value of a random variable.
- Formula:
  - Discrete: $E[X] = \sum_x x\, p(x)$
  - Continuous: $E[X] = \int x f(x) dx$
- Expectation is **linear**, even if variables are dependent.
- Used widely in ML for **loss minimization, risk analysis, and parameter estimation**.

Variance of Random Variables

1. Conceptual Overview

While **expectation** tells us the *average* value of a random variable, **variance** tells us how much the values **spread out** around the mean.

- High variance → data points are spread far from the mean.
- Low variance → data points are clustered near the mean.
- In ML:
  - Variance measures model stability (high variance → overfitting).
  - Used in risk analysis, error estimation, and uncertainty quantification.

---

2. Mathematical Foundation

For a random variable X with mean $\mu = E[X]$:

- **Definition**:

$$\mathrm{Var}(X) = E\left[(X-\mu)^2\right]$$

- **Discrete Case**:

$$\mathrm{Var}(X) = \sum_x (x-\mu)^2\, p_X(x)$$

- **Continuous Case**:

$$\mathrm{Var}(X) = \int_{-\infty}^{\infty} (x-\mu)^2\, f_X(x)\, dx$$

- **Shortcut Formula**:

$$\mathrm{Var}(X) = E[X^2] - (E[X])^2$$

- **Standard Deviation (σ)**:

$$\sigma = \sqrt{\mathrm{Var}(X)}$$

Gives spread in the same units as X.

Example (Die Roll):

For a fair die (X ∈ {1,2,3,4,5,6}):

- E[X]=3.5E[X] = 3.5

- Compute:

$$E[X^2] = \frac{1^2 + 2^2 + 3^2 + 4^2 + 5^2 + 6^2}{6} = \frac{91}{6} \approx 15.17$$

$$\mathrm{Var}(X) = E[X^2] - (E[X])^2 = 15.17 - (3.5)^2 = 2.92$$

---

3. Practical Example (Python)

```
import numpy as np

# Example: Variance of a fair die
die_faces = np.array([1, 2, 3, 4, 5, 6])
probabilities = np.repeat(1/6, 6)

# Expected value
expected_value = np.sum(die_faces * probabilities)

# E[X^2]
expected_square = np.sum((die_faces**2) * probabilities)

# Variance (theoretical)
variance = expected_square - expected_value**2
print("Theoretical Variance:", variance)

# Simulation
samples = np.random.choice(die_faces, size=100000, p=probabilities)
print("Simulated Variance:", np.var(samples))
```

---

4. Key Takeaways

- **Variance** measures the **spread** around the mean.
- Formula:
  - $\mathrm{Var}(X) = E[(X - \mu)^2]$

- $\mathrm{Var}(X) = E[X^2] - (E[X])^2$

- **Standard deviation** = the square root of variance.
- In ML, variance is crucial for understanding **model generalization**, **bias-variance tradeoff**, and **uncertainty**.

Conditional Expectation (with examples)

---

1. Conceptual Overview

Conditional Expectation is the **expected value of a random variable given some condition or additional information**.

- It refines our expectation when we **restrict the sample space**.
- Intuitively:
  - Unconditional expectation → average over the whole distribution.
  - Conditional expectation → average **only over outcomes that satisfy the condition**.
- Applications in ML & statistics:

  1. Regression (Most Direct Use)

  - In **supervised learning**, especially regression, the true function we want to approximate is:

    f(x) = E[Y|X=x]

  - Meaning: The **best predictor** of the target Y given features X, under squared error loss, is the **conditional expectation**.
  - Example: Predicting house prices given size and location → $E[\mathrm{Price}|\mathrm{Size}, \mathrm{Location}]$.

---

2. Mathematical Foundation

For a random variable X:

- **Discrete case**:

$$E[X|A] = \sum_x x\, P(X = x|A)$$

- **Continuous case**:

$$E[X|A] = \int_{-\infty}^{\infty} x\, f_{X|A}(x)\, dx$$

- More generally, if X and Y are RVs:

$$E[X|Y = y] = \sum_x x\, P(X = x|Y = y) \quad \text{(discrete)}$$

$$E[X|Y = y] = \int_{-\infty}^{\infty} x\, f_{X|Y}(x|y)\, dx \quad \text{(continuous)}$$

Example 1: Fair Die (Condition on Even Roll)

- Let X = die roll outcome.
- Condition: roll is even → {2,4,6}.
- Expected value:

$$E[X|X \text{ is even}] = \frac{2+4+6}{3} = 4$$

Example 2: Mixed Dice (Condition on Choice)

- Choose one die at random: 4-faced or 6-faced.
- Roll the chosen die.
- If we know the **6-faced die was chosen**:

$$E[X|\text{6-faced}] = \frac{1+2+3+4+5+6}{6} = 3.5$$

- If we don't know which die was chosen:

$$E[X] = \frac{1}{2} \cdot 2.5 + \frac{1}{2} \cdot 3.5 = 3$$

3. Practical Example (Python)

```
import numpy as np

# Example 1: Conditional expectation of a die roll given even outcome
die = np.array([1,2,3,4,5,6])
prob = np.repeat(1/6, 6)

# Expected value given even roll
even_outcomes = np.array([2,4,6])
expected_even = np.mean(even_outcomes)
print("E[X | X even] =", expected_even)

# Example 2: Mixed dice
expected_4 = np.mean([1,2,3,4])
expected_6 = np.mean([1,2,3,4,5,6])

# Without knowing the die
expected_mix = 0.5*expected_4 + 0.5*expected_6
print("E[X if die unknown] =", expected_mix)
```

---

4. Key Takeaways

- **Conditional expectation** refines our average when we restrict the sample space or know partial information.
- Discrete case: weighted sum of values using conditional probabilities.
- Continuous case: integral using conditional PDFs.
- Widely used in ML: E[Y | X] is the **best predictor of Y given X** under squared error loss.

Hypothesis Testing: Binary Classification Problem

---

1. Conceptual Overview

Hypothesis Testing is a statistical framework for making decisions under uncertainty.

- In a **binary hypothesis test**, we want to decide between **two competing explanations** of the data:
  - **Null Hypothesis (H_0)**: The “default” or “normal” state.
  - **Alternative Hypothesis (H_1)**: The “changed” or “abnormal” state.
- This maps directly to **binary classification** in ML:
  - H_0 = Class 0 (e.g., "normal")
  - H_1 = Class 1 (e.g., "fraud", "disease", "failure").
- Examples in real life:
  - Medical test → Does the patient have the disease (H_1) or not (H_0)?
  - Predictive maintenance → Is the machine failing (H_1) or working normally (H_0)?
  - Signal detection → Is a signal present (H_1) or just noise (H_0)?

---

2. Mathematical Foundation

We observe data y and must choose between:

- $H_0 : y \sim p(y|H_0)$
- $H_1 : y \sim p(y|H_1)$

Decision Rule:

We compare the likelihoods under both hypotheses.

- The optimal test (Neyman–Pearson Lemma) is the **Likelihood Ratio Test (LRT)**: $\Lambda(y) = \frac{p(y|H_1)}{p(y|H_0)}$
- Rule:
  - If $\Lambda(y) > \eta$ → choose H_1

- If $\Lambda(y) \leq \eta$ → choose H_0

  where η is a threshold depending on the acceptable error rate.

---

3. Practical Example (Python)

Suppose we want to detect whether a measurement yy comes from:

- $H_0 : y \sim N(0, 1)$ (normal machine)
- $H_1 : y \sim N(2, 1)$ (failing machine)

```
import numpy as np
from scipy.stats import norm

# Observed value
y = 1.2

# Likelihoods under each hypothesis
p_H0 = norm.pdf(y, loc=0, scale=1)   # N(0,1)
p_H1 = norm.pdf(y, loc=2, scale=1)   # N(2,1)

# Likelihood ratio
LR = p_H1 / p_H0

# Decision threshold (eta = 1 here for simplicity)
eta = 1
decision = "H1 (Failing)" if LR > eta else "H0 (Normal)"
print(f"Observed y={y}, Likelihood Ratio={LR:.3f}, Decision={decision}")
```

---

4. Key Takeaways

- Hypothesis testing = **statistical decision making under uncertainty**.
- Binary hypothesis testing is directly analogous to **binary classification**.
- **Likelihood Ratio Test (LRT)** is the mathematically optimal way to decide between H_0 and H_1.

- Applications in ML:
  - Medical diagnostics, anomaly detection, fraud detection, spam filtering.
- This sets the stage for **errors (Type I and Type II)** and **Gaussian special case testing**.

Null and Alternative Hypotheses

1. Conceptual Overview

When performing **hypothesis testing**, we start by clearly defining two competing claims:

- **Null Hypothesis (H_0)**
  - Represents the **status quo**, "no effect," or "normal" condition.
  - Example: "The machine is working normally," "The drug has no effect," "The email is not spam."
- **Alternative Hypothesis (H_1)**
  - Represents the **change**, "effect," or "abnormal" condition.
  - Example: "The machine is failing," "The drug improves recovery," "The email is spam."

In ML terms:

- $H_0 \equiv$ Class 0 (baseline/negative class).
- $H_1 \equiv$ Class 1 (positive class).

The **goal** is to use observed data to decide whether to **reject** H_0 in favor of H_1.

---

2. Mathematical Foundation

- Null hypothesis:

  $$H_0 : \theta = \theta_0$$
- Alternative hypothesis (can take different forms):

- **Two-sided**:

  $H_1 : \theta \neq \theta_0$

- **One-sided (greater)**:

  $H_1 : \theta > \theta_0$

- **One-sided (less)**:

  $H_1 : \theta < \theta_0$

- The decision is based on test statistics derived from data.
- Example in Gaussian setting:
  - $H_0 : y \sim N(0, 1)$
  - $H_1 : y \sim N(2, 1)$

---

3. Practical Example (Python)

Let’s test whether a sample mean differs significantly from a hypothesized mean (μ_0 = 50).

```
import numpy as np
from scipy import stats

# Sample data (e.g., product weights)
data = np.array([49, 51, 52, 50, 48, 53, 47, 50])

# Null hypothesis: mean = 50
mu0 = 50

# Perform one-sample t-test
t_stat, p_value = stats.ttest_1samp(data, mu0)

print("t-statistic:", t_stat)
print("p-value:", p_value)

if p_value < 0.05:
```

```
    print("Reject H0 (evidence for H1)")
else:
    print("Fail to reject H0 (no strong evidence against H0)")
```

---

4. Key Takeaways

- H_0**: Null hypothesis** → baseline assumption.
- H_1**: Alternative hypothesis** → competing claim we want to test.
- Types of H_1: one-sided or two-sided.
- The decision is made by comparing the test statistic against the threshold (or the p-value against the significance level).
- In ML, this is the foundation of **binary classification** and **anomaly detection**.

Gaussian Hypotheses in Binary Testing

1. Conceptual Overview

A very common setting in hypothesis testing is when data is assumed to come from a **Gaussian (Normal) distribution**.

- Many natural phenomena (measurement errors, sensor noise, biological data) are approximately Gaussian.
- In **binary testing**, each hypothesis corresponds to a different Gaussian distribution.

Example:

- H_0 : Sensor reading comes from N(μ_0, σ^2) → normal condition.
- H_1 : Sensor reading comes from N(μ_1, σ^2) → fault condition.

The challenge: the two distributions may **overlap**, so decisions are uncertain.

---

2. Mathematical Foundation

- Hypotheses:

$$H_0 : Y \sim N(\mu_0, \sigma^2), \quad H_1 : Y \sim N(\mu_1, \sigma^2)$$

- **Likelihood Ratio Test (LRT):**

$$\Lambda(y) = \frac{f(y|H_1)}{f(y|H_0)}$$

For Gaussian PDFs:

$$f(y|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-\mu)^2}{2\sigma^2}}$$

So:

$$\Lambda(y) = \exp\left(\frac{(y-\mu_0)^2 - (y-\mu_1)^2}{2\sigma^2}\right)$$

- Decision Rule:
  - If $\Lambda(y) > \eta$, choose H_1.
  - Else choose H_0.
- With equal priors and same variance, this simplifies to a **threshold test**:

$$y \tau y \underset{H_0}{\overset{H_1}{\gtrless}} \tau$$

  where threshold $\tau = \frac{\mu_0 + \mu_1}{2}$.

---

3. Practical Example (Python)

```
import numpy as np
from scipy.stats import norm

# Parameters
mu0, mu1, sigma = 0, 2, 1
threshold = (mu0 + mu1) / 2
```

```
# Observation
y = 1.3

# Decision rule
decision = "H1 (Fault)" if y > threshold else "H0 (Normal)"
print(f"Observation y={y}, Threshold={threshold}, Decision={decision}")
```

4. Key Takeaways

- In **Gaussian binary testing**, each hypothesis corresponds to a different Normal distribution.
- The **Likelihood Ratio Test** reduces to a **simple threshold rule** when variances are equal.
- Threshold is typically the midpoint between means ($\frac{\mu_0+\mu_1}{2}$).
- Overlap between Gaussians causes uncertainty → leads to **errors** (Type I & II).
- Used widely in **signal detection, quality control, and anomaly detection in ML**.

Likelihood Ratio Test (LRT)

1. Conceptual Overview

The **Likelihood Ratio Test (LRT)** is one of the most powerful and widely used methods in hypothesis testing.

- It compares how likely the observed data is under two competing hypotheses:
  - Null hypothesis (H_0)
  - Alternative hypothesis (H_1)
- Intuition:
  - If the data looks **much more likely under** H_1 than under H_0, we reject H_0.
  - Otherwise, we stick with H_0.

In ML, this is analogous to **choosing the class label with higher likelihood** → a fundamental principle in **Bayesian classification and Maximum Likelihood Estimation (MLE)**.

---

2. Mathematical Foundation

For an observation y:

- **Likelihood Ratio**:

  $\Lambda(y) = \frac{p(y|H_1)}{p(y|H_0)}$

- **Decision Rule**:

  $$\Lambda(y) \underset{H_0}{\overset{H_1}{\gtrless}} \eta$$

  where η is a threshold determined by error tolerance or prior probabilities.

- With **equal priors** and **equal misclassification costs**:

  $\eta = 1$

- Special case: **Gaussian hypotheses with the same variance** → test simplifies to comparing observation y against a **threshold**:

  $\underset{H_0}{\overset{H_1}{\gtrless}} \frac{\mu_0+\mu_1}{2}$

---

3. Practical Example (Python)

```
import numpy as np
from scipy.stats import norm

# Hypotheses
mu0, mu1, sigma = 0, 2, 1

# Observation
y = 1.2

# Likelihood under H0 and H1
```

```
p_H0 = norm.pdf(y, mu0, sigma)
p_H1 = norm.pdf(y, mu1, sigma)

# Likelihood ratio
LR = p_H1 / p_H0

# Decision with threshold eta=1
decision = "H1" if LR > 1 else "H0"
print(f"y={y}, LR={LR:.3f}, Decision={decision}")
```

4. Key Takeaways

- LRT compares likelihoods under H_0 and H_1.
- Rule: $\Lambda(y) > \eta$ → choose H_1, else H_0.
- With Gaussian hypotheses (same variance), test reduces to **thresholding the observation**.
- LRT is **mathematically optimal** (Neyman–Pearson Lemma) for binary hypothesis testing.
- In ML, LRT is conceptually linked to **MLE**, **Naive Bayes classifier**, and **Bayesian decision theory**.

LRT for Gaussian Random Variables & Threshold Decision Rule

1. Conceptual Overview

When the two hypotheses correspond to **Gaussian distributions with equal variance**, the **Likelihood Ratio Test (LRT)** simplifies greatly.

- Instead of computing a ratio of Gaussian PDFs, the decision rule becomes a **simple threshold test**.
- This makes Gaussian hypothesis testing **computationally easy** and widely used in **signal detection, anomaly detection, and classification**.

2. Mathematical Foundation

Let:

- $H_0 : Y \sim N(\mu_0, \sigma^2)$
- $H_1 : Y \sim N(\mu_1, \sigma^2)$

The likelihood ratio is:

$$\Lambda(y) = \frac{f(y|\mu_1, \sigma^2)}{f(y|\mu_0, \sigma^2)}$$

Simplify:

$$\Lambda(y) = \exp\left(\frac{(y-\mu_0)^2 - (y-\mu_1)^2}{2\sigma^2}\right)$$

Decision Rule (LRT):

- If $\Lambda(y) > \eta$, choose H_1.
- Else choose H_0.

With **equal priors** (η = 1) and equal variance:

- Rule reduces to **thresholding y**:

$y \underset{H_0}{\overset{H_1}{\gtrless}} \tau$

where

$\tau = \frac{\mu_0 + \mu_1}{2}$

3. Practical Example (Python)

```
import numpy as np
from scipy.stats import norm

# Gaussian parameters
mu0, mu1, sigma = 0, 2, 1
```

```
threshold = (mu0 + mu1) / 2  # Midpoint threshold

# Observed value
y = 1.3

# Decision rule
decision = "H1 (Abnormal)" if y > threshold else "H0 (Normal)"
print(f"Observation y={y}, Threshold={threshold}, Decision={decision}")
```

---

4. Key Takeaways

- For Gaussian distributions with the **same variance**, LRT reduces to a **threshold test**.
- The threshold is usually the **midpoint between the two means**.
- This is widely used in practice (e.g., signal detection, fault monitoring).
- Overlap between distributions still causes **errors** (false alarms, missed detections).

Types of Errors: Type I and Type II

1. Conceptual Overview

In hypothesis testing (and binary classification), decisions are made under **uncertainty**.

Sometimes, we will **make mistakes** when choosing between H_0 and H_1.

There are two fundamental types of errors:

- **Type I Error (False Positive / False Alarm)**
  - Reject H_0 when H_0 is actually true.
  - Example: Declaring a machine faulty when it is actually normal.
- **Type II Error (False Negative / Missed Detection)**
  - Fail to reject H_0 when H_1 is actually true.

- Example: Declaring a machine normal when it is actually faulty.

💡 In ML terms:

- **Type I Error** = False Positive
- **Type II Error** = False Negative

---

2. Mathematical Foundation

Let H_0 = normal, H_1 = abnormal.

- **Type I Error probability (α)**:

$$\alpha = P(\text{Decide } H_1 \mid H_0 \text{ is true})$$

Also called **significance level**.

- **Type II Error probability (β)**:

$$\beta = P(\text{Decide } H_0 \mid H_1 \text{ is true})$$

- **Power of the test**:

$$1 - \beta$$

Represents the probability of correctly detecting H_1.

---

3. Practical Example (Python)

Let's simulate errors when testing:

- $H_0 : Y \sim N(0, 1)$
- $H_1 : Y \sim N(2, 1)$
- Threshold: $\tau = 1$.

```
import numpy as np

# Parameters
mu0, mu1, sigma = 0, 2, 1
threshold = 1
N = 100000
```

```
# Simulate samples
samples_H0 = np.random.normal(mu0, sigma, N)
samples_H1 = np.random.normal(mu1, sigma, N)

# Type I Error: H0 true but decide H1
alpha = np.mean(samples_H0 > threshold)

# Type II Error: H1 true but decide H0
beta = np.mean(samples_H1 <= threshold)

print(f"Type I Error (alpha): {alpha:.3f}")
print(f"Type II Error (beta): {beta:.3f}")
print(f"Power of the test: {1-beta:.3f}")
```

---

4. Key Takeaways

- **Type I Error (α)** = Reject H_0 incorrectly = False Positive.
- **Type II Error (β)** = Fail to reject H_0 incorrectly = False Negative.
- **Power (1−β)** = Probability of correctly detecting H_1.
- There is often a **trade-off**: reducing α increases β, and vice versa.
- In ML:
  - Precision ↔ Avoiding Type I errors.
  - Recall ↔ Avoiding Type II errors.

Computing Probabilities of Error

1. Conceptual Overview

When doing hypothesis testing, the **probability of error** quantifies how often we make **wrong decisions** due to overlap between distributions.

- **Type I Error (α)**: Probability of rejecting H_0 when it is true (false alarm).

- **Type II Error (β)**: Probability of failing to reject H_0 when H_1 is true (missed detection).

The **total probability of error** depends on:

1. The **distributions** under each hypothesis.
2. The **decision threshold**.
3. The **prior probabilities** of H_0 and H_1.

---

2. Mathematical Foundation

- Suppose priors:
  - $P(H_0) = \pi_0,$
  - $P(H_1) = \pi_1.$
- Total probability of error:

$$P_e = \pi_0 \cdot P(\text{decide } H_1 | H_0) + \pi_1 \cdot P(\text{decide } H_0 | H_1)$$

$$P_e = \pi_0 \cdot \alpha + \pi_1 \cdot \beta$$

- For Gaussian hypothesis testing with equal priors and equal variance:
  - Threshold $\tau = \frac{\mu_0 + \mu_1}{2}$.
  - Type I Error (α):

$$\alpha = P(Y > \tau | H_0) = 1 - \Phi\left(\frac{\tau - \mu_0}{\sigma}\right)$$

  - Type II Error (β):

$$\beta = P(Y \leq \tau | H_1) = \Phi\left(\frac{\tau - \mu_1}{\sigma}\right)$$

  - Where $\Phi(\cdot)$ = standard normal CDF.

---

3. Practical Example (Python)

```
import numpy as np
from scipy.stats import norm

# Parameters
mu0, mu1, sigma = 0, 2, 1
pi0, pi1 = 0.5, 0.5  # Equal priors
tau = (mu0 + mu1) / 2  # Threshold

# Type I error (alpha)
alpha = 1 - norm.cdf((tau - mu0) / sigma)

# Type II error (beta)
beta = norm.cdf((tau - mu1) / sigma)

# Total probability of error
P_e = pi0 * alpha + pi1 * beta

print(f"Type I Error (alpha): {alpha:.3f}")
print(f"Type II Error (beta): {beta:.3f}")
print(f"Total Probability of Error: {P_e:.3f}")
```

4. Key Takeaways

- **Error probability** comes from overlapping distributions.
- Formula:

$$P_e = \pi_0 \alpha + \pi_1 \beta$$

- In Gaussian testing: errors are computed using **normal CDF**.
- Adjusting the **threshold** trades off between α and β.
- In ML terms, this is the same as balancing **precision vs recall** using thresholds.