Distributions, Central Limit Theorem, and Estimation in ML

Random Variables: Discrete vs Continuous

A **random variable** is a function that assigns a numerical value to each outcome of a random experiment.

What is a Random Variable?

Instead of thinking about abstract outcomes like "heads" or "tails," a random variable converts them to numbers.

Example:

```
Coin flip experiment:
- Outcome: Heads
- Random variable X: X = 1 (assign 1 to heads)
- Or: X = 0 (assign 0 to tails)
```

Example: Rolling a die

```
Outcome: Rolling a 4
Random variable X = value shown = 4
```

Why use random variables? They let us apply mathematical operations (addition, multiplication, calculus) to probabilistic events.

Discrete Random Variables

A **discrete random variable** takes on a **finite or countably infinite** set of values.

Characteristics:

- Specific, separated values (gaps between possible values)
- Can be listed or counted
- Probability described by **probability mass function (PMF)**

Examples:

- Number of heads in 10 coin flips: $X \in \{0, 1, 2, 3, ..., 10\}$
- Number of emails received per hour: $X \in \{0, 1, 2, 3, ...\}$
- Test score (if graded on points): $X \in \{0, 1, 2, ..., 100\}$

- Product defects in a batch: $X \in \{0, 1, 2, 3, ...\}$

Probability Mass Function (PMF):

```
P(X = k) = probability that X equals exactly k

Properties:
- 0 ≤ P(X = k) ≤ 1
- Σ P(X = k) = 1 (probabilities sum to 1)
```

Example: Fair die

```
P(X = 1) = 1/6
P(X = 2) = 1/6
...
P(X = 6) = 1/6
Sum = 6 × (1/6) = 1 ✓
```

Continuous Random Variables

A **continuous random variable** can take on **any value in an interval** (uncountably infinite values).

Characteristics:

- Any real number in a range
- Cannot list all possible values
- Probability described by **probability density function (PDF)**

Examples:

- Temperature tomorrow: $X \in [-50°C, 50°C]$
- Height of a person: $X \in [0cm, 300cm]$
- Stock price: $X \in [0, \infty)$
- Response time of a website: $X \in [0, \infty)$

Probability Density Function (PDF):

```
f(x) = probability density at value x

Properties:
- f(x) ≥ 0 (always non-negative)
- ∫ f(x) dx = 1 (total area under curve is 1)
- P(a ≤ X ≤ b) = ∫[a to b] f(x) dx (probability is area under curve)
```

Key difference: For continuous variables, P(X = exact value) = 0 (infinitesimal).

- P(temperature = 25.000000°C exactly) = 0
- But P(24.5 ≤ temperature ≤ 25.5) > 0

We can only talk about probability in intervals or ranges.

Visual Comparison

Discrete (PMF):

```
P(X=k)
 |
 |  ▄
 |  █  ▄
 |  █  █  ▄
 |  █  █  █  ▄
 |____|___|___|___|___
   1  2  3  4  k
```

Probability is at specific points (bars).

Continuous (PDF):

```
f(x)
 |
 |     /\
 |    /  \
 |   /    \
 |  /      \
 |____/_________\____
   a       b  x

P(a ≤ X ≤ b) = area under curve
```

Probability is the area under the smooth curve.

Cumulative Distribution Function (CDF)

For both discrete and continuous, the **CDF** gives cumulative probability:

```
F(x) = P(X ≤ x)
```

Properties:

- $0 \leq F(x) \leq 1$
- Monotonically increasing (never decreases)
- $F(-\infty) = 0$, $F(\infty) = 1$

Example: Fair die

```
F(1) = P(X ≤ 1) = 1/6
F(2) = P(X ≤ 2) = 2/6
F(3) = P(X ≤ 3) = 3/6 = 1/2
...
F(6) = P(X ≤ 6) = 1
```

Applications in ML

Discrete RVs: Classification (probability of each class), counting events (defects, clicks)

Continuous RVs: Regression (predicting real values), measurements, time, distance

Most real-world ML problems involve continuous random variables, but many also model discrete outcomes (categorical classification).

Common Distributions: Bernoulli, Binomial, Normal, Poisson

Understanding these distributions is essential—they model most real-world data.

Bernoulli Distribution

The **Bernoulli distribution** models a single trial with two possible outcomes (success/failure, yes/no).

Parameter: p = probability of success ($0 \leq p \leq 1$)

PMF:

```
P(X = 1) = p        (success)
P(X = 0) = 1 - p    (failure)
```

Mean and Variance:

```
E[X] = p
Var(X) = p(1 - p)
```

Examples:

- Coin flip: $p = 0.5$
- Email is spam: $p = 0.3$
- Customer converts: $p = 0.02$
- Loan defaults: $p = 0.05$

In ML: Binary classification output, click-through rates, success indicators.

Binomial Distribution

The **Binomial distribution** models multiple independent Bernoulli trials.

Parameters:

- n = number of trials
- p = probability of success in each trial

PMF:

```
P(X = k) = C(n,k) × p^k × (1-p)^(n-k)

Where C(n,k) = n! / (k! × (n-k)!) is "n choose k"
```

Intuition: Of n trials, exactly k succeed (with probability p), the rest fail (with probability 1-p). The C(n,k) term counts how many ways this can happen.

Mean and Variance:

```
E[X] = n × p
Var(X) = n × p × (1 - p)
```

Example: 5 coin flips, how many heads?

```
n = 5, p = 0.5

P(X = 3) = C(5,3) × (0.5)^3 × (0.5)^2
         = 10 × 0.125 × 0.25
         = 0.3125 (31.25% chance of 3 heads)

E[X] = 5 × 0.5 = 2.5 (expect 2.5 heads on average)
Var(X) = 5 × 0.5 × 0.5 = 1.25
```

Example: Ad clicks

```
n = 1000 ad impressions, p = 0.02 click-through rate

E[clicks] = 1000 × 0.02 = 20 clicks
Var(clicks) = 1000 × 0.02 × 0.98 = 19.6
```

In ML: Number of successes in fixed trials, conversion counts, defect counts in batches.

Normal Distribution

The **Normal (Gaussian) distribution** is the most important distribution in statistics and ML.

Parameters:

- μ (mu) = mean (center of distribution)
- σ (sigma) = standard deviation (spread)

PDF:

```
f(x) = (1 / (σ√(2π))) × exp(-(x - μ)² / (2σ²))
```

This intimidating formula creates the famous **bell curve**.

Mean and Variance:

```
E[X] = μ
Var(X) = σ²
```

Properties:

- Symmetric around μ (mean = median = mode)
- 68% of data within μ ± σ

- 95% of data within μ ± 2σ
- 99.7% of data within μ ± 3σ

Visualizing normal distributions:

```
f(x)
 |
 |        μ=0, σ=1
 |          ╱╲
 |         ╱  ╲      μ=0, σ=2
 |        ╱    ╲    ╱  ╲
 |       ╱      ╲__╱    ╲
 |______╱_______________↑____
    -3  -2  -1  0  1  2  3

Narrower curve: smaller σ (more concentrated)
Wider curve: larger σ (more spread out)
```

Standard Normal Distribution (Z):

```
Z ~ N(0, 1)  [mean = 0, std dev = 1]

Any normal variable can be standardized:
Z = (X - μ) / σ
```

Example: Heights

```
Adult male heights: X ~ N(175 cm, 8 cm)

P(X < 183) = P(Z < (183-175)/8) = P(Z < 1) ≈ 0.84 (84%)
P(165 < X < 185) = P(-1.25 < Z < 1.25) ≈ 0.79 (79%)
```

Why Normal Distribution?

- **Central Limit Theorem** (see next section): averages of many independent variables approach normal
- Many natural phenomena follow normal distribution (heights, test scores, measurement errors)
- Mathematically convenient (easy to work with)
- Many ML algorithms assume normality (linear regression, Gaussian processes)

In ML:

- Modeling measurement errors
- Prior distributions in Bayesian methods
- Assuming normally distributed features improves some algorithms
- Regularization often assumes normally distributed weights

Poisson Distribution

The **Poisson distribution** models the number of events occurring in a fixed time/space interval.

Parameter: λ (lambda) = average rate of events

PMF:

```
P(X = k) = (e^(-λ) × λ^k) / k!
```

Mean and Variance:

```
E[X] = λ
Var(X) = λ
```

Unique property: **mean equals variance** (unusual!).

Example: Customer arrivals

```
λ = 10 customers per hour

P(exactly 5 arrive) = (e^(-10) × 10^5) / 5! ≈ 0.038 (3.8%)
P(more than 15 arrive) = ?

E[X] = 10 (expect 10 on average)
Var(X) = 10 (std dev ≈ 3.16)
```

Examples:

- Emails received per hour
- Website traffic per minute
- Accident rates per month
- Defects per meter of material
- Calls to a help desk

When to use Poisson:

- Events occur independently
- Events happen at constant average rate
- Time/space intervals are independent

In ML: Modeling count data, anomaly detection (unusual spike in counts), forecasting event frequencies.

Distribution Selection Guide

Distribution	Situation	Parameter(s)
Bernoulli	Single yes/no event	p (success rate)
Binomial	Fixed number of trials, count successes	n, p
Normal	Measurement, aggregated data, errors	μ, σ
Poisson	Count of events in fixed interval	λ (rate)

Central Limit Theorem (CLT) and Why It Matters in ML

The **Central Limit Theorem** is arguably the most important theorem in statistics and ML.

The Theorem Statement

If you take repeated samples of size n from ANY distribution and compute the sample mean, those means will form a Normal distribution, regardless of the original distribution's shape.

Mathematically:

```
If X₁, X₂, ..., Xₙ are independent samples from any distribution with mean μ and variance σ²

Then the sample mean X̄ = (X₁ + X₂ + ... + Xₙ) / n

Approaches N(μ, σ²/n) as n → ∞
```

The distribution of means is approximately **normal** with:

- Mean = μ (same as original)
- Variance = σ^2/n (decreases as sample size increases!)
- Standard error = $\sigma / \sqrt{n}$

Visual Intuition

Original distribution (could be any shape):

```
f(x)
|    ▄▄▄▄▄▄
|  ▄▀    ▀▄
| █       █
|█         █
|█_____________█
(Example: uniform or exponential)
```

Distribution of sample means:

```
f(X̄)
|      /\
|     /  \
|    /    \      (Normal, regardless of original!)
|   /      \
|__/________\____
```

Effect of sample size:

```
n=10:    /\
        /  \    (wider, more uncertain)

n=100:  |\/|
        |    (narrower, more certain)

n=1000: ||   (very narrow, very certain)
```

As n increases, distribution of means gets narrower and more concentrated.

Why This Matters: Three Key Reasons

1. Generalization from Theory to Practice Many ML algorithms assume normally distributed data. CLT justifies this:

- If you aggregate many independent factors, the result is approximately normal
- Most real measurements are aggregates of many small effects

2. Confidence Intervals and Hypothesis Testing Because sample means are normal, we can use normal distribution properties:

```
95% Confidence Interval for true mean μ:
[X̄ - 1.96 × SE, X̄ + 1.96 × SE]

Where SE = σ / √n (standard error)
```

Example: Average customer lifetime value

```
Sample of 100 customers: X̄ = $500
Sample std dev: σ = $200
SE = $200 / √100 = $20

95% CI = [500 - 1.96×20, 500 + 1.96×20]
       = [$460.80, $539.20]

We're 95% confident true mean is between $460.80 and $539.20
```

3. Sample Size Calculation CLT determines required sample size:

```
SE = σ / √n

To halve standard error (be twice as confident):
Need √4 = 4 times as many samples
```

If 100 samples give ±$20, need 400 samples for ±$10.

CLT in Machine Learning Practice

1. Model Averaging

```
Final prediction = average of many model predictions
By CLT: prediction uncertainty decreases with more models
```

2. Confidence in Parameter Estimates

```
Model parameter w estimated from data
Distribution of w estimates across different samples approaches normal
Standard error tells us confidence in the estimate
```

3. Regularization Justification

```
Many ML algorithms find weights that minimize average error
By CLT: error distribution is normal
Regularization assumes normal distribution of errors
```

4. Batch Normalization in Neural Networks

```
Each mini-batch is a sample
Normalizing based on batch statistics relies on CLT reasoning
```

5. Bootstrap Confidence Intervals

```
Resample data many times, compute statistic for each sample
Distribution of statistics approaches normal by CLT
```

Example: Website Conversion Rate

Your website converts visitors to customers. You want to estimate true conversion rate.

Data collection:

```
Sample 1: 100 visitors, 10 conversions → p̂₁ = 0.10
Sample 2: 100 visitors, 15 conversions → p̂₂ = 0.15
Sample 3: 100 visitors, 12 conversions → p̂₃ = 0.12
...
Sample 100: 100 visitors, 11 conversions → p̂₁₀₀ = 0.11
```

Distribution of sample proportions:

```
By CLT, these proportions form a normal distribution!

Mean of proportions: (0.10 + 0.15 + 0.12 + ... + 0.11) / 100 ≈ 0.127
Standard error: σ/√n

95% CI: [0.127 - 1.96×SE, 0.127 + 1.96×SE]
```

This is how we estimate true conversion rate with confidence.

When CLT Fails

CLT assumes:

- **Independence**: Samples don't influence each other

- **Large enough n**: Rule of thumb: $n \geq 30$ (or more if original distribution very skewed)
- **Finite variance**: Original distribution must have finite variance (some distributions don't)

With these conditions met, CLT is remarkably robust.

Point Estimation of Parameters

When you have data, you need to estimate the distribution's parameters (like μ and σ for normal).

What is Point Estimation?

Point estimation is computing a single value (point estimate) as an estimate of an unknown parameter.

Examples:

```
True population mean μ (unknown) → Estimate with sample mean x̄
True population variance σ² (unknown) → Estimate with sample variance s²
True probability p (unknown) → Estimate with sample proportion p̂
```

Common Point Estimators

Sample Mean (estimates population mean μ):

```
x̄ = Σxᵢ / n

Unbiased: E[x̄] = μ ✓
Efficient: Low variance, CLT applies
Most commonly used
```

Sample Variance (estimates population variance σ^2):

```
s² = Σ(xᵢ - x̄)² / (n - 1)

Unbiased: E[s²] = σ² ✓
Note: divide by (n-1), not n [Bessel's correction]
```

Sample Proportion (estimates probability p):

```
p̂ = (number of successes) / n

Unbiased: E[p̂] = p ✓
Used for binary events
```

Sample Median (estimates population median):

```
Sort data, take middle value
Robust to outliers (unlike mean)
```

Properties of Good Estimators

1. Unbiasedness

```
E[estimator] = true parameter
```

On average, the estimate equals the truth (no systematic over/under estimation).

Example: Sample mean is unbiased for μ

```
E[x̄] = μ
```

2. Consistency

```
As n → ∞, estimator → true parameter (with probability 1)
```

With more data, estimate gets better.

3. Efficiency

```
Smaller variance is better
```

Among unbiased estimators, choose the one with lowest variance.

Example:

```
Sample mean: Var(x̄) = σ²/n
Sample median: Var(median) ≈ 1.25 × σ²/n (less efficient)
```

4. Robustness

```
Not sensitive to outliers or violations of assumptions
```

Sample median is robust; sample mean is not.

Maximum Likelihood Estimation (MLE)

Maximum Likelihood Estimation finds parameters that make observed data most likely.

Idea: Given data, what parameter values would make this data most probable?

Likelihood function:

```
L(θ | data) = P(data | θ)

Probability of observing this data given parameter θ
```

MLE:

```
θ̂ = argmax L(θ | data)
```

Find parameter that maximizes likelihood.

Example: Coin flips

Observed: 7 heads in 10 flips. What's p (probability of heads)?

```
Likelihood: L(p | 7H, 3T) = p^7 × (1-p)^3

L(0.5) = 0.5^7 × 0.5^3 = 0.001
L(0.7) = 0.7^7 × 0.3^3 = 0.0097
L(0.8) = 0.8^7 × 0.2^3 = 0.0167
L(0.9) = 0.9^7 × 0.1^3 = 0.000729

Maximum at p = 0.7
```

MLE says p ≈ 0.7 (which equals 7/10, the observed proportion!).

Practical Estimation

In practice with real data:

Step 1: Determine likely distribution (histogram, domain knowledge)

Step 2: Compute parameter estimates:

```python
# If normal distribution
μ_hat = mean(data)
σ_hat = std(data)

# If binomial
n = total_trials
p_hat = successes / total_trials

# If Poisson
λ_hat = mean(data)
```

Step 3: Compute confidence intervals (using CLT):

```
95% CI = [estimate - 1.96 × SE, estimate + 1.96 × SE]
```

Example: Customer spending

```
Data: [50, 75, 100, 60, 90, 80, 110, 95, 85, 100] (10 customers)

x̄ = 845 / 10 = $84.5
s² = Σ(xᵢ - x̄)² / 9 = 1592.5 / 9 ≈ 176.9
s = √176.9 ≈ $13.3

SE = s / √n = 13.3 / √10 ≈ $4.2

95% CI = [84.5 - 1.96×4.2, 84.5 + 1.96×4.2]
       = [$76.3, $92.7]
```

We're 95% confident true mean spending is between $76.3 and $92.7.

Introduction to Monte Carlo Simulation

Monte Carlo simulation estimates quantities by randomly sampling and computing.

Core Idea

Instead of solving a problem analytically (with formulas), simulate it with random numbers.

Process:

1. Generate random samples from a distribution

2. Apply a function or rule to each sample
3. Aggregate results (average, count, etc.)
4. Use aggregate as estimate

Why Monte Carlo?

Many problems are hard or impossible to solve analytically:

- Complex probability calculations
- High-dimensional integrals
- Non-linear systems
- Real-world simulations

Monte Carlo provides approximate answers.

Simple Example: Estimating π

Problem: Estimate π using random sampling.

Method:

1. Generate random points in a 1×1 square
2. Check if each point is inside a quarter-circle (radius 1)
3. Ratio: (points in circle) / (total points) ≈ (π/4) / 1 = π/4
4. Multiply by 4 to get π

```
Algorithm:
For i = 1 to N:
  x = random(0, 1)
  y = random(0, 1)
  if x² + y² ≤ 1:
    count_inside += 1

π_estimate = 4 × count_inside / N
```

Results with different sample sizes:

```
N = 100: π_estimate ≈ 3.16
N = 1,000: π_estimate ≈ 3.14
N = 10,000: π_estimate ≈ 3.1415
N = 100,000: π_estimate ≈ 3.14159...
```

With more samples, estimate approaches true value ($\pi \approx 3.14159$).

Computing Expected Values

Problem: Find $E[g(X)]$ for complex function g.

Analytical approach: $E[g(X)] = \int g(x) \times f(x)\, dx$ (hard integral!)

Monte Carlo approach:

```
1. Sample X₁, X₂, ..., Xₙ from distribution of X
2. Compute g(X₁), g(X₂), ..., g(Xₙ)
3. Average: Ê[g(X)] = Σg(Xᵢ) / n
```

By law of large numbers, this average converges to true expected value.

Example: Option pricing

Black-Scholes formula for option price is complex. Monte Carlo approach:

```
for i = 1 to N:
  simulate stock price path to maturity
  payoff = max(stock_price - strike, 0)
  payoffs.append(payoff)

option_price = average(payoffs) × discount_factor
```

This works even for complex, non-linear models.

Monte Carlo in Machine Learning

1. Bayesian Inference

```
Sample from posterior distribution
Use samples to estimate credible intervals
```

2. Uncertainty Quantification

```
Propagate input uncertainty through model
Estimate output distribution
Quantify prediction uncertainty
```

3. Model Evaluation

```
Resample from data distribution
Train model on each resample
Estimate distribution of performance metrics
```

4. Hyperparameter Tuning

```
Randomly sample hyperparameter combinations
Train and evaluate on each
Find best combination
```

5. Reinforcement Learning

```
Monte Carlo tree search: sample possible action sequences
Estimate value of each path
Choose highest-value action
```

Variance Reduction Techniques

Monte Carlo estimates improve with more samples, but sampling is expensive.

Importance Sampling:

```
Sample from different (easier) distribution
Weight samples by likelihood ratio
Reduces variance without more samples
```

Stratified Sampling:

```
Divide population into strata
Sample proportionally from each stratum
Ensures representative sampling
Reduces variance
```

Control Variates:

```
Use a correlated variable with known expected value
Reduce variance by exploiting correlation
```

Practical Monte Carlo Example: Risk Assessment

Scenario: Uncertain project completion time due to multiple tasks.

```
Task A: 5-7 days (uniform)
Task B: 8-12 days (uniform)
Task C: 4-6 days (uniform)

Total time = A + B + C

What's distribution of total time?
```

Monte Carlo approach:

```
for i = 1 to 10,000:
  a = random(5, 7)
  b = random(8, 12)
  c = random(4, 6)
  total = a + b + c
  times.append(total)

Results:
Mean completion time: E[total] ≈ 17.5 days
5th percentile: 14.2 days (90% confident of finishing by then)
95th percentile: 20.8 days
```

From 10,000 simulations, we estimate the complete distribution without analytical formulas.

Convergence and Accuracy

Accuracy improves with √N:

```
Error ∝ 1/√N
```

To get 10x more accuracy, need 100x more samples. This is the fundamental tradeoff.

Standard error of estimate:

```
SE ≈ σ / √N

σ = std dev of function values
N = number of samples
```

This follows from CLT—the average of samples is normally distributed!

Pseudocode: General Monte Carlo

```
function monte_carlo(N, distribution, function):
  samples = 0
  for i = 1 to N:
    x = random_sample_from(distribution)
    samples += function(x)

  return samples / N
```

Simple algorithm, but powerful.

Putting It All Together: A Complete Example

Scenario: Customer Lifetime Value (CLV) Estimation

You want to estimate customer lifetime value, which depends on:

- Annual spending: $S \sim \text{Normal}(\mu=\$500, \sigma=\$100)$
- Retention rate: $r \sim \text{Beta}(\alpha=2, \beta=1)$ [higher probability of staying]
- Customer lifetime: T = geometric sum based on r
- Discount rate: $d = 5\%$ per year

Analytical approach: Complex (involves infinite series)

Monte Carlo approach:

```
for i = 1 to 100,000:
  annual_spend = sample from Normal(500, 100)
  retention_rate = sample from Beta(2, 1)

  clv = 0
  for year in 1 to 20:  # assume max 20 years
    probability_staying = retention_rate ^ year
    discounted_value = annual_spend / (1.05 ^ year)
    clv += probability_staying × discounted_value

  clv_estimates.append(clv)

Results:
Mean CLV: E[CLV] ≈ average(clv_estimates)
Std Dev: σ ≈ std(clv_estimates)
95% CI: [percentile(5), percentile(95)]
```

Output:

```
Mean CLV: $4,250
95% CI: [$3,800, $4,700]
Median: $4,200
Std Dev: $450
```

Monte Carlo lets us estimate CLV without closed-form formulas, handling complex uncertainty.

Summary: From Theory to Simulation

Random variables convert outcomes to numbers, enabling mathematics.

Common distributions (Bernoulli, Binomial, Normal, Poisson) describe most real phenomena.

Central Limit Theorem explains why normal distributions are ubiquitous and justifies confidence intervals.

Point estimation finds best-guess parameters from data using sample statistics.

Monte Carlo simulation estimates probabilities and expected values through random sampling.

Together, these form the statistical foundation of machine learning:

- Understanding distributions helps choose appropriate models
- CLT justifies statistical inference and confidence intervals

- Point estimation fills in unknown parameters
- Monte Carlo simulates complex systems and estimates uncertainty

Master these concepts, and you understand the probabilistic foundations underlying all of modern data science and machine learning.