TP2 PEyA

Rodrigo Goñi

June 12, 2025

1 Ejercicio 1

Primero, definimos la distribución de probabilidad para la variable aleatoria discreta X:

X	Probabilidad $P(X = x)$
0	$\frac{2\theta}{3}$
1	$\frac{4\theta}{3}$
2	$\frac{1-2\theta}{3}$
3	$\frac{2(1-2\theta)}{3}$

Para que estas sean probabilidades válidas, se deben cumplir dos condiciones fundamentales:

1. $P(X = x) \geq 0$ para todo x.
2. $\sum_x P(X = x) = 1$.

Verificamos la segunda condición (la suma de probabilidades debe ser 1):

$$\frac{2\theta}{3} + \frac{4\theta}{3} + \frac{1-2\theta}{3} + \frac{2(1-2\theta)}{3} = \frac{2\theta + 4\theta + 1 - 2\theta + 2 - 4\theta}{3} = \frac{3}{3} = 1$$

La segunda condición se cumple.

De la primera condición ($P(X = x) \geq 0$), obtenemos el **rango válido de θ**:

- $\frac{2\theta}{3} \geq 0 \implies \theta \geq 0$
- $\frac{4\theta}{3} \geq 0 \implies \theta \geq 0$
- $\frac{1-2\theta}{3} \geq 0 \implies 1 - 2\theta \geq 0 \implies 1 \geq 2\theta \implies \theta \leq \frac{1}{2}$
- $\frac{2(1-2\theta)}{3} \geq 0 \implies 1 - 2\theta \geq 0 \implies \theta \leq \frac{1}{2}$

Por lo tanto, el rango válido para θ es $0 \leq \theta \leq \frac{1}{2}$.

Contamos las veces que aparece cada dato en la muestra:

- $X = 0$: aparece **4 veces**
- $X = 1$: aparece **2 veces**
- $X = 2$: aparece **1 vez**
- $X = 3$: aparece **3 veces**

Ahora, escribimos la función de verosimilitud:

$$L(\theta) = P(X = 0)^4 \times P(X = 1)^2 \times P(X = 2)^1 \times P(X = 3)^3$$

Sustituyendo las probabilidades:

$$L(\theta) = \left(\frac{2\theta}{3}\right)^4 \times \left(\frac{4\theta}{3}\right)^2 \times \left(\frac{1-2\theta}{3}\right)^1 \times \left(\frac{2(1-2\theta)}{3}\right)^3$$

Simplificando la expresión:

$$\begin{aligned}
L(\theta) &= \frac{(2\theta)^4}{3^4} \times \frac{(4\theta)^2}{3^2} \times \frac{(1-2\theta)^1}{3^1} \times \frac{(2(1-2\theta))^3}{3^3} \\
&= \frac{16\theta^4}{81} \times \frac{16\theta^2}{9} \times \frac{1-2\theta}{3} \times \frac{8(1-2\theta)^3}{27} \\
&= \frac{2048\theta^6(1-2\theta)^4}{59049}
\end{aligned}$$

Para facilitar la derivación, tomamos el logaritmo natural de la función de verosimilitud:

$$\begin{aligned}
\ln(L(\theta)) &= \ln\left(\frac{2048\theta^6(1-2\theta)^4}{59049}\right) \\
&= \ln(2048) + \ln(\theta^6) + \ln((1-2\theta)^4) - \ln(59049) \\
&= \ln(2048) + 6\ln(\theta) + 4\ln(1-2\theta) - \ln(59049)
\end{aligned}$$

Derivamos la función log-verosimilitud con respecto a θ y la igualamos a cero para encontrar el punto crítico:

$$\begin{aligned}
\frac{d}{d\theta}\ln(L(\theta)) &= \frac{d}{d\theta}(\ln(2048) + 6\ln(\theta) + 4\ln(1-2\theta) - \ln(59049)) \\
&= 0 + \frac{6}{\theta} + 4 \times \frac{1}{1-2\theta} \times (-2) - 0 \\
&= \frac{6}{\theta} - \frac{8}{1-2\theta}
\end{aligned}$$

Igualamos a cero para encontrar el MLE de θ:

$$\frac{6}{\theta} - \frac{8}{1-2\theta} = 0$$

$$\frac{6}{\theta} = \frac{8}{1-2\theta}$$

$$6(1-2\theta) = 8\theta$$

$$6 = 8\theta + 12\theta$$

$$6 = 20\theta$$

$$\theta = \frac{3}{10}$$

El valor obtenido para θ es $\frac{3}{10} = 0.3$. Este valor se encuentra dentro del rango válido que establecimos al principio: $0 \leq 0.3 \leq 0.5$.

—

Estimación Numérica utilizando Python

Para complementar el cálculo analítico, utilizamos un enfoque numérico para encontrar el MLE de θ usando Python y la librería ‘scipy.optimize‘.

```python
import numpy as np
from scipy.optimize import minimize

def log_likelihood_analytical(theta, counts):
    """
    Calcula el logaritmo de la funcion de verosimilitud utilizando la expresion analitica
    simplificada derivada, excluyendo terminos constantes que no afectan la optimizacion.

    Parametros:
    theta (float): El parametro del modelo.
    counts (dict): Un diccionario con las frecuencias observadas de cada valor de X.

    Retorna:
    float: El valor negativo del log-verosimilitud
    """

    if theta <= 0 or theta >= 0.5:
        return np.inf

    log_L = 6 * np.log(theta) + 4 * np.log(1 - 2 * theta)

    return -log_L

observed_counts = {
    0: 4,
    1: 2,
    2: 1,
    3: 3
}

initial_theta = 0.1

# Definimos los limites para theta [epsilon, 0.5 - epsilon] para evitar log(0)
bounds = [(1e-10, 0.5 - 1e-10)]

# Usamos scipy.optimize.minimize para encontrar el theta que minimiza -log_likelihood.
result = minimize(log_likelihood_analytical, initial_theta, args=(observed_counts,),
    method='L-BFGS-B', bounds=bounds)

# --- Mostrar Resultados ---
print("--- Resultados de la Optimizacion Numerica ---")
if result.success:
    estimated_theta = result.x[0]
    print(f"Estimador de Maxima Verosimilitud (MLE) para theta (numerico): {estimated_theta:.4f}")
    print(f"Nuestro calculo manual dio: {3/10:.4f} (0.3)")

    constant_term = np.log(2048) - np.log(59049)
    max_log_likelihood = 6 * np.log(estimated_theta) + 4 * np.log(1 - 2 * estimated_theta) +
        constant_term
    print(f"Log-Verosimilitud maximizada (completa): {max_log_likelihood:.4f}")

else:
    print("La optimizacion no fue exitosa.")
    print(result.message)

print("\n--- Verificacion de las probabilidades con el MLE estimado numericamente ---")
final_theta = estimated_theta
final_theta = max(1e-10, min(0.5 - 1e-10, final_theta))
```

```python

p_x0_calc = (2 * final_theta) / 3
p_x1_calc = (4 * final_theta) / 3
p_x2_calc = (1 - 2 * final_theta) / 3
p_x3_calc = (2 * (1 - 2 * final_theta)) / 3

print(f"P(X=0) = {p_x0_calc:.4f}")
print(f"P(X=1) = {p_x1_calc:.4f}")
print(f"P(X=2) = {p_x2_calc:.4f}")
print(f"P(X=3) = {p_x3_calc:.4f}")
print(f"Suma de probabilidades: {p_x0_calc + p_x1_calc + p_x2_calc + p_x3_calc:.4f}")
```

Listing 1: Código Python para la estimación numérica del MLE.

Resultados de la Ejecución del Código Python

A continuación, se presentan los resultados obtenidos al ejecutar el script de Python:

```
--- Resultados de la Optimización Numérica ---
Estimador de Máxima Verosimilitud (MLE) para theta: 0.3000
Nuestro cálculo manual dio: 0.3000 (0.3)
Log-Verosimilitud maximizada (completa): -14.2505

--- Verificación de las probabilidades con el MLE ---
P(X=0) = 0.2000
P(X=1) = 0.4000
P(X=2) = 0.1333
P(X=3) = 0.2667
Suma de probabilidades: 1.0000
```

—

2 Ejercicio 2

El modelo propuesto es:

$$Y_i = a + bX_i + cX_i^2$$

La suma de los cuadrados de los residuos (S) es:

$$S = \sum_{i=1}^{n} (Y_i - (a + bX_i + cX_i^2))^2$$

Donde n es el número de observaciones. En este caso, $n = 5$.

Para minimizar S, calculamos las derivadas parciales de S con respecto a a, b y c y las igualamos a cero.

Derivada con respecto a a:

$$\begin{aligned}
\frac{\partial S}{\partial a} &= \sum_{i=1}^{n} 2(Y_i - a - bX_i - cX_i^2)(-1) = 0 \\
&\implies \sum_{i=1}^{n} (Y_i - a - bX_i - cX_i^2) = 0 \\
&\implies \sum Y_i - na - b\sum X_i - c\sum X_i^2 = 0 \\
&\implies na + b\sum X_i + c\sum X_i^2 = \sum Y_i \quad \text{(Ecuación 1)}
\end{aligned}$$

Derivada con respecto a b:

$$
\begin{aligned}
\frac{\partial S}{\partial b} &= \sum_{i=1}^{n} 2(Y_i - a - bX_i - cX_i^2)(-X_i) = 0 \\
&\implies \sum_{i=1}^{n} (Y_iX_i - aX_i - bX_i^2 - cX_i^3) = 0 \\
&\implies \sum Y_iX_i - a\sum X_i - b\sum X_i^2 - c\sum X_i^3 = 0 \\
&\implies a\sum X_i + b\sum X_i^2 + c\sum X_i^3 = \sum Y_iX_i \quad \text{(Ecuación 2)}
\end{aligned}
$$

Derivada con respecto a c:

$$
\begin{aligned}
\frac{\partial S}{\partial c} &= \sum_{i=1}^{n} 2(Y_i - a - bX_i - cX_i^2)(-X_i^2) = 0 \\
&\implies \sum_{i=1}^{n} (Y_iX_i^2 - aX_i^2 - bX_i^3 - cX_i^4) = 0 \\
&\implies \sum Y_iX_i^2 - a\sum X_i^2 - b\sum X_i^3 - c\sum X_i^4 = 0 \\
&\implies a\sum X_i^2 + b\sum X_i^3 + c\sum X_i^4 = \sum Y_iX_i^2 \quad \text{(Ecuación 3)}
\end{aligned}
$$

Calcular las Sumatorias Necesarias

- $\sum X_i = 65$
- $\sum Y_i = 995$
- $\sum X_i^2 = 1071$
- $\sum X_i^3 = 20123$
- $\sum X_i^4 = 406275$
- $\sum X_iY_i = 18741$
- $\sum X_i^2Y_i = 378417$

Sustituir los Valores en el Sistema de Ecuaciones Normales

$$
\begin{gathered}
5a + 65b + 1071c = 995 \quad \text{(Ecuación 1)} \\
65a + 1071b + 20123c = 18741 \quad \text{(Ecuación 2)} \\
1071a + 20123b + 406275c = 378417 \quad \text{(Ecuación 3)}
\end{gathered}
$$

Resolver el Sistema de Ecuaciones Lineales con python Resolviendo el sistema se obtienen los siguientes valores aproximados:

```python
import numpy as np

# 5a + 65b + 1071c = 995
# 65a + 1071b + 20123c = 18741
# 1071a + 20123b + 406275c = 378417

A = np.array([
    [5, 65, 1071],
    [65, 1071, 20123],
    [1071, 20123, 406275]
])
```

```python

B = np.array([995, 18741, 378417])

try:
    solucion = np.linalg.solve(A, B)
    a_estimado = solucion[0]
    b_estimado = solucion[1]
    c_estimado = solucion[2]

    print(f"Estimador de a: {a_estimado}")
    print(f"Estimador de b: {b_estimado}")
    print(f"Estimador de c: {c_estimado}")

    print("\nVerificacion:")
    print(
        f"Ecuacion 1: {A[0,0]*a_estimado + A[0,1]*b_estimado + A[0,2]*c_estimado} (Esperado:
            {B[0]})")
    print(
        f"Ecuacion 2: {A[1,0]*a_estimado + A[1,1]*b_estimado + A[1,2]*c_estimado} (Esperado:
            {B[1]})")
    print(
        f"Ecuacion 3: {A[2,0]*a_estimado + A[2,1]*b_estimado + A[2,2]*c_estimado} (Esperado:
            {B[2]})")

except np.linalg.LinAlgError as e:
    print(f"Error al resolver el sistema: {e}")
    print("La matriz de coeficientes podria ser singular o no invertible.")
```

Listing 2: Código Python para resolucion de ecuaciones lineales.

Resultados de la Ejecución del Código Python

A continuación, se presentan los resultados obtenidos al ejecutar el script de Python:

```
Estimador de a: -8.715780854166221
Estimador de b: 1.372834514700814
Estimador de c: 0.8864095805931635

Verificación:
Ecuación 1: 995.0 (Esperado: 995)
Ecuación 2: 18740.999999999996 (Esperado: 18741)
Ecuación 3: 378417.0 (Esperado: 378417)
```

3 Ejercicio 3

La Distribución a Priori Beta con parámetros $\alpha_0 = 2$ y $\beta_0 = 3$. La función de densidad de probabilidad de una distribución Beta(α_0, β_0) es:

$$f(p) = \frac{1}{B(\alpha_0, \beta_0)} p^{\alpha_0 - 1}(1-p)^{\beta_0 - 1}$$

donde $B(\alpha_0, \beta_0) = \frac{\Gamma(\alpha_0)\Gamma(\beta_0)}{\Gamma(\alpha_0+\beta_0)}$ es la función Beta.

Don Francisco tiene $n = 8$ clientes, y de ellos, $k = 3$ están en mora. Este escenario se modela con una distribución Binomial.

La función de masa de probabilidad de la distribución Binomial(n, p) es:

$$P(k \mid p) = \binom{n}{k} p^k (1-p)^{n-k}$$

En nuestro caso, la verosimilitud es:

$$L(p \mid k=3, n=8) = \binom{8}{3} p^3 (1-p)^{8-3} = \binom{8}{3} p^3 (1-p)^5$$

La distribución a posteriori $P(p \mid \text{datos})$ es proporcional al producto de la distribución a priori y la verosimilitud:

$$P(p \mid \text{datos}) \propto P(p) \times L(p \mid \text{datos})$$

$$P(p \mid \text{datos}) \propto \left(\frac{1}{B(\alpha_0, \beta_0)} p^{\alpha_0 - 1} (1-p)^{\beta_0 - 1} \right) \times \left(\binom{n}{k} p^k (1-p)^{n-k} \right)$$

Quitando las constantes de proporcionalidad (que no dependen de p):

$$P(p \mid \text{datos}) \propto p^{\alpha_0 - 1 + k} (1-p)^{\beta_0 - 1 + n - k}$$

Esta forma corresponde a una distribución Beta con nuevos parámetros α' y β':

$$\alpha' = \alpha_0 + k$$

$$\beta' = \beta_0 + n - k$$

Sustituyendo los valores:

$$\alpha' = 2 + 3 = 5$$

$$\beta' = 3 + 8 - 3 = 3 + 5 = 8$$

Por lo tanto, la distribución a posteriori del porcentaje de morosidad p es una **distribución Beta($\alpha' = 5, \beta' = 8$)**.

La función de densidad de probabilidad a posteriori es:

$$P(p \mid \text{datos}) = \frac{1}{B(5,8)} p^{5-1} (1-p)^{8-1} = \frac{1}{B(5,8)} p^4 (1-p)^7$$

La media de una distribución Beta(α', β') está dada por la fórmula:

$$E[p \mid \text{datos}] = \frac{\alpha'}{\alpha' + \beta'}$$

Sustituyendo los valores de α' y β':

$$E[p \mid \text{datos}] = \frac{5}{5+8} = \frac{5}{13}$$

Como valor decimal:

$$E[p \mid \text{datos}] \approx 0.3846$$

El porcentaje de morosidad es aproximadamente del 38.46%. La varianza de una distribución Beta(α', β') está dada por la fórmula:

$$\text{Var}[p \mid \text{datos}] = \frac{\alpha' \beta'}{(\alpha' + \beta')^2 (\alpha' + \beta' + 1)}$$

Sustituyendo los valores de α' y β':

$$\text{Var}[p \mid \text{datos}] = \frac{5 \times 8}{(5+8)^2 (5+8+1)}$$

$$\text{Var}[p \mid \text{datos}] = \frac{40}{2366}$$

Como valor decimal:

$$\text{Var}[p \mid \text{datos}] \approx 0.016906$$

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

alpha_0 = 2
beta_0 = 3

mean_prior = alpha_0 / (alpha_0 + beta_0)
variance_prior = (alpha_0 * beta_0) / \
    ((alpha_0 + beta_0)**2 * (alpha_0 + beta_0 + 1))

print(f"--- Distribucin a Priori ---")
print(f"Parametros a priori (alpha_0, beta_0): ({alpha_0}, {beta_0})")
print(f"Media a priori: {mean_prior:.4f}")
print(f"Varianza a priori: {variance_prior:.6f}")
print("-" * 40)

n_clientes = 8
k_morosos = 3

print(f"--- Datos Observados ---")
print(f"Numero total de clientes (n): {n_clientes}")
print(f"Numero de clientes morosos (k): {k_morosos}")
print("-" * 40)

alpha_prime = alpha_0 + k_morosos
beta_prime = beta_0 + (n_clientes - k_morosos)

print(f"--- Distribucion a Posteriori ---")
print(
    f"Nuevos parametros a posteriori (alpha', beta'): ({alpha_prime}, {beta_prime})")

mean_posterior = alpha_prime / (alpha_prime + beta_prime)
print(
    f"Media a posteriori (E[p|datos]): {mean_posterior:.4f} (aprox. {mean_posterior*100:.2f}%)")

variance_posterior = (alpha_prime * beta_prime) / \
    ((alpha_prime + beta_prime)**2 * (alpha_prime + beta_prime + 1))
print(f"Varianza a posteriori (Var[p|datos]): {variance_posterior:.6f}")
print("-" * 40)

p_values = np.linspace(0, 1, 500)

pdf_prior = stats.beta.pdf(p_values, alpha_0, beta_0)

pdf_posterior = stats.beta.pdf(p_values, alpha_prime, beta_prime)

plt.figure(figsize=(10, 6))
plt.plot(p_values, pdf_prior,
        label=f'A Priori Beta({alpha_0}, {beta_0})', linestyle='--', color='blue')
plt.plot(p_values, pdf_posterior,
        label=f'A Posteriori Beta({alpha_prime}, {beta_prime})', color='red')
plt.axvline(mean_prior, color='blue', linestyle=':',
           label=f'Media a Priori ({mean_prior:.2f})')
plt.axvline(mean_posterior, color='red', linestyle=':',
           label=f'Media a Posteriori ({mean_posterior:.2f})')

plt.title('Distribucion A Priori y A Posteriori del Porcentaje de Morosidad')
```

```python
plt.xlabel('Porcentaje de Morosidad (p)')
plt.ylabel('Densidad de Probabilidad')
plt.legend()
plt.grid(True, linestyle=':', alpha=0.7)
plt.show()
```

Listing 3: Código Python para resolucion inferencia bayesiana.

Resultados de la Ejecución del Código Python

A continuación, se presentan los resultados obtenidos al ejecutar el script de Python:

```
Parámetros a priori (alpha_0, beta_0): (2, 3)
Media a priori: 0.4000
Varianza a priori: 0.040000
----------------------------------------
--- Datos Observados ---
Número total de clientes (n): 8
Número de clientes morosos (k): 3
----------------------------------------
--- Distribución a Posteriori ---
Nuevos parámetros a posteriori (alpha’, beta’): (5, 8)
Media a posteriori (E[p|datos]): 0.3846 (aprox. 38.46%)
Varianza a posteriori (Var[p|datos]): 0.016906
----------------------------------------
```

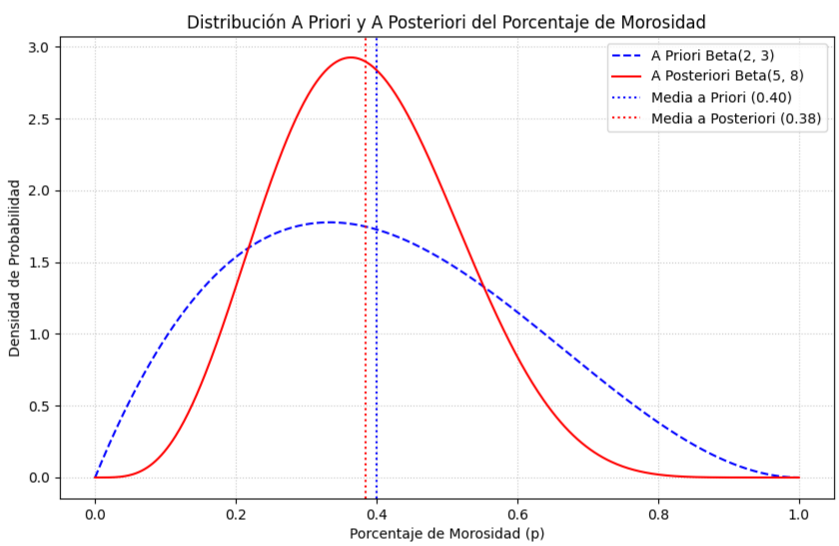

Figure 1: Distribución A Priori y A Posteriori del Porcentaje de Morosidad.