C1 ESTIMACIÓN DE MÁXIMA VEROSIMILITUD EN CONFIABILIDAD

November 19, 2025

1 Contexto

En un centro de datos de alto rendimiento, una **ingeniera especialista en Confiabilidad** es responsable de la estabilidad de un clúster crítico de bases de datos. La especialista ha recolectado registros (logs) precisos sobre los tiempos operativos de las instancias desde su despliegue hasta que ocurre un fallo (crash) o se requiere un reinicio forzado.

El equipo de ingeniería busca ir más allá del simple cálculo de promedios; el objetivo es modelar matemáticamente la curva de confiabilidad del sistema. Este análisis permitirá a la ingeniera y a su equipo responder preguntas estratégicas, tales como:

- ¿Cuál es la probabilidad de que un servidor se mantenga operativo por más de 30 días continuos?
- ¿Los fallos observados corresponden a defectos de fabricación ("mortalidad infantil") o a un desgaste natural por uso intensivo?

Para resolver esto, se procederá a ajustar una **Distribución de Weibull**, herramienta estándar en el análisis de supervivencia, utilizando métodos de optimización numérica avanzada.

2 Estimación de máxima verosimilitud

La función de densidad de probabilidad (PDF) utilizada para modelar el tiempo $t \geq 0$ hasta el fallo es:

$$f(t; k, \lambda) = \frac{k}{\lambda} \left(\frac{t}{\lambda} \right)^{k-1} e^{-(t/\lambda)^k}, \; t \geq 0$$

Donde los parámetros que la ingeniera debe estimar son:

- k (forma/shape): Indica la física del fallo. Si $k < 1$, la tasa de fallos decrese (problemas tempranos); si $k > 1$, la tasa aumenta (desgaste).
- λ (escala/scale): Representa la vida característica del componente.

Para encontrar los parámetros k y λ que mejor expliquen los datos recolectados, se utiliza el método de **estimación de máxima verosimilitud (MLE)**. En términos computacionales, esto se traduce en minimizar la función de **menos log-verosimilitud**:

$$-L(k,\lambda) = -\log\left(\prod_{i=1}^{n} f(t_i;k,\lambda)\right) = -n\log(k) + nk\log(\lambda) - (k-1)\sum_{i=1}^{n}\log(t_i) + \sum_{i=1}^{n}\left(\frac{t_i}{\lambda}\right)^k$$

3 Implementación

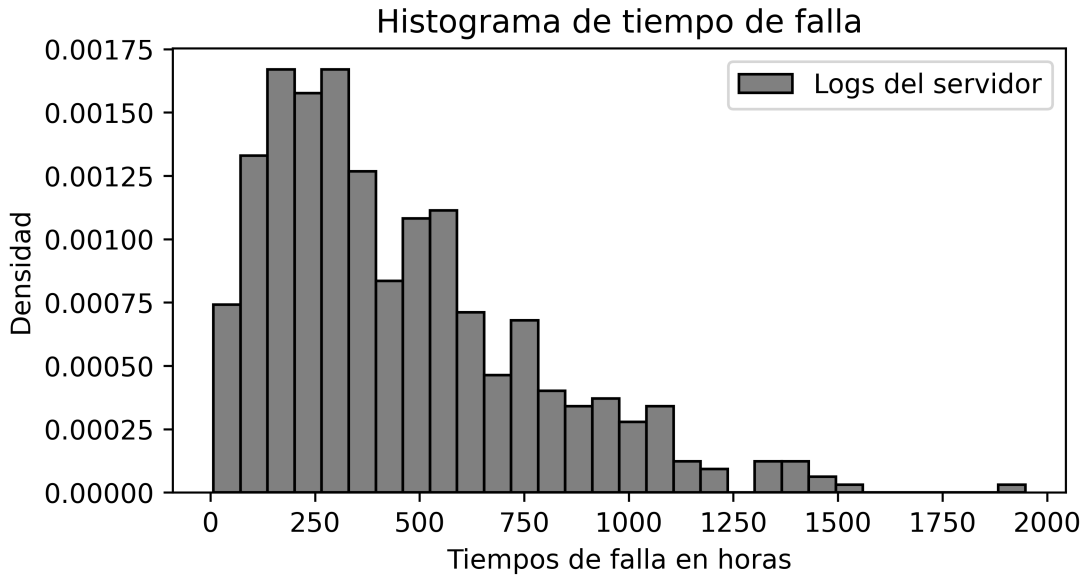
[1]:
```
# Cargar librerías necesarias
# ---------------------------
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import minimize
from scipy.stats import weibull_min

# Configuraciones iniciales
# -------------------------
plt.rcParams["figure.dpi"] = 600
```

[2]:
```
# Generación de datos simulados (logs del servidor)
# -------------------------------------------------
np.random.seed(2025)

# Parámetros reales
k_verdadero = 1.5
lambda_verdadero = 500
n_muestras = 500

# Generamos tiempos de falla (datos observables)
tiempos_falla = weibull_min.rvs(k_verdadero, scale = lambda_verdadero, size =␣
 ↪n_muestras)

print(f"Datos generados: {n_muestras} observaciones.")
print(f"Ejemplos de tiempos de falla en horas: {tiempos_falla[:5]}")
```

```
Datos generados: 500 observaciones.
Ejemplos de tiempos de falla en horas: [138.37520056 842.67357769 968.82377546
351.65121059 311.36141317]
```

[3]:
```
# Histograma de los tiempos de falla
# ----------------------------------
fig, ax = plt.subplots(figsize = (6, 3))
ax.hist(tiempos_falla, bins = 30, density = True,
        color = "gray", edgecolor = "black",
        label = "Logs del servidor")
ax.set_title("Histograma de tiempo de falla")
ax.set_xlabel("Tiempos de falla en horas")
ax.set_ylabel("Densidad")
ax.legend()
```

```
plt.show()
```



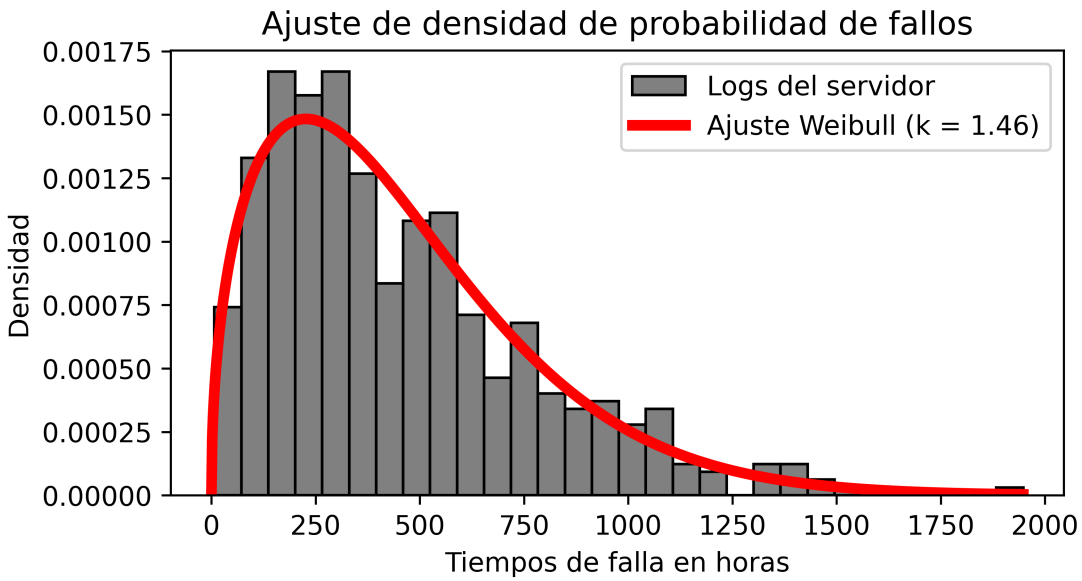
[4]:
```
# Definición del modelo de optimización
# -------------------------------------
def menos_log_verosimilitud(parametros, tiempos):
    k = np.exp(parametros[0])
    lam = np.exp(parametros[1])
    n = len(tiempos)
    sum_log_t = np.sum(np.log(tiempos))
    sum_t_lambda = np.sum((tiempos / lam) ** k)
    mlv = -n * np.log(k) + n * k * np.log(lam) - (k - 1) * sum_log_t +␣
 ↪sum_t_lambda
    return mlv
```

[5]:
```
# Proceso de optimización
# -----------------------
parametros_inicial = np.array([0.0, 0.0]) # Corresponde a k = 1 y lambda = 1

resultado = minimize(fun = menos_log_verosimilitud,
                     x0 = parametros_inicial,
                     args = (tiempos_falla),
                     method = "Nelder-Mead")

# Recuperar parámetros
k_estimado = np.exp(resultado.x[0])
lambda_estimado = np.exp(resultado.x[1])
```

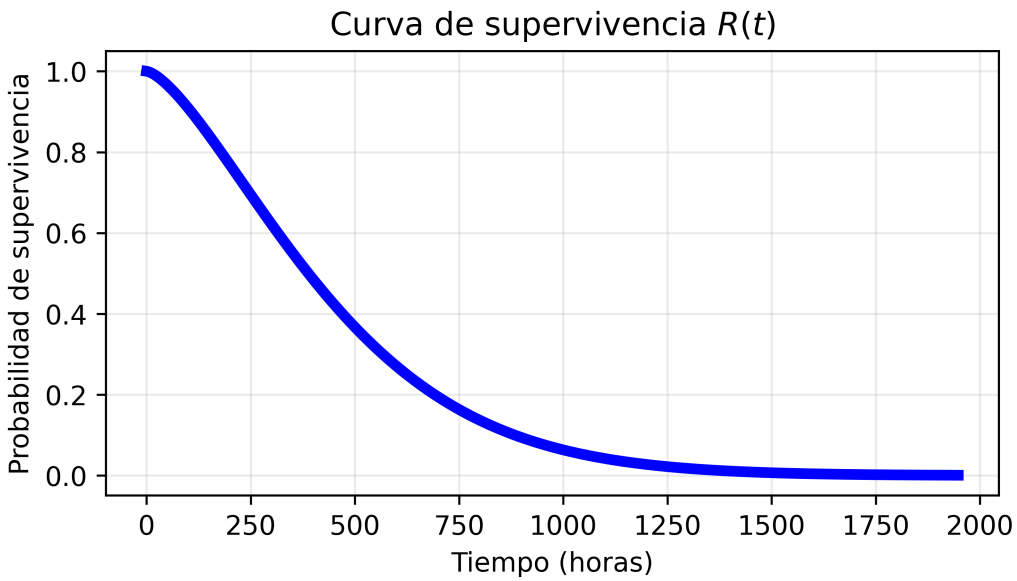
```
print("--- Resultados de la Estimación ---")
print(f"Éxito: {resultado.success}")
print(f"Iteraciones: {resultado.nit}")
print(f"Parámetros Reales:    k = {k_verdadero:.4f}, lambda = {lambda_verdadero:
  ↪.4f}")
print(f"Parámetros Estimados: k = {k_estimado:.4f}, lambda = {lambda_estimado:.
  ↪4f}")
```

```
--- Resultados de la Estimación ---
Éxito: True
Iteraciones: 93
Parámetros Reales:    k = 1.5000, lambda = 500.0000
Parámetros Estimados: k = 1.4610, lambda = 499.3229
```

[6]:
```
# Ajuste de densidad de probabilidad de fallos
# ---------------------------------------------
fig, ax = plt.subplots(figsize = (6, 3))
ax.hist(tiempos_falla, bins = 30, density = True,
        color = "gray", edgecolor = "black",
        label = "Logs del servidor")

x_ajuste = np.linspace(0, max(tiempos_falla), 1000)
y_ajuste = weibull_min.pdf(x_ajuste, k_estimado, scale = lambda_estimado)
ax.plot(x_ajuste, y_ajuste, 'r-', lw = 4,
        label = f"Ajuste Weibull (k = {k_estimado:.2f})")

ax.set_title("Ajuste de densidad de probabilidad de fallos")
ax.set_xlabel("Tiempos de falla en horas")
ax.set_ylabel("Densidad")
ax.legend()
plt.show()
```



[7]:
```
# Función de Supervivencia (Reliability Function R(t))
# R(t) = 1 - CDF(t)
fig, ax = plt.subplots(figsize = (6, 3))

reliability = np.exp(-(x_ajuste / lambda_estimado) ** k_estimado)

ax.plot(x_ajuste, reliability, "b-", lw = 4)
ax.set_title("Curva de supervivencia $R(t)$")
ax.set_xlabel("Tiempo (horas)")
ax.set_ylabel("Probabilidad de supervivencia")
ax.grid(True, alpha = 0.25)
plt.show()
```



4 Análisis de resultados

- El parámetro de forma estimado es $k \approx 1.5$. Al ser mayor que 1, la ingeniera confirma la hipótesis de que los servidores están fallando por desgaste operativo.
- **Implicancia técnica:** El riesgo de fallo aumenta con el tiempo. Esto indica al equipo de operaciones que no basta con reiniciar los servidores solo cuando fallan; se deben programar mantenimientos preventivos antes de que la confiabilidad caiga por debajo de un umbral crítico.

Juan F. Olivares Pacheco (`jfolivar@uda.cl`)

Universidad de Atacama, Facultad de Ingeniería, Departamento de Matemática