MARTINGALAS.

Nexus-Probability

CURSO 3 (PROCESOS ESTOCÁSTICOS I)

PARTE 1 / LECCIÓN 1

En esta sección, definiremos el concepto de martingala en tiempo discreto, el cual es fundamental para la comprensión de la segunda parte de este trabajo.

1. Filtraciones

En lo que sigue $(\Omega, \mathcal{F}, \mathbb{P})$ es un espacio de probabilidad, $\{\mathcal{F}_n\}_{n\in\mathbb{N}}$ es una filtración; esto es, una sucesión creciente de sub-σ-álgebras de $\mathcal{F}$: $\mathcal{F}_1 \subseteq \mathcal{F}_2 \subseteq \cdots \subseteq \mathcal{F}$, $\mathcal{F}_\infty := \sigma\left(\bigcup_{n\in\mathbb{N}} \mathcal{F}_n\right)$ y $X = (X_n)_{n\in\mathbb{N}}$ una sucesión de variables aleatorias. Nótese que si se define

$$\mathcal{F}_n^X := \sigma(X_1, \ldots, X_n),\ n \in \mathbb{N},$$

entonces $\{\mathcal{F}_n^X\}_{n\in\mathbb{N}}$ es una filtración; ésta se denomina la filtración natural de X.

Definición 1 (Martingala) *Se dice que $X = \{X_n, \mathcal{F}_n\}_{n\in\mathbb{N}}$ es una **martingala**, si para cada $n \in \mathbb{N}$ se verifica:*

1. $\mathbb{E}[|X_n|] < \infty$
2. X_n *es* $\mathcal{F}-$*medible.*
3. $X_n = \mathbb{E}[X_{n+1}|\mathcal{F}_n]$.

Nota 1 *Cuando el punto (3) se cambia por $X_n \geq \mathbb{E}[X_{n+1}|\mathcal{F}_\backslash]$, se dice que X es una **supermartingala**, y cuando se cambia por $X_n \leq \mathbb{E}[X_{n+1}|\mathcal{F}_\backslash]$, entonces se dice que X es una **submartingala.***

Si $X = \{X_n, \mathcal{F}_n\}_{n\in\mathbb{N}}$ es una submartingala y $f : \mathbb{R} \to \mathbb{R}$ es una función convexa creciente tal que $f(X_n)$ es integrable para cada $n \in \mathbb{N}$, entonces

$$f(X) = \{f(X_n), \mathcal{F}_n\}_{n\in\mathbb{N}}$$

es una submartingala. En efecto, por la desigualdad de Jensen,

$$X_n \leq E(X_{n+1}|\mathcal{F}_n) \text{ implica } f(X_n) \leq f(E(X_{n+1}|\mathcal{F}_n)) \leq E(f(X_{n+1})|\mathcal{F}_n).$$

En particular,

$$\{X_n^+, \mathcal{F}_n\}_{n\in\mathbb{N}}$$

es una submartingala.

Ejercicio 1 *El ejemplo más sencillo y fundamental de martingala se obtiene definiendo $X_n = E(Y|\mathcal{F}_n)$, donde Y es una variable aleatoria integrable y $\{\mathcal{F}_n\}_{n\in\mathbb{N}}$ una filtración. En efecto, por las propiedades de la esperanza condicionada, se tiene*

$$E(|X_n|) = E(|E(Y|\mathcal{F}_n)|) \leq E(E(|Y||\mathcal{F}_n)) = E(|Y|) < \infty,$$

y por la definición de esperanza condicionada X_n es $\mathcal{F}_n$-medible. Por último, por la propiedad de la esperanza condicionada se llega a

$$E(X_{n+1}|\mathcal{F}_n) = E(E(Y|\mathcal{F}_{n+1})|\mathcal{F}_n) = E(Y|\mathcal{F}_n) = X_n.$$

En este ejemplo la variable aleatoria Y no es necesariamente $\mathcal{F}_\infty$-medible; pero si definimos $X_\infty = E(Y|\mathcal{F}_\infty)$, entonces X_∞ es $\mathcal{F}_\infty$-medible e integrable y, por la propiedad de la esperanza condicionada, para todo $n \in \mathbb{N}$, se tiene

$$X_n = E(Y|\mathcal{F}_n) = E(E(Y|\mathcal{F}_\infty)|\mathcal{F}_n) = E(X_\infty|\mathcal{F}_n).$$

Más aún, si Z es otra variable aleatoria $\mathcal{F}_\infty$-medible tal que $X_n = E(Z|\mathcal{F}_n)$, $n \in \mathbb{N}$, entonces $Z = X_\infty$ c.t.p..

Este ejemplo motiva la siguiente definición.

Definición 2 (Elemento Final de X) *Sea $X = \{X_n, \mathcal{F}_n\}_{n\in\mathbb{N}}$ una martingala (submartingala, supermartingala). Si existe una variable aleatoria integrable y $\mathcal{F}_\infty$-medible X_∞ tal que para todo $n \in \mathbb{N}$ se tenga $X_n = E(X_\infty|\mathcal{F}_n)$ (respectivamente $X_n \leq E(X_\infty|\mathcal{F}_n)$, $X_n \geq E(X_\infty|\mathcal{F}_n)$), entonces se dice que X_∞ es último elemento de X o elemento final de X.*

De acuerdo con lo dicho arriba, si X_∞ y Z son elementos finales de una martingala $X = \{X_n, \mathcal{F}_n\}_{n\in\mathbb{N}}$, entonces $X_\infty = Z\, c.t.p.$ No es este el caso cuando X es una submartingala (supermartingala). En efecto, si $X_n \leq E(X_\infty|\mathcal{F}_n)$, $n \in \mathbb{N}$ ($X_n \geq E(X_\infty|\mathcal{F}_n)$, $n \in \mathbb{N}$) y c es una constante positiva, entonces $X_n \leq E(X_\infty + c|\mathcal{F}_n)$, $n \in \mathbb{N}$ ($X_n \geq E(X_\infty - c|\mathcal{F}_n)$, $n \in \mathbb{N}$). El concepto clave para decidir sobre la existencia de último elemento para una martingala es el de integrabilidad uniforme, recomendamos investigar mas sobre esto.

Ejercicios

Ejercicio 2 *Veremos como simular con* numpy *la martingala. Esta sera una estrategia de apuestas que esta popularizada en la ruleta y tiene la siguiente estructura:*

- *Se comienza con una apuesta Inicial. (Se apuesta siempre al negro.)*
- *Si se gana, se vuelve a la apuesta inicial.*
- *Si se pierde, se dobla la apuesta.*

Solución:

La esencia de esta estrategia es que cuando perdemos, doblamos la siguiente apuesta para intentar recuperar la cantidad perdida. En la teoría, suena bien.

Podemos implementar esta estrategia de la siguiente forma. Se define una cantidad `inicial`, una `apuesta`, una probabilidad de ganar `prob_ganar` y una cantidad `objetivo` después de la cual se para la simulación.

```
import random

def martingala(inicial, apuesta, prob_ganar, objetivo):
    balance = inicial
    bet = apuesta

    while balance >0 and balance < inicial + objetivo:
        rojo = random.random() < prob_ganar
        if rojo:
            balance -= bet
            bet *= 2
        else:
            balance += bet
            bet = apuesta

        if bet>balance:
            break
    return balance
```

Como puedes ver el código usa `random` para generar el evento. El `while` hace que continuamente se apueste mientras haya balance y no hayamos llegado al objetivo. Puedes ver los dos escenarios claramente definidos:

- Si sale Rojo perdemos. Entonces doblamos la apuesta `bet`
- Si sale **Negro** ganamos. Volvemos a la apuesta inicial.

Ahora podemos usar nuestra función con unos parámetros concretos. Usamos un balance inicial de 1000 Pesos, usando una apuesta inicial de 10 pesos, con una probabilidad de ganar del 35 % y un objetivo de 2000 pesos. Vemos el resultados

```
    print(martingala(inicial=1000,
                     apuesta=10,
                     prob_ganar=0.35,
                     objetivo=2000))
```

Nota 2 *Como puedes comprobar ejecutando la simulación múltiples veces, en la mayoría de las ocasiones se pierde. Esto es debido a que una racha de mala suerte hace que la apuesta crezca exponencialmente, y llegue un punto donde no se puede afrontar.*

Modifica los valores para buscar una estrategia que reduzca el riesgo de bancarrota.

Ejercicio 3 *Considera un proceso de caminata aleatoria simétrica* $\{X_n\}_{n\in\mathbb{N}}$ *definido de la siguiente manera:*

- $X_0 = 0$.
- *Para cada* $n \geq 1$, $X_n = X_{n-1} + Y_n$, *donde* Y_n *es una variable aleatoria que toma los valores* $+1$ *y* -1 *con probabilidad* $\frac{1}{2}$ *cada uno.*

Muestre que $\{X_n\}_{n\in\mathbb{N}}$ *es una martingala con respecto a su filtración natural* $\{\mathcal{F}_n\}_{n\in\mathbb{N}}$, *donde* $\mathcal{F}_n = \sigma(X_1, X_2, \ldots, X_n)$.

Solución:

A continuación, se presenta un código en Python que simula la caminata aleatoria y verifica la propiedad de martingala.

```
  import numpy as np
import matplotlib.pyplot as plt

# Parametros
n_steps = 1000
n_simulations = 10

# Funcion para simular la caminata aleatoria
def simulate_random_walk(n_steps):
    steps = np.random.choice([-1, 1], size=n_steps)
    walk = np.cumsum(steps)
    return walk

```

```
# Verificacion de la propiedad de martingala
def verify_martingale_property(walk):
    for i in range(1, len(walk)):
        expected_value = walk[i-1]
        if walk[i] != expected_value:
            return False
    return True

# Simulacion y verificacion
for _ in range(n_simulations):
    walk = simulate_random_walk(n_steps)
    is_martingale = verify_martingale_property(walk)
    print(f"Simulacion {_+1}:  Es  martingala? {is_martingale}")
    plt.plot(walk)

plt.title("Simulacion de una Caminata Aleatoria (Martingala)")
plt.xlabel("Pasos")
plt.ylabel("Posicion")
plt.show()
```

```
Simulacion 1: ¿Es martingala? True
Simulacion 2: ¿Es martingala? True
Simulacion 3: ¿Es martingala? True
Simulacion 4: ¿Es martingala? True
Simulacion 5: ¿Es martingala? True
```

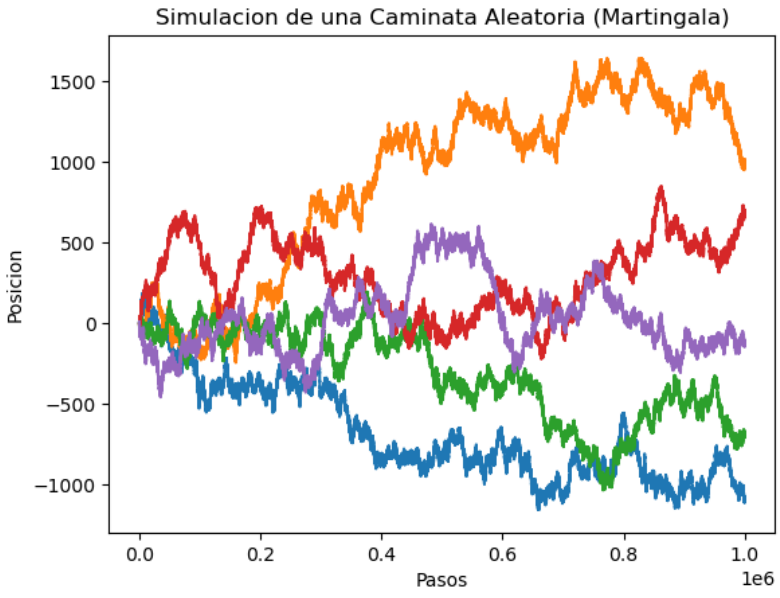

Figura 1:

Explicación:Simulación.

- **Simulación de la Caminata Aleatoria:** La función `simulate_random_walk` genera una caminata aleatoria simétrica donde en cada paso se suma o resta 1 con igual probabilidad.
- **Verificación de la Propiedad de Martingala:** La función `verify_martingale_property` verifica que el valor esperado de X_n dado $\mathcal{F}_{n-1}$ es X_{n-1}, lo cual es una propiedad clave de las martingalas.
- **Visualización:** Se realizan múltiples simulaciones y se grafican las caminatas aleatorias para visualizar el comportamiento del proceso.

Este código demuestra que la caminata aleatoria simétrica es una martingala, ya que cumple con la propiedad de que el valor esperado futuro dado el presente es igual al valor presente.

Ejercicio 4 (Finanzas.) *Supongamos que el precio de una acción S_n sigue un modelo discreto en el tiempo $n = 0, 1, 2, \ldots$, donde:*

- $S_0 = 100$ *(precio inicial).*
- $S_n = S_{n-1} \cdot e^{Y_n}$, *con Y_n una variable aleatoria que toma valores $+0.05$ o -0.05 con probabilidad $\frac{1}{2}$.*

Demuestra que el proceso $\{S_n\}_{n\geq 0}$ es una martingala bajo la medida de probabilidad neutral al riesgo. Luego, simula 100 trayectorias del precio de la acción para 30 días y verifica estadísticamente que el valor esperado condicional cumple $E(S_{n+1}|\mathcal{F}_n) = S_n$.

Solución:

```
import numpy as np
import matplotlib.pyplot as plt

# Par metros
n_days = 30
n_simulations = 100
initial_price = 100
delta = 0.05  # Variaci n diaria (5%)

# Funci n para simular una trayectoria del precio
def simulate_stock_price(n_days):
    price = [initial_price]
    for _ in range(n_days):
```

```
        # Generar movimiento +5% o -5% con igual probabilidad
        movement = np.random.choice([1 + delta, 1 - delta])
        price.append(price[-1] * movement)
    return price

# Simulaci n de m ltiples trayectorias
simulations = np.array([simulate_stock_price(n_days) for _ in range(n_simulations)])

# Verificaci n de la propiedad de martingala
errors = []
for n in range(n_days):
    # Calcular E(S_{n+1} | S_n) para todas las simulaciones
    current_prices = simulations[:, n]
    next_prices = simulations[:, n + 1]
    expected_next = current_prices * (0.5 * (1 + delta) + 0.5 * (1 - delta))  # E[movement] = 1
    errors.extend(next_prices - expected_next)  # Deber a aproximarse a 0

# An lisis estad stico de los errores
mean_error = np.mean(errors)
std_error = np.std(errors)
print(f"Error medio: {mean_error:.6f}")
print(f"Desviaci n est ndar del error: {std_error:.6f}")

# Visualizaci n de 10 trayectorias
plt.figure(figsize=(10, 6))
for i in range(10):
    plt.plot(simulations[i], lw=1)
plt.title("Simulaci n del Precio de una Acci n (Martingala)")
plt.xlabel("D as")
plt.ylabel("Precio (USD)")
plt.grid(True)
plt.show()
```

```
Error medio: 0.098455
Desviación estándar del error: 5.143335
```

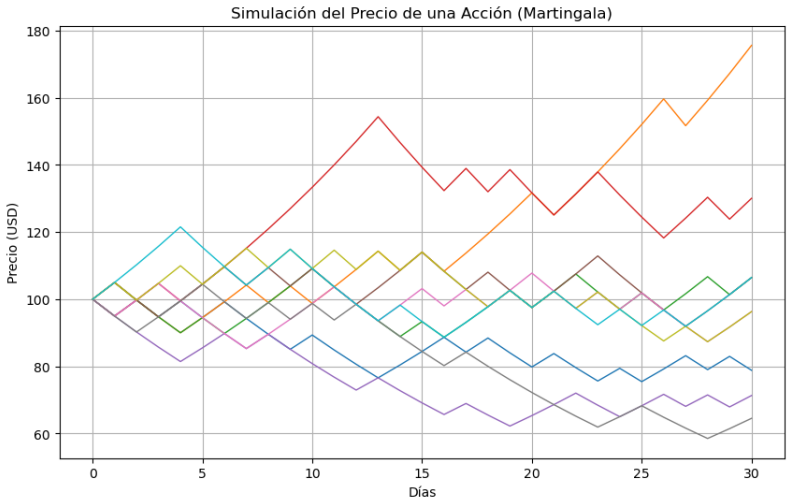

Figura 2: Simulación.

Explicación:

- **Modelo del Precio:** El precio se actualiza multiplicativamente con cambios de $\pm 5\,\%$. Bajo la medida neutral al riesgo, el factor esperado de crecimiento es $E[e^{Y_n}] = 1$, lo que implica $E(S_{n+1}|\mathcal{F}_n) = S_n$, cumpliendo la propiedad de martingala.
- **Verificación Estadística:**
  - Se calcula la diferencia entre S_{n+1} y $E(S_{n+1}|S_n)$ para todos los días y simulaciones.
  - Si el error medio es cercano a cero y la desviación estándar refleja solo ruido aleatorio, se valida la propiedad.
- **Visualización:** Las trayectorias muestran un comportamiento oscilatorio sin tendencia, consistente con una martingala.

Conclusión: El código demuestra que, en promedio, el precio futuro condicional es igual al precio actual, validando que $\{S_n\}$ es una martingala.