Guía para diseñar, mapear y resolver tu propio problema de decisión estática bajo incertidumbre (con ejemplo y código en Python)

Instrucciones

Este documento describe un proceso **paso a paso** para que los estudiantes **formulen** y **resuelvan** un **problema de decisión estática** bajo incertidumbre, incorporando información parcial (proxy), heurísticas y un pequeño ejemplo programado en Python.

La idea es que el lector aplique los conceptos en un **problema original** (inventado por sí mismo), definiendo:

1. **Qué se decide**
2. **Cuál es la incertidumbre** y el espacio de estados
3. **Qué información parcial** (proxy) se dispone
4. **Cómo se calcula** la utilidad o el costo
5. **Qué criterios** o heurísticas se usan para la decisión con pocos datos

Para ilustrar, desarrollaremos un **ejemplo lúdico**: tomar la decisión de **enviar (o no) un mensaje** a la persona que nos gusta, bajo incertidumbre acerca de sus intenciones y sentimientos, basándonos en ciertas **"señales"** (proxy) que podemos observar en redes sociales. Se añadirán elementos anecdóticos (gaslighting, etc.) para que sirvan de inspiración.

Al final, presentaremos **código en Python** (en forma de jupyter_notebook, scripts, txt y markdown) que demuestra cómo **definir el modelo**, **implementarlo** y **visualizar** el proceso de decisión.

Entregable

PROF

Entregable: Piensa en un problema de tu dia a dia, y utilizando lo que sabes formulalo como un problema de teoria de desicion estatica (con incertidumbre), y agrega codigo para entender mejor el problema y simularlo. Pueden usar cuantos scripts.py, jupyters y .md o .txt requierran (dentro de lo sensato) siempre y cuando este disponible online (ya sea github, drive, onedrive, colab, etc..). Tota tu informacion tiene que ser accesible y online para que la pueda calificar. Al ser tu proyecto tu decides la estructura, visualizaciones, heuristicas, el problema, etc.

- Definicion matematica y clara del problema (puedes usar latex, mardown o subir una foto clara y pegarla en un archivo markdown o directo en un jupyter notebook). Desde la idea general, cuales son los estados, probabilidades, funciones de perdida, etc. La idea es que des una explicacion muy detallada y formal del problema que tu quieras.
- Codigo donde se define el problema, se visualiza su definicion.
- Por lo menos dos heuristicas/solucione para el problema. Tienes que describir por que las tomaste (o como se inspiraron). Compara las heuristicas usando varios ejemplos e interpreta.
- **Importante** La descripcion del problema, como se mapea a un problema de decision, las heuristicas, metricas y comparaciones tienen que ser muy detalldas pues es lo mas importante.

1. Planeación y mapeo del problema

1.1 Preguntas de prompting para iniciar

Antes de plantear un problema de decisión bajo incertidumbre, conviene hacerse preguntas clave (puedes usar esto como lista de *prompts* para ti mismo):

1. **Objetivo**: ¿Qué quiero lograr y cuál es la acción principal a decidir?
2. **Contexto**: ¿Dónde ocurre la decisión? ¿Qué factores externos influyen?
3. **Incertidumbre**: ¿Qué aspectos del mundo (o de la contraparte) no conozco con certeza y podrían afectar mi resultado?
4. **Métricas de éxito**: ¿Cómo valoro o cuantifico el resultado de mi acción? ¿Gano algo? ¿Pierdo algo?
5. **Información parcial (proxy)**: ¿Tengo datos, señales o indicios (aunque ruidosos) que puedan guiar mi decisión?
6. **Recursos o restricciones**: ¿Hay límites de tiempo, capacidad, dinero o credibilidad?
7. **Posibles heurísticas**: ¿Si no puedo modelarlo todo con precisión, qué reglas aproximadas (basadas en experiencia o datos parciales) puedo usar?

1.2 Definición conceptual de un problema genérico

Siguiendo el documento teórico previo, un **problema de decisión estática bajo incertidumbre** incluye:

1. **Decisión**: Conjunto de acciones (D)
2. **Estados de la naturaleza** (\Omega): Configuraciones posibles del mundo
3. **Información proxy** (Z): Observaciones ruidosas correlacionadas con (\Omega)
4. **Función de utilidad** (U(\omega, d)) o costo
5. **Modelo probabilístico**: (p(\omega)) o (p(\omega \mid Z))

Tu objetivo:
[
d^* = \arg\max_{d \in D}; \mathbb{E}[U(\omega, d)].
]
(O el criterio que definas, p. ej. maximin, etc.)

PROF

---

2. Ejemplo ilustrativo: "¿Le escribo o no a la persona que me gusta?"

Para mostrar cómo se aplica el proceso, usaremos un **escenario** hipotético: Tienes un "crush" y deseas **decidir** si **enviarle** un mensaje (o no) con el fin de conseguir cierta respuesta positiva. Sin embargo, desconoces su verdadero **interés** o intención. Usarás señales en redes sociales para inferir la probabilidad de éxito.

2.1 Elementos del ejemplo

1. **Decisiones** $((D))$:
    - (d_0): No enviar mensaje

- (d_1): Enviar mensaje de texto (WhatsApp)
- (Podrías agregar más decisiones: mandar meme, story reply, pedir cita directa, etc.)

2. **Estados de la naturaleza** ((\Omega)):

   - (\omega_0): La persona no está interesada en ti en absoluto
   - (\omega_1): La persona está "indiferente/ambigua"
   - (\omega_2): La persona siente cierto interés
   - (\omega_3): La persona es "malevolente/gaslighter" (enredará la situación)

   (Aquí, (\omega_3) es un estado peculiar: si la persona actúa con manipulación, tal vez tu iniciativa lleve a un outcome negativo.)

3. **Información proxy** ((Z)):
   Observaciones (señales en redes) antes de tomar la decisión:

   - Si ve tus estados de Instagram de inmediato
   - Si da like/comparte tus publicaciones de TikTok
   - Si te comenta tus historias con emojis
   - Si te envía indirectas o reacciona a tu contenido
   - Si (al contrario) hace ghosting o reacciona de forma extraña

   Podríamos codificar estas señales en un vector (por ejemplo, (\text{Proxy} = [1,1,0,1]) = "te ve, te likea, no te comparte, te comenta un emoji").

4. **Función de utilidad** (U(\omega, d)):

   - Si **no envías** el mensaje ((d_0)), tu utilidad es pequeña pero estable (evitas rechazo, pero no avanzas)
   - Si **envías** ((d_1)) y la persona está interesada ((\omega_2)), tu utilidad es alta (mayor "felicidad")
   - Si la persona te ignora o te hace "ghosting" ((\omega_0)), tu utilidad puede ser negativa (sientes rechazo)
   - Si la persona es **gaslighter** ((\omega_3)), el desenlace puede conllevar estrés, manipulación, etc. (peor que el simple rechazo)

   PROF

   Ejemplo sencillo de tabla:

Estado ((\omega))	**Decisión (d_0) (No envío)**	**Decisión (d_1) (Envío)**
(\omega_0) (0% int)	(U=0)	(U=-5) (rechazo/ghost)
(\omega_1) (ambig)	(U=0)	(U=1) (respuesta tibia)
(\omega_2) (int)	(U=0)	(U=+10) (gran acercam.)
(\omega_3) (gaslighter)	(U=0)	(U=-8) (manipulación)

5. **Modelo probabilístico**:

- A priori, (\omega) tiene ciertas probabilidades (p. ej., la persona con un 25% de estar interesada, 25% ambig, 45% sin interés, 5% gaslighter)
- Con la **información proxy** ({Z=z}), ajustamos la distribución. Por ejemplo, si "te reacciona a historias + likes + te invita a un plan", podría aumentar la probabilidad de (\omega_2); si "no contesta nada" pero te da un like "extraño", quizá (\omega_1) o (\omega_3) crezcan.

2.2 Heurísticas con pocos datos

Si solamente tienes "2-3 interacciones" en tu historial y poca claridad, podrías:

- **Regla de dedo**: "Si la persona ve mis historias y me escribe, asumo un 70% de probabilidad de interés ((\omega_2)) y decido enviar mensaje. Caso contrario, no lo envío."
- **Maximin**: Evito la decisión que podría resultar en el peor outcome (gaslighting = -8). Así no envío nunca (d_0)
- **Most likely state**: Observa la señal e infiere el estado más probable. Toma la decisión que maximice la utilidad para ese estado "ganador"

Conforme reúnas más evidencias (más casos de "a cuántas personas les fue bien si vieron likes y reacciones"), irás refinando la probabilidad.

---

3. Guía de diseño para tu propio problema

A continuación, una lista detallada de pasos para que diseñes **tu** problema de decisión estática:

1. **Elige un dominio** que te interese (ej.: producción, marketing, relaciones personales, logística, etc.)
2. **Identifica las decisiones principales** ((D)) (p. ej., fabricar 10 o 20 unidades, enviar o no un mensaje, invertir en x o y...)
3. **Enumera los estados de la naturaleza** ((\Omega)) (p. ej., alta demanda, baja demanda, la persona interesada o no, etc.)
4. **Especifica la utilidad**: ¿Cómo es el resultado si tomas cada decisión en cada estado? (puede ser monetario o subjetivo)
5. **Describe la incertidumbre**: ¿Por qué no conoces (\omega)? ¿Cómo la modelas (distribución)?
6. **Establece la información parcial** (proxy) que **observas** antes de decidir: ¿datos de clima, redes, mercado, etc.?
7. **Proponte** un criterio de decisión: ¿maximizar valor esperado? ¿maximin? ¿regla de umbral?
8. **Considera heurísticas** si no puedes estimar todo con precisión
9. **Opcional**: si el problema es demasiado grande, recorta la complejidad (número de estados, de variables proxy, etc.)

PROF

---

4. Código de ejemplo en Python

Para **concretar** cómo llevar tu problema a la práctica, daremos una mini-demostración en Python usando el ejemplo de "enviar o no el mensaje". Puedes ejecutarlo en un *notebook* (Jupyter, Google Colab) o en un archivo `.py`.

4.1 Estructura básica

```
import numpy as np

# 1. Definir estados y decisiones
estados = ["No int", "Ambig", "Int", "Gaslighter"]
decisiones = ["No enviar", "Enviar"]

# 2. Utilidad en formato de matriz (fila=estado, columna=decision)
U = np.array([
    [0,  -5],  # omega_0: no interesado
    [0,   1],  # omega_1: ambiguo
    [0,  10],  # omega_2: interesado
    [0,  -8]   # omega_3: gaslighter
])

# 3. Probabilidades a priori de cada estado
p_omega = np.array([0.45, 0.25, 0.25, 0.05])  # deben sumar 1

# 4. Valor esperado de cada decision sin proxy (simple)
VE_no_enviar = np.sum(p_omega * U[:,0])
VE_enviar = np.sum(p_omega * U[:,1])
print("Valor esperado DECISION SIN OBSERVACIONES:")
print(f"print("\nValor esperado DECISION DADO z=1:")
print(f" - No enviar: {VE_no_env_z1:.2f}")
print(f" - Enviar   : {VE_env_z1:.2f}")

best_dec_z1 = "No enviar" if VE_no_env_z1 > VE_env_z1 else "Enviar"
print(f"Mejor decision si te da like = {best_dec_z1}")
```

Explicación

PROF

1. Calculamos la **probabilidad** de que el "like" ocurra
2. Con la **regla de Bayes**, actualizamos la probabilidad de cada estado al ver (z=1)
3. Recalculamos el **valor esperado** de cada decisión bajo esa distribución posterior
4. Comparamos y tomamos la mejor decisión si "sí hay like"

4.2 Informacion Proxy

```
# 1. Matriz p(z=1 | omega)
p_z_given_omega = np.array([0.05, 0.30, 0.70, 0.20])

# 2. p(z=1) = sum over all states of p(z=1|omega)*p(omega)
p_z1 = np.sum(p_z_given_omega * p_omega)
# 3. Posterior p(omega|z=1) via Bayes
#    p(omega_i|z=1) = p(z=1|omega_i)*p(omega_i) / p(z=1)
post_omega_z1 = (p_z_given_omega * p_omega) / p_z1
```

```
print("Probabilidad a posteriori dada z=1 (me da like):")
for i, st in enumerate(estados):
    print(f"  {st}: {post_omega_z1[i]:.2f}")

# 4. Valor esperado de cada decision dado z=1
VE_no_env_z1 = np.sum(post_omega_z1 * U[:,0])
VE_env_z1    = np.sum(post_omega_z1 * U[:,1])

print("\nValor esperado DECISION DADO z=1:")
print(f" - No enviar: {VE_no_env_z1:.2f}")
print(f" - Enviar   : {VE_env_z1:.2f}")

best_dec_z1 = "No enviar" if VE_no_env_z1 > VE_env_z1 else "Enviar"
print(f"Mejor decision si te da like = {best_dec_z1}")
```

4.3 Extensión a varias señales

Si tienes más señales, podrías crear un vector `p_z_given_omega` multidimensional o factorizar las probabilidades. Con pocos datos, es probable que tengas que usar **heurísticas** (p. ej., "si la persona da me gusta y responde mensajes, subo la probabilidad de 'int' a 0.8").

4.4 Visualizaciones

Para ayudar a la **interpretación**, podrías graficar cómo cambia la utilidad esperada según probabilidad de interés. Ejemplo:

```
import matplotlib.pyplot as plt

# Ejemplo: variemos la probabilidad de que esté interesado
# y veamos la utilidad esperada de Enviar vs No Enviar.
p_int_range = np.linspace(0,1,50)
VE_noenv = []
VE_env = []

for p_int in p_int_range:
    # supondremos p_omega = [p_noint, p_ambig, p_int, p_gas]
    # por simplicidad, fijamos p_noint+p_ambig+p_gas=1 - p_int
    # digamos p_noint=0.4(1-p_int), p_ambig=0.3(1-p_int), p_gas=0.3(1-
p_int)
    p_noint  = 0.4*(1-p_int)
    p_ambig  = 0.3*(1-p_int)
    p_gas    = 0.3*(1-p_int)
    p_curr = np.array([p_noint, p_ambig, p_int, p_gas])

    ve0 = np.sum(p_curr * U[:,0])
    ve1 = np.sum(p_curr * U[:,1])
    VE_noenv.append(ve0)
    VE_env.append(ve1)
```

PROF

```
plt.plot(p_int_range, VE_noenv, label="No enviar")
plt.plot(p_int_range, VE_env, label="Enviar")
plt.xlabel("Probabilidad de estar interesado (p_int)")
plt.ylabel("Valor esperado de la decisión")
plt.legend()
plt.title("Comparación de decisiones vs prob. interés")
plt.show()
```

Este tipo de gráfico te permite ver en qué rango de probabilidad de interés conviene "enviar" o "no enviar".

---

5. Conclusiones y próximos pasos

1. **Diseño**: Definir de forma clara los **estados**, **decisiones**, **utilidad** y **proxy** es fundamental
2. **Heurísticas y pocos datos**: Cuando no podemos estimar con precisión las distribuciones, usamos **reglas aproximadas**:
    - Reglas de umbral
    - Máxima verosimilitud
    - Ajustes basados en conteos empíricos
3. **Implementación**: Con Python (o cualquier otro lenguaje), puedes **prototipar** tus suposiciones, ir probando distintos valores y **visualizar** resultados (por ejemplo, gráficas de valor esperado)
4. **Iterar**: A medida que obtienes más datos o feedback real, **actualiza** tus probabilidades y la utilidad si aparecen nuevos escenarios (ej. un estado extra, "la persona salió con otra persona", etc.)

5.1 Recomendaciones para tu ejercicio final

- **Elige un tema** que te resulte interesante o útil: puede ser social, empresarial, creativo, etc.
- **Reduce** el número de estados y decisiones para que sea manejable (3-5 estados, 2-4 decisiones)
- **Define la utilidad** en una matriz (estados x decisiones)
- **Crea** un pequeño **script en Python** que:
    1. Declara las probabilidades a priori de los estados
    2. Define la utilidad
    3. Calcula el **valor esperado** de cada decisión
    4. (Opcional) Integra la **información proxy** y actualiza la distribución a posteriori
    5. Determina la **decisión** que maximiza tu criterio
- **Haz pruebas** con diferentes entradas y observa cómo cambian los resultados

PROF

5.2 Ideas para extender tu ejercicio

- **Múltiples proxys**: Incluir más señales y estimar (p(z_1, z_2 \mid \omega))
- **Costos extras**: Ej. costo emocional de insistir, costo de inventario en el caso de un problema de stock, etc.

- **Simulaciones**: Generar escenarios aleatorios y ver cuántas veces tu decisión heurística da buen resultado
- **Añadir comportamientos**: por ejemplo, si la otra persona es gaslighter, tu utilidad puede variar con más matices

---

6. Comentario final

Este **modelo** (y el ejemplo del "mensaje al crush") no deja de ser una **simplificación** de la realidad. La **fortaleza** de la teoría de la decisión estática bajo incertidumbre radica en que te obliga a:

- **Estructurar** el problema
- **Hacer explícita** la incertidumbre
- **Reflexionar** sobre cómo la información parcial afecta tus creencias
- **Analizar** tus preferencias (utilidades)

Siguiendo estos pasos y usando Python para experimentos, **los estudiantes** pueden **crear** su propio **ejercicio**, ver cómo la teoría se traduce en código, y entender mejor la relevancia de la **incertidumbre** y las **heurísticas** en la toma de decisiones.

¡Anímate a definir tu propio escenario y a compartir tu código y conclusiones con tus compañeros!

_Nota: Este documento está diseñado para ayudarte a estructurar y resolver problemas de decisión bajo incertidumbre. Recuerda que el ejemplo del "crush" es solo ilustrativo - puedes aplicar estos conceptos a cualquier dominio que te interese.