# 3 Razonamiento inductivo: inferencia y probabilidad

## 3.1 Introducción al razonamiento inductivo



En informático, al igual que en muchos ámbitos de la vida, se usa el pensamiento lógico para razonar. Cuando se habla del razonamiento inductivo, se trata de un tipo de pensamiento lógico usado para inferir conclusiones generales a partir de observaciones o evidencias específicas. Este razonamiento se contrapone en cierta manera al razonamiento deductivo, del cual se diferencia en que las conclusiones se derivan de premisas específicas de manera necesaria (deductivo) mientras que el inductivo implica llegar a conclusiones probables basadas en la información disponible.

En otras palabras, se podría resumir de la siguiente forma:

- **Razonamiento deductivo**: se basa en la demostración de una teoría o hipótesis previa. Por ejemplo, si veo que una tostada lleva mucho tiempo en la tostadora, puedo deducir que se va a quemar.
- **Razonamiento inductivo**: se basa en la observación de hechos particulares para llegar a una generalización. Por ejemplo, si veo una tostada quemada, puedo inferir que ha estado mucho tiempo en la tostadora.

En este tema, se estudiará el razonamiento inductivo.

Hay algunos conceptos clave del razonamiento inductivo que se deben tener en cuenta antes de comenzar el tema, como son la generalización, la probabilidad, la incertidumbre y las hipótesis.

**Definiciones**

**Generalización**: se basa en la observación de casos particulares para usar esa información para llevar a cabo una conclusión que se aplique a un rango amplio de situaciones. Por ejemplo, si todos los gatos que

he tenido son cariñosos, puedo concluir que todos los gatos lo son.
**Probabilidad**: estas conclusiones o afirmaciones no tienen que ser absolutas ni verdaderas en todos los casos, por lo que se tienen que estudiar bajo la probabilidad e inferencia. En el ejemplo, no se puede estar seguro al 100% de que todos los gatos que existen son cariñosos, sino basarnos en una probabilidad alta de que así sean.
**Incertidumbre**: las conclusiones se basan en un número limitado de observaciones, por lo que hay un grao de incertidumbre en la conclusión. Cuantos más datos se tengan, más probable será la afirmación. Por ejemplo, el número de gatos que he tenido es muy limitado, por lo que en un futuro, la tenencia de nuevos gatos pueden cambiar la conclusión inductiva, por lo que hay cierto grado de incertidumbre.
**Hipótesis**: el razonamiento inductivo lleva a la formulación de hipótesis que deben ser probadas con análisis adicionales. Si mi hipótesis es que los gatos son cariñosos, tengo que comprobar que los gatos de las personas de mi alrededor también los son, para tener más pruebas de ello.

En las siguientes secciones se analizará con más detalle la inferencia y probabilidad en este tipo de razonamiento.

## 3.2 Inferencia en sistemas estocásticos

En el mundo real, los sistemas y procesos no se pueden predecir con certeza, es decir, no son completamente deterministas, comportándose de manera probabilística o aleatoria. Para representar y comprender estos sistemas, se utilizan modelos estocásticos, que incorporan elementos de incertidumbre y aleatoriedad.

A continuación, se analizan algunas de las técnicas más conocidas para inferir en sistemas estocásticos, dentro de las categorías de filtrado, muestreo, predicción, estimación y aprendizaje.
**Nota**: la inferencia bayesiana aparece en el capítulo 3.3.5, para una mayor comprensión tras explicar el Teorema de Bayes.

### 3.2.1 Filtrado estocástico: Kalman y Partículas

En ciertos problemas de la inteligencia artificial, como pueden ser la estimación del estado oculto en modelos de series temporales o en la localización de sistemas de navegación, se suelen usar técnicas de filtrado estocástico. Hay dos muy conocidas, y que sirven como primera toma de contacto: filtros de Kalman y filtro de partículas.

#### Filtro de Kalman

El filtro de Kalman es una técnica recursiva de estimación que se utiliza para estimar el estado de un sistema dinámico a partir de observaciones ruidosas. Proporciona una manera eficiente de combinar información de medidas pasadas y actuales para obtener una estimación precisa del estado presente que evoluciona con el tiempo. Sus principales aplicaciones

son en la localización GPS y en el cálculo de PIB.
A nivel de cálculo, el modelo se basa en una ecuación para el estado a estudiar y en otra para las observaciones de ese estado, que serían combinación lineal* del estado y del ruido (variable aleatoria perturbadora de las observaciones). Además, el algoritmo estima con una minimización del error cuadrático medio (diferencia entre el valor de estimación y el real).

Entre las bondades del algoritmo se encuentra la ausencia de almacenamiento de toda la información sobre las observaciones pasadas, puesto que trata con estimaciones lineales, y su sencillez para resolver problemas complejos de estimación.

Como se ha comentado antes, se trata de un algoritmo recursivo, donde se repiten dos fases principales: predicción y corrección (o actualización).

def kalmanFilter($X_{t-1}, P_{t-1}, u_t, z_t$)
1.- Etapa de predicción:
$\overline{x}_t = A_t x_{t-1} + B_t u_t$ %Proyección del estado hacia adelante
$\overline{P}_t = A_t P_{t-1} A_t^T + R_t$ %Proyección del error hacia adelante
2.- Etapa de corrección:
$K_t = \overline{P}_t C_t^T (C_t \overline{P}_t C_t^T + Q_t)^{-1}$ %Computo de la ganancia de Kalman
$x_t = \overline{x}_t + K_t(z_t - C_t \overline{x}_t)$ %Corrección del estado de la medida ($z_t$)
$P_t = (I - K_t C_t)\overline{P}_t)$ %Corrección de la convarianza del error
%Devuelve $x_t, P_t$

Donde:
$A_t$ es la matriz nxn que relaciona el estado en el instante t-1 con el estado en el instante t, en ausencia de señales de control.
$B_t$ es la matriz nxl que relaciona las señales de control (opcionales) con el estado actual.
$R_t$ es la matriz nxn que representa a la convarianza del error del ruido del proceso.
$C_t$ es la matriz mxn que relaciona el estado actual con las observaciones del entorno. $Q_t$ es la matriz nxn que representa a la covarianza del error del ruido de las observaciones.
$K_t$ es la matriz nxm que representa a la ganancia de Kalman.

**¿Qué es la ganancia de Kalman?** Cuando se tienen observaciones y una predicción del estado del sistema, la ganancia de Kalman indica cuánto peso se debe dar a cada una de estas fuentes de información para obtener una estimación óptima del estado actual del sistema. Un valor alto de ganancia significa que las observaciones son fiables (baja incertidumbre), mientras que si son bajas, el filtro confiará más en la predicción del estado (alta incertidumbre).

Para poner en práctica y comprender mejor este algoritmo, se pone un ejemplo práctico a continuación:

* Existe otra versión para sistemas estocásticos no lineales: Filtro de Kalman extendido

**Paso 1: Inicialización**

Se inicializa el filtro con dos componentes principales: el estado inicial y la matriz de covarianza inicial. El estado inicial es la mejor estimación inicial del estado del sistema, y la matriz de covarianza inicial recoge la incertidumbre asociada a esa estimación.

```
#Importar libreria numpy
import numpy as np

#Inicializacion del estado inicial y la matriz de covarianza inicial
estado_inicial = np.array([0, 0])  #Estado inicial [0, 0]
mat_cov_inicial = np.eye(2)  #La matriz de covarianza inicial es la
                                identidad
```

**Paso 2: Predicción del estado**

Para predecir el estado futuro y su matriz de covarianza se hace uso del modelo dinámico del sistema. En el caso de disponer de controles externos, se pueden agregar en este paso.

```
#Modelo dinamico simple (por ejemplo, movimiento rectilineo constante)
A = np.array([[1, 1], [0, 1]])  #Matriz de transicion de estado
B = np.array([0.5, 1])  #Vector de control (opcional)

#Prediccion del estado futuro
estado_predicho = np.dot(A, estado_inicial) + np.dot(B, controles)
mat_cov_predicha = np.dot(np.dot(A, mat_cov_inicial), A.T)
```

**Paso 3: Corrección (actualización) a partir de las observaciones**

Comparativa de las predicciones y las observaciones actuales para mejorar la estimación del estado y su incertidumbre.

```
#Observaciones actuales (simuladas)
observacion = np.array([2.2, 1.8])

#Modelo de observacion (por ejemplo, medicion lineal)
H = np.array([[1, 0], [0, 1]])  #Matriz de observacion
med_ruido = np.eye(2) * 0.1  #Ruido de medicion

#Calculo de la ganancia de Kalman
K = np.dot(np.dot(mat_cov_predicha, H.T), np.linalg.inv(np.dot(np.dot(
                                H, mat_cov_predicha), H.T) +
                                med_ruido))

#Actualizacion del estado estimado y la matriz de covarianza
estado_actualizado = estado_predicho + np.dot(K, observacion - np.dot(
                                H, estado_predicho))
mat_cov_actualizada = np.dot(np.eye(2) - np.dot(K, H),
                                mat_cov_predicha)
```

**Paso 4: Repetición del proceso**

Este proceso se repite en cada paso t de tiempo, utilizando las observaciones más recientes para mejorar la estimación del estado del sistema.

**Filtro de Partículas**

Aunque el filtro de Kalman extendido viene a cubrir la problemática en sistemas estocásticos no lineales y no gaussianos, existe otro método que se usa para ello, siendo muy sencillo y fácil de implementar. Se trata del algoritmo de filtro de partículas (Particle Filter).
Es un método de estimación estadística usado en la inferencia bayesiana para estimar la posición, estado o parámetros de un sistema dinámico que evoluciona en el tiempo y está sujeto a incertidumbre.

Entre sus principales aplicaciones está la localización y seguimiento de objetos (vehículos, robots, etc) a través del un entorno (abierto o cerrado), los sistemas de navegación GPS, el seguimiento de objetos o personas (reconocimiento facial) en videos, o la estimación del estado de sistemas basados en sensores.

A continuación, se explican los pasos del algoritmo, con un ejemplo sencillo en lenguaje Python:

**Paso 1: Inicialización de partículas**

Se inicia con un conjunto de partículas que representan diferentes hipótesis sobre el estado del sistema. Cada partícula tiene una posición, estado o conjunto de parámetros asociados.

```
import random

#Numero de particulas
num_particulas = 100

#Inicializacion de particulas aleatorias en el rango de la posicion
particulas = [random.uniform(0, 100) for _ in range(num_particulas)]
```

**Paso 2: Predicción**

Se predice cómo evolucionarán las partículas en el siguiente paso de tiempo utilizando un modelo de transición que describe cómo el sistema cambia con el tiempo. Se agrega ruido para representar la incertidumbre en las predicciones.

```
def modelo_de_transicion(particula):
    #Se simula un movimiento aleatorio positivo con una cierta
                                velocidad constante
    velocidad = random.uniform(0, 2)
    return particula + velocidad

#Se aplica el modelo de transicion a todas las particulas
particulas = [modelo_de_transicion(p) for p in particulas]
```

**Paso 3: Actualización**

Se compara cada partícula con las observaciones reales del sistema. Cada partícula obtiene un peso que refleja cómo de bien se ajusta a las

observaciones. Las partículas que se ajustan mejor a las observaciones reciben pesos más altos.

```
posicion_real = 27  #Posicion real del objeto

def calcular_peso(particula, medicion, desviacion_estandar):
    #Se calcula la probabilidad de que la particula sea consistente con la medicion
    probabilidad = 1 / (desviacion_estandar * (2 * 3.14159) ** 0.5) * \
                math.exp(-(particula - medicion) ** 2 / (2 * desviacion_estandar ** 2))
    return probabilidad

desviacion_estandar = 4  #Desviacion estandar del ruido de medicion

#Se calculan los pesos de las particulas en funcion de la medicion
pesos = [calcular_peso(p, posicion_real, desviacion_estandar) for p in particulas]

#Se normalizan los pesos para que sumen 1
suma_pesos = sum(pesos)
pesos_normalizados = [p / suma_pesos for p in pesos]
```

**Paso 4: Resampling**

Las partículas se muestrean de acuerdo a sus pesos. Las partículas con pesos más altos tienen una mayor probabilidad de ser seleccionadas, mientras que las partículas con pesos bajos tienen menos probabilidad. Este paso ayuda a enfocar el conjunto de partículas en las regiones más prometedoras del espacio de estado.

```
def resampling(particulas, pesos):
    nueva_generacion = []
    num_particulas = len(particulas)

    for _ in range(num_particulas):
        #Se selecciona una particula en base a sus pesos normalizados
        particula_seleccionada = random.choices(particulas, pesos_normalizados)[0]
        nueva_generacion.append(particula_seleccionada)

    return nueva_generacion

particulas = resampling(particulas, pesos_normalizados)
```

**Paso 5: Estimación del estado**

Se estima el estado del sistema combinando las partículas muestreadas. Esto se hace calculando la media ponderada de las posiciones de las partículas.

```
estado_estimado = sum(particulas) / len(particulas) #Media ponderada
print("Estado estimado:", estado_estimado)
```

Por último, aclarar que este algoritmo se engloba también como un método de Montecarlo, del tipo secuencial. En la siguiente sección se explican otras técnicas de muestro de Montecarlo.

### 3.2.2 Muestreo de MonteCarlo

Como ya se ha anticipado anteriormente con el filtro de partículas, existen unos métodos de muestreo llamados Métodos de Monte Carlo, que son técnicas estadísticas para estimar propiedades de sistemas complejos. Su funcionamiento, como se ha visto, se basa en generar muestras aleatorias de una distribución de probabilidad para aproximar la solución deseada.

Además del **filtro de partículas**, que se usa principalmente en el seguimiento de objetos por visión artificial y para la localización en robótica, existen otros métodos más o menos usados, que tienen sus aplicaciones concretas.
Por ejemplo, el **muestro estándar** se suele usar en matemáticas para calcular una integral numéricamente, o para estimar el valor esperado de una función, similar al método de **Integración por Monte Carlo**. En **muestro de importancia** se usa cuando las muestras que se generan directamente de la distribución son difíciles de extraer. El **bootstrap** se usa en métodos de remuestreo para la validación de modelos.
No obstante, el método de muestreo más usado en inteligencia artificial serían las **Cadenas de Markov Monte Carlo (MCMC).**

El proceso para el muestro de MCMC se basa en cadenas de Markov para explorar todo el espacio de muestro de manera correcta y eficiente, obteniendo muestras que se acercan poco a poco a la distribución objetivo. Se usa en multitud de problemáticas de la IA, como en el aprendizaje bayesiano y profundo (problemas de clasificación, regresión, clustering, redes neuronales bayesianas, etc), rastreo de objetos en entornos complejos, procesamiento del lenguaje natural (traducción automática) o el aprendizaje por refuerzo.
Para ello, existen una serie de técnicas y algoritmos que sirven para el muestro de distribuciones de probabilidad complejas y de alta dimensión. En este punto se explican dos algoritmos muy usados: el algoritmo Metropolis-Hastings y el muestreo de Gibbs.

#### Metropolis-Hastings

El algoritmo de Monte Carlo Metropolis-Hastings (MH) es una técnica de muestreo de Monte Carlo Markov Chain (MCMC) que se usa para generar muestras de una distribución de probabilidad deseada cuando no es fácil muestrear directamente de esa distribución. Se basa en la idea de construir una cadena de Markov que converja a la distribución de probabilidad objetivo como su distribución estacionaria. En otras palabras, explora progresivamente el espacio de muestra, generando muestras que se asemejan a la distribución objetivo. A continuación, se describen los pasos generales del algoritmo, haciendo uso de un pequeño ejemplo en Python.

```
1     import numpy as np
2     import matplotlib.pyplot as plt
3
4     #Funcion objetivo: Distribucion normal con media desconocida
5     def objetivo(x, media):
6         return np.exp(-(x - media) ** 2 / 2)
7
```

```
8   #Algoritmo de Metropolis-Hastings
9   def metropolis_hastings(iteraciones, media_propuesta,
                            desviacion_estandar_propuesta,
                            muestra_inicial):
10      muestra_actual = muestra_inicial #PASO 1: Se inicializa desde un
                            punto inicial en el espacio de
                            muestra
11      muestras = [muestra_actual]
12
13      for _ in range(iteraciones): #Se repite un numero suficiente de
                            veces para obtener muestras que se
                            distribuyan de acuerdo con la
                            distribucion objetivo
14          #PASO 2: Generar un candidato de la distribucion propuesta,
                            que puede ser simetrica o no. Buscar
                            una que sea facil de muestrear (por
                            ejemplo, la normal)
15          candidato = np.random.normal(muestra_actual,
                            desviacion_estandar_propuesta)
16
17          #PASO 3: Calcular la relacion de aceptacion (la razon entre la
                             densidad de probabilidad en la
                            propuesta y la densidad de
                            probabilidad en el valor actual)
18          ratio_aceptacion = objetivo(candidato, media_propuesta) /
                            objetivo(muestra_actual,
                            media_propuesta)
19
20          #PASO 4: Aceptar o rechazar el candidato. Si el cociente de
                            aceptacion es mayor o igual a 1, se
                            acepta la propuesta. Sino se acepta
                            la propuesta con una probabilidad
                            igual al cociente de aceptacion
21          if np.random.rand() < ratio_aceptacion:
22              muestra_actual = candidato
23
24          #PASO 5: Actualizacion del valor actual de la variable
                            aleatoria con el nuevo valor
                            propuesto. Si se rechaza la propuesta
                            , el valor actual se conserva
25          muestras.append(muestra_actual)
26
27      return muestras
28
29  #Parametros del problema
30  media_real = 5.0
31  desviacion_estandar_propuesta = 1.0
32  iteraciones = 10000
33
34  #Ejecutar el algoritmo de Metropolis-Hastings
35  muestra_inicial = 0.0  #Punto de inicio arbitrario
36  muestras = metropolis_hastings(iteraciones, media_real,
                            desviacion_estandar_propuesta,
                            muestra_inicial)
37
38  #Visualizar las muestras generadas
39  plt.hist(muestras, bins=50, density=True, alpha=0.6, color='b')
40  x = np.linspace(-5, 15, 100)
41  plt.plot(x, objetivo(x, media_real), 'r-', lw=2)
42  plt.xlabel('Muestra')
43  plt.ylabel('Densidad')
44  plt.title('Muestreo con Metropolis-Hastings')
45  plt.show()
```

**Muestreo de Gibbs**

El Muestreo de Gibbs es una técnica simple de muestreo en distribuciones conjuntas multidimensionales. Es especialmente útil en inferencia bayesiana y modelado probabilístico, y cuando es difícil muestrear directamente de la distribución conjunta, pero es más fácil muestrear de las distribuciones condicionales.

El objetivo principal de este algoritmo es la generación de muestras de una distribución conjunta cuando se conoce la distribución condicional de cada variable dada el valor de las demás variables.

Para una comprensión en profundidad, lo mejor es verlo en un ejemplo simple bidimensional en Python. No obstante, el interés radica cuando se tienen modelos probabilísticos más complejos, como las redes Bayesianas, donde puede haber muchas más variables y distribuciones condicionales.

```
import numpy as np

#Definir una distribucion conjunta (por ejemplo, una distribuci n
                                   gaussiana bidimensional)
media = np.array([0, 0])
covarianza = np.array([[1, 0.5], [0.5, 1]])

#Numero de muestras a generar
num_muestras = 1000

#PASO 1: Inicializar las variables de interes con valores iniciales
x = 0
y = 0
muestras = []

#Realizar el muestreo de Gibbs
for _ in range(num_muestras): #Despues de un numero suficiente de
                                   iteraciones, las muestras generadas
                                   convergen hacia la distribucion
                                   conjunta deseada
    #PASO 2: Muestrear x dado y (distribucion condicional de x)
    x = np.random.normal(media[0] + covarianza[0, 1] * (y - media[1])
                                   / covarianza[1, 1], np.sqrt(
                                   covarianza[0, 0] - covarianza[0, 1]**
                                   2 / covarianza[1, 1]))

    #PASO 3: Muestrear y dado x (distribucion condicional de y)
    y = np.random.normal(media[1] + covarianza[0, 1] * (x - media[0])
                                   / covarianza[0, 0], np.sqrt(
                                   covarianza[1, 1] - covarianza[0, 1]**
                                   2 / covarianza[0, 0]))

    muestras.append([x, y])
```

### 3.2.3 Propagación de creencias (predicción y estimación)

Junto a la Cadena de Markov Monte Carlo (MCMC), el algoritmo de propagación de creencias (BP) es uno de los más populares para realizar inferencia computacional.

Esta inferencia en sistemas estocásticos implica predecir eventos futuros o estimar estados no observados, algo realmente útil en problemas como el pronóstico del clima, donde las condiciones futuras son inciertas y deben estimarse a partir de datos históricos y modelos estocásticos.

Se trata de un algoritmo de paso de mensajes para realizar inferencia sobre modelos gráficos, que permite analizar grandes sistemas dividiéndolos en partes más pequeñas y asegurándose de que todas las soluciones sean coherentes entre sí. Un ejemplo sencillo sería el modelado de la propagación de una enfermedad viral por contacto cercano. Los científicos no tendrían que estudiar toda la red de contacto, sino analizar y saber qué pasará con las personas con las que entren en contacto, no todas en su la red, de forma recursiva (se analizan sus contactos y así sucesivamente).

Como se ha comentado, MCMC es un método probabilístico exacto que, sin embargo, a menudo es realmente lento. Por el contrario, la técnica de propagación de creencias es un método determinista rápido, empíricamente muy exitoso, pero que en situaciones de bucles puede reducir su eficacia.

Su funcionamiento es muy sencillo: se pretende realizar consultas para identificar como varia la distribución de probabilidad dada nueva información, actualizando la probabilidad de las variables en función de estas nuevas observaciones.
Esta nueva información (o evidencia) permite dos clasificaciones:

- *Hard evidence*: se especifican nuevos valores sobre variables de la red.
- *Soft evidence*: se especifica una nueva distribución de probabilidad para una o varias variables de la red.

Al ser un algoritmo muy estudiando, hay multitud de variaciones y clasificaciones en función de las necesidades concretas del problema a tratar, como algoritmos de inferencia exacta (cálculo de probabilidad exacta por repetición) o aproximada (estimación de valores aproximados de la probabilidad para un muestreo global). También se distinguen entre versiones del algoritmo en función del tipo de consulta, como consultas de probabilidad condicional (CPQ) o consultas de máximo a posteriori (MAP).

Finalmente, mencionar algunos de los algoritmos que se usan en la actualidad con más fecuencia: *LS Message Passing, Logic Sampling y Likelihood Weighting*.

Finalmente, a modo de esquema general (puesto que hay muchas versiones) se pueden definir unos pasos para la realización de este algoritmo:

1. Definición de la Red: por ejemplo, si es una red bayesiana con nodos que representan variables y aristas que representan las relaciones probabilísticas entre ellas, cada nodo tiene asociada una probabilidad condicional que describe cómo la variable en ese nodo depende de las variables en sus nodos padres.
2. Inicialización: todas las variables de la red se sitúan en un estado inicial, como por ejemplo, una suposición o una estimación inicial de las probabilidades marginales de cada variable.

3. Propagación hacia Adelante: se propaga la información desde los nodos de entrada hacia los nodos de salida. En cada uno de ellos se calcula una creencia provisional (una estimación) de la probabilidad marginal de la variable en ese nodo basada en la información de sus nodos padres.
4. Actualización de Creencias: se actualizan las creencias en cada nodo partiendo de la información de sus nodos vecinos, mediante una combinación de las creencias de estos y las probabilidades condicionales asociadas con esas aristas.
5. Propagación hacia Atrás: actualizadas las creencias en los nodos de salida, el algoritmo empieza a propagar información desde los nodos de salida hacia los nodos de entrada, fluyendo la información en ambas direcciones a través de la red.
6. Iteración: se repiten los pasos 3, 4 y 5 varias veces hasta que las creencias convergen o hasta que se alcance un cierto criterio de convergencia. En cada iteración, las creencias se refinan y se vuelven más precisas.
7. Resultados Finales: se obtienen estimaciones más precisas de las probabilidades marginales de todas las variables en la red, que reflejan cómo las variables están relacionadas en la red y cómo se influyen mutuamente.

### 3.2.4 Aprendizaje estocástico

En el aprendizaje automático, la inferencia en sistemas estocásticos también se aplica al entrenar modelos. Los algoritmos de aprendizaje pueden ser estocásticos, lo que significa que incorporan aleatoriedad en el proceso de actualización de los parámetros del modelo. Gracias a ello, se dota de flexibilidad y adaptación al modelo frente a situaciones donde la incertidumbre y la variabilidad son inherentes en los datos o en el proceso de modelado.

Se usa en una amplia variedad de aplicaciones en estadísticas, aprendizaje automático y más allá para abordar problemas del mundo real, como podría ser el descenso de gradiente estocástico (Stochastic Gradient Descent, SGD), que es una técnica de optimización ampliamente utilizada, con propiedades de suavizado, que utiliza muestras aleatorias de datos para ajustar los parámetros del modelo. Su funcionamiento se basa en calcular el gradiente en un subconjunto aleatorio de los datos en cada iteración, introduciendo variabilidad en el proceso de optimización y ayudando a converger más rápido y evitar mínimos locales.
Un ejemplo de código en Python del algoritmo de SGD puede ser el siguiente:

```
1    import numpy as np
2
3    #Se generan los datos de ejemplo donde X es la variable independiente
                                  e Y es la variable dependiente con un
                                   poco de ruido aleatorio
4    np.random.seed(0)
5    x = 2 * np.random.rand(100, 1)
6    y = 4 + 3 * x + np.random.randn(100, 1)
7
8    #Inicializa los parametros del modelo
```

```
    theta = np.random.randn(2, 1)  #Variable del modelo de manera
                                       aleatoria
    tasa_aprendizaje = 0.01  #Tasa de aprendizaje (controla la velocidad
                                       de convergencia del algoritmo)

    #Numero de  iteraciones (epocas)
    n_epocas = 1000

    #Implementacion del Descenso Estocastico de Gradiente (SGD)
    for epoch in range(n_epocas):
        for i in range(len(X)):
            indice_aleatorio = np.random.randint(len(X))  #Selecciona un
                                       punto de datos aleatorio
            xi = X[indice_aleatorio:indice_aleatorio+1]
            yi = y[indice_aleatorio:indice_aleatorio+1]
            gradiente = xi.T.dot(xi.dot(theta) - yi)  #Calcula el
                                       gradiente
            theta = theta - tasa_aprendizaje * gradiente  #Actualiza los
                                       parametros

    #Los parametros finales del modelo son almacenados en 'theta'
    print("Parametro final del modelo:")
    print(theta)
```

Existen múltiples versiones de esta técnica, como pueden ser AdaGrad o Adam, que se adaptan a necesidades específicas de la aplicación que se desarrolle.

## 3.3 Probabilidad

En el razonamiento computacional bajo incertidumbre, la probabilidad es vital para estimar esa incertidumbre. La teoría de la probabilidad se estudia y es conocida en gran parte, por lo que en este tema se comienza con un resumen/recordatorio de los conceptos clave de la probabilidad para pasar a aplicarlo en modelos de inteligencia artificial.

### 3.3.1 Conceptos básicos de la probabilidad

**Definiciones**

**Probabilidad**: La medida de la posibilidad de que ocurra un evento o un conjunto de eventos, expresada numéricamente como una relación entre el número de casos favorables y el número total de casos posibles.
**Fuente:** Oxford Languages

Esta definición de probabilidad refleja el concepto fundamental de probabilidad como una medida cuantitativa de la incertidumbre o qué grado de certeza se tiene de que ocurra un evento concreto en relación con el conjunto de todos los eventos posibles. La probabilidad se expresa normalmente como un número real comprendido entre 0 y 1, donde 0 indica que un evento es imposible de ocurrir y 1 indica que es seguro que ocurrirá. Todos aquellos valores intermedios representan grados de incertidumbre, expresados en porcentaje para una mayor claridad.
Por ejemplo, si se juega al parchis se necesita sacar un 5 con un dado

para poder salir de casa. Como hay 6 eventos posibles (del 1 al 6), la probabilidad es de 1/6, por lo que el grado de certidumbre sería de en torno al 16.6%.

A continuación, se definen otros conceptos básicos previos a entrar en más detalle.

**Definiciones**

**Población**: conjunto de individuos o elementos que son objeto de estudio.
**Muestra**: grupo reducido de la población que se usa en el estudio. Sus elementos deben ser escogidos cuidadosamente para que representen a toda la población.
**Espacio muestral**: conjunto de todos los posibles resultados individuales o eventos que pueden ocurrir en un proceso estocástico.
**Evento**: subconjunto específico del espacio muestral. Es un resultado o conjunto de resultados posibles dentro de un experimento estocástico.
**Variable**: Característica de la población que interesa en el estudio.
**Fuente:** Oxford Languages

Por ejemplo, si se desea predecir con IA en una fase temprana del Alzheimer a personas menores de 65 años, se podría decir que:

- La población es el conjunto de personas menores de 65 años. Por ejemplo, si el número de personas fuese 1000 millones, ese sería el tamaño de la población.
- Se selecciona un conjunto reducido de personas en estudio que cumplan con los requisitos (muestra), por ejemplo 1000 (tamaño muestral).
- Ha dos posibles resultados en este estudio: tiene o no Alzheimer, luego ese sería el espacio muestral.
- Si se centran en las personas que tienen Alzheimer, ese sería un evento.
- Alzheimer sería una variable cualitativa nominal, cuyos valores carecen de orden y puede tomar los valores "si" o "no".

Las variables se pueden clasificar en dos tipos (cuantitativas o cualitativas) y en dos subtipos cada uno de ellos.
Una variable cualitativa es aquella que representa categorías, clases o etiquetas en lugar de valores numéricos (no se pueden cuantificar de esta manera). Por ejemplo, la fecha de nacimiento. Tienen dos subtipos: nominal y ordinal. Las norminales se refieren a categorías sin ningún orden intrínseco (por ejemplo, los colores), y las ordinales tienen un orden o jerarquía natural (por ejemplo, la edad).
Dentro de las variables cuantitativas, representan cantidades medibles y se expresan con valores numéricos. Es decir, pueden ser objeto de operaciones matemáticas. Hay dos subtipos: discretas y continuas. La diferencia principal radica en que las discretas representan valores numéricos que son contables y generalmente enteros (número de hermanos de una persona), y las continuas expresan valores numéricos que pueden tomar cualquier valor dentro de un rango determinado (por ejemplo, la altura de una persona).

### Tablas y medidas estadísticas

En la estadística descriptiva dispone de tablas estadísticas y medidas que son herramientas útiles en estadística para organizar, resumir y presentar datos de manera ordenada y comprensible. En este apartado se incluyen las definiciones, fórmulas y un ejemplo de cada una de ellas.

**Definiciones**

**Frecuencia absoluta ($f_i$)**: número de veces que se repite (frecuencia) cada categoría o valor en una variable.
**Frecuencia relativa ($f_r$)**: porcentaje que representa cada categoría con respecto al total. Se calcula mediante el cociente entre la frecuencia absoluta y el tamaño muestral: $f_r = \frac{f_i}{n}$.
**Frecuencia acumulada**: representa la acumulación de frecuencias (conteo de observaciones) en una distribución, a medida que se avanza a través de los valores de una variable. Hay dos tipos: la frecuencia absoluta acumulada ($F_i = \sum_{k=1}^{i} f_k$) y la frecuencia relativa acumulada ($F_i^r = \sum_{k=1}^{i} f_k^r$).
**Fuente:** Oxford Languages

Un ejemplo muy sencillo sería analizar los minutos que pasa parado un coche en los semáforos. Se toma como muestra los 7 semáforos por los que para un coche en un trayecto de casa al trabajo en un día concreto (muestra). Los datos de la muestra son: 3, 2, 2, 3, 1, 3 y 3 minutos. La tabla de frecuencias quedaría de la siguiente forma:

**Table 3.1:** Tabla de frecuencias.

| Tiempo parado | $f_i$ | $f_r$ | $F_i$ | $F_r$ |
|---|---|---|---|---|
| 1 minuto | 1 | 0.14 | 1 | 0.14 |
| 2 minutos | 2 | 0.29 | 3 | 0.43 |
| 3 minutos | 4 | 0.57 | 7 | 1 |

La tabla de frecuencias también es posible hacerlo con variables cuantitativas continuas, tomando como amplitud del intervalo la diferencia entre los límites del intervalo (por ejemplo, una parada de 1 a 5 minutos tendría una amplitud de 4 minutos), y como marca del intervalo el punto medio del intervalo representado (en el mismo ejemplo, la marca sería 3 minutos).

Por otra parte, para el análisis de las características de una distribución de frecuencias, existen diferentes medidas que permiten sintetizar toda la información de estas tablas, clasificadas en medidas de posición, dispersión y forma. A continuación, se analizan las más usadas de cada tipo.

**Medidas de posición**

**Media aritmética:** representa el valor central o promedio de un conjunto de datos numéricos (punto de referencia para representar el "centro" de los datos). Se calcula así: $\overline{x} = \frac{\sum_{i=1}^{n} x_i}{n}$
**Media ponderada:** es similar a la media aritmética, pero está se usa

cuando no todas las categorías ($w$) tienen el mismo peso. Se calcula como $\overline{x_p} = \frac{\sum_{i=1}^{k} w_i x_i}{\sum_{i=1}^{k} w_i}$

**Media geométrica:**: calculo del valor central en un conjunto de datos cuando los datos representan tasas de crecimiento, razones o proporciones. Cálculo: $\overline{x_g} = \sqrt[n]{\prod_{i=1}^{n} x_i}$

**Media armónica**: cálculo del valor central en un conjunto de datos cuando los datos representan tasas, velocidades o relaciones recíprocas. Cálculo: $\overline{x_h} = \frac{n}{\sum_{i=1}^{n} \frac{1}{x_i}}$

**Mediana**: es el valor central en un conjunto de datos ordenados. No se ve afectada por valores extremos (atípicos), a diferencia de la media aritmética.

**Moda ($f_i$)**: se trata de la categoría que más se repite en la muestra. Puede haber más de una moda.

**Cuantiles**: dividen un conjunto de datos ordenados en partes iguales o en porciones específicas para identificar valores que se encuentran en posiciones particulares dentro de una distribución de datos. Muy útiles para comprender la dispersión y la distribución de los datos. Los más conocidos son los percentiles y los cuartiles.

En el ejemplo anterior (tiempo de parada del coche), la media aritmética sería 2.42 minutos, la moda sería la parada de 3 minutos (4 veces), y la mediana sería 3 minutos.

**Medidas de dispersión**

**Rango:** es la diferencia entre los valores máximo y mínimo de la muestra, y sirve para tener una idea general de la dispersión de los datos. Es demasiado sensible a valores atípicos. Se calcula restando el valor mínimo y máximo del conjunto de valores.

**Varianza:** mide cómo de dispersos están los valores de un conjunto de datos en relación con su media. Se calcula como $\sigma^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i - \overline{x})^2$

**Desviación típica:** tiene el mismo objetivo que la varianza (conocer la variabilidad en los datos), pero es una medida más interpretable y fácil de comprender, ya que está en las mismas unidades que los datos originales. Se calcula como $\sigma = \sqrt{\sigma^2}$

Si se tienen que dos coches han obtenido la misma media de tiempo de parada, pero el primero tiene una desviación típica de 1 minuto y el segundo tiene 3 minutos, ¿cuál media es más fiable/representativa? En estos casos, cuanto menor sea el valor de la desviación típica, menos dispersos estarán los datos respecto a su media, y más real y representativa será.

Hay más medidas de dispersión, como el rango intercuartílico, semiintercuartílico o desviación mediana, entre otras. Sin embargo, las medidas definidas son las más representativas y usadas.

**Medidas de forma**

**Coeficiente de asimetría de Pearson:** sirve para cuantificar la asimetría (o sesgo) en la distribución de un conjunto de datos. Indica si los

datos están sesgados hacia la izquierda (valor negativo), hacia la derecha (valor positivo) o si la distribución es aproximadamente simétrica (cerca de cero). Es sensible a valores atípicos. Su fórmula es $A_S = \frac{3(\overline{x}-mediana)}{\sigma}$
**Coeficiente de asimetría de Fisher:** sirve para estudiar si la distribución está centrada en torno a su media, similar a Pearson. La diferencia principal la utilidad cuando se busca una caracterización detallada de la asimetría de la distribución. Se calcula así: $A_F = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i-\overline{x})^3}{\sigma^3}$
**Coeficiente de curtosis:** mide la concentración de los datos en relación con la media y proporciona información sobre la presencia de valores atípicos y la forma de las colas de la distribución. Si el valor es negativo es una distribución platicúrtica, leptocúrtica si es positivo, y mesocúrtico si es igual a cero. Se calcula así: $K = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i-\overline{x})^4}{\sigma^4}$

Finalmente, destacar que dentro de cada campo existen multitud de medidas que son interesantes estudiar según el caso, como podría ser en las ciencias sociales el Índice de Gini.

### Teoría de conjuntos e independencia de eventos

A lo largo de los capítulos, se han definido y explicado las diferencias entre fenómenos determinísticos (resultados conocidos con exactitud) y aleatorios (sin exactitud, con incertidumbre).
Esa incertidumbre se puede analizar mediante probabilidad, siendo vital la definición previa de conceptos básicos de teoría de conjuntos y operaciones con ellas.
Definido el espacio muestral de un experimento aleatorio (conjunto de todos los posibles resultados), aclarar que hay dos tipos de espacios muestrales: finito (valores discretos) e infinito no numerable (valores continuos).
También se ha definido un evento (o suceso), que no es más que una coleccion de posibles resultados de un experimento (un subconjunto del espacio muestral).
Dicho esto, se definen las siguientes cuatro operaciones básicas con eventos/sucesos/conjuntos:

**Unión.** Los puntos que están en los conjuntos A o B. Se denota como $A \cup B$

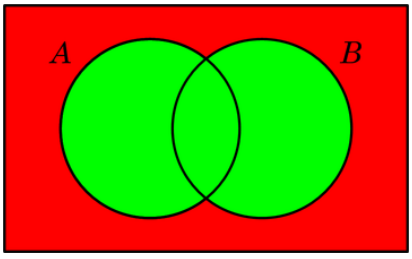


**Figure 3.1:** Operación Unión. La parte verde coloreada sería la unión de los conjuntos A y B.

**Intersección.** Los puntos que están en los conjuntos A y B. Se denota como $A \cap B$

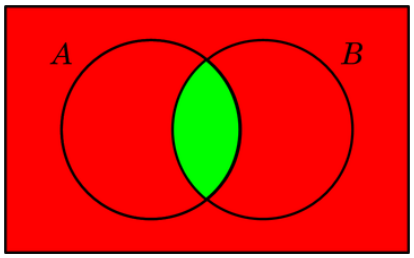


**Figure 3.2:** Operación Intersección. La parte verde coloreada sería la intersección de los conjuntos A y B.

**Complementario.** Aquellos puntos que no están en el conjunto A. Se denota como $A^C$

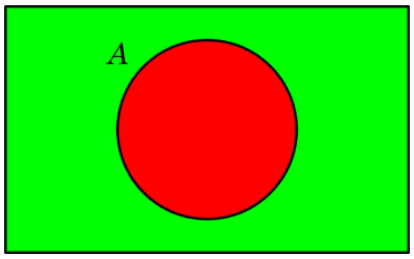


**Figure 3.3:** Operación Complementario. La parte verde coloreada sería el complementario de A.

**Diferencia.** Aquellos puntos que están en el conjunto A y no en el conjunto B. Se denota como $A - B$

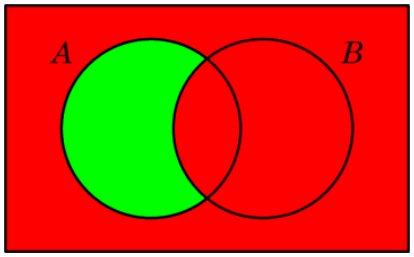


**Figure 3.4:** Operación Diferencia. La parte verde coloreada sería la diferencia de los conjuntos A menos B.

**Independencia** o conjuntos disjuntos es cuando la unión de los conjuntos A y B es el conjunto vacío: $A \cap B = \emptyset$.

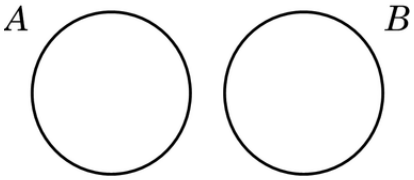


**Figure 3.5:** Independencia. Representación gráfica de dos conjuntos independientes A y B.

**Regla del producto y regla de la suma.**

En la resolución de problemas de combinatoria y probabilidad hay dos reglas fundamentales que se deben conocer: regla del producto y regla de la suma.
El objetivo principal de ambas reglas es contar y enumerar objetos o eventos en conjuntos finitos.

**Regla del Producto.** Se usa cuando se quiere determinar el número de formas en que dos o más eventos o elecciones independientes pueden ocurrir en secuencia. Si un evento A puede ocurrir de $x$ maneras diferentes y, de manera independiente, otro evento B puede ocurrir de $y$ maneras

diferentes, entonces el número total de formas en que ambos eventos pueden ocurrir en secuencia es $x * y$.
Por ejemplo, si se quiere elegir un menú con 4 entrantes diferentes y 5 platos principales diferentes, entonces hay 4*5 = 10 menús diferentes que se pueden elegir.

**Regla de la Suma.** Se usa cuando se quiere conocer el número total de formas en que un evento o elección puede ocurrir, y este evento se puede lograr de diferentes maneras mutuamente excluyentes.
Si un evento A se puede conseguir de $x$ maneras diferentes y, de manera independiente, y otro evento B se puede de $y$ maneras diferentes (donde las dos opciones son mutuamente excluyentes, es decir, no pueden ocurrir simultáneamente), entonces el número total de formas en que uno de los dos eventos puede ocurrir es x+y.
Por ejemplo, si para ir a trabajar se puede ir en metro o bus, y hay 4 buses diferentes y 5 metros diferentes que van al trabajo, entonces hay 4 + 5 = 9 formas diferentes de llegar al trabajo.

Estas reglas son muy usadas en diversos campos, como estadística, teoría de la probabilidad o la teoría de grafos, entre otros.

**Distribuciones de Probabilidad**

Las distribuciones de probabilidad representan la forma en que se distribuyen las probabilidades de ocurrencia de distintos resultados en un experimento aleatorio o en un conjunto de datos. Dicho de otra forma, una distribución de probabilidad describe todas las posibles valores que puede tomar una variable aleatoria y la probabilidad asociada a cada uno de esos valores.
Se usan para modelar y comprender una variedad de fenómenos del mundo real, y la elección de la distribución de probabilidad adecuada depende de la naturaleza del fenómeno que se está estudiando y de los datos disponibles.
Hay dos tipos principales: discretas y continuas. A continuación se explican las más conocidas de cada tipo.

***Distribución de probabilidad discreta***. En este tipo de distribuciones, las variables aleatorias toman valores discretos o separados.

| **Distribución de Bernoulli** |
| --- |
| **Definición:** sirve para modelar un experimento aleatorio con dos posibles resultados: éxito y fracaso (o sí y no, 1 y 0, etc.). Se denota comúnmente como *Bernoulli(p)*, donde "p" es la probabilidad de éxito y "q" es la probabilidad de fracaso (q = 1 - p). Esta distribución describe un solo ensayo o prueba de un evento binario.<br>**Notación:** $P[X = x] = p^x(1-p)^{(1-x)}$, siendo x=0 ó 1 |

| **Distribución Binomial** |
| --- |
| **Definición:** sirve para modelar el número de éxitos (resultados posi- |

tivos) en una serie de ensayos independientes e idénticamente distribuidos, que siguen una distribución de Bernoulli. La distribución binomial se denota como *Binomial(n, p)*, donde "n" es el número de ensayos o pruebas y "p" es la probabilidad de éxito en cada ensayo.
**Notación:** $P[X = k] = \binom{n}{k} p^k (1-p)^{(n-k)}$, siendo k el número de éxitos en n ensayos

**Distribución de Poisson**

**Definición:** se utiliza para modelar el número de eventos raros o inusuales que ocurren en un intervalo de tiempo o espacio fijo. Los eventos ocurren de manera independiente en un intervalo continuo. Si hay muchos ensayos con probabilidad de éxito bajo en cada ensayo, se puede usar la distribución de Poisson para aproximar la distribución binomial.
**Notación:** se calcula como $P[X = k] = \frac{e^{-\lambda}\lambda^k}{k!}$

Función de masa de probabilidad (PMF). Esta función (Probability Mass Function en inglés) se usa para describir la distribución de probabilidad discreta de una variable aleatoria. Es decir, la PMF proporciona la probabilidad de que una variable aleatoria discreta tome un valor concreto. Por ejemplo, si la probabilidad de lanzar un dado tiene 6 resultados posibles, la probabilidad sera de 1/6, siendo la suma de todas estas probabilidades 1.

La PMF permite comprender cómo se distribuyen las probabilidades entre los diferentes valores posibles de la variable, y es útil para calcular probabilidades específicas como la probabilidad de que la variable tome un valor particular o la probabilidad de que caiga en un rango específico. Por lo tanto, se convierte en una herramienta fundamental para modelar la incertidumbre en diferentes problemas.

***Distribución de probabilidad continua***. En este tipo de distribuciones, las variables aleatorias pueden tomar cualquier valor en un rango continuo.

**Distribución Normal**

**Definición:** también conocida como distribución gaussiana, se caracteriza por tener una forma de campana simétrica y se usa para modelar datos continuos y fenómenos donde la variación es continua y suave. Está completamente caracterizada por su media y su desviación estándar, donde se describe una distribución de valores en torno a una media.
**Notación:** la fórmula es $P[X = x]\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$, pero se suele utilizar tipificaciones $z = \frac{x-\mu}{\sigma}$

**Distribución Exponencial**

**Definición:** sirve para modelar el tiempo que debe transcurrir hasta

que ocurra un evento raro y aleatorio. En otras palabras, se usa para modelar el tiempo entre eventos sucesivos en un proceso de Poisson.
**Notación:** $P[X = k] = 1 - e^{-\lambda x}$

Función de densidad de probabilidad (PDF). Se trata de una función matemática que se utiliza en teoría de la probabilidad para describir la distribución de probabilidad de una variable aleatoria continua. La PDF es esencial para comprender y analizar cómo se distribuyen los valores de una variable aleatoria continua en un rango específico de valores. Tiene una formulación general, que se adapta a cada una de las distribuciones continuas.

**Remark 3.3.1** Función de densidad de probabilidad.

$$f(x) \geq 0$$

$$\int_{-\infty}^{\infty} f(x)\, dx = 1$$

donde $x$ es la variable aleatoria continua.

Por ejemplo, la función de densidad de probabilidad de la distribución exponencial sería $f(x) = \lambda e^{-\lambda x}$.

### 3.3.2 Modelos probabilísticos gráficos: teorema de Bayes y redes bayesianas

Estos modelos probabilísticos se tratan de una una herramienta usada en la IA para representar y analizar las relaciones probabilísticas entre un conjunto de variables aleatorias. Estos modelos utilizan grafos para representar las dependencias condicionales entre variables y se dividen principalmente en dos tipos: las Redes Bayesianas y los Modelos de Markov. Anteriormente se han comentado algunos modelos de Markov, como Belief Propagation (BP), y en esta sección se va a profundizar un poco más en las redes bayesianas.

#### Teorema de Bayes

Este teorema describe cómo actualizar nuestras creencias sobre un evento cuando se obtiene nueva evidencia relevante, siendo clave en la toma de decisiones bajo incertidumbre. Antes de nada, se muestra la formulación del teorema de Bayes:

**Remark 3.3.2** Teorema de Bayes

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}, \text{ donde}$$

P(A) es la probabilidad a priori de que ocurra el evento A.

- P(B) es la probabilidad a priori de que ocurra el evento B.
- P(A | B) es la probabilidad condicional de que ocurra el evento A dado que ha ocurrido previamente el evento B.
- P(B | A) es la probabilidad condicional de que ocurra el evento B dado que ha ocurrido previamente el evento A.

¿Qué quiere decir esto? Se puede simplificar diciendo que permite actualizar lo que ya creíamos inicialmente (representado por $P(A)$) en función de nueva evidencia (representada por $P(B|A)$) para calcular la probabilidad revisada de que ocurra A (representada por $P(A|B)$). Un ejemplo sería:
La probabilidad de que un periodista publique una noticia falsa por error es del 20% (P(A) = 0.2). El periodista comprueba sus fuentes y que la noticia sea verídica el 90% de las veces (P(B) = 0.9). La probabilidad de que compruebe sus fuentes condicionado a que ha publicado una noticia falsa previamente es del 40% (P(B | A)) ¿Cuál es la probabilidad de que el periodista publique una noticia falsa habiendo comprobado sus fuentes previamente (P(A | B))? Sería $\frac{0.4*0.2}{0.9} = 8.8\%$

En la IA se usa mucho en el aprendizaje automático (particularmente en el contexto de redes bayesianas), como se verá en las siguientes secciones.

**Aprendizaje de parámetros y estructura en redes bayesianas**

Las redes bayesianas son modelos probabilísticos gráficos que representan relaciones de dependencia condicionales entre variables aleatorias utilizando un grafo dirigido acíclico (DAG).
Un ejemplo sencillo de red bayesiana sería la búsqueda de relaciones entre los síntomas que presente un paciente en una consulta médica y las posibles enfermedades que pueda tener. Es decir, dados los síntomas (conocido) se infiere la probabilidad de que padezca ciertas enfermedades.

En un DAG, cada nodo en el grafo representa una variable y las aristas indican las relaciones probabilísticas. Entrando en mayor detalle, las redes bayesianas son grafos acíclicos dirigidos donde los nodos o vertices representan variables aleatorias desde un punto de vista condicional de Bayes: pueden ser cantidades que pueden ser observadas, variables latentes, parámetros no conocidos o hipótesis. Y las aristas representan dependencias condicionales, donde los nodos que no están conectados (aislados) representan variables independientes. Cada nodo está vinculado a una fórmula o regla que se usa para calcular la probabilidad de que ese nodo en particular tenga ciertos valores, considerando lo que se sabe acerca de las variables conectadas a él. Estas fórmulas toman como entrada información sobre las variables que influyen en el nodo en cuestión (sus 'variables padres') y, en base a esa información, generan un número que representa la probabilidad de que la variable representada por el nodo tenga un valor específico.
Por último, si la red es no dirigida y/o cíclica, se suelen usar redes de Markov.

Ya se han visto algunos algoritmos de inferencia que se pueden aplicar a redes bayesianas, como el muestreo de Gibbs o la propagación de creencias. A continuación, se analizan las tres tareas principales de inferencia para redes bayesianas:

**Deducción de variables no observadas**. Es una tarea que sirve para responder a las consultas de probabilidad acerca de la red. En otras palabras, se busca estimar o predecir los valores de variables que no son directamente observadas, pero que están relacionadas con las variables observadas en la red. Esto implica utilizar la información disponible en la red para obtener estimaciones razonables de las variables no observadas.

Por ejemplo, la red se puede utilizar para averiguar el conocimiento actualizado del estado de un subconjunto de variables cuando otras variables (que serían las llamadas "variables de evidencia") se observan. Este proceso de cálculo de la distribución posterior de las variables dada la evidencia es lo que se llama inferencia probabilística. Se usa mucho en aplicaciones de detección, para elegir los valores de la variable de un subconjunto que minimizan alguna función de pérdida esperada (por ejemplo, la probabilidad de error de decisión).

Respecto a las técnicas, hay dos clasificaciones: inferencias exactas y aproximadas:

- Exactas:
  - Eliminación de variables: son técnicas para simplificar y mejorar la eficiencia de estas redes sin perder la capacidad de representación del problema. El proceso es eliminar variables no observadas mediante integración o suma. Hay muchas técnicas conocidas, como eliminación de variables irrelevantes, redundantes, por causalidad, o por poda, entre otras.
  - Propagación en un árbol clique (clique tree propagation): se basa en la transformación de una red bayesiana en un árbol de cliques, que es una estructura de datos que simplifica los cálculos de inferencia, almacenando la información/consultas probabilísticas en una "caché" para consultarlo más de una vez y propagar rápidamente las nuevas evidencias.
  - Condicionamiento recursivo (recursive conditioning): sirve para modelar y aprender relaciones causales en sistemas complejos que evolucionan con el tiempo o en situaciones secuenciales. Se usa en sistemas biológicos, predicción en series temporales, análisis de datos financieros y cualquier otro contexto en el que las relaciones causales sean importantes y cambiantes con el tiempo.
  - Búsqueda AND/OR: son técnicas que aportan equilibrio espacio-temporal eliminando eficientemente variables cuando se usa el suficiente espacio. Se usan en redes Bayesianas con estructuras que involucran nodos AND y nodos OR, buscando la probabilidad de evidencia o calcular probabilidades marginales en estas redes, teniendo en cuenta las estructuras de decisión que contienen dichos nodos.
- Aproximadas:
  - Muestreo de importancia (importance sampling): se usa para estimar la distribución posterior de las variables ocultas dadas las observaciones disponibles sin tener que realizar un muestreo directo de esa distribución. La elección adecuada de la distribución de importancia es clave para obtener

estimaciones precisas y confiables.

- Eliminación mini-bucket: sirve para aproximarse a la distribución posterior de las variables latentes dadas las observaciones disponibles, mediante pequeños grafos llamandos mini-buckets donde se calcula un potencial que representa la distribución conjunta de las variables en ese conjunto, teniendo en cuenta las observaciones y las relaciones probabilísticas de la red bayesiana original. Posteriormente se agrupan y se eliminan variables. Con ello, se reduce la complejidad computacional al dividir el problema en subproblemas más pequeños y manejables (es importante tener en cuenta que esta técnica se basa en aproximaciones y puede no ser exacta en todos los casos).
- Loopy Belief Propagation (LBP): se usa para abordar problemas de inferencia que involucran variables no observadas o latentes, y en redes con ciclos. Se trata de una técnica iterativa que utiliza mensajes para propagar creencias y estimar la distribución posterior en la red bayesiana.
- Generalized Belief Propagation (GBP): es una extensión de BP diseñada para mejorar la convergencia en comparación con BP en estructuras complejas, incluidos grafos con ciclos.
- Métodos variacionales: sos métodos variacionales que se centran en encontrar una distribución aproximada que sea más fácil de manejar que la distribución posterior exacta, usando optimizaciones como la Divergencia de Kullback-Leibler.
- MCMC: ya se ha comentado en capítulos anteriores.

**Aprendizaje de parámetros**.

Para especificar por completo la red bayesiana, y representar la distribución de probabilidad conjunta correctamente, se necesita especificar para cada nodo la distribución de probabilidad condicionada dados sus padres. Esta distribución puede tener múltiples formas, como las gausianas. Cuando se tienen solo restricciones sobre una distribución conocida, se usa el principio de máxima entropía. Si las distribuciones condicionales tienen parámetros desconocidos y hay que estimarlos a partir de los datos, se usa el enfoque de máxima probabilidad.
Y dependiendo de diferentes casuísticas, existen diferentes técnicas para aprender estos parámetros en función de las restricciones que se presentan. A continuación, se explican brevemente dos técnicas muy usadas para el aprendizaje de parámetros:

- Aprendizaje de Parámetros por Máxima Verosimilitud: rste enfoque busca estimar las tablas de probabilidad condicional (TPC) en una red bayesiana de manera que maximice la probabilidad de los datos observados. Es eficaz cuando se tienen suficientes datos.
- Algoritmo de expectación-maximización (expectation-maximization): alterna los valores esperados calculados de las variables condicionales no observadas y observadas, maximizando con la probabilidad total, suponiendo que los valores calculados previamente son correctos.

**Aprendizaje de estructuras**.

Es la tarea más sencilla de las tres. Es cuando una red bayesiana se especifica por un experto para inferencia. Dicha tarea determina la topología o estructura de una red bayesiana, decidiendo qué variables están conectadas directamente entre sí, es decir, qué variables influyen en otras en el grafo de la red. Se busca la estructura más adecuada que explique las relaciones de dependencia condicional entre las variables observadas.
Para llevar a cabo esta tarea, hay diferentes técnicas que son interesantes, explicando brevemente las más usadas a continuación:

- Aprendizaje de estructura por algoritmos de búsqueda: se basan en aprender la estructura de una red bayesiana a partir de datos observados. Estos métodos exploran diferentes estructuras posibles y seleccionan la que mejor se ajusta a los datos. Los más usados son: algoritmo de K2, el algoritmo de Hill-Climbing y el algoritmo de Búsqueda de pasos orientados.
- Aprendizaje bayesiano de estructura: se basan en enfoques bayesianos para aprender la estructura de una red bayesiana, incorporando información a priori y calculando la probabilidad posterior de diferentes estructuras.
- Aprendizaje activo: en lugar de depender únicamente de datos pasivos, el aprendizaje activo implica seleccionar de manera inteligente las observaciones o experimentos adicionales que proporcionarán la información más valiosa para mejorar el modelo de la red bayesiana.
- Aprendizaje de estructura incremental: se agregan nodos y aristas gradualmente a una red bayesiana existente a medida que se obtienen nuevos datos, lo que es útil en escenarios de crecimiento continuo de datos

**Actualización de creencias a través de la inferencia bayesiana**

Se trata de un enfoque poderoso para manejar la incertidumbre, que permite ajustar nuestras creencias o conocimientos sobre un evento o situación a medida que se obtienen nuevos datos o evidencias. Este proceso se basa en el teorema de Bayes y se utiliza para calcular la distribución posterior de una variable o conjunto de variables dadas las observaciones y una distribución previa.

A continuación, se describe cómo funciona la actualización de creencias a través de la inferencia bayesiana en una serie de pasos generales:

**Paso 1: Distribución Previa (Prior)**

Antes de observar nuevos datos, se establece una distribución previa que refleja nuestras creencias o conocimientos iniciales sobre la variable o variables de interés. La distribución previa representa la incertidumbre inicial acerca de los parámetros o valores de la variable y se elige de acuerdo con la información disponible y las suposiciones previas.

**Paso 2: Recopilación de Datos (Observaciones)**

Se obtienen nuevos datos o evidencias que proporcionan información adicional sobre la variable o variables de interés. Estos datos pueden ser observaciones experimentales, resultados de mediciones, encuestas, etc.

**Paso 3: Teorema de Bayes**

El teorema de Bayes permite actualizar la distribución previa a la distribución posterior teniendo en cuenta los datos observados. Usaremos la fórmula $P(\theta|data) = \frac{P(data|\theta)*P(\theta)}{P(data)}$, donde:

- $P(\theta|data)$ es la distribución posterior, la distribución actualizada después de observar los datos.
- $P(data|\theta)$ es la función de verosimilitud, que representa la probabilidad de observar los datos dado un valor particular de los parámetros ($\theta$).
- $P(\theta)$ es la distribución previa, que representa nuestras creencias iniciales sobre los parámetros.
- $P(data)$ es la probabilidad marginal de los datos, que actúa como una constante de normalización.

**Paso 4: Cálculo de la Distribución Posterior**

Utilizando el teorema de Bayes, se calcula la distribución posterior que refleja nuestras creencias actualizadas después de observar los datos. La distribución posterior proporciona información sobre la incertidumbre actualizada y es la base para tomar decisiones o hacer inferencias adicionales.

**Paso 5: Iteración y Actualización Continua**

A medida que se obtienen nuevos datos, el proceso de actualización de creencias se repite iterativamente. Cada vez que se observan nuevos datos, la distribución posterior se actualiza para reflejar las creencias más actuales y se utiliza como distribución previa para la próxima iteración.

Supongamos un ejemplo sencillo: se establece una distribución previa que refleje nuestras creencias iniciales sobre la probabilidad de obtener caras en una moneda. Supongamos que creemos que la probabilidad es igualmente probable para caras y cruces, por lo que usamos una distribución uniforme como prior. De 10 monedas, se obtienen 7 caras y 2 cruces:

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3  from scipy.stats import binom
4
5  #Paso 1: Distribucion Previa (Prior)
6  prior = np.array([0.5, 0.5])  #Probabilidad uniforme para caras y
                                    cruces
7
8  #Visualizacion de la distribucion previa
```

```
    plt.bar(['Cara', 'Cruz'], prior)
    plt.title('Distribucion Previa (Prior)')
    plt.show()

    #Paso 2: Datos Observados (Observaciones)
    data_caras = 7
    data_cruces = 3

    #Pasos 3 y 4
    likelihood = binom.pmf(data_caras, data_caras + data_cruces, 0.5)  #
                                        Funcion de verosimilitud
    posterior = likelihood * prior
    posterior /= posterior.sum()

    #Visualizacion de la distribucion posterior
    plt.bar(['Cara', 'Cruz'], posterior)
    plt.title('Distribucion Posterior')
    plt.show()

    #Solo quedaria iterar cuando se vayan obteniendo nuevos datos.
```

**Estimación puntual y estimación de intervalos de credibilidad**

Se trata de dos enfoques diferentes para resumir y comunicar la incertidumbre en un parámetro. Por un lado, la estimación puntual proporciona una única estimación del valor de un parámetro, mientras que la estimación de intervalos de credibilidad proporciona una medida de la incertidumbre en torno a ese valor estimado. La elección entre uno u otro depende de la necesidad de comunicar la incertidumbre en los resultados y de la interpretación que se requiere en un contexto específico. Los intervalos de credibilidad son especialmente útiles en el enfoque bayesiano, donde se pueden calcular directamente a partir de la distribución posterior.

*Estimación Puntual.* Se refiere a la obtención de un único valor que representa la mejor suposición o estimación del valor de un parámetro o estadística de interés en un conjunto de datos. Un estimador puntual se elige de manera que resuma de la mejor manera posible la información contenida en los datos observados. Por ejemplo, un buen estimador de la media poblacional (imposible de calcular en la gran mayoría de problemas) es la media muestral. La estimación puntual no proporciona información sobre la incertidumbre en torno al valor estimado, y se interpreta como una única "mejor conjetura" del parámetro (aproximación).

*Estimación de Intervalos de Credibilidad.* También conocidos como intervalos de confianza bayesianos, permiten establecer una medida de la incertidumbre alrededor de un parámetro o estadística de interés. En lugar de proporcionar un solo valor estimado, un intervalo de credibilidad incluye un rango de valores posibles para el parámetro, junto con una medida de la probabilidad o credibilidad asociada con ese rango. Siguiendo el ejemplo anterior, de intervalo de credibilidad tomaríamos el intervalo del 95% de credibilidad para la media poblacional está entre X y Y (los límites del intervalo). Los intervalos de credibilidad reflejan la incertidumbre y permiten una interpretación más completa de los resultados, ya que indican la variabilidad posible en el parámetro.

**Estimación de la incertidumbre en Redes Bayesianas**

Para medir la incertidumbre, ya se han visto algunas medidas que permiten estimarla, como la entropía, varianza, MCMC, intervalos de confianza bayesianos BP, etc. Sin embargo, es solo la punta del iceberg, puesto que existen decenas de medidas, unas más específicas y otras más generales, para resolver distintos problemas. A continuación, se explican brevemente algunas de ellas, siendo necesario profundizar en la que se adapte mejor al problema que se quiera tratar:

1. Análisis de Sensibilidad: identifica cuáles son los factores más influyentes en los resultados del modelo y cómo las variaciones en estos factores afectan las conclusiones del análisis.
2. Análisis de Sensibilidad Global: evalúa el impacto de las variaciones en múltiples parámetros.
3. Análisis de Sensibilidad Local: se centra en un solo parámetro o variable a la vez.
4. Bootstrap Bayesiano: se pueden realizar múltiples remuestreos de los datos para evaluar cómo varían las conclusiones cuando se introducen diferentes muestras de datos.
5. Análisis de Propagación de Errores: cuando se trabaja con modelos que involucran incertidumbre en las entradas o parámetros, se ha de analizar y comprender cómo las incertidumbres se propagan a través de los modelos, lo que es esencial para tomar decisiones informadas y confiables
6. Análisis en Distribuciones No Paramétricas: para modelos no paramétricos, como procesos Gaussianos o procesos de Dirichlet, se pueden aplicar técnicas específicas para cuantificar la incertidumbre como pruebas de Wilcoxon,de Kruskal-Wallis, de Mann-Whitney U o de Kolmogorov-Smirnov, entre otros.
7. Muestreo Reversible Jump (RJMCMC): explora diferentes estructuras de red, permitiendo la estimación de la incertidumbre en la topología de la red.
8. Distribución a Posteriori Empírica (EPD): se basa en el muestreo de Monte Carlo y permite aproximar la distribución posterior mediante muestras.
9. Divergencia de Kullback-Leibler (KL): mide la discrepancia entre dos distribuciones de probabilidad, como la distribución posterior y la distribución previa.
10. Estimación de Momentos de Orden Superior: estimar momentos de orden superior para cuantificar aspectos más detallados de la incertidumbre.
11. Estimación de Incertidumbre en Redes Temporales: técnicas específicas para estimar la incertidumbre en los estados futuros.
12. Estimación de Incertidumbre en Modelos de Regresión Bayesianos: medidas como los intervalos de predicción y los intervalos de credibilidad para estimar la incertidumbre en las predicciones.
13. Estimación de Incertidumbre en Redes Bayesianas Dinámicas: técnicas de filtrado y suavizado para estimar la incertidumbre en estados ocultos.
14. Redes Bayesianas con Datos Faltantes: técnicas de imputación bayesiana para manejar datos faltantes y estimar la incertidumbre asociada con los valores imputados.

15. Estimación de Incertidumbre en Inferencia de Causalidad: técnicas de bootstrap y remuestreo para estimar la incertidumbre en las relaciones causales.
16. Muestreo por Importancia Secuencial (SIS): sirven para estimar la distribución posterior a medida que se obtienen nuevas observaciones.

Por último, destacar que las métricas clásicas usadas en los problemas de clasificación en IA, cómo precisión (precision), recuperación (recall), sensibilidad (sensitivity), especificidad (specificity) y la puntuación F1 (F1-score) no se utilizan directamente para medir la incertidumbre en un modelo de clasificación. Estas métricas se usan principalmente para evaluar el rendimiento de un modelo en términos de su capacidad para clasificar correctamente las instancias de datos en función de sus predicciones y las etiquetas reales.

La incertidumbre se refiere a la falta de confianza o certeza en las predicciones del modelo. Mientras que las métricas de evaluación miden la precisión de las predicciones, no proporcionan información directa sobre la incertidumbre.

Para medir la incertidumbre en un modelo de clasificación, es necesario considerar otras de las métricas mencionadas que evalúen la confianza o probabilidad asociada con las predicciones.

### 3.3.3 Aprendizaje supervisado con incertidumbre

En el contexto del aprendizaje supervisado con incertidumbre, las técnicas de aprendizaje automático (machine learning) reconoce y modela la incertidumbre asociada con los datos etiquetados y las predicciones realizadas por un modelo. En lugar de tratar las etiquetas como valores precisos y definitivos, este enfoque considera que las etiquetas pueden ser inciertas o ruidosas y busca incorporar esta incertidumbre en el proceso de aprendizaje y predicción.

Se deben tener en cuenta ciertos aspectos, como que la incertidumbre puede surgir de varias fuentes, como errores de medición, etiquetas ambiguas, datos ruidosos o insuficiencia de datos. Identificar y entender las fuentes de incertidumbre se convierte en algo fundamental.
Los modelos a usar se basan en enfoques probabilísticos para representar la incertidumbre en las etiquetas y las predicciones. Esto implica modelar la probabilidad de que una etiqueta sea correcta o la probabilidad de que una predicción sea precisa.
Respecto a la clasificación, se pueden usar dos enfoques. Uno es el probabilístico, que ya se ha visto con anterioridad, donde en lugar de asignar una única clase a una instancia de datos, se asigna una distribución de probabilidad sobre las clases posibles. Por ejemplo, en lugar de decir "esto es un gato", el modelo podría decir "hay un 80% de probabilidad de que sea un gato y un 20% de probabilidad de que sea un perro". El otro enfoque es la regresión y clasificación bayesiana:

**Regresión y Clasificación bayesiana**

**Regresión bayesiana**: se **modela** la relación entre una variable de entrada (características) y una variable de salida continua (objetivo) de una manera probabilística, mediante una distribución de probabilidad condicional. El **objetivo** es estimar la distribución de probabilidad de la variable objetivo dado un conjunto de características. El **resultado** es una distribución de probabilidad sobre el valor de la variable objetivo (en lugar de una predicción puntual, se obtiene una estimación de la variable objetivo con su incertidumbre). La **evaluación** implica métricas que reflejan la calidad de la distribución de probabilidad de la variable objetivo, como el error cuadrático medio, el coeficiente de determinación ($R^2$) y los intervalos de predicción. Su **uso** es predecir valores numéricos, como pronóstico financiero, estimación de precios, modelado de series temporales y análisis de datos continuos.

**Clasificación bayesiana:** se aplica a problemas de clasificación, donde se **modela** la probabilidad de pertenencia a cada clase dada una instancia de datos de entrada. El **objetivo** es calcular las probabilidades de clase y asignar la instancia a la clase con la probabilidad más alta. El resultado es una distribución de probabilidad sobre las clases posibles. Se proporciona una probabilidad para cada clase, y la instancia se asigna a la clase más probable. Se **evalúa** mediante métricas como la precisión, la sensibilidad, la especificidad y la entropía de clase, que miden la calidad de las probabilidades de clase asignadas. Se **usa** en problemas de clasificación, como diagnóstico médico, procesamiento de lenguaje natural (clasificación de texto), reconocimiento de patrones y filtrado de spam.

En este tipo de aprendizaje, también se ha de tener en cuenta la estimación de intervalos de predicción para cuantificar la incertidumbre en las predicciones. Estos intervalos representan rangos en los que se espera que caiga el valor verdadero con cierta confianza.

Finalmente, la evaluación de modelos tendrá en cuenta la incertidumbre en las predicciones, sabiendo que las métricas de rendimiento, como la precisión, pueden no ser adecuadas por sí solas, y es importante considerar métricas que reflejen la calidad de las predicciones probabilísticas.

Otro aspecto relevante que no se ha mencionado, es que el aprendizaje automático supervisado puede flexibilizarse como semi-supervisado, combinando datos etiquetados y no etiquetados para mejorar el rendimiento del modelo y reducir la incertidumbre. Los datos no etiquetados pueden proporcionar información adicional para hacer predicciones más precisas.
Además, se puede dotar al modelo de cierta inteligencia en la selección de las instancias de datos más significativas a etiquetar, lo que ayuda a reducir la incertidumbre en las áreas más importantes del espacio de características. A esto se le denomina aprendizaje activo con incertidumbre.

### 3.3.4 Aprendizaje por refuerzo probabilístico

Del inglés *Probabilistic Reinforcement Learning*, se trata de una variante del aprendizaje por refuerzo, que se centra en modelar la incertidumbre inherente a las interacciones entre un agente y su entorno. En lugar de tratar los procesos de decisión como completamente deterministas, el aprendizaje por refuerzo probabilístico considera que hay incertidumbre en las transiciones de estado y en las recompensas, y busca modelar y tomar decisiones óptimas en función de esta incertidumbre.
Es clave en aplicaciones donde la incertidumbre es inherente y su consideración crítica, como puede ser la navegación de robots en entornos desconocidos o en la toma de decisiones en juegos con información incompleta, por ejemplo.

En estos modelos, se representa el entorno y las transiciones de estado y no se tiene certeza completa sobre las transiciones y las recompensas, llevando a trabajar con distribuciones de probabilidad sobre estos valores (políticas estocásticas). Es decir, se pueden tener incertidumbre en las transiciones y en las recompensas:

- Incertidumbre en las Transiciones: las transiciones no son deterministas como en el aprendizaje por refuerzo tradicional. Esto significa que no se sabe con certeza cuál será el próximo estado después de tomar una acción en un estado dado, teniendo que trabajar con probabilidades.
- Incertidumbre en las Recompensas: se pueden recibir recompensas con distribuciones de probabilidad en lugar de valores deterministas.

Respecto al equilibrio entre explotación (seguir la mejor acción conocida) y exploración (probar nuevas acciones para aprender) que tienen estos tipos de modelos (también llamada búsqueda en anchura o profundidad), la gestión se realiza probabilísticamente, decidiendo en cada momento cuándo explorar y cuándo explotar.

Por último, la forma de modular la incertidumbre en las transiciones y recompensas se suele llevar a cabo mediante técnicas como procesos de decisión de Markov (MDP) parcialmente observados, modelos gráficos probabilísticos y métodos de inferencia bayesiana.

Al igual que en otros apartados, existen muchos algoritmos diferentes de aprendizaje por refuerzo (reinforcing learning, RL) como Temporal Difference RL, Policy-Based RL, Value-Based RL, Model-Based RL, Deep Reinforcement Learning, Imitation Learning, Continuous RL, etc.
Ante la imposibilidad de explicar con detalle todos, vamos a ver un ejemplo más detallado de un algoritmo muy usado, Q-learning, que es un tipo de algoritmo de refuerzo con diferencia temporal (Temporal Difference RL).

#### Ejemplo de Reinforcing Learning: Q-learning

Q-learning es un algoritmo de aprendizaje por refuerzo (RL) usado para aprender una política óptima en un entorno discreto basado en la teoría de Markov, donde el agente toma decisiones secuenciales para

maximizar una recompensa acumulativa. La principal característica del Q-learning es que aprende una función llamada *Q-función* que estima el valor esperado de tomar una acción en un estado específico y seguir una política óptima a partir de ese punto.

El ejemplo que vamos a ver es simple: un personaje que camina por un laberinto para encontrar la salida. Para resolverlo, se implementa el algoritmo Q-learning para que el personaje que camina por un laberinto simple encuentre la salida (S es el estado inicial, G es el estado objetivo y H representa una pared). El personaje aprende a tomar acciones para llegar al estado objetivo mientras evita las paredes.

El código muestra cómo se inicializa la tabla Q, cómo se seleccionan las acciones epsilon-greedy, cómo se actualiza la tabla Q y cómo se encuentra el camino óptimo desde el estado inicial hasta el objetivo utilizando la política aprendida.

```
import numpy as np

#Definir el laberinto como una matriz
#'S' representa el estado inicial
#'G' representa el estado objetivo
#'H' representa una pared
#'0'representa un espacio libre
maze = np.array([
['S', 'H', '0', '0', '0'],
['0', 'H', '0', 'H', '0'],
['0', '0', '0', '0', '0'],
['0', 'H', '0', 'H', '0'],
['0', '0', '0', 'G', '0']
])

#Definir las acciones posibles (arriba, abajo, izquierda, derecha)
actions = [(0, -1), (0, 1), (-1, 0), (1, 0)]

#Tasa de aprendizaje y factor de descuento
alpha = 0.1
gamma = 0.9

#Inicializar la tabla Q con ceros
n_states = np.prod(np.shape(maze))
n_actions = len(actions)
Q = np.zeros((n_states, n_actions))

#Obtener la posicion del estado inicial y objetivo
start_state = np.argwhere(maze == 'S')[0]
goal_state = np.argwhere(maze == 'G')[0]

#Funcion para convertir coordenadas (x, y) a un unico numero de estado
def state_to_idx(state):
    return state[0] * maze.shape[1] + state[1]

#Funcion para determinar si un estado es valido (no fuera del
                                laberinto ni en una pared)
def is_valid_state(state):
    return 0 <= state[0] < maze.shape[0] and 0 <= state[1] < maze.
                                shape[1] and maze[state[0], state[1]]
                                 != 'H'

#Algoritmo Q-learning
num_episodes = 1000

```

```
    for _ in range(num_episodes):
        state = start_state
        done = False

    while not done:
        #Seleccion de una accion epsilon-greedy
        if np.random.rand() < 0.2:  #Exploracion con probabilidad epsilon
            action = np.random.randint(n_actions)
        else:
            action = np.argmax(Q[state_to_idx(state), :])

        #Tomar la accion y obtener el proximo estado y recompensa
        next_state = (state[0] + actions[action][0], state[1] + actions[
                                    action][1])

        if is_valid_state(next_state):
            if maze[next_state[0], next_state[1]] == 'G':
                reward = 1  #Recompensa por llegar al objetivo
                done = True
            else:
                reward = 0  #Recompensa peque a por moverse

            #Actualizar la tabla Q
            Q[state_to_idx(state), action] += alpha * (reward + gamma * np
                                    .max(Q[state_to_idx(next_state), :])
                                    - Q[state_to_idx(state), action])

            state = next_state

    #Encontrar el camino optimo desde el estado inicial hasta el objetivo
    path = [tuple(start_state)]
    state = start_state

    while not (state[0] == goal_state[0] and state[1] == goal_state[1]):
        action = np.argmax(Q[state_to_idx(state), :])
        next_state = (state[0] + actions[action][0], state[1] + actions[
                                    action][1])
        path.append(tuple(next_state))  #Convertir a tupla
        state = next_state

    #Imprimir el laberinto con el camino optimo
    for row in range(maze.shape[0]):
        for col in range(maze.shape[1]):
            if (row, col) == tuple(start_state):
                print('S', end=' ')
            elif (row, col) == tuple(goal_state):
                print('G', end=' ')
            elif maze[row, col] == 'H':
                print('H', end=' ')
            elif any(coord == (row, col) for coord in path):    #Verificar
                                    si (row, col) esta en path
                print('*', end=' ')
            else:
                print('0', end=' ')
        print()
```

Pero, ¿este ejemplo tiene incertidumbre? No, es determinista. Por lo tanto, ¿cómo podríamos incorporar incertidumbre en este ejemplo? Pues vamos a hacer que las transiciones entre estados sean estocásticas y agregar una variabilidad en las recompensas.

- Estocasticidad en las transiciones: en lugar de realizar transiciones deterministas, se puede hacer que las transiciones entre estados sean probabilísticas. Por ejemplo, podemos agregar una probabilidad de

que el agente no alcance su objetivo deseado y tome una dirección diferente a la deseada, incluso si eligió la acción óptima. Esto se hace modificando la lógica de transición de estados en el bucle principal del algoritmo Q-learning.

- Variabilidad en las recompensas: podemos introducir incertidumbre en las recompensas al hacer que las recompensas sean aleatorias en lugar de constantes. Por ejemplo, en lugar de dar una recompensa de 1 cuando el agente llega al estado objetivo, se puede asignar una recompensa que sigue una distribución de probabilidad, como una distribución normal o uniforme.

```
...

    #Algoritmo Q-learning con incertidumbre
    num_episodes = 1000
    max_steps_per_episode = 100  #Numero maximo de pasos por episodio

    for episode in range(num_episodes):
        state = start_state
        done = False
        step = 0  #Contador de pasos

        while not done and step < max_steps_per_episode:
            #Seleccion de una accion epsilon-greedy
            epsilon = 0.2  #Probabilidad de exploracion
            if np.random.rand() < epsilon:
                action = np.random.randint(n_actions)  #Exploracion
                                aleatoria
            else:
                action = np.argmax(Q[state_to_idx(state), :])  #
                                Explotacion

            next_state = (state[0] + actions[action][0], state[1] +
                                actions[action][1])

            if is_valid_state(next_state):
                #Introducir variabilidad en las recompensas
                stochastic_reward = np.random.normal(loc=0.9, scale=0.1)
                                #Recompensa aleatoria
                reward = stochastic_reward

                #Actualizar la tabla Q usando la ecuacion de Q-learning
                Q[state_to_idx(state), action] += alpha * (reward + gamma
                                * np.max(Q[state_to_idx(next_state),
                                :]) - Q[state_to_idx(state), action])

                state = next_state
                step += 1

                if maze[next_state[0], next_state[1]] == 'G':
                done = True

...
```

### 3.3.5 Incertidumbre en la toma de decisiones

La incertidumbre en la toma de decisiones se produce cuando falta conocimiento completo o información precisa sobre un conjunto de eventos o situaciones futuras que pueden influir en el resultado de una decisión. En muchas situaciones de la vida real, no se tiene información perfecta y hay que tomar decisiones basadas en datos incompletos,

imprecisos o inciertos. La incertidumbre puede surgir debido a varios factores, como falta de datos, variabilidad inherente en los sistemas, ruido en los datos, que los modelos estén simplificados o, incluso, cambios inesperados.

Frente a ello, se utilizan técnicas como la teoría de la decisión bayesiana para tomar decisiones racionales en situaciones inciertas. A esto se le denomina Toma de decisiones bajo incertidumbre.

A continuación, se describen brevemente algunas de las técnicas más usadas:

**Teoría de la decisión bayesiana**. Esta técnica combina la probabilidad bayesiana con la teoría de la decisión. Se basa en la actualización de creencias y probabilidades a medida que se recopila nueva información. Los modelos bayesianos permiten tomar decisiones racionales considerando la incertidumbre y la información disponible.

**Árboles de decisión**. Los árboles de decisión son herramientas gráficas que representan decisiones secuenciales y sus consecuencias. Cada nodo en el árbol representa una decisión o un evento, y las ramas representan las opciones disponibles y sus probabilidades asociadas. Los árboles de decisión son útiles para visualizar y analizar escenarios complejos.

**Valor esperado**. El valor esperado es una medida que calcula la "bondad" esperada de una decisión dada la incertidumbre. Se calcula como la suma ponderada de los posibles resultados multiplicados por sus probabilidades. Las decisiones se eligen con base en el valor esperado más alto.

**Teoría de juegos**. En situaciones en las que múltiples partes toman decisiones estratégicas, la teoría de juegos se utiliza para modelar y analizar interacciones competitivas o cooperativas bajo incertidumbre. Esto es común en economía y estrategia empresarial.

**Optimización robusta**. En lugar de buscar soluciones óptimas en un solo escenario, la optimización robusta busca soluciones que sean buenas en una amplia gama de escenarios posibles, teniendo en cuenta la incertidumbre.