

Tema 1. Introducción: Razonamiento y Representación del Conocimiento

Conceptos clave

La representación del conocimiento y el razonamiento es un área de la inteligencia artificial cuyo objetivo principal es representar el conocimiento de manera que facilite la inferencia, es decir, la capacidad de sacar conclusiones a partir de dicho conocimiento. Esto implica el uso de sistemas de símbolos que representan un dominio de discurso (aquello sobre lo que se puede hablar), junto con funciones que permitan razonar formalmente sobre los objetos. (Fuente: Wikipedia)

En este contexto, se emplean lógicas formales para proporcionar una semántica precisa que define cómo se aplican las funciones de razonamiento a los símbolos dentro de un dominio.

Incertidumbre en el conocimiento

En la práctica, el dominio de discurso no siempre es completamente observable, lo que genera incertidumbre. Esto puede ocurrir debido a:

- Información inexistente o inaccesible.
- Ambigüedad en los datos.
- Representaciones imprecisas o inconsistentes.

Representación del conocimiento

En ciencias de la computación e inteligencia artificial, se han diseñado diversas estructuras para representar información, conocidas colectivamente como “representación del conocimiento”. Estas representaciones están diseñadas para ser procesadas por ordenadores modernos.

Originalmente, la lógica fue la herramienta principal para representar y razonar sobre el conocimiento. Esto incluye:

- Almacén de hechos.
- Establecimiento de relaciones entre hechos mediante reglas de producción.
- Inferencia de nuevos hechos basados en hechos existentes y reglas.

Agentes lógicos

Los agentes inteligentes utilizan la representación del conocimiento para realizar tareas. Su funcionamiento general incluye:

1. Obtener una percepción y registrarla como un nuevo hecho.
2. Incorporar esta percepción a la base de conocimiento.
3. Actualizar la base de conocimiento usando inferencia para decidir la siguiente acción.
4. Realizar la acción y actualizar la base de conocimiento si es necesario.

Expansión del campo

A medida que la inteligencia artificial avanza, la representación del conocimiento debe trascender la lógica formal para abordar problemas complejos. Ejemplos incluyen:

- La representación del entorno de un robot móvil para evitar obstáculos mientras alcanza un destino.
- Cómo los chatbots estructuran información para responder rápida y adecuadamente.

Conocimiento incierto

Los sistemas inteligentes a menudo operan bajo condiciones de incertidumbre, que pueden deberse a:

- Falta de información completa.
- La naturaleza no observable del universo de discurso.

La racionalidad, en ausencia de incertidumbre, asegura decisiones óptimas. Sin embargo, bajo incertidumbre se recurre a:

- Grados de creencia.
- Evaluación de la utilidad de las acciones posibles.

Una forma común de representar la incertidumbre es a través de la teoría de la probabilidad.

El mundo Wumpus

El “mundo Wumpus” es un juego de lógica que ilustra el uso de agentes inteligentes en entornos inciertos.

- **Tablero:** Representa localizaciones por las que se mueve el agente.
- **Objetivo:** Encontrar un tesoro escondido en una casilla desconocida.
- **Obstáculos:**
 - El “Wumpus”, una criatura que devora al agente si este entra en su casilla.
 - Agujeros que hacen que el agente caiga al vacío.

Características del agente

El agente posee:

- **Actuadores:**
 - Puede moverse en direcciones Norte, Sur, Este y Oeste.
 - Dispone de un arco y una flecha para matar al Wumpus si este se encuentra en su trayectoria.
- **Sensores:**
 - Detecta el hedor del Wumpus en casillas adyacentes.
 - Percibe brisas cerca de agujeros.
 - Oye gritos si hiere al Wumpus.
 - Ve el tesoro al entrar en su casilla.

Ejemplo

Un escenario típico en el mundo Wumpus:

1. **Percepción inicial:** {nada, nada, nada, nada}.
 - Acción: Moverse a (2,1).
2. **Percepción:** {nada, brisa, nada, nada}.
 - Acción: Evitar casillas peligrosas usando inferencia lógica.
3. **Percepción:** {hedor, nada, nada, nada}.
 - Acción: Analizar probables posiciones del Wumpus y actuar en consecuencia.

Lógica Formal: Razonamiento y Representación del Conocimiento

Introducción

La **lógica formal** es una herramienta esencial en el campo de la inteligencia artificial y la computación, ya que proporciona un medio para representar el conocimiento de manera precisa, sin ambigüedades ni redundancias. El ser humano ha utilizado históricamente el **lenguaje natural** para transmitir y representar conocimiento. Sin embargo, el lenguaje natural presenta problemas como:

- **Redundancias:** Expresiones que repiten información innecesariamente.
 - Ejemplo: "Sube arriba" (la acción de subir ya implica ir hacia arriba).
- **Ambigüedades:** Frases que pueden interpretarse de múltiples maneras.
 - Ejemplo: "Vi un elefante en mi patio" (¿Quién estaba en el patio, el hablante o el elefante?).

Para que una **computadora** pueda procesar y utilizar el conocimiento de manera efectiva, es necesario representarlo en un formato que elimine estas redundancias y ambigüedades. La lógica formal cumple este propósito al:

- Proporcionar un **lenguaje formal** con reglas estrictas de sintaxis y semántica.
- Permitir la **inferencia lógica** a través de principios bien definidos.
- Facilitar la **representación del conocimiento** de manera estructurada y coherente.

Tipos de Lógica Formal

Existen principalmente dos tipos de lógica formal que son relevantes en la representación del conocimiento:

1. Lógica Proposicional:

- Elemento básico: **Proposición**.
- Se enfoca en las relaciones entre proposiciones a través de conectivas lógicas.
- Limita su capacidad de expresión al no poder representar relaciones internas de las proposiciones.

2. Lógica de Primer Orden:

- Elementos básicos: **Términos** y **Predicados**.
- Extiende la lógica proposicional al permitir cuantificación y referencia a objetos individuales.
- Ofrece mayor expresividad y capacidad para representar conocimientos más complejos.

Lógica Proposicional

La **lógica proposicional** es el sistema más básico de lógica formal, centrado en el análisis y manipulación de **proposiciones** y sus relaciones mediante conectivas lógicas.

Lenguaje del Cálculo Proposicional

Enunciados y Proposiciones

- **Enunciado:** Pensamiento expresable por palabras o por escrito.
 - Puede ser una afirmación, pregunta, orden, etc.
 - Ejemplos: "¿Tienes hambre?", "Buenas tardes".
- **Proposición:** Enunciado que puede ser asignado un valor de **verdadero (V)** o **falso (F)**, pero no ambos simultáneamente.
 - Debe ser una declaración clara y sin ambigüedad en cuanto a su veracidad.
 - Ejemplos:
 - "Llueve."
 - "Es tarde."
 - "Hace calor."
 - "El coche es rojo."

Conectivas Lógicas

Las **conectivas lógicas** son operadores que permiten construir nuevas proposiciones a partir de otras existentes. Las principales conectivas son:

1. Negación Lógica (\neg):

- Invierte el valor de verdad de una proposición.
- Si p es una proposición, entonces su negación se denota como $\neg p$ y se lee "no p ".
- Ejemplos:
 - Si p : "Llueve", entonces $\neg p$: "No llueve".

2. Conjunción Lógica (\wedge):

- La proposición p y q es verdadera si y solo si ambas p y q son verdaderas.
- Se denota $p \wedge q$.
- Ejemplo:
 - p : "El coche es rojo."
 - q : "El coche es un deportivo."
 - $p \wedge q$: "El coche es un deportivo rojo."

3. Disyunción Lógica (\vee):

- **Disyunción Inclusiva:** La proposición p o q es verdadera si al menos una de p o q es verdadera.
 - Se denota $p \vee q$.
 - Ejemplo:
 - "El coche es rojo o es un deportivo."
- **Disyunción Exclusiva (XOR):** p o q es verdadera si exactamente una de p o q es verdadera, pero no ambas.
 - Se denota $p \text{ XOR } q$.

Proposiciones Simples y Compuestas

• Proposiciones Simples:

- No contienen conectivas lógicas.
- Representadas por letras minúsculas o mayúsculas: p, q, r, A, B .

• Proposiciones Compuestas:

- Formadas a partir de proposiciones simples mediante conectivas lógicas.
- Ejemplos:
 - $\neg p, p \vee q, p \wedge q$.

Conectivas Lógicas Secundarias

Además de las conectivas básicas, existen conectivas lógicas secundarias que permiten expresar implicaciones y equivalencias:

1. Implicación Material (\rightarrow):

- La proposición $p \rightarrow q$ se lee "si p entonces q ".
- Es falsa solo cuando p es verdadera y q es falsa.
- Equivalencia lógica: $p \rightarrow q \equiv \neg p \vee q$.
- Ejemplo:
 - p : "Llueve."
 - q : "Me mojo."
 - $p \rightarrow q$: "Si llueve, entonces me mojo."

2. Implicación Recíproca (\leftarrow):

- La proposición $p \leftarrow q$ se interpreta como "si q entonces p ".
- Es la implicación en sentido contrario.

3. Bicondicional (\leftrightarrow):

- La proposición $p \leftrightarrow q$ se lee "p si y solo si q".
- Es verdadera cuando p y q tienen el mismo valor de verdad.
- Representa una equivalencia lógica entre p y q .

Tablas de Verdad

Las **tablas de verdad** son herramientas que permiten determinar el valor de verdad de proposiciones compuestas en función de los valores de verdad de sus componentes.

Ejemplos de Tablas de Verdad

1. Negación:

p	$\neg p$
V	F
F	V

2. Conjunción:

p	q	$p \wedge q$
V	V	V
V	F	F
F	V	F
F	F	F

3. Disyunción Inclusiva:

p	q	$p \vee q$
V	V	V
V	F	V
F	V	V
F	F	F

4. Implicación Material:

p	q	$p \rightarrow q$
V	V	V
V	F	F
F	V	V
F	F	V

Tautologías, Contradicciones y Contingencias

- **Tautología:**
 - Una proposición que es siempre verdadera, independientemente de los valores de verdad de sus componentes.
 - Ejemplo: $p \vee \neg p$ es siempre verdadera.
- **Contradicción:**
 - Una proposición que es siempre falsa.
 - Ejemplo: $p \wedge \neg p$ es siempre falsa.
- **Contingencia:**
 - Una proposición que puede ser verdadera o falsa dependiendo de los valores de sus componentes.
 - Ejemplo: $p \vee q$ puede ser verdadera o falsa.

Ejemplo:

Determinar la naturaleza de la proposición $p \vee \neg p$:

p	$\neg p$	$p \vee \neg p$
V	F	V
F	V	V

La proposición $p \vee \neg p$ es una **tautología**.

Ejercicios de Lógica Proposicional

Determinar el valor de verdad de las siguientes expresiones lógicas:

- $p \rightarrow (\neg p \rightarrow p)$
 - Valor de verdad: Siempre **verdadera** (tautología).
- $(p \wedge \neg q) \wedge q$
 - Valor de verdad: Siempre **falsa** (contradicción).
- $(r \vee p) \wedge \neg(q \vee p)$
 - Valor de verdad: Dependiente de los valores de p, q, r (contingencia).

Lógica de Primer Orden

La **lógica de primer orden (LPO)**, también conocida como **cálculo de predicados**, amplía la lógica proposicional al incorporar **cuantificadores** y la capacidad de referirse a objetos individuales y sus propiedades.

Limitaciones de la Lógica Proposicional

La lógica proposicional es insuficiente para expresar ciertas afirmaciones que implican relaciones entre objetos o generalizaciones. Por ejemplo:

- Ejemplo:**
 - Conocimiento: "Confucio es un hombre."
 - Regla general: "Todos los hombres son mortales."
 - Deducción: "Por lo tanto, Confucio es mortal."

La lógica proposicional no puede expresar esta inferencia de manera directa, ya que no puede representar la estructura interna de las proposiciones ni cuantificar sobre individuos.

Elementos de la Lógica de Primer Orden

Alfabeto

El alfabeto de la LPO está compuesto por:

1. Símbolos de Variables:

- Representan objetos o individuos.
- Denotados por letras: x, y, z, x_1, y_1 , etc.

2. Símbolos de Constantes:

- Representan objetos específicos.
- Ejemplos: a, b, c , Pedro, Confucio.

3. Símbolos de Función:

- Representan funciones que toman objetos y producen otro objeto.
- Denotados por f, g, h .
- Aridad:** Número de argumentos que toma una función.

4. Símbolos de Predicado:

- Representan propiedades o relaciones entre objetos.
- Denotados por $P, Q, R, ES_JEFE, ES_MORTAL$.
- Pueden ser de aridad n , es decir, pueden tomar n argumentos.

5. Conectivas Lógicas:

- Igual que en la lógica proposicional: $\neg, \vee, \wedge, \rightarrow, \leftrightarrow$.

6. Cuantificadores:

- **Cuantificador Universal (\forall):**
 - Representa "para todo" o "para cada".
 - Ejemplo: $\forall x$ significa "para todo x ".
- **Cuantificador Existencial (\exists):**
 - Representa "existe al menos uno".
 - Ejemplo: $\exists x$ significa "existe un x ".

7. Símbolos de Puntuación:

- Paréntesis y comas para agrupar expresiones.

Términos y Fórmulas Bien Formadas

• Términos:

- Objetos de los que hablamos.
- Pueden ser:
 - **Constantes:** a, b, c .
 - **Variables:** x, y, z .
 - **Funciones aplicadas a términos:** $f(x), g(a, b)$.

• Fórmulas Bien Formadas (fbf):

- Combinaciones de términos y símbolos conforme a las reglas sintácticas de la LPO.
- Pueden ser:
 - **Átomos:** Aplicación de un predicado a términos: $P(x, y)$.
 - **Compuestas:** Combinaciones de átomos mediante conectivas y cuantificadores.

Ejemplos

1. Predicado con Función:

- Representar: "Uno sumado a dos es igual a tres."
- Términos:
 - "Uno": 1.
 - "Dos": 2.
 - "Tres": 3.
 - "Uno sumado a dos": $\text{sumadoa}(1, 2)$.
- Predicado:
 - "Es igual a": $ES_IGUAL(a, b)$.
- Fórmula:
 - $ES_IGUAL(\text{sumadoa}(1, 2), 3)$.

2. Predicado Binario:

- Representar: "Pedro es el jefe de Luis."
- Términos:
 - "Pedro": Pedro.
 - "Luis": Luis.
- Predicado:
 - "Es jefe de": $ES_JEFE(a, b)$.
- Fórmula:
 - $ES_JEFE(\text{Pedro}, \text{Luis})$.

Cuantificadores

Cuantificador Universal (∀)

- Se interpreta como "para todo x " o "para cada x ".
- Utilizado para expresar afirmaciones generales.
- **Ejemplo:**
 - "Todos los seres humanos son mortales."
 - Representación:
 - $\forall x [\text{ES_HUMANO}(x) \rightarrow \text{ES_MORTAL}(x)]$.

Cuantificador Existencial (∃)

- Se interpreta como "existe al menos un x ".
- Utilizado para afirmar la existencia de al menos un objeto que cumple cierta propiedad.
- **Ejemplo:**
 - "Hay por lo menos un satélite."
 - Representación:
 - $\exists x [\text{SATELITE}(x)]$.

Expresividad y Potencial de la Lógica de Primer Orden

La LPO es extremadamente potente y permite:

- **Representar cualquier conocimiento** de manera formal y estructurada.
- **Realizar inferencias lógicas** para deducir nueva información a partir de conocimientos existentes.
- **Validar argumentos** y comprobar la **validez** de razonamientos.
- **Aprender y descubrir patrones** en datos al utilizarla como base para algoritmos de aprendizaje automático simbólicos.

Ejemplo Completo

- **Conocimiento:**
 - "Todos los hombres son mortales."
 - "Sócrates es un hombre."
- **Representación en LPO:**
 - $\forall x [\text{HOMBRE}(x) \rightarrow \text{MORTAL}(x)]$.
 - $\text{HOMBRE}(\text{Socrates})$.
- **Deducción:**
 - Podemos inferir que $\text{MORTAL}(\text{Socrates})$, es decir, "Sócrates es mortal."

Conclusiones

La **lógica formal** es una herramienta fundamental para la **representación del conocimiento** y el **razonamiento** en sistemas de inteligencia artificial. La lógica proposicional ofrece una base sólida para manejar proposiciones y conectivas lógicas, pero es limitada en su capacidad de expresión.

La **lógica de primer orden** supera estas limitaciones al introducir cuantificadores y la capacidad de referirse a objetos y relaciones específicas. Esto permite representar conocimientos de manera más rica y realizar inferencias más poderosas.

Al comprender y aplicar principios de lógica formal, es posible:

- **Eliminar ambigüedades** y **redundancias** presentes en el lenguaje natural.
- **Formalizar conocimientos** y reglas de inferencia que pueden ser procesadas por computadoras.
- **Desarrollar agentes inteligentes** capaces de razonar, aprender y tomar decisiones basadas en conocimientos representados lógicamente.

Este resumen ha cubierto los conceptos esenciales de la lógica formal, enfatizando la importancia de la lógica proposicional y la lógica de primer orden en la representación y manipulación del conocimiento. Comprender estos fundamentos es crucial para avanzar en el campo de la inteligencia artificial y desarrollar sistemas capaces de interactuar de manera inteligente con el mundo.

Los Sistemas Expertos: Una Revolución en la Inteligencia Artificial

Introducción

Los sistemas expertos representan uno de los avances más significativos en el campo de la inteligencia artificial, surgiendo como una respuesta a la necesidad de replicar el conocimiento y el proceso de toma de decisiones de expertos humanos en dominios específicos. A diferencia de los sistemas de propósito general, estos sistemas se centran en áreas de conocimiento concretas, lo que ha demostrado ser una estrategia más efectiva para obtener resultados prácticos y aplicables.

¿Qué es un Sistema Experto?

Un sistema experto busca “clonar” el conocimiento de un experto humano en un campo específico. Estos sistemas se caracterizan por:

- Resolver problemas complejos mediante razonamiento simbólico.
- Manejar situaciones con incertidumbre.
- Proporcionar interfaces naturales para la interacción con usuarios.
- Incorporar mecanismos de aprendizaje.
- Justificar sus decisiones y recomendaciones.

Arquitectura de un Sistema Experto

La arquitectura fundamental de un sistema experto se compone de tres elementos principales:

1. Base de Conocimiento

- Almacena la experiencia y el saber del experto.
- Organizada en hechos y reglas estructuradas en formato IF-THEN.
- Ejemplo:
 - **Regla:** “Si el coche no arranca Y la batería muestra menos de 10 voltios, ENTONCES el problema es una batería defectuosa”.

2. Motor de Inferencia

- Procesa la información contenida en la base de conocimiento.
- Llega a conclusiones utilizando algoritmos de búsqueda y matching.
- **Algoritmo Rete:** eficiente para manejar grandes conjuntos de reglas y hechos.

3. Interfaz de Usuario

- Permite la interacción en lenguaje natural.
- Facilita la entrada de datos y la visualización de resultados.

Subsistema de Explicación

Una característica fundamental de los sistemas expertos es su capacidad para explicar su razonamiento:

- Justifican sus decisiones.
- Proporcionan transparencia sobre cómo se alcanzaron las conclusiones.

Ventajas de los Sistemas Expertos

- **Disponibilidad:** Operan 24/7.
- **Consistencia:** Respuestas uniformes y sin variabilidad.
- **Preservación del conocimiento:** Facilitan su distribución y reutilización.

Limitaciones

- Carecen de creatividad y adaptabilidad comparados con expertos humanos.
- Dependencia de la calidad del conocimiento inicial.
- Pueden presentar problemas de rendimiento con grandes volúmenes de datos.

Aplicaciones

Los sistemas expertos se han implementado exitosamente en campos como:

- **Diagnóstico médico:** Analizan síntomas y resultados de pruebas.
- **Asesoría financiera:** Evaluación de variables de mercado.
- **Diagnóstico automotriz:** Identificación de fallos en vehículos.

Desarrollo de un Sistema Experto

El desarrollo incluye las siguientes fases:

1. **Adquisición del conocimiento:** Extraer y estructurar conocimiento experto.
2. **Diseño:** Planificar la estructura del sistema.
3. **Pruebas:** Validar su desempeño en escenarios reales.
4. **Documentación:** Registrar procesos y resultados.
5. **Mantenimiento:** Actualizar y mejorar el sistema.

Futuro de los Sistemas Expertos

A medida que la tecnología avanza, los sistemas expertos evolucionan integrando nuevas técnicas de inteligencia artificial y aprendizaje automático. Su capacidad para manejar conocimiento específico los convierte en herramientas clave en numerosos campos profesionales y científicos.

Conclusión

Los sistemas expertos representan una manifestación práctica y exitosa de la inteligencia artificial aplicada, demostrando cómo el conocimiento experto puede ser capturado, procesado y utilizado de manera efectiva. Su continua evolución asegura un futuro prometedor en la automatización del conocimiento experto.

Tema 5. Razonamiento con Incertidumbre y Teorema de Bayes

Introducción a la Incertidumbre

La incertidumbre en la inteligencia artificial se presenta cuando no se tiene un conocimiento completo del mundo o de las consecuencias de las acciones. Modelar la incertidumbre permite manejar situaciones complejas sin necesidad de describir exhaustivamente todas las reglas. En este contexto, se asigna un grado de creencia a las proposiciones, con valores entre 0 y 1, lo que permite expresar la confianza en la verdad de una proposición.

Grado de Creencia

El grado de creencia se asigna a una proposición para indicar su probabilidad de ser verdadera o falsa. Un valor de 0 significa que la proposición es inequívocamente falsa, mientras que un valor de 1 indica que es inequívocamente verdadera. Los valores intermedios reflejan niveles intermedios de creencia. Es importante diferenciar entre grado de creencia y grado de pertenencia, que es un concepto de la lógica difusa.

Decisiones Racionales

En situaciones de incertidumbre, un agente debe tomar decisiones racionales basadas en la teoría de la utilidad, que combina la teoría de la probabilidad y la utilidad. La utilidad mide el beneficio que un estado particular ofrece al agente, y el agente preferirá estados con mayor utilidad. La teoría de la decisión permite seleccionar acciones que maximicen la utilidad esperada.

Teoría de la Probabilidad

Notación Básica

Las variables aleatorias representan elementos del mundo cuyo estado es desconocido. Estas variables pueden ser:

- **Booleanas:** con dominio <cierto, falso>
- **Discretas:** con un dominio contable, como <soleado, lluvioso, nublado>
- **Continuas:** con dominio en los números reales o un subconjunto de estos.

Probabilidad a Priori o Incondicional

La probabilidad a priori de una proposición es el grado de creencia asignado en ausencia de información adicional. Por ejemplo, la probabilidad de que ocurra un choque puede escribirse como $P(\text{Chocar}) = 0.2$. Las probabilidades a priori se derivan de la observación y registro de eventos.

Distribución de Probabilidad

Una distribución de probabilidad asigna probabilidades a todos los posibles valores de una variable aleatoria. En una distribución conjunta, se consideran todas las combinaciones posibles de valores de un conjunto de variables.

Probabilidad Condicional

La probabilidad condicional $P(a|b)$ indica la probabilidad de a dado que se conoce b . La regla del producto establece que $P(a \wedge b) = P(a|b)P(b) = P(b|a)P(a)$.

Axiomas de la Probabilidad

1. Las probabilidades están entre 0 y 1: $0 \leq P(a) \leq 1$.
2. Las proposiciones necesariamente ciertas tienen probabilidad 1, y las falsas 0.
3. La probabilidad de una disyunción se calcula como $P(a \vee b) = P(a) + P(b) - P(a \wedge b)$.

Inferencia Probabilística

La inferencia probabilística se basa en la distribución conjunta completa de las variables. Permite calcular la probabilidad marginal y condicional de eventos complejos.

Independencia

Dos proposiciones a y b son independientes si $P(a|b) = P(a)$, lo que simplifica los cálculos de probabilidad y reduce el tamaño de representación del problema.

Teorema de Bayes

El Teorema de Bayes relaciona las probabilidades condicionales y marginales de eventos. Se deriva de la regla del producto:

$$P(b|a) = \frac{P(a|b)P(b)}{P(a)}$$

Este teorema permite actualizar la probabilidad de una hipótesis a medida que se obtiene nueva evidencia.

Planteamiento del Ejercicio

Un robot móvil dispone de un sensor para detectar obstáculos. Se nos proporcionan los siguientes datos:

- $P(\text{Sensor+} \mid \text{Choque}) = 0.85$ (El sensor detecta el obstáculo el 85% de las veces que hay choque)
- $P(\text{Choque}) = 0.20$ (Probabilidad de choque del 20%)
- $P(\text{Sensor+}) = 0.25$ (Probabilidad de que el sensor detecte un obstáculo)

Objetivos

1. Calcular $P(\text{Choque} \mid \text{Sensor+})$: Probabilidad de choque cuando el sensor detecta un obstáculo
2. Calcular $P(\text{No_Choque} \mid \text{Sensor-})$: Probabilidad de no chocar cuando el sensor no detecta obstáculo

Resolución

1. Cálculo de $P(\text{Choque} \mid \text{Sensor+})$

Aplicando el Teorema de Bayes:

$$P(\text{Choque} \mid \text{Sensor+}) = \frac{P(\text{Sensor+} \mid \text{Choque}) \times P(\text{Choque})}{P(\text{Sensor+})}$$
$$P(\text{Choque} \mid \text{Sensor+}) = \frac{0.85 \times 0.20}{0.25} = 0.68$$

Esto significa que cuando el sensor detecta un obstáculo, hay un 68% de probabilidad de que el robot choque.

2. Cálculo de $P(\text{No_Choque} \mid \text{Sensor-})$

Para este cálculo, necesitamos primero algunos valores adicionales:

a) $P(\text{No_Choque}) = 1 - P(\text{Choque}) = 1 - 0.20 = 0.80$

b) $P(\text{Sensor-}) = 1 - P(\text{Sensor+}) = 1 - 0.25 = 0.75$

c) $P(\text{Sensor-} \mid \text{No_Choque})$ podemos calcularlo usando la información dada:

$$P(\text{Sensor-} \mid \text{No_Choque}) = \frac{P(\text{Sensor-}) - P(\text{Sensor-} \mid \text{Choque}) \times P(\text{Choque})}{P(\text{No_Choque})}$$
$$P(\text{Sensor-} \mid \text{No_Choque}) = \frac{0.75 - 0.15 \times 0.20}{0.80} = 0.9125$$

Ahora podemos aplicar Bayes:

$$P(\text{No_Choque} \mid \text{Sensor-}) = \frac{P(\text{Sensor-} \mid \text{No_Choque}) \times P(\text{No_Choque})}{P(\text{Sensor-})}$$
$$P(\text{No_Choque} \mid \text{Sensor-}) = \frac{0.9125 \times 0.80}{0.75} = 0.97$$

Interpretación de Resultados

1. Cuando el sensor detecta un obstáculo (lectura positiva), hay un 68% de probabilidad de que el robot choque.
2. Cuando el sensor no detecta obstáculos (lectura negativa), hay un 97% de probabilidad de que el robot no choque.

Conclusiones del Ejercicio

- El sensor es más fiable cuando no detecta obstáculos (97% de acierto) que cuando los detecta (68% de acierto).
- Esto sugiere que el sensor tiene una tasa de falsos positivos relativamente alta.
- Para aplicaciones críticas de seguridad, podría ser necesario complementar este sensor con otros métodos de detección.

Implicaciones Prácticas

Este tipo de análisis es fundamental en robótica y sistemas autónomos para:

- Evaluar la fiabilidad de los sensores
- Tomar decisiones basadas en lecturas de sensores
- Diseñar sistemas de seguridad redundantes
- Calibrar la sensibilidad de los sensores

Tema 6: Redes Bayesianas y Razonamiento bajo Incertidumbre:

Introducción al Razonamiento Probabilístico y Redes Bayesianas

El razonamiento bajo incertidumbre es uno de los pilares fundamentales en la inteligencia artificial moderna y representa uno de los desafíos más significativos en este campo. Las redes bayesianas emergen como una solución elegante y poderosa para modelar y manipular conocimiento probabilístico en entornos inciertos, proporcionando un marco robusto para la representación y manipulación del conocimiento probabilístico.

1. Fundamentos Teóricos y Conceptuales

1.1 El Teorema de Bayes: Base Teórica

El pilar fundamental de las redes bayesianas descansa en el Teorema de Bayes, una fórmula matemática que permite actualizar nuestras creencias a medida que obtenemos nueva evidencia. Este teorema se expresa matemáticamente como:

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)} = \frac{P(X|Y)P(Y)}{\sum_i P(X|Y_i)P(Y_i)}$$

Esta expresión, aunque aparentemente simple, encapsula un principio profundo: la capacidad de actualizar nuestro conocimiento probabilístico basándonos en nueva información. Esta actualización de creencias es fundamental en sistemas que deben tomar decisiones en entornos inciertos.

1.2 Conceptos Fundamentales

1.2.1 Independencia Condicional

La independencia condicional es un concepto crucial que permite simplificar significativamente las representaciones probabilísticas. Cuando dos eventos son condicionalmente independientes dado un tercero, podemos tratarlos de manera separada, reduciendo la complejidad computacional de nuestros cálculos.

Las ecuaciones que representan la independencia condicional son:

$$P(X, Y|Z) = P(X|Z)P(Y|Z) \\ P(X|Y, Z) = P(X|Z)$$

1.2.2 Regla de la Cadena

La regla de la cadena es una herramienta útil para descomponer probabilidades conjuntas en productos de probabilidades condicionales:

$$P(X, Y) = P(Y|X)P(X) \\ P(X, Y, Z) = P(Z|X, Y)P(Y|X)P(X)$$

2. Estructura y Representación de Redes Bayesianas

2.1 Definición Formal

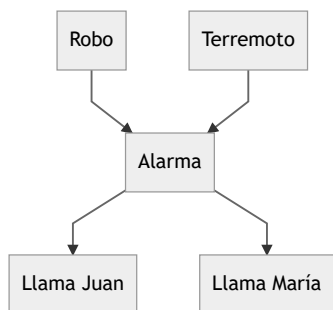
Una red bayesiana es un grafo dirigido acíclico (DAG) donde:

- Nodos:** Representan variables aleatorias.
- Aristas:** Indican dependencias directas entre variables.
- Tablas de Probabilidad Condicional (CPTs):** Cada nodo tiene asociada una CPT que cuantifica las relaciones probabilísticas con sus padres en el grafo.

Este marco permite representar de manera compacta distribuciones de probabilidad conjuntas.

2.2 Ejemplo Detallado: Sistema de Alarma

Consideremos el ejemplo clásico de un sistema de alarma que puede ser activado por un robo o un terremoto. Dos vecinos, Juan y María, pueden llamar para reportar la alarma.



Tablas de Probabilidad Condicional (CPTs)

Probabilidad a priori:

- $P(\text{Robo}) = 0.001$
- $P(\text{Terremoto}) = 0.002$

Probabilidad condicional de la Alarma dado Robo y Terremoto:

Robo	Terremoto (T)	$P(\text{Alarma} \mid R, T)$
V	V	0.95
V	F	0.94
F	V	0.29
F	F	0.001

Probabilidad condicional de que Juan y María llamen dado que la alarma está sonando:

- $P(\text{Llama Juan} \mid \text{Alarma})$:
 - Alarma verdadera: 0.90
 - Alarma falsa: 0.05
- $P(\text{Llama María} \mid \text{Alarma})$:
 - Alarma verdadera: 0.70
 - Alarma falsa: 0.01

Este ejemplo captura las relaciones causales entre eventos (por ejemplo, un robo puede activar la alarma), y cuantifica estas relaciones mediante tablas de probabilidad condicional.

3. Métodos de Inferencia y Razonamiento

La verdadera potencia de las redes bayesianas se manifiesta en su capacidad para realizar inferencias y actualizar creencias basadas en nueva evidencia.

3.1 Inferencia Exacta

3.1.1 Método de Enumeración

El método de enumeración, aunque computacionalmente costoso, proporciona resultados precisos calculando:

$$P(X|e) = \alpha P(X, e) = \alpha \sum_y P(X, e, y)$$

Donde:

- (X) : Variable(s) de interés.
- (e) : Evidencia observada.
- (y) : Todas las demás variables no observadas.
- (α) : Factor de normalización para asegurar que las probabilidades sumen 1.

Ejemplo Práctico:

Calcular $P(\text{Robo} \mid \text{Llama Juan}=V, \text{Llama María}=V)$:

Para $(\text{Robo} = V)$:

$$P(r, j = V, m = V) = P(r) \sum_t P(t) P(a|r, t) P(j|a) P(m|a)$$

Computando los términos:

- ($P@ = 0.001$)
- ($P(t) = \{0.002, 0.998\}$)
- ($P(a|r, t)$) según la tabla anterior.
- ($P(j|a=V) = 0.9$), ($P(j|a=F) = 0.05$)
- ($P(m|a=V) = 0.7$), ($P(m|a=F) = 0.01$)

Entonces,

$$\text{resultado} = P(r) * [P(t=V) * P(a|r, t=V) + P(t=F) * P(a|r, t=F)] * P(j|a) * P(m|a)$$

Este cálculo se repetirá para ($\text{Robo} = F$), y luego se normalizarán los resultados para obtener la probabilidad buscada.

3.2 Inferencia Aproximada

Debido a la complejidad computacional de la inferencia exacta en redes grandes, se emplean métodos de inferencia aproximada.

3.2.1 Muestreo Directo

1. **Generar N muestras aleatorias** de acuerdo con las distribuciones de probabilidad de la red.
2. **Contar** las muestras que coinciden con la evidencia y el evento de interés.
3. **Estimar la probabilidad** como la proporción de muestras favorables respecto al total.

Ejemplo:

Si generamos 1000 muestras y 284 de ellas cumplen con el evento de interés dado la evidencia, la probabilidad estimada es:

```
probabilidad = muestras_positivas / muestras_totales
# probabilidad = 284 / 1000 = 0.284
```

3.2.2 Muestreo por Rechazo

Mejora el muestreo directo al descartar muestras que no sean consistentes con la evidencia observada.

Algoritmo:

1. Generar muestras aleatorias.
2. **Descartar** aquellas que no coincidan con la evidencia.
3. Calcular la probabilidad con las muestras restantes.

3.3 Otros Métodos de Muestreo

- **Muestreo de Gibbs:** Un método de cadenas de Markov que actualiza iterativamente las variables no evidentes.

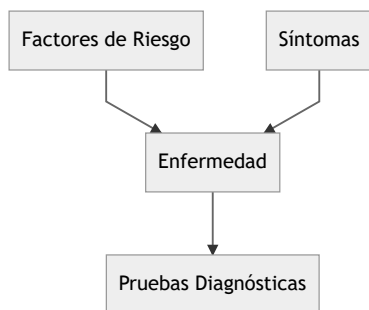
4. Aplicaciones Prácticas y Consideraciones

Las redes bayesianas encuentran aplicaciones en diversos campos:

4.1 Diagnóstico Médico

Modelan relaciones entre síntomas, enfermedades y factores de riesgo, ayudando en el diagnóstico y pronóstico.

Ejemplo: Red de Diagnóstico Médico



4.2 Sistemas de Recomendación

Capturan preferencias de usuarios y patrones de comportamiento para realizar recomendaciones personalizadas.

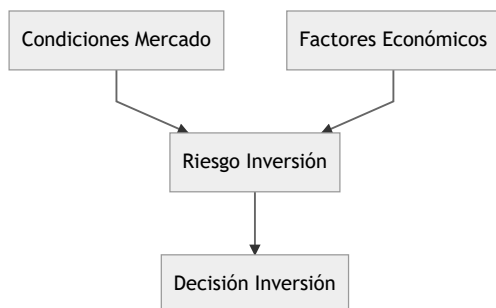
Ejemplo: Sistema de Recomendación

Variables:

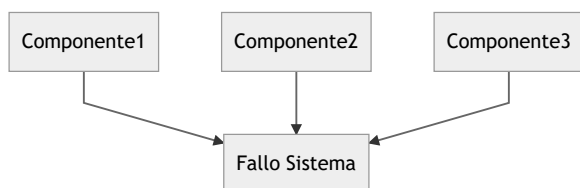
- **Preferencias_Usuario**
- **Historial_Compras**
- **Contexto_Temporal**
- **Recomendación_Final**

4.3 Análisis de Riesgos Financieros

Evalúan probabilidades de eventos adversos en proyectos o sistemas complejos.



4.4 Diagnóstico de Fallos en Sistemas



5. Ventajas y Limitaciones

5.1 Ventajas

1. **Representación Intuitiva de Causalidad:** Las redes bayesianas permiten modelar relaciones causales de manera natural.
2. **Manejo Eficiente de Incertidumbre:** Integran evidencia y actualizan probabilidades de forma coherente.
3. **Capacidad de Actualización:** Permiten incorporar nueva evidencia y ajustar las creencias en consecuencia.
4. **Interdisciplinariedad:** Aplicables en diversos campos, desde medicina hasta finanzas.

5.2 Limitaciones

1. **Complejidad Computacional:** La inferencia exacta en redes grandes puede ser intratable.
2. **Datos Precisos Requeridos:** Necesitan estimaciones precisas de probabilidades condicionales, lo cual puede ser difícil en la práctica.

6. Implementación Práctica

6.1 Ejemplo de Código en Python (usando pgmpy)

```
from pgmpy.models import BayesianNetwork
from pgmpy.factors.discrete import TabularCPD

# Definir la estructura del modelo
model = BayesianNetwork([
    ('Robo', 'Alarma'),
    ('Terremoto', 'Alarma'),
    ('Alarma', 'Llama_Juan'),
    ('Alarma', 'Llama_Maria')
])

# Definir las tablas de probabilidad condicional (CPDs)
cpd_robo = TabularCPD(variable='Robo', variable_card=2, values=[[0.999], [0.001]])
cpd_terremoto = TabularCPD(variable='Terremoto', variable_card=2, values=[[0.998], [0.002]])

cpd_alarma = TabularCPD(
    variable='Alarma', variable_card=2,
    values=[
        [0.001, 0.29, 0.94, 0.95], # Alarma = Falso
        [0.999, 0.71, 0.06, 0.05]  # Alarma = Verdadero
    ],
    evidence=['Robo', 'Terremoto'],
    evidence_card=[2, 2]
)

cpd_llama_juan = TabularCPD(
    variable='Llama_Juan', variable_card=2,
    values=[[0.05, 0.9], [0.95, 0.1]],
    evidence=['Alarma'],
    evidence_card=[2]
)

cpd_llama_maria = TabularCPD(
    variable='Llama_Maria', variable_card=2,
    values=[[0.01, 0.7], [0.99, 0.3]],
    evidence=['Alarma'],
    evidence_card=[2]
)

# Añadir CPDs al modelo
model.add_cpds(cpd_robo, cpd_terremoto, cpd_alarma, cpd_llama_juan, cpd_llama_maria)

# Comprobar si el modelo es válido
assert model.check_model()
```

6.2 Inferencia con el Modelo

```
from pgmpy.inference import VariableElimination

infer = VariableElimination(model)

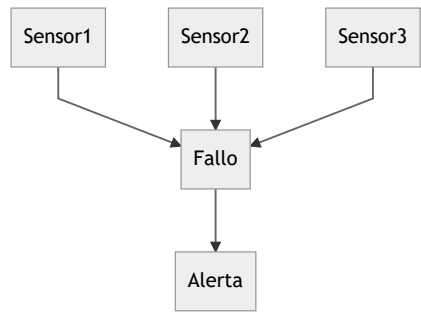
# Calcular P(Robo | Llama_Juan=Verdadero, Llama_Maria=Verdadero)
posterior_robo = infer.query(
    variables=['Robo'],
    evidence={'Llama_Juan': 1, 'Llama_Maria': 1}
)

print(posterior_robo)
```

7. Casos de Estudio

7.1 Sistema de Diagnóstico de Fallos

En sistemas complejos, identificar la causa raíz de un fallo es crucial. Las redes bayesianas pueden modelar las relaciones probabilísticas entre componentes y fallos.



7.2 Previsión Meteorológica

Modelando influencias entre variables climáticas para predecir condiciones futuras.

Variables:

- **Presión Atmosférica**
- **Temperatura**
- **Humedad**
- **Precipitación**

8. Tendencias Futuras y Desarrollos

8.1 Integración con Aprendizaje Profundo

- **Redes Bayesianas Profundas:** Combinación con arquitecturas de redes neuronales profundas para manejar datos complejos y de alta dimensionalidad.
- **Inferencia Variacional:** Métodos que aproximan distribuciones posteriores complejas, facilitando la inferencia en modelos grandes.
- **Modelos Híbridos:** Integración de modelos probabilísticos gráficos con técnicas de aprendizaje automático.

8.2 Aplicaciones Emergentes

1. **Internet de las Cosas (IoT):** Gestión de incertidumbre en redes de sensores y dispositivos conectados.
2. **Sistemas Autónomos:** Toma de decisiones bajo incertidumbre en vehículos autónomos y robots.
3. **Medicina Personalizada:** Modelado de datos genómicos y clínicos para tratamientos individualizados.

Conclusiones y Perspectivas Futuras

Las redes bayesianas continúan siendo una herramienta fundamental en inteligencia artificial, combinando elegancia teórica con aplicabilidad práctica. Su capacidad para manejar incertidumbre de manera sistemática y para combinar conocimiento experto con datos empíricos las hace invaluable en numerosos campos de aplicación.

Los avances en técnicas de inferencia aproximada y la creciente disponibilidad de datos prometen expandir aún más su utilidad. Sin embargo, el desafío permanente es equilibrar la complejidad del modelo con la tractabilidad computacional, manteniendo la precisión necesaria para aplicaciones prácticas.

Mientras la inteligencia artificial continúa evolucionando, las redes bayesianas seguirán siendo esenciales para modelar y razonar sobre la incertidumbre en sistemas inteligentes, contribuyendo significativamente al desarrollo de sistemas de toma de decisiones más robustos y confiables.

Razonamiento Probabilístico en el Tiempo: Filtros de Kalman y Localización

Introducción al Razonamiento y Representación del Conocimiento en el Tiempo

El **razonamiento probabilístico en el tiempo** es fundamental para mantener y actualizar el conocimiento sobre entornos que evolucionan dinámicamente. En inteligencia artificial y robótica, es común enfrentarse a sistemas que cambian con el tiempo y donde las mediciones pueden ser ruidosas o inciertas.

Por qué es Importante

- **Evolución del Entorno:** En muchas aplicaciones, necesitamos mantener nuestro conocimiento actualizado sobre el estado del mundo a medida que cambia.
- **Incertidumbre Creciente:** Sin nuevas evidencias, nuestra confianza en el estado actual disminuye con el tiempo.
- **Reducción de Incertidumbre:** Al incorporar nuevas evidencias, podemos reducir la incertidumbre sobre variables específicas del entorno.

Aplicaciones Comunes

- **Modelos Físicos:** Predicción del clima, corrientes marinas, movimientos sísmicos.
- **Seguimiento de Objetos:** Localización y navegación de robots móviles.
- **Sistemas Económicos:** Análisis y predicción de mercados financieros.
- **Sistemas Dinámicos:** Cualquier sistema que evolucione temporalmente y requiera predicciones.

Enfoques para el Razonamiento Probabilístico en el Tiempo

Existen dos enfoques principales:

1. **Modelos de Markov y Modelos Ocultos de Markov (HMMs).**
2. **Filtros de Kalman y su extensión, Filtros de Kalman Extendidos (EKF).**

Ambos enfoques son casos particulares de **Redes Bayesianas Dinámicas**, pero en este resumen nos centraremos en los **Filtros de Kalman**, ampliamente utilizados en robótica para la localización de robots móviles y en la resolución del problema de SLAM (Simultaneous Localization and Mapping).

Revisión de Distribuciones Gaussianas

Antes de adentrarnos en los filtros de Kalman, es importante revisar las **distribuciones gaussianas**, ya que los filtros de Kalman asumen modelos gaussianos en su funcionamiento.

Distribución Normal Univariada

Para una variable aleatoria continua x , la distribución normal univariada con media μ y varianza σ^2 se define como:

$$g(x) \sim \mathcal{N}(\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2} \left(\frac{x-\mu}{\sigma} \right)^2}$$

Distribución Normal Multivariada

Para un vector aleatorio \mathbf{x} de dimensión n , con media $\boldsymbol{\mu}$ y matriz de covarianza Σ , la distribución normal multivariada es:

$$g(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^n \det(\Sigma)}} \exp \left(-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^\top \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right)$$

El Problema de la Localización de Robots Móviles

La localización de robots móviles es un ejemplo clásico en inteligencia artificial donde el razonamiento probabilístico en el tiempo es esencial.

Descripción del Problema

- **Mapa del Entorno:** Es imprescindible disponer de un mapa del entorno en el que se moverá el robot.
- **Posición Inicial:**
 - **Conocida:** Se trata de un problema de seguimiento (**tracking**), es decir, localización local.
 - **Desconocida:** El robot debe determinar su ubicación sin conocimiento previo (localización global).
- **Sensores:** El robot cuenta con sensores para realizar observaciones del entorno.
- **Incertidumbre:**
 - **Movimiento:** Existen errores (ruido) en el movimiento del robot, incluso si se ejecuta un comando preciso.
 - **Sensores:** Las lecturas de los sensores también son imprecisas debido al ruido.

Representación de la Incertidumbre

- **Pose del Robot:** En un plano bidimensional, la pose se representa como (X, Y, ϕ) , donde ϕ es la orientación.
- **Modelado Gaussiano:** La incertidumbre se representa mediante una distribución gaussiana sobre cada variable.

Filtro de Kalman

El **Filtro de Kalman** es un algoritmo recursivo que permite estimar el estado de un sistema dinámico lineal en presencia de ruido gaussiano.

Componentes del Filtro de Kalman

El filtro se define mediante matrices que modelan la evolución del estado y las observaciones:

1. **Matriz de Transición de Estado (A_t):**
 - Es una matriz $n \times n$ que describe cómo el estado cambia de $t - 1$ a t en ausencia de comandos de control o ruido.
 - Modela la dinámica del sistema.
2. **Matriz de Control (B_t):**

- Es una matriz $n \times l$ que describe cómo los comandos de control \mathbf{u}_t afectan al estado.
- Permite incorporar acciones o movimientos realizados.

3. Matriz de Observación (C_t):

- Es una matriz $k \times n$ que mapea el estado \mathbf{x}_t al espacio de las observaciones \mathbf{z}_t .
- Modela cómo el estado se refleja en las mediciones.

4. Ruido de Proceso (ϵ_t):

- Representa la incertidumbre en el modelo de movimiento.
- Se asume que ϵ_t sigue una distribución normal con covarianza R_t .

5. Ruido de Observación (δ_t):

- Representa la incertidumbre en las mediciones de los sensores.
- Se asume que δ_t sigue una distribución normal con covarianza Q_t .

Modelo del Sistema

El estado y las observaciones evolucionan según:

• Evolución del Estado:

$$\mathbf{x}_t = A_t \mathbf{x}_{t-1} + B_t \mathbf{u}_t + \epsilon_t$$

• Modelo de Observación:

$$\mathbf{z}_t = C_t \mathbf{x}_t + \delta_t$$

Ciclo Predicción-Corrección

El filtro de Kalman opera en dos pasos iterativos:

1. Predicción:

- Estima el estado posterior dado el comando de control y el modelo de transición.
- **Predicción de la media:**

$$\hat{\boldsymbol{\mu}}_t = A_t \boldsymbol{\mu}_{t-1} + B_t \mathbf{u}_t$$

- **Predicción de la covarianza:**

$$\hat{\Sigma}_t = A_t \Sigma_{t-1} A_t^\top + R_t$$

- La incertidumbre aumenta debido al ruido en el movimiento.

2. Corrección (Actualización):

- Actualiza la estimación del estado incorporando las nuevas observaciones.
- **Cálculo de la Ganancia de Kalman (K_t):**

$$K_t = \hat{\Sigma}_t C_t^\top (C_t \hat{\Sigma}_t C_t^\top + Q_t)^{-1}$$

- **Actualización de la media:**

$$\mu_t = \hat{\mu}_t + K_t(\mathbf{z}_t - C_t\hat{\mu}_t)$$

- **Actualización de la covarianza:**

$$\Sigma_t = (\mathbf{I} - K_t C_t) \hat{\Sigma}_t$$

- La incertidumbre disminuye gracias a la información adicional de las observaciones.

Características del Filtro de Kalman

- **Óptimo para Sistemas Lineales Gaussianos:** Provee estimaciones óptimas cuando el sistema y el modelo de observación son lineales y los ruidos son gaussianos.
- **Forma Cerrada:** Las distribuciones de probabilidad se mantienen gaussianas en cada iteración.
- **Eficiente Computacionalmente:** El costo es polinomial respecto a las dimensiones del estado y las observaciones.

Filtro de Kalman Extendido (EKF)

Para sistemas no lineales, el filtro de Kalman estándar no es aplicable. El **Filtro de Kalman Extendido (EKF)** es una extensión que permite trabajar con modelos no lineales.

Problema de No Linealidad

Muchos sistemas reales, especialmente en robótica, no pueden ser descritos mediante modelos lineales. Por ejemplo, el movimiento de un robot móvil a menudo es mejor representado por funciones no lineales debido a la naturaleza de su dinámica y sensores.

Modelo del Sistema No Lineal

- **Evolución del Estado:**

$$\mathbf{x}_t = \mathbf{g}(\mathbf{u}_t, \mathbf{x}_{t-1}) + \boldsymbol{\varepsilon}_t$$

- **Modelo de Observación:**

$$\mathbf{z}_t = \mathbf{h}(\mathbf{x}_t) + \delta_t$$

Donde \mathbf{g} y \mathbf{h} son funciones no lineales.

Linearización mediante Expansión de Taylor

El EKF aproxima las funciones no lineales \mathbf{g} y \mathbf{h} mediante su **expansión en serie de Taylor de primer orden** alrededor de la estimación actual.

- **Jacobianos:**

- **Matriz Jacobiana del Modelo de Transición ((\mathbf{G}_t)):**

$$G_t = \left. \frac{\partial \mathbf{g}}{\partial \mathbf{x}} \right|_{\mathbf{x}=\hat{\mu}_{t-1}, \mathbf{u}=\mathbf{u}_t}$$

- **Matriz Jacobiana del Modelo de Observación ((\mathbf{H}_t)):**

$$H_t = \left. \frac{\partial \mathbf{h}}{\partial \mathbf{x}} \right|_{\mathbf{x}=\hat{\mu}_t}$$

Algoritmo del EKF

El algoritmo sigue un proceso similar al filtro de Kalman estándar, pero utilizando las aproximaciones lineales.

Pasos del EKF

1. Predicción del Estado:

- $\hat{\mu}_t = \mathbf{g}(\mathbf{u}_t, \mu_{t-1})$
- $\hat{\Sigma}_t = G_t \Sigma_{t-1} G_t^\top + R_t$

2. Cálculo de la Ganancia de Kalman:

- $K_t = \hat{\Sigma}_t H_t^\top (H_t \hat{\Sigma}_t H_t^\top + Q_t)^{-1}$

3. Actualización del Estado:

- $\mu_t = \hat{\mu}_t + K_t (\mathbf{z}_t - \mathbf{h}(\hat{\mu}_t))$

4. Actualización de la Covarianza:

- $\Sigma_t = (\mathbf{I} - K_t H_t) \hat{\Sigma}_t$

Notas Importantes

- Aproximación Local:** La linearización es válida en un entorno cercano a la estimación actual.
- No Óptimo:** El EKF no es óptimo para sistemas no lineales, pero suele proporcionar buenos resultados en la práctica.
- Sensibilidad:** La precisión del EKF depende de qué tan lineal sea el sistema en la región de operación.

Visualización de la Linearización

La siguiente gráfica ilustra cómo la expansión de Taylor aproxima la función no lineal (\mathbf{g}) mediante una línea tangente en el punto de operación:

Localización Mediante EKF

La **localización EKF** es una aplicación práctica del filtro de Kalman extendido para mantener y actualizar la estimación de la pose de un robot móvil.

Objetivos

- Mantenimiento de la Pose:** Mantener una representación probabilística (gaussiana) de la pose del robot a lo largo del tiempo.
- Media y Covarianza:**
 - La media (μ_t) indica la posición más probable del robot.
 - La covarianza (Σ_t) refleja la incertidumbre asociada a esa estimación.

Funcionalidad

- Predicción:** Basada en el modelo de movimiento y los comandos ejecutados.

- **Corrección:** Incorporando las observaciones de los sensores para ajustar la estimación.

Caso de Posición Inicial Conocida

- **Seguimiento (Tracking):** Si la posición inicial es conocida, el EKF puede mantener la estimación de la pose del robot a medida que se mueve.
- **Reducción de Incertidumbre:** Las observaciones periódicas permiten reducir la incertidumbre en la estimación.

Localización Global

Cuando la posición inicial del robot es desconocida, la localización mediante EKF no es suficiente debido a que la distribución inicial debería ser una gaussiana con varianza infinita, lo cual es impracticable. En estos casos, se utilizan algoritmos de **localización de Markov** o filtros de partículas.

Algoritmo de Localización de Markov

- **Distribución de Creencia (bel):** Mantiene una función de probabilidad sobre todas las poses posibles.
- **Actualización Basada en Observaciones:** A medida que el robot obtiene nuevas observaciones, actualiza la distribución de creencia para reducir la incertidumbre.
- **Ejemplo 1D:** En un entorno unidimensional, la distribución de creencia puede visualizarse como una función sobre la línea, ajustándose en cada paso temporal.

 Localización de Markov 1D

Conclusiones

El razonamiento probabilístico en el tiempo es esencial para sistemas que requieren mantener un conocimiento actualizado y preciso en entornos dinámicos e inciertos. Los filtros de Kalman y su extensión, los filtros de Kalman extendidos, proporcionan herramientas poderosas para estimar el estado de un sistema cuando las dinámicas y observaciones pueden ser modeladas lineal o no linealmente.

Ventajas del Filtro de Kalman

- **Estimación Óptima:** En sistemas lineales gaussianos, provee estimaciones óptimas del estado.
- **Eficiencia Computacional:** Es computacionalmente eficiente y adecuado para aplicaciones en tiempo real.
- **Extensibilidad:** El EKF permite manejar sistemas no lineales mediante aproximaciones locales.

Limitaciones

- **Linealidad:** El filtro de Kalman estándar no es adecuado para sistemas no lineales.
- **Aproximaciones en EKF:** La linearización en el EKF puede introducir errores si el sistema es altamente no lineal.
- **Inicialización:** Requiere estimaciones iniciales de la media y la covarianza, lo cual puede ser un desafío en problemas de localización global.

Aplicaciones en Robótica

- **Localización de Robots Móviles:** Seguimiento preciso de la pose del robot en entornos conocidos.
- **SLAM (Simultaneous Localization and Mapping):** Construcción del mapa del entorno mientras se estima la pose del robot.
- **Navegación Autónoma:** Permite a los robots tomar decisiones informadas basadas en estimaciones precisas de su estado.

Representación de Conocimiento en Robótica Móvil

Introducción

La **robótica móvil autónoma** es un campo de la inteligencia artificial y la ingeniería que se enfoca en el diseño y desarrollo de robots capaces de desplazarse por entornos dinámicos sin intervención humana directa. La capacidad de estos robots para **conocer y entender su entorno** es fundamental para garantizar su operatividad y seguridad.

Objetivos Principales de los Robots Móviles

- Desplazamiento Seguro:** Los robots deben moverse evitando colisiones, caídas y cualquier acción que pueda provocar daños a sí mismos, a las personas o al entorno.
- Interacción con el Entorno:** Los robots móviles necesitan interactuar con su entorno para realizar tareas específicas, como buscar víctimas en situaciones de emergencia, guiar personas o ayudar a quienes lo necesiten.

Importancia del Conocimiento del Entorno

Para cumplir con estos objetivos, es **esencial que los robots móviles tengan un conocimiento preciso del entorno en el que operan**. Este conocimiento se adquiere y actualiza gracias a una variedad de sensores que proporcionan información sobre:

- El propio estado del robot:** posición, orientación, velocidad, etc.
- El entorno circundante:** obstáculos, características geográficas, objetos de interés, etc.

Principales Tareas en Robótica Móvil

A Nivel Local

- Navegación:** Desplazamiento eficiente y seguro en el entorno inmediato.
- Evitación de Obstáculos:** Detección y evasión de obstáculos para prevenir colisiones.

A Nivel Global

- Localización:** Determinar la posición del robot en un mapa global.
- Planificación de Trayectorias:** Definir rutas óptimas para alcanzar objetivos específicos.
- Construcción de Mapas:** Generar representaciones del entorno para fines de navegación y planificación.

Aplicaciones Comunes

- Búsqueda y Rescate:** Localización de víctimas en situaciones de emergencia.
- Guiado de Personas:** Asistencia en entornos como aeropuertos, hospitales o centros comerciales.
- Asistencia Personal:** Acompañamiento y ayuda a personas con necesidades especiales.

Sensores en Robótica Móvil

Los **sensores son los dispositivos que permiten a los robots móviles percibir su propio estado y el entorno que les rodea**. Se dividen principalmente en:

- **Sensores Internos:** Proporcionan información sobre el estado interno del robot.
- **Sensores Externos:** Capturan datos del entorno externo al robot.

Sensores Internos

Los sensores internos son esenciales para el **control y seguimiento del movimiento** del robot.

Odometría

La **odometría** es el método que estima la posición y orientación del robot basándose en la información de los sensores internos. Los componentes clave son:

- **Encoders:**
 - **Encoders Rotatorios:** Miden la rotación de las ruedas o ejes del robot.
 - **Funcionamiento:** Al contar las pulsaciones o cambios de estado, se determina cuánto ha girado cada rueda.
- **Sensores Inerciales:**
 - **Giroscopios:** Miden la velocidad angular o cambios en la orientación.
 - **Acelerómetros:** Miden cambios en la velocidad lineal.

Problemas Asociados

- **Acumulación de Errores:** Los pequeños errores en la medición se acumulan con el tiempo, lo que lleva a **derivas significativas en la estimación de la posición**.
- **Ruido:** Los sensores internos son propensos al ruido y a las imprecisiones debido a factores como vibraciones, deslizamiento de ruedas, y condiciones del terreno.

Sensores Inerciales (IMU)

Una **Unidad de Medición Inercial (IMU)** es un dispositivo que combina giroscopios y acelerómetros para proporcionar una estimación más precisa del movimiento.

- **Mediciones en Tres Ejes:** Proporciona datos de aceleración y rotación en los ejes X, Y y Z.
- **Aplicaciones:** Navegación inercial, estabilización y control de actitud del robot.

Sensores Externos

Los sensores externos permiten al robot **interactuar y entender su entorno**. Se clasifican en:

- **Sensores de Posicionamiento:** Determinan la posición del robot en relación con un sistema de referencia externo.
- **Sensores de Rango:** Miden la distancia a objetos y obstáculos.
- **Sensores de Visión:** Capturan imágenes del entorno para su análisis y procesamiento.

Sensores de Posicionamiento

Posicionadores Globales

- **GPS (Sistema de Posicionamiento Global):**

- **Funcionamiento:** Utiliza señales de satélites para determinar la posición en latitud, longitud y altitud.
- **Principio Básico:** Calcula la distancia a varios satélites en órbita midiendo el tiempo que tardan las señales en llegar.

$$\text{distancia} = \text{velocidad de la señal} \times \text{tiempo de viaje}$$

- **Precisión:**

- **Antiguamente:** Existía un error aleatorio introducido (Selective Availability) que limitaba la precisión.
- **Actualidad:** La precisión típica es de **2 a 3 metros**.
- **GPS Diferencial (DGPS):** Mejora la precisión utilizando una estación base con posición conocida para corregir errores.

- **Limitaciones:**

- **No Funciona en Interiores:** Las señales de GPS no penetran edificios u obstáculos sólidos.
- **Obstrucciones:** Edificios altos, árboles densos o estructuras subterráneas impiden el funcionamiento adecuado.

Posicionadores Locales

- **Balizas Activas (Active Beacons):**

- **Funcionamiento:** Utilizan señales emitidas por dispositivos fijos en el entorno (balizas) para determinar la posición del robot.
- **Aplicaciones:** Entornos interiores donde el GPS no es viable.

- **Redes Inalámbricas:**

- **Wi-Fi:** Uso de puntos de acceso y la intensidad de la señal para estimar la posición.
- **Redes Móviles:** Basado en la triangulación de señales de torres de telefonía.

Sensores de Rango

Los sensores de rango son **críticos para la detección de obstáculos y el mapeo del entorno**. Miden la distancia entre el sensor y los objetos circundantes.

Tecnologías Utilizadas

1. **Apertura de la Señal:**

- **Principio:** Medición basada en el patrón de propagación de la señal.
- **Limitaciones:** Menor precisión y alcance limitado.

2. **Tiempo de Vuelo (ToF - Time of Flight):**

- **Directo:**

- **Principio:** Mide el tiempo que tarda una señal en viajar desde el emisor al objeto y regresar al receptor.

- **Ecuación Básica:**

$$\text{distancia} = \frac{\text{velocidad} \times \text{tiempo}}{2}$$

- **Aplicaciones:** Sensores de ultrasonidos (sonar), radares.

- **Indirecto:**

- **Principio:** En lugar de medir el tiempo de vuelo, se mide el desplazamiento de fase entre la señal emitida y la recibida.
- **Aplicaciones:** Sensores láser de alta precisión (LIDAR).

Sensores Comunes

1. Sensores Infrarrojos:

- **Características:** Económicos, ideales para distancias cortas.
- **Limitaciones:** Sensibles a las condiciones de iluminación y propiedades de las superficies.

2. Sonar (SOund Navigation And Ranging):

- **Funcionamiento:**

1. **Emisión:** El sensor emite un haz de ultrasonidos (típicamente a 50 kHz) con una apertura de 15 grados.
2. **Espera:** Cambia a modo receptor y mide el tiempo hasta recibir el eco.
3. **Cálculo de Distancia:** Utilizando la velocidad del sonido en el aire (aproximadamente 340 m/s).

- **Ventajas:** Económicos y fáciles de implementar.

- **Limitaciones:**

- **Ángulo de Incidencia:** Superficies inclinadas pueden desviar el haz y producir lecturas erróneas.
- **Amplitud del Cono:** La apertura amplia puede causar problemas de resolución espacial.
- **Dobles Rebotes:** Reflexiones múltiples pueden confundirse con objetos más cercanos.

3. Sensores Láser:

- **LIDAR (Light Detection and Ranging):**

- **Funcionamiento:** Utilizan láseres para medir distancias con alta precisión, midiendo el tiempo de vuelo indirecto.
- **Características:**
 - **Alta Resolución:** Capaz de detectar detalles finos en el entorno.
 - **Alcance:** Puede abarcar desde pocos metros hasta varios kilómetros.
- **Aplicaciones:** Mapeo 3D, vehículos autónomos, drones.

- **Ejemplo:** LIDAR de 360 grados que gira para escanear el entorno completo alrededor del robot.

- **Interferencias Ambientales:** Condiciones climáticas, partículas en el aire o superficies reflectantes pueden afectar las mediciones.
- **Limitaciones Físicas:** Alcance limitado, ángulos muertos y resoluciones limitadas en función de la tecnología utilizada.

Sensores de Visión

Los sensores de visión proporcionan **información rica y detallada del entorno** mediante la captura de imágenes.

Tipos de Sensores

- **Cámaras CCD (Charge-Coupled Device):** Ofrecen alta calidad de imagen, utilizadas en aplicaciones que requieren precisión.
- **Cámaras CMOS (Complementary Metal-Oxide-Semiconductor):** Más económicas y consumen menos energía, ampliamente utilizadas en dispositivos móviles.

Características

- **Sistema de Cámara Oscura (Pin-Hole):**
 - **Principio:** La luz entra a través de una pequeña apertura y se proyecta invertida en el sensor.
 - **Limitación:** Se pierde información de profundidad, lo que hace necesario el uso de técnicas adicionales para obtenerla.
- **Discretización:** La imagen se divide en píxeles, lo que implica una pérdida de información continua pero permite su procesamiento digital.

Aplicaciones

- **Localización y Mapeo Visual:** Utilizando referencias visuales para determinar la posición y orientación del robot.
- **Reconocimiento de Objetos y Entornos:** Identificación y clasificación de objetos, señales, obstáculos y características del entorno.
- **Interacción con el Entorno:** Detección de personas, seguimiento de movimientos y otras interacciones avanzadas.

Espectro de Captura

- **Espectro Visible:** Imágenes en colores o en escala de grises que representan lo que el ojo humano puede percibir.
- **Espectro No Visible:**
 - **Infrarrojo:** Captura información basada en el calor emitido por objetos, útil en condiciones de poca luz o para detectar seres vivos.
 - **Ultravioleta:** Utilizado en aplicaciones específicas como detección de materiales o sustancias particulares.

Integración y Uso de Sensores en Robótica Móvil

La **combinación de diferentes sensores** permite a los robots móviles tener una percepción más completa y confiable del entorno. Algunos aspectos clave son:

Fusión de Datos

- **Complementariedad:** Los datos de diferentes sensores pueden complementarse para superar las limitaciones individuales.
- **Redundancia:** La información redundante de múltiples sensores aumenta la confiabilidad de las mediciones.
- **Algoritmos de Fusión:** Técnicas como filtros de Kalman, filtros de partículas y redes neuronales se utilizan para integrar los datos sensoriales.

Navegación y Localización

- **Sistemas SLAM (Simultaneous Localization and Mapping):** Los robots construyen mapas del entorno mientras se localizan en él, haciendo uso intensivo de sensores de rango y visión.
- **Planificación de Rutas:** Utilizan la información sensorial para generar trayectorias eficientes y seguras.

Desafíos en la Integración Sensorial

- **Sincronización:** Alinear temporalmente los datos de sensores diferentes es crucial para una interpretación correcta.
- **Calibración:** Los sensores deben estar correctamente calibrados para asegurar la precisión en las mediciones y evitar errores sistemáticos.
- **Procesamiento en Tiempo Real:** El procesamiento de grandes cantidades de datos sensoriales requiere algoritmos eficientes y capacidad computacional adecuada.

Conclusiones

La representación y manejo del conocimiento en robótica móvil dependen en gran medida de los **sensores que proporcionan información sobre el estado interno del robot y su entorno**. La correcta selección, integración y procesamiento de los datos de estos sensores son fundamentales para el desarrollo de robots móviles autónomos eficaces y seguros.

Importancia de la Sensórica

- **Percepción Precisa:** Es la base para todas las tareas de navegación, localización, evasión de obstáculos y toma de decisiones.
- **Interacción Inteligente:** Permite a los robots comprender y adaptarse al entorno, interactuando de manera efectiva con objetos y personas.

Futuras Tendencias

- **Sensores Más Avanzados:** Desarrollo de sensores más precisos, económicos y con mayor capacidad para operar en diferentes condiciones ambientales.
 - **Integración Sensorial Más Profunda:** Uso de inteligencia artificial y aprendizaje automático para mejorar la fusión y el análisis de datos sensoriales.
 - **Mayor Autonomía:** Avances en la sensórica y la representación del conocimiento permitirán robots móviles con niveles más altos de autonomía y adaptabilidad.
-

Representación Métrica del Entorno en Robótica Móvil

Introducción

La **representación métrica del entorno** es fundamental en robótica móvil y en sistemas de inteligencia artificial que interactúan con espacios físicos. Los **mapas métricos** capturan las propiedades geométricas del entorno, proporcionando información detallada necesaria para tareas como navegación, evitación de obstáculos, localización y mapeo (SLAM).

Enfoques Principales

Existen dos enfoques principales para la representación métrica del entorno:

1. **Rejillas de Ocupación:** Dividen el espacio en una rejilla regular de celdas, donde cada celda almacena información sobre la ocupación del espacio.
2. **Representaciones Poliédricas:** Utilizan primitivas geométricas como líneas, planos y poliedros para representar el entorno de manera más compacta.

La elección del tipo de mapa depende del objetivo específico:

- **Navegación y Evitación de Obstáculos:** Se requiere información detallada del entorno, por lo que las **rejillas de ocupación** son más adecuadas.
- **Localización o Mapping:** Interesa identificar partes significativas del entorno que sean útiles para los objetivos, donde las **representaciones poliédricas** son más eficientes.

Rejillas de Ocupación

Las **rejillas de ocupación** representan el espacio como una rejilla de grano fino donde cada celda corresponde a una ubicación específica en el entorno.

Características

- **Estados de las Celdas:**
 - **Ocupado:** La celda contiene un objeto o está bloqueada.
 - **Libre:** La celda está despejada y puede ser atravesada.
 - **Desconocido:** No se ha obtenido información sobre la celda.
- **Información Probabilística:** La probabilidad de ocupación de cada celda se actualiza mediante observaciones y mediciones de los sensores.

Modelo Probabilístico

Asumiendo que la posición del robot $x_{1:t}$ y las lecturas de los sensores $z_{1:t}$ son conocidas, la probabilidad a posteriori del mapa m se obtiene como:

$$P(m|z_{1:t}, x_{1:t}) = \prod_i p(m_i|z_{1:t}, x_{1:t})$$

Donde m_i representa la celda i -ésima del mapa.

Ventajas

- **Conocimiento Detallado:** Proporcionan una representación exhaustiva del entorno, útil para navegación precisa y planificación detallada.
- **Simplicidad de Implementación:** Fácil de entender y programar.

Desventajas

- **Alto Consumo de Memoria:** Requiere almacenar información para cada celda, lo que es ineficiente en entornos de gran tamaño o dimensiones.
- **Escalabilidad Limitada:** No es práctico para áreas extensas o entornos tridimensionales complejos.

Para abordar estas limitaciones, se utilizan estructuras de datos más eficientes como **QuadTrees** y **OcTrees**.

QuadTrees

Los **QuadTrees** son estructuras de datos en forma de árbol que permiten una división no homogénea del espacio bidimensional. Aprovechan áreas contiguas con las mismas características para reducir el uso de memoria.

Funcionamiento

- **División Recursiva:** El espacio se divide en cuatro cuadrantes (de ahí "Quad") de forma recursiva.
- **Nodos:**
 - **Nodo Hoja:** Representa una región homogénea (todas las celdas tienen el mismo estado).
 - **Nodo Interno:** Tiene cuatro hijos correspondientes a los cuadrantes superior izquierdo, superior derecho, inferior izquierdo e inferior derecho.

Construcción del QuadTree

1. **Inicialización:** Se parte de una lista de puntos 2D adquiridos por los sensores del robot.
2. **Definición de Región:** Cada nodo está delimitado por dos puntos (esquina superior izquierda y esquina inferior derecha).
3. **Evaluación de Homogeneidad:**
 - Si todos los puntos dentro de la región tienen el mismo estado, el nodo es una hoja.
 - Si hay diversidad, se subdivide el nodo en cuatro cuadrantes y se repite el proceso para cada uno.

Algoritmo Simplificado

```
QuadTree(p1, p2, listaPuntos):
    Nodo = crearNodo(p1, p2)
```



```
si mismoTipo(Nodo, listaPuntos):
    Nodo.tipo = obtenerTipo(Nodo, listaPuntos)
sino:
    pMedio = puntoMedio(Nodo)
    dividir listaPuntos en cuatro listas según pMedio
    Nodo.nodoSI = QuadTree(p1, pMedio, puntosSI)
    Nodo.nodoSD = QuadTree((pMedio.x, p1.y), (p2.x, pMedio.y), puntosSD)
    Nodo.nodoII = QuadTree((p1.x, pMedio.y), (pMedio.x, p2.y), puntosII)
    Nodo.nodoID = QuadTree(pMedio, p2, puntosID)
retornar Nodo
```

Ventajas

- **Eficiencia en Memoria:** Reduce significativamente el uso de memoria en comparación con las rejillas de ocupación convencionales.
- **Adaptabilidad:** Se adapta dinámicamente a la complejidad del entorno, subdividiendo más en áreas con mayor detalle.

Aplicaciones

- **Mapas 2D:** Ideales para representar entornos bidimensionales en robótica móvil y gráficos por computadora.

OcTrees

Las **OcTrees** son la extensión tridimensional de los QuadTrees, utilizadas para representar entornos en 3D.

Características

- **División Espacial en 3D:** Cada nodo se divide en ocho octantes (de ahí “Oc”, de “Octree”).
- **Nodos:**
 - **Nodo Hoja:** Representa un volumen en el espacio con estado homogéneo.
 - **Nodo Interno:** Tiene ocho hijos correspondientes a los subvolúmenes.

Uso en Robótica Móvil

- **Mapeo 3D:** Permite almacenar mapas tridimensionales obtenidos de sensores de rango 3D, como LIDAR o cámaras estéreo.
- **Eficiencia:** Conserva memoria al representar áreas homogéneas sin necesidad de almacenar cada punto individualmente.

Representaciones Poliédricas

Las **representaciones poliédricas** utilizan primitivas geométricas (líneas, planos, poliedros) para modelar el entorno. Son útiles cuando se requiere una representación más abstracta y menos detallada.

Características

- **Primitivas Geométricas:**
 - **Líneas y Planos:** Para entornos 2D y superficies planas.
 - **Poliedros:** Representación de volúmenes en 3D.
 - **Círculos y Cilindros:** Para objetos curvos o redondeados.
- **Construcción:**
 - Puede ser construida manualmente o mediante algoritmos de ajuste a datos sensoriales.
 - Se utilizan métodos de ajuste como **mínimos cuadrados** para encontrar las primitivas que mejor se ajustan a los datos.

Proceso de Extracción de Primitivas

1. **Recolección de Datos:** Obtenemos una nube de puntos a partir de sensores como LIDAR o cámaras estéreo.
2. **Selección de Vecindad:** Para cada punto, se define una vecindad (puntos cercanos).
3. **Ajuste de Primitivas:** Se intenta ajustar una primitiva geométrica (por ejemplo, un plano) a los puntos de la vecindad.
4. **Validación:** Se verifica si el ajuste es adecuado según un criterio de error.
5. **Construcción del Mapa:** Se generan los elementos geométricos que representan el entorno.

Ventajas

- **Reducción de Datos:** Al representar grandes conjuntos de puntos con unas pocas primitivas geométricas, se reduce la cantidad de información a manejar.
- **Adecuado para Entornos Simples:** Es eficiente en entornos donde las estructuras geométricas son predominantes (edificios, habitaciones cuadradas).

Desventajas

- **Complejidad Computacional:** El proceso de ajuste puede ser costoso, especialmente en tiempo real, ya que la complejidad es $O(k \cdot n^2)$ (donde n es el número de puntos y k es el número de vecindarios).
- **Limitación en Entornos Complejos:** Dificultad para representar entornos con formas irregulares o gran variabilidad.

Kd-Trees

Los **Kd-Trees** (K-dimensional Trees) son estructuras de datos tipo árbol binario que permiten búsquedas eficientes en espacios multidimensionales.

Características

- **Nodos:**

- Cada nodo representa un punto en un espacio de (k) dimensiones.
- Los nodos dividen el espacio mediante hiperplanos perpendiculares a los ejes de coordenadas.

- **División del Espacio:**

- En cada nivel del árbol, se elige un eje de división (ciclando entre las (k) dimensiones).
- El hiperplano de división pasa por el punto asociado al nodo y divide el espacio en dos mitades.

Construcción del Kd-Tree

La construcción del árbol se realiza de forma recursiva:

1. **Inicio:** Comenzar con una lista de puntos y una profundidad inicial ($\text{depth} = 0$).

2. **Selección del Eje:**

- El eje de división es ($\text{axis} = \text{depth} \bmod k$).

3. **Ordenamiento de Puntos:** Ordenar los puntos según el eje seleccionado.

4. **Selección del Nodo:** Elegir el punto medio como el nodo actual.

5. **División de Puntos:**

- Los puntos antes del punto medio forman el subárbol izquierdo.
- Los puntos después forman el subárbol derecho.

6. **Recursión:** Aplicar el proceso recursivamente para los subárboles incrementando (depth).

Algoritmo Simplificado

```
KdTree(pointList, depth):  
    si pointList está vacío:  
        retornar null  
    axis = depth mod k  
    ordenar pointList según axis  
    median = punto medio de pointList  
    nodo = nuevo Nodo(median)  
    nodo.left = KdTree(puntos antes de median, depth + 1)  
    nodo.right = KdTree(puntos después de median, depth + 1)  
    retornar nodo
```

Operaciones Comunes

- **Búsqueda del Vecino Más Cercano:** Encontrar el punto más cercano a un punto dado.
- **Búsqueda por Rango:** Encontrar todos los puntos dentro de un rango especificado.

Complejidad

- **Construcción:** $(O(n \log n))$, donde (n) es el número de puntos.
- **Búsqueda:** En promedio $(O(\log n))$, aunque en el peor caso puede ser $(O(n))$.

Ventajas

- **Eficiencia en Búsquedas:** Permite búsquedas rápidas en espacios de alta dimensionalidad.
- **Versatilidad:** Aplicable en diversas áreas como robótica, gráficos por computadora y reconocimiento de patrones.

Aplicaciones en Robótica

- **Localización y Mapeo:** Almacenar y acceder eficientemente a puntos de interés en el entorno.
- **Planificación de Trayectorias:** Encontrar rutas óptimas evitando obstáculos representados en el espacio multidimensional.

Conclusiones

La representación métrica del entorno es crucial para que los robots móviles puedan navegar, mapear y operar de manera autónoma en entornos desconocidos o dinámicos.

Resumen de Técnicas

- **Rejillas de Ocupación:** Ofrecen una representación detallada pero pueden ser ineficientes en términos de memoria.
- **QuadTrees y Octrees:** Mejoran la eficiencia de almacenamiento aprovechando la homogeneidad en el espacio, ideales para entornos 2D y 3D respectivamente.
- **Representaciones Poliédricas:** Simplifican el entorno en primitivas geométricas, reduciendo la complejidad pero a costa de detalles.
- **Kd-Trees:** Proporcionan una estructura eficiente para manejar y buscar puntos en espacios multidimensionales, esencial para diversas operaciones en robótica.

Elección de la Representación

La elección de la representación depende de:

- **Objetivos Específicos:** Navegación detallada vs. localización global.
- **Características del Entorno:** Complejidad, tamaño y dimensionalidad.
- **Recursos Disponibles:** Capacidad de memoria y potencia de cómputo.

Retos y Oportunidades

- **Optimización de Recursos:** Buscar un equilibrio entre detalle y eficiencia.
- **Entornos Dinámicos:** Adaptar las representaciones para manejar cambios en el entorno.
- **Integración de Técnicas:** Combinar diferentes representaciones para aprovechar sus respectivas ventajas.

Representación Topológica del Entorno en Robótica Móvil

Introducción

La **representación topológica del entorno** es un enfoque clave en robótica móvil y en sistemas de inteligencia artificial que requieren una abstracción eficiente y útil del espacio en el que operan. A diferencia de las representaciones métricas que capturan las propiedades geométricas detalladas del entorno, las representaciones topológicas se centran en la **estructura y conectividad** de los espacios, proporcionando una vista más abstracta y simplificada.

Mapas Topológicos

- **Definición:** Un mapa topológico es una representación basada en grafos que captura la relación entre diferentes lugares o estados en el entorno.
- **Componentes:**
 - **Nodos:** Representan **lugares distintivos** o estados del entorno que son relevantes para las tareas del robot.
 - **Aristas (Arcos):** Indican **relaciones** o **conectividades** entre los lugares, como caminos transitables o transiciones posibles.

Importancia en Robótica Móvil

- **Simplificación del Entorno:** Los mapas topológicos permiten manejar entornos complejos mediante la abstracción de información esencial.
- **Eficiencia:** Reducen la complejidad computacional al evitar el manejo de detalles geométricos innecesarios para ciertas tareas.
- **Aplicaciones:**
 - **Navegación Global:** Planificación de rutas a través de lugares clave.
 - **Localización:** Identificación del lugar actual basándose en observaciones.
 - **SLAM (Simultaneous Localization and Mapping):** Construcción de mapas simultáneamente mientras se localiza el robot.

Falta de Consenso

No existe un acuerdo unánime sobre cómo deben construirse los mapas topológicos o qué elementos específicos deben incluir. Esto se debe a que los mapas topológicos se adaptan a las necesidades particulares de cada aplicación y pueden variar en:

- **Significado de Nodos y Aristas:** Dependiendo del contexto, un nodo puede representar desde un punto físico hasta un estado abstracto del robot.
- **Granularidad:** La cantidad de detalle y la definición de lugares distintivos pueden cambiar según los objetivos.

Elementos Comunes en Mapas Topológicos

A pesar de las variaciones, existen elementos comunes en la construcción y estructura de los mapas topológicos:

Identificación de Nodos

- **Lugares Distintivos:** Se identifican utilizando los sensores del robot, que pueden ser características del entorno, lecturas sensoriales únicas o patrones detectados.
- **Estados Distintivos (ds):** Asociados a vistas o lecturas específicas de los sensores que permiten distinguir diferentes lugares.

Relaciones entre Nodos (Aristas)

- **Conectividad:** Se establecen relaciones de conexión entre los nodos basadas en la capacidad del robot para moverse de un lugar a otro.
- **Acciones Asociadas:** Las transiciones pueden estar etiquetadas con las acciones necesarias para moverse entre lugares.

Información Métrica Asociada

- Aunque los mapas topológicos son abstractos, es común asociar información métrica local a las aristas, como distancias aproximadas o ángulos de giro, para mejorar la navegación.

Generación Automática de Mapas Topológicos

La construcción automática de mapas topológicos implica la transformación de las experiencias del robot en una representación estructurada del entorno.

Secuencia de Experiencias

El robot acumula una secuencia de experiencias en forma de:

$$v_0, a_0, v_1, a_1, \dots, a_{n-1}, v_n$$

- v_i : Vista o lectura sensorial en el estado i .
- a_i : Acción ejecutada para pasar de v_i a v_{i+1} .

Estados Distintivos

- **Definición:** Un estado del entorno en el que el robot ha tomado datos específicos (una vista o lectura sensorial).
- **Características:**
 - **Posición y Orientación:** Cada estado distintivo está asociado a una posición y orientación en el mapa.
 - **Identificación Única:** Las vistas permiten distinguir estados distintivos, aunque pueden darse casos donde la misma vista ocurre en diferentes lugares.

Acciones

- **Tipos de Acciones:**

- **Giros (Turns):** Acciones que cambian la orientación del robot sin cambiar su posición (ejemplo: girar 90 grados a la derecha).
- **Desplazamientos (Travels):** Acciones que mueven al robot de un lugar a otro (ejemplo: avanzar 5 metros).

Esquemas de Transición

La secuencia de experiencias se transforma en un conjunto de **esquemas**, que representan transiciones entre estados distintivos:

$$\{(v_i, ds_i), a_i, (v_{i+1}, ds_{i+1})\}$$

- **(ds_i):** Estado distintivo asociado a la vista (v_i).
- Los esquemas capturan cómo las acciones llevan al robot de un estado distintivo a otro.

Agrupación en Lugares y Regiones

- **Lugares:** Conjunto de estados distintivos que están conectados por acciones de giro (turns). Representan posiciones del robot donde puede cambiar su orientación.
- **Regiones:** Agrupan varios lugares, permitiendo crear mapas topológicos **jerárquicos**.

Construcción del Mapa Topológico

Objetivo

- **Minimizar** el conjunto de caminos topológicos y lugares dados los datos de experiencia.
- Crear una representación eficiente que capture las conexiones esenciales sin redundancias innecesarias.

Caminos Topológicos

- Un **camino** establece una relación entre dos lugares mediante una acción de desplazamiento (travel) sin incluir acciones de giro.
- Los caminos tienen una **dirección**, lo que significa que ir de (A) a (B) puede ser diferente que ir de (B) a (A).
- **Rutas:** Son caminos entre dos regiones, permitiendo planificación a mayor escala.

Predicados y Relaciones

Para formalizar las relaciones entre los distintos elementos, se utilizan **predicados**:

- **on(pa, p):** El lugar (p) está en el camino (pa).
- **order(pa, dir, p, q):** El lugar (p) está antes que (q) en el camino (pa) siguiendo la dirección (dir).
- **at(ds, p):** El estado distintivo (ds) está en el lugar (p).
- **along(ds, pa, dir):** El estado distintivo (ds) está a lo largo del camino (pa) en dirección (dir).
- **teq(ds1, ds2):** Los estados distintivos (ds1) y (ds2) son topológicamente equivalentes (representan el mismo lugar o posición en el mapa).

Axiomas y Teoremas

- **Conectividad de Acciones:**

- Dos acciones de desplazamiento **travel** consecutivas comparten un mismo camino.
- Una acción de **giro en U (turn-around)** permite al robot recorrer un camino en la dirección opuesta.
- Las acciones de **giro a la izquierda (turn-left)** o **giro a la derecha (turn-right)** conducen al robot a caminos diferentes.

Ejemplos Prácticos

Ejemplo 1: Camino Lineal con Retorno

Secuencia de Esquemas:

1. $\langle a, \text{travel}, b \rangle$
2. $\langle b, \text{turnAround}, c \rangle$
3. $\langle c, \text{travel}, d \rangle$

Análisis:

- **Lugares Topológicos:**

- $A = \{a\}$
- $B = \{b, c\}$ (mismo lugar con orientación diferente)
- $C = \{d\}$

- **Camino Topológico:**

- Pa : camino que conecta a con d pasando por b y c .
- Al ejecutar $\langle d, \text{turnAround}, a' \rangle$, donde a' es equivalente a a , completamos el ciclo.

Ejemplo 2: Camino con Bifurcaciones

Secuencia de Esquemas:

1. $\langle a, \text{turnRight}, b \rangle$
2. $\langle b, \text{travel}, c \rangle$
3. $\langle c, \text{turnAround}, d \rangle$
4. $\langle d, \text{travel}, e \rangle$
5. $\langle e, \text{turnRight}, a' \rangle$
6. $\langle a', \text{turnRight}, b' \rangle$

Análisis:

- **Lugares Topológicos:**

- $P = \{a, b\}$
- $Q = \{c, d\}$
- $R = \{e, a', b'\}$
- Se observa que a y a' son equivalentes, al igual que b y b' , por lo que $P = R$.

Ejemplo 3: Intersecciones y Nuevos Caminos

Secuencia de Esquemas:

1. $\langle ds1, travel, ds2 \rangle$
2. $\langle ds2, turnRight, ds3 \rangle$
3. $\langle ds3, travel, ds4 \rangle$
4. $\langle ds4, turnLeft, ds5 \rangle$
5. $\langle ds5, travel, ds6 \rangle$

Análisis:

- **Lugares:**
 - $(A = \{ ds1 \})$
 - $(B = \{ ds2, ds3 \})$
 - $(C = \{ ds4, ds5 \})$
 - $(D = \{ ds6 \})$
- **Caminos Topológicos:**
 - (Pa) : conecta $(ds1)$ con $(ds2)$.
 - $(Pa1)$: conecta $(ds3)$ con $(ds4)$.
 - $(Pa2)$: conecta $(ds5)$ con $(ds6)$.

Ejemplo 4: Ambigüedad en el Mapa

Situación:

- El robot visita lugares (A, B, C, D, E, F, C) en orden.
- Las vistas en las intersecciones, especialmente en (B) y (C) , son muy similares o idénticas.

Problema:

- El robot puede confundir los lugares debido a vistas similares, generando ambigüedad en el mapa topológico.

Solución:

- Utilizar **información métrica local** adicional (como distancias recorridas o ángulos de giro) para desambiguar lugares que pueden tener vistas similares.

Uso de Información Métrica Local

Para mejorar la precisión y resolver posibles ambigüedades en el mapa topológico, se asocia información métrica local a los elementos del mapa.

Asociación de Información Métrica

- **Caminos:**
 - Se almacena la distancia aproximada entre lugares a lo largo del camino.

- Ayuda a estimar posiciones relativas y tiempos de recorrido.
- **Lugares:**
 - Se asocia el ángulo o dirección en la que comienzan los caminos desde ese lugar.
 - Permite al robot orientar sus acciones basándose en referencias métricas.

Ventajas

- **Desambiguación:** Al incorporar medidas métricas, es posible distinguir entre lugares con vistas similares pero posiciones diferentes.
- **Navegación Más Precisa:** Se mejora la planificación de movimientos y se reducen errores en la ejecución de acciones.

Ejemplo de Desambiguación

- Dos intersecciones con vistas similares pueden diferenciarse si el robot sabe que una está a 10 metros de su posición actual y la otra a 30 metros.
- Al comparar las distancias recorridas, el robot puede determinar en cuál de las dos se encuentra.

Conclusiones

La representación topológica del entorno es una herramienta poderosa en robótica móvil, especialmente para tareas que requieren una abstracción del espacio y una navegación eficiente sin necesidad de detalles geométricos precisos.

Beneficios de los Mapas Topológicos

- **Eficiencia Computacional:** Reducen la cantidad de datos a procesar y almacenar.
- **Adaptabilidad:** Pueden ajustarse fácilmente a diferentes entornos y escalas.
- **Simplificación:** Facilitan la planificación y ejecución de movimientos basados en transiciones entre lugares clave.

Desafíos

- **Ambigüedad en Lugares:** Vistas similares pueden llevar a confusiones en la localización.
- **Necesidad de Información Adicional:** En algunos casos, es necesario complementar con información métrica o características únicas del entorno.

Integración con Otras Representaciones

- **Mapas Híbridos:** Combinar mapas topológicos con métricos para aprovechar las ventajas de ambos enfoques.
- **Jerarquías:** Crear mapas topológicos jerárquicos donde las regiones se organizan en niveles, facilitando la navegación a diferentes escalas.

Representación Semántica del Entorno en Robótica Móvil

Introducción

La **representación semántica del entorno** es un enfoque avanzado en robótica móvil e inteligencia artificial que busca dotar a los robots no solo de una comprensión geométrica o topológica del espacio en el que operan, sino también de una **comprensión a nivel conceptual y cualitativo**. Los **mapas semánticos** son una herramienta clave en este ámbito, permitiendo a los robots interactuar de manera más natural y efectiva tanto con su entorno como con los seres humanos.

Mapas Semánticos

- **Definición:** Un mapa semántico es una representación del entorno en forma de grafo donde los nodos y aristas no solo reflejan posiciones y conexiones, sino que también incorporan información sobre **conceptos significativos**. Esta información puede incluir tipos de objetos, categorías de espacios, relaciones entre elementos, etc.
- **Relación entre Conceptos:** Los mapas semánticos establecen relaciones entre diferentes conceptos y entidades presentes en el entorno, facilitando una **descripción cualitativa** y de alto nivel.

Importancia en Robótica Móvil

- **Abstracción del Espacio:** Permiten al robot tener una comprensión más abstracta y humana del entorno, más allá de coordenadas y métricas precisas.
- **Interacción Humano-Robot (HRI):** Facilitan la comunicación con los humanos, ya que el robot puede entender y utilizar conceptos comunes (por ejemplo, "cocina", "mesa", "pasillo").
- **Interpretación de Escenas:** Ayudan al robot a interpretar y actuar en su entorno basándose en significados y funciones de los espacios y objetos.

Mapas Semánticos en Robótica

Características Clave

- **Identificación de Signos y Símbolos:** Los mapas semánticos registran signos y símbolos que contienen conceptos significativos para los humanos.

- **Información Geométrica Mejorada:** No solo incluyen datos geométricos o topológicos, sino que también enriquecen estos datos con características cualitativas de alto nivel.
- **Representación Mejorada del Entorno:** Proporcionan una visión más completa y útil para tareas avanzadas de robótica.

Ventajas

- **Comunicación Efectiva:** Permiten que los robots comprendan y utilicen lenguaje natural y conceptos humanos en sus operaciones.
- **Planificación Avanzada:** Facilitan la planificación de tareas complejas al entender el propósito y las relaciones entre los objetos y lugares.
- **Adaptabilidad:** Mejoran la capacidad del robot para adaptarse a cambios en el entorno y comprender situaciones nuevas.

Construcción Automática de Mapas Semánticos

La construcción automática de mapas semánticos es un reto importante en la robótica moderna. Si bien la **construcción de mapas geométricos** es un problema ampliamente abordado y resuelto utilizando técnicas como SLAM (Simultaneous Localization and Mapping), la **incorporación de semántica** requiere métodos más complejos que combinan percepción avanzada, aprendizaje automático y razonamiento.

Fuentes de Datos

- **Datos Métricos:** Información proveniente de sensores como LIDAR, cámaras de profundidad, que proporciona la estructura geométrica del entorno.
- **Información Visual:** Imágenes y videos utilizados para reconocer y clasificar objetos y escenas.
- **Sensores Múltiples:** Combinación de diferentes tipos de sensores para enriquecer la percepción.

Metodologías de Construcción

La construcción de mapas semánticos puede caracterizarse según varios criterios:

Caracterización Según la Escala

Métodos Indoors

- **Escena Única:**
 - **Alcance:** Entornos reducidos, como una habitación o una parte específica de un edificio.

- **Métodos:** Se enfocan en reconocer y etiquetar objetos y elementos dentro de una única escena.
- **Aplicaciones:** Robots asistentes en el hogar, reconocimiento de objetos para manipulación.
- **Gran Escala:**
 - **Alcance:** Edificios completos, plantas enteras o entornos interiores amplios.
 - **Métodos:** Requieren técnicas más avanzadas para manejar información extensa, incluyendo la integración de datos provenientes de múltiples ubicaciones y tiempos.
 - **Desafíos:** Manejo de grandes volúmenes de datos, mantenimiento de coherencia en la representación.

Métodos Outdoors

- **Alcance:** Entornos exteriores como calles, parques, ciudades enteras.
- **Metodologías:**
 - Incorporación de datos de GPS, imágenes aéreas y otros sensores adecuados para exteriores.
 - Reconocimiento de elementos como calles, edificios, señales de tráfico.
- **Desafíos:** Variabilidad del entorno, condiciones climáticas, iluminación, escala masiva.

Uso de Mapas Topológicos

La construcción de mapas semánticos a menudo se apoya en mapas topológicos, que pueden ser:

Sin Restringir

- **Descripción:** Los mapas topológicos se utilizan como base, y la semántica se añade sobre los nodos existentes sin alterar la estructura.
- **Ventaja:** Flexibilidad para añadir información semántica sin modificar la representación topológica.

Restringidos

- **Descripción:** La construcción del mapa topológico está guiada por la información semántica, es decir, la semántica influye en cómo se forman los nodos y aristas.
- **Ventaja:** Mejora la relevancia y utilidad del mapa para tareas específicas al integrar semántica desde el inicio.

Uso de Coherencia Temporal

La **coherencia temporal** se refiere a la relación y consistencia de las observaciones a lo largo del tiempo.

- **Razonamiento Probabilístico en el Tiempo:** Utilización de modelos probabilísticos que tienen en cuenta la evolución temporal, como:
 - **Modelos Ocultos de Markov (HMMs).**
 - **Filtros de Kalman Extendidos (EKF).**
- **Beneficios:**
 - Mejoran la precisión en la identificación y seguimiento de objetos y lugares a lo largo del tiempo.
 - Ayudan a resolver ambigüedades y a mantener una representación coherente del entorno dinámico.

Tipo de Percepción

La percepción para la construcción de mapas semánticos puede clasificarse según:

Entrada Única

- **Anotación Manual:**
 - El operador humano etiqueta manualmente las partes del mapa o las imágenes.
 - **Limitaciones:** No es escalable, consume mucho tiempo y es susceptible a errores humanos.
- **Etiquetado de Píxeles:**
 - Uso de técnicas de visión por computadora para etiquetar cada píxel de una imagen con una clase semántica.
 - **Herramientas:** Redes neuronales convolucionales (CNNs), segmentación semántica.

Entrada Múltiple

- **Varias Fuentes de Datos:**
 - Combinación de datos de diferentes sensores (cámaras RGB, cámaras de profundidad, LIDAR, etc.) para enriquecer la percepción.
 - **Ventaja:** Permite obtener una representación más robusta y completa del entorno.
 - **Desafío:** La fusión de datos heterogéneos requiere métodos sofisticados para asegurar la coherencia y precisión.

Segmentación Semántica Usando Deep Learning

La **segmentación semántica** es una técnica esencial en la construcción de mapas semánticos, que consiste en asignar una etiqueta semántica a cada píxel de una imagen.

Uso de Redes Neuronales Convolucionales (CNNs)

- **Funcionamiento:**

- Las CNNs aprenden a reconocer patrones en las imágenes a través de un proceso de entrenamiento supervisado.
- Se utilizan para clasificar cada píxel en una categoría específica (por ejemplo, “pared”, “puerta”, “persona”).

- **Arquitecturas Comunes:**

- **FCN (Fully Convolutional Networks):** Adaptación de CNNs para producir mapas de segmentación completos.
- **U-Net:** Arquitectura diseñada para segmentación en imágenes médicas, adaptada a otras áreas.
- **DeepLab:** Utiliza técnicas como atrous convolutions y CRFs para mejorar la segmentación.

Ventajas del Deep Learning

- **Precisión:** Capaces de lograr altos niveles de precisión en la clasificación y detección de objetos.
- **Escalabilidad:** Pueden entrenarse con grandes conjuntos de datos y generalizar a nuevas escenas.
- **Aprendizaje de Características:** No requieren la definición manual de características, ya que aprenden representaciones relevantes directamente de los datos.

Desafíos

- **Necesidad de Grandes Conjuntos de Datos:** Requieren cantidades significativas de datos etiquetados para entrenar eficientemente.
- **Computación Intensiva:** El entrenamiento y la inferencia pueden requerir hardware especializado (GPUs).
- **Generalización:** Mantener la precisión en ambientes no vistos o condiciones cambiantes puede ser difícil.

Aplicaciones de Mapas Semánticos en Robótica

Navegación Semántica

- Permite al robot planificar rutas no solo basándose en distancias, sino también considerando la semántica de los lugares (por ejemplo, “ir a la cocina”).

Interacción Humano-Robot

- Facilita la comprensión de comandos de lenguaje natural y la ejecución de tareas basadas en instrucciones humanas.

Manipulación de Objetos

- Ayuda en la identificación y manipulación de objetos específicos en el entorno, mejorando tareas como recoger y colocar objetos.

Entornos Dinámicos

- Los mapas semánticos permiten al robot adaptarse a cambios en el entorno, como nuevos objetos o modificaciones en la disposición de espacios.

Conclusiones

La **representación semántica del entorno** es un paso fundamental hacia la creación de robots más inteligentes y capaces de interactuar eficazmente con el mundo que los rodea. Los mapas semánticos enriquecen la percepción del robot, permitiéndole comprender conceptos de alto nivel y relacionarlos con su experiencia sensorial y acciones.

Beneficios Clave

- **Comunicación Mejorada:** Facilita interacciones más naturales entre humanos y robots.
- **Planificación y Razonamiento:** Mejora la capacidad de los robots para planificar y ejecutar tareas complejas.
- **Adaptabilidad y Autonomía:** Permite a los robots operar eficientemente en entornos dinámicos y desconocidos.

Desafíos Futuros

- **Integración de Datos:** Necesidad de métodos avanzados para fusionar y aprovechar diversas fuentes de información.
- **Aprendizaje Continuo:** Desarrollo de sistemas que puedan actualizar y mejorar sus mapas semánticos en tiempo real.
- **Interoperabilidad:** Crear estándares y frameworks que faciliten el intercambio y uso de mapas semánticos entre diferentes sistemas y plataformas.

Conocimiento en Aprendizaje: Representación en Redes Neuronales

Introducción

El **aprendizaje automático** es una rama fundamental de la inteligencia artificial que se centra en desarrollar algoritmos y modelos que permitan a las máquinas **aprender** de los datos y mejorar su desempeño en tareas específicas sin ser programadas explícitamente para ello. Existen diversos paradigmas de aprendizaje automático:

- **Aprendizaje Supervisado:** El modelo aprende a partir de un conjunto de datos etiquetados, es decir, ejemplos de entrada y salida deseada. Su objetivo es aprender una función que mapee las entradas a las salidas correctas.
- **Aprendizaje No Supervisado:** El modelo trabaja con datos no etiquetados, buscando patrones o estructuras ocultas dentro de los datos, como agrupamientos o reducciones de dimensionalidad.
- **Aprendizaje por Refuerzo:** El modelo aprende a tomar decisiones secuenciales mediante interacciones con un entorno, recibiendo recompensas o castigos que guían su comportamiento hacia la maximización de una función de recompensa.
- **Aprendizaje Genético:** Basado en algoritmos evolutivos, este enfoque utiliza conceptos de selección natural y genética para evolucionar soluciones a problemas optimizando una población de individuos a lo largo de generaciones.

Representación del Conocimiento en Aprendizaje Automático

Un aspecto crucial en el aprendizaje automático es la **representación del conocimiento** que el modelo adquiere durante el proceso de entrenamiento. Surgen preguntas importantes:

- **¿Cómo se representa el conocimiento aprendido?**
- **¿Podemos inferir relaciones causales entre el resultado del entrenamiento y el conocimiento adquirido?**

Estas cuestiones son fundamentales, ya que comprender la representación interna del conocimiento permite interpretar, explicar y confiar en los modelos, especialmente en aplicaciones críticas.

Enfoque en Redes Neuronales

Entre los diversos métodos de aprendizaje automático, las **redes neuronales** destacan por su capacidad para aproximar funciones complejas y por su amplio uso en diversas aplicaciones. Sin embargo, uno de los principales desafíos asociados con las redes neuronales es entender cómo se representa el conocimiento dentro de ellas y cómo este conocimiento se relaciona con los datos de entrenamiento y las predicciones realizadas.

Redes Neuronales

¿Qué son las Redes Neuronales?

Las **redes neuronales artificiales** (ANNs) son modelos computacionales inspirados en la estructura y funcionamiento del cerebro humano y de otros seres vivos. Su objetivo es imitar la capacidad de aprendizaje y abstracción del cerebro mediante sistemas **conexionistas**, donde el conocimiento se representa a través de conexiones fortalecidas o debilitadas entre elementos básicos llamados **neuronas**.

Neurona: Modelo Biológico vs. Computacional

Modelo Biológico

- **Entradas (Dendritas):** Las dendritas son extensiones que reciben señales de otras neuronas.
- **Integración (Soma):** La neurona integra las señales recibidas. Funciona como un dispositivo de “todo o nada”; solo se activa si la suma de las señales entrantes supera un cierto umbral.

- **Salidas (Axón):** El axón transmite la señal de salida a otras neuronas a través de sinapsis.

Modelo Computacional

- **Entradas:** Representadas por números reales x_i que corresponden a las características o variables de entrada.
- **Integración:** Se calcula una suma ponderada de las entradas mediante **pesos sinápticos** w_i :

$$\text{net} = \sum_i w_i x_i$$

- **Función de Activación:** Se aplica una función $f(\text{net})$ al valor integrado para obtener la salida de la neurona.
- **Salida:** El resultado $y = f(\text{net})$ es la respuesta de la neurona, que puede ser utilizado como entrada para otras neuronas o como salida final del modelo.

El Perceptrón Simple

El **perceptrón** es uno de los modelos más simples y básicos de neuronas artificiales, introducido por Frank Rosenblatt en 1958.

Estructura y Funcionamiento

- **Integración de Entradas:** Calcula la suma ponderada de las entradas:

$$\text{net} = \sum_i w_i x_i$$

- **Función de Activación:** Utiliza una función escalón (también conocida como función de Heaviside):

$$y = f(\text{net}) = \begin{cases} 1 & \text{si } \text{net} > \Theta \\ 0 & \text{otro caso} \end{cases}$$

Donde Θ es el **umbral** (bias) que determina cuándo la neurona se activa.

- **Aprendizaje:** El proceso de ajuste de los pesos w_i para minimizar el error entre las salidas deseadas y las obtenidas.

Simplificación con Sesgo Integrado

Es común integrar el umbral Θ en el modelo agregando un peso adicional w_0 y una entrada fija $x_0 = 1$:

- **Nueva Integración:**

$$\text{net} = \sum_{i=1}^N w_i x_i - \Theta = \sum_{i=0}^N w_i x_i$$

Donde $w_0 = -\Theta$ y $x_0 = 1$.

- **Función de Activación Actualizada:**

$$y = f(\text{net}) = \begin{cases} 1 & \text{si } \text{net} > 0 \\ 0 & \text{otro caso} \end{cases}$$

Representación del Conocimiento

- Los **pesos sinápticos** (w_i) almacenan el conocimiento del perceptrón.
- Modificando los pesos, se altera la salida del perceptrón para una misma entrada, lo que permite que el modelo aprenda a clasificar correctamente los datos.

Interpretación Geométrica del Perceptrón

El perceptrón puede interpretarse geoméricamente como un modelo que define un **hiperplano** en el espacio de características. Este hiperplano separa los datos en dos regiones:

- **Ecuación del Hiperplano:**

$$\sum_{i=1}^N w_i x_i - \Theta = 0$$

- **Clasificación:**

- Si $\sum_i w_i x_i > \Theta$, el punto se clasifica en la clase positiva ($y = 1$).
- Si $\sum_i w_i x_i \leq \Theta$, el punto se clasifica en la clase negativa ($y = 0$).

Ejemplo con N=2

- **Hiperplano en 2D:**

$$w_1 x_1 + w_2 x_2 - w_0 = 0$$

Esta ecuación representa una línea recta que divide el plano en dos regiones.

Limitaciones

- El perceptrón simple solo puede resolver problemas **linealmente separables**, es decir, aquellos en los que las clases pueden separarse mediante un hiperplano.

Entrenamiento del Perceptrón

El objetivo del entrenamiento es encontrar los pesos (w_i) que permitan al perceptrón clasificar correctamente los ejemplos del conjunto de datos.

Proceso de Entrenamiento

1. **Inicialización:** Se asignan valores iniciales (generalmente aleatorios) a los pesos.
2. **Presentación de Ejemplos:** Se presentan los ejemplos de entrenamiento uno a uno.
3. **Cálculo de la Salida:** Para cada ejemplo, se calcula la salida del perceptrón:

$$y = f\left(\sum_i w_i x_i\right)$$

4. **Actualización de Pesos:** Si la salida (y) es diferente de la salida deseada (y_d), se actualizan los pesos:

$$w_i \leftarrow w_i + \Delta w_i$$

Donde:

$$\Delta w_i = \eta(y_d - y)x_i$$

η (η) es la tasa de aprendizaje.

Convergencia

- El algoritmo de aprendizaje del perceptrón **converge** si y solo si los datos son linealmente separables.
- Si los datos no son linealmente separables, el algoritmo puede no converger o no clasificar correctamente todos los ejemplos.

Ejemplo Práctico

Supongamos un conjunto de datos de dos clases que son linealmente separables. Se busca entrenar un perceptrón para clasificarlos:

- **Datos de Entrenamiento:**

x_1	x_2	Clase (y_d)
2	2	1
7	-3	0
1	-1	0

- **Proceso:**
 - Inicializar pesos (w_0, w_1, w_2).
 - Para cada ejemplo, calcular (y) y actualizar los pesos si ($y \neq y_d$).
 - Repetir hasta que el perceptrón clasifique correctamente los ejemplos o hasta alcanzar un número máximo de iteraciones.

Limitaciones del Perceptrón Simple

- No puede resolver problemas como el **EXOR** (función lógica XOR), donde las clases no son linealmente separables.
- Para abordar este tipo de problemas, es necesario utilizar arquitecturas más complejas, como el **perceptrón multicapa**.

Perceptrón Multicapa

El **perceptrón multicapa** (MLP) es una extensión del perceptrón simple que introduce una o más **capas ocultas** entre las entradas y las salidas. Esto permite al modelo aprender representaciones más complejas y manejar problemas no linealmente separables.

Estructura del MLP

- **Capas:**
 - **Capa de Entrada:** Recibe los datos de entrada.
 - **Capas Ocultas:** Procesan las entradas a través de neuronas interconectadas que aplican funciones de activación no lineales.
 - **Capa de Salida:** Genera las predicciones del modelo.
- **Conectividad:**
 - Se suele utilizar una arquitectura **totalmente conectada** (fully connected), donde cada neurona de una capa está conectada con todas las neuronas de la siguiente capa.

Funciones de Activación

Se utilizan funciones no lineales para permitir que la red aprenda relaciones complejas:

- **Sigmoide:**

$$\sigma(\text{net}) = \frac{1}{1 + e^{-\text{net}}}$$

- **ReLU (Rectified Linear Unit):**

$$f(\text{net}) = \max(0, \text{net})$$

- **TanH (Tangente Hiperbólica):**

$$f(\text{net}) = \frac{e^{\text{net}} - e^{-\text{net}}}{e^{\text{net}} + e^{-\text{net}}}$$

Proceso de Aprendizaje: Retropropagación del Error

El algoritmo de **backpropagation** es el método estándar para entrenar redes neuronales multicapa.

Pasos del Algoritmo

1. Propagación Hacia Adelante:

- Los datos se introducen en la capa de entrada y se propagan a través de las capas ocultas hasta llegar a la capa de salida.

2. Cálculo del Error:

- Se calcula el error en la salida comparando las predicciones del modelo (y) con las salidas deseadas (y_d):

$$E = \frac{1}{2} \sum_k (y_{d,k} - y_k)^2$$

3. Retropropagación del Error:

- El error se propaga hacia atrás a través de la red, calculando gradientes parciales respecto a los pesos.

4. Actualización de Pesos:

- Los pesos se ajustan en función de los gradientes calculados y una tasa de aprendizaje (η):

$$w_{ij} \leftarrow w_{ij} - \eta \frac{\partial E}{\partial w_{ij}}$$

Capacidad de Aprendizaje

- Gracias a las funciones de activación no lineales y las capas ocultas, el MLP puede aproximar funciones complejas y resolver problemas como el EXOR.

Interpretación Geométrica

- Cada capa oculta transforma el espacio de entrada en una nueva representación, permitiendo que la red establezca **fronteras de decisión** más complejas que no son posibles con un perceptrón simple.
- Las **fronteras de decisión** generadas por un MLP pueden ser no lineales y adaptarse a la distribución de los datos.

Representación del Conocimiento en Redes Neuronales

Interpretabilidad y Transparencia

- Las redes neuronales, especialmente las profundas, son comúnmente consideradas **cajas negras**, ya que es difícil interpretar los patrones y relaciones que han aprendido.
- El conocimiento está distribuido en los **pesos sinápticos** de la red, y la interacción compleja entre numerosas neuronas y capas dificulta extraer reglas o explicaciones claras.

Desafíos

- Comprensión del Aprendizaje:** Entender cómo se representa internamente el conocimiento y cómo cada peso influye en la salida es un desafío.
- Causalidad:** Inferir relaciones causales directas entre las estructuras internas de la red y las decisiones que toma requiere análisis sofisticados.
- Transferibilidad:** El conocimiento adquirido en una tarea específica puede no ser fácilmente transferible a otra, o puede requerir ajustes adicionales.

Enfoques para Mejorar la Interpretabilidad

Visualización de Pesos y Activaciones

- **Mapas de Activación:** Visualizar cómo responden las neuronas a diferentes entradas puede dar pistas sobre qué características están aprendiendo.
- **Filtros y Características:** En redes convolucionales, es posible visualizar los filtros para entender qué patrones detectan.

Extracción de Reglas

- Intentar extraer reglas lógicas o modelos más simples que aproximen el comportamiento de la red.
- **Árboles de Decisión:** En algunos casos, es posible aproximar la red mediante árboles de decisión que son más interpretables.

Redes Neuronales Discretas

- Utilizar funciones de activación discretas o simplificar la arquitectura para facilitar el análisis.

Métodos de Atención

- Incorporar mecanismos de atención que realcen las partes más relevantes de la entrada para la predicción.

Importancia de la Interpretabilidad

- **Confiabilidad:** En aplicaciones críticas (medicina, finanzas, seguridad), es esencial entender y confiar en las decisiones tomadas por el modelo.
- **Ética y Transparencia:** Para cumplir con regulaciones y garantizar un uso ético de la inteligencia artificial.
- **Diagnóstico y Mejora:** Detectar errores, sesgos o áreas de mejora en el modelo.

Conclusiones

Las **redes neuronales** son una herramienta poderosa en el aprendizaje automático, capaces de modelar relaciones complejas y aprender de grandes cantidades de datos. Sin embargo, representan un desafío en términos de comprensión y representación del conocimiento adquirido.

Puntos Clave

- **Representación del Conocimiento:** En las redes neuronales, el conocimiento se distribuye en los pesos sinápticos, y aprender implica ajustar estos pesos a través del entrenamiento.
- **Limitaciones del Perceptrón Simple:** Solo puede resolver problemas linealmente separables. Los problemas más complejos requieren arquitecturas más sofisticadas.
- **Perceptrón Multicapa:** Introduce capas ocultas y funciones de activación no lineales, permitiendo resolver problemas no linealmente separables y aprender representaciones más complejas.
- **Interpretabilidad:** Es un área activa de investigación. Comprender cómo se representa y procesa el conocimiento en las redes neuronales es crucial para mejorar la confianza y eficacia de estos modelos.

Reflexión Final

Entender la **representación del conocimiento** en el aprendizaje automático es fundamental para avanzar hacia sistemas de inteligencia artificial más transparentes, fiables y éticos. Aunque las redes neuronales presentan desafíos en este aspecto, los esfuerzos continuos en investigación y desarrollo de nuevas técnicas de interpretabilidad y explicabilidad son esenciales para aprovechar plenamente el potencial de estas tecnologías en beneficio de la sociedad.

Redes Neuronales Convolucionales (CNNs): Razonamiento y Representación del Conocimiento

Introducción

Las **redes neuronales** (NN) han demostrado ser herramientas poderosas en el ámbito del aprendizaje automático, permitiendo resolver problemas de clasificación, regresión y otras tareas complejas. El proceso de **aprendizaje** en estas redes implica encontrar una frontera en un espacio de **k dimensiones** que separa adecuadamente los elementos de diferentes clases.

Limitaciones de las Redes Neuronales Clásicas con Imágenes

Cuando trabajamos con datos de baja dimensionalidad, las NN tradicionales funcionan satisfactoriamente. Sin embargo, si la entrada a una red neuronal es una **imagen**, el número de dimensiones del espacio de soluciones es igual al número de píxeles de la imagen:

- **Imagen de 64x64 píxeles:** 4,096 dimensiones.
- **Imagen de 100x100 píxeles:** 10,000 dimensiones.
- **Imagen de 1024x768 píxeles:** 786,432 dimensiones.

Este aumento exponencial en las dimensiones genera problemas conocidos como la **maldición de la dimensionalidad**, haciendo que las redes neuronales tradicionales sean ineficaces para procesar imágenes de alta resolución debido al alto requerimiento computacional y la dificultad para entrenar modelos con tantos parámetros.

Necesidad de Reducir la Dimensionalidad: Filtrado de Imágenes

Para abordar este desafío, es necesario **reducir la información** presente en las imágenes sin perder las características esenciales que permiten reconocer los objetos y patrones en ellas.

Idea Fundamental

- **Filtrar las Imágenes:** Aplicar técnicas que resalten características relevantes, como bordes, esquinas y texturas, que son fundamentales para la interpretación y reconocimiento de imágenes.

Primeros Enfoques en Filtrado de Imágenes

Filtros Clásicos

- **Filtros de Gabor:** Detectan bordes y texturas en diferentes orientaciones y frecuencias espaciales.
- **Filtro de Canny:** Identifica bordes en imágenes detectando cambios abruptos en la intensidad de los píxeles.

Problema: Aunque estos filtros extraen características relevantes, no mejoran significativamente los resultados del aprendizaje cuando se utilizan en conjunto con redes neuronales tradicionales.

Limitaciones de los Filtros Predefinidos

- **Selección Manual:** La elección de los filtros es realizada por expertos, lo que introduce sesgos y limita la capacidad del modelo para adaptarse a diversas tareas.
- **No Universalidad:** Un conjunto de filtros puede no ser adecuado para todas las imágenes o aplicaciones.

Pregunta Clave: ¿Y si, en lugar de elegir los filtros 'a mano', dejamos que sea el **algoritmo** el que decida qué filtros son los más adecuados?

Convoluciones en Redes Neuronales

Las **convoluciones** permiten que el modelo aprenda automáticamente los filtros más efectivos para extraer características relevantes de las imágenes.

Convolución en Imágenes

- **Convolución 2D:** Operación matemática que combina una imagen de entrada con un **kernel** o filtro para producir una nueva imagen que resalta ciertas características.

Definición Matemática

Para una imagen (I) y un kernel (K), la convolución (S) se define como:

$$S(i, j) = \sum_m \sum_n I(i - m, j - n) \cdot K(m, n)$$

¿Cómo Funcionan las Convoluciones?

Ejemplo 1D

- **Entrada y Kernel:** Vectores unidimensionales.
- **Resultado:** Otro vector donde cada elemento es el producto escalar de una ventana de la entrada con el kernel.

Extensión a 2D

- La convolución 2D se extiende considerando matrices en lugar de vectores, donde se superpone el kernel sobre la imagen y se calcula la suma ponderada.

Capas Convolucionales en Redes Neuronales

Las **capas convolucionales** son el núcleo de las redes neuronales convolucionales (CNNs), diseñadas para procesar datos con estructuras de cuadrícula, como imágenes.

Limitaciones de Capas Totalmente Conectadas

Al tratar imágenes como entradas para **capas totalmente conectadas** (fully connected), surge la necesidad de **aplanar** la imagen, transformando la matriz bidimensional en un vector unidimensional. Esto implica:

- **Pérdida de Información Espacial:** La estructura de la imagen se pierde, ya que no se conserva la relación entre píxeles vecinos.
- **Cantidad Masiva de Parámetros:** Cada neurona de la primera capa oculta está conectada con cada píxel de la imagen, lo que resulta en un número exorbitante de pesos a entrenar.

Concepto de Capas Locales

Para resolver estos problemas, introducimos el concepto de **capas locales**:

- **Parche (Receptive Field):** Es una pequeña ventana de la imagen (por ejemplo, 3x3 píxeles) que corresponde a una región específica.
- **Conexiones Locales:** Cada neurona recibe entradas solo de una región limitada de la imagen, preservando la información espacial y reduciendo el número de parámetros.

Implementación de Capas Convolucionales

- **Convolución como Operación Lineal:** Cada neurona aplica una convolución sobre su parche correspondiente.
- **Compartimiento de Pesos:** Todas las neuronas que aplican el mismo filtro (kernel) comparten los mismos pesos, lo que reduce drásticamente el número de parámetros y permite que el modelo detecte características independientemente de su posición en la imagen.

Propiedades Clave

Equivarianza a la Translación

- **Definición:** La salida de la convolución cambia de la misma manera que la entrada cuando esta se desplaza.
- **Implicación:** Si un patrón aparece en diferentes posiciones de la imagen, la capa convolucional puede detectarlo de manera consistente.

Matrices Toeplitz

- **Uso:** Las convoluciones pueden representarse matemáticamente utilizando matrices Toeplitz, lo que ayuda a entender la operación como una multiplicación matricial.
- **Ventaja:** Permite aprovechar optimizaciones en cálculos matriciales y entender la convolución en términos de operaciones lineales.

Estructura Interna de una Capa Convolucional

- **Entrada:** Una imagen o un conjunto de mapas de características de dimensiones $(H \times W \times D)$, donde H y W son alto y ancho, y D es la profundidad (número de canales, por ejemplo, RGB tiene $D = 3$).
- **Kernel:** Un filtro de tamaño $(k \times k \times D)$.
- **Salida:** Mapas de características resultantes de aplicar múltiples filtros a la entrada.

Composición de Convoluciones

- Aplicar múltiples capas convolucionales consecutivas es análogo a aplicar un filtro con un mayor campo receptivo, pero tiene la ventaja de introducir no linealidades a través de **funciones de activación** entre capas, lo que permite al modelo aprender características más complejas.

Reducción de Dimensionalidad: Capas de Pooling

Aunque las convoluciones preservan el tamaño de la entrada (suponiendo cierto padding y stride), es deseable reducir gradualmente las dimensiones para:

- **Disminuir el Costo Computacional:** Reduciendo el tamaño de los mapas de características.
- **Capturar Información a Diferentes Escalas:** Al disminuir la resolución espacial, las capas posteriores pueden enfocarse en características más globales.

Capas de Pooling

Max-Pooling

- **Definición:** Divide los mapas de características en regiones no solapadas (por ejemplo, 2x2) y toma el valor máximo de cada región.
- **Efecto:** Reduce las dimensiones a la mitad en cada dirección, preservando las características más destacadas.

Average Pooling

- **Definición:** Similar a Max-Pooling, pero en lugar del máximo, calcula el promedio de los valores en cada región.
- **Uso:** Menos común en CNNs modernas, ya que Max-Pooling suele producir mejores resultados en términos de detección de características prominentes.

Beneficios de Pooling

- **Reducción de Overfitting:** Al reducir el número de parámetros y la sensibilidad a pequeñas variaciones.
- **Invariancia a Pequeñas Translaciones y Distorsiones:** Ayuda al modelo a ser más robusto a cambios locales en la entrada.

Arquitectura de una Red Neuronal Convolucional (CNN)

Las CNNs combinan capas convolucionales y de pooling para extraer y condensar características de las imágenes, seguidas de capas totalmente conectadas para clasificar o realizar predicciones basadas en esas características.

Estructura General

1. **Capas Convolucionales:** Aplicar múltiples filtros para extraer características locales desde bajo nivel (bordes, texturas) hasta alto nivel (partes de objetos, objetos completos).
2. **Capas de Activación:** Funciones no lineales (ReLU, sigmoid, tanh) aplicadas después de cada convolución para introducir no linealidad al modelo.
3. **Capas de Pooling:** Reducen las dimensiones espaciales y consolidan la información.
4. **Capas Totalmente Conectadas:** Al final de la red, las características extraídas se utilizan para clasificar o predecir mediante una o más capas totalmente conectadas.

Ejemplo de Arquitectura CNN

Clasificador de 10 Clases (por ejemplo, dígitos del 0 al 9):

1. **Entrada:** Imagen de tamaño $28 \times 28 \times 1$ (escala de grises).

2. **Primera Capa Convolucional:**

- Número de filtros: 32
- Tamaño del kernel: 3×3
- Salida: $28 \times 28 \times 32$

3. **Función de Activación:**

- ReLU aplicada a la salida de la convolución.

4. **Primera Capa de Pooling:**

- Tipo: Max-Pooling
- Tamaño: 2×2
- Salida: $14 \times 14 \times 32$

5. **Segunda Capa Convolucional:**

- Número de filtros: 64
- Tamaño del kernel: 3×3
- Salida: $14 \times 14 \times 64$

6. **Función de Activación:**

- ReLU

7. **Segunda Capa de Pooling:**

- Max-Pooling 2×2
- Salida: $7 \times 7 \times 64$

8. **Aplanamiento:**

- Convertir el tensor en un vector de tamaño $7 \times 7 \times 64 = 3,136$

9. **Capa Totalmente Conectada:**

- Número de neuronas: 128
- Función de Activación: ReLU

10. **Capa de Salida:**

- Número de neuronas: 10 (una por clase)
- Función de Activación: Softmax para obtener probabilidades de clase.

Entrenamiento de la CNN

- **Función de Pérdida:** Entropía cruzada categórica para problemas de clasificación multiclase.

- **Optimizador:** Algoritmos como SGD, Adam, RMSprop.
- **Regularización:** Técnicas como dropout o weight decay para prevenir overfitting.

Ventajas de las CNNs

- **Reducción de Parámetros:** Gracias al compartimiento de pesos y conexiones locales.
- **Captura de Características Espaciales:** Preservan y explotan la estructura espacial de las imágenes.
- **Escalabilidad:** Pueden manejar imágenes de alta resolución de manera más eficiente que las redes totalmente conectadas.

Conclusiones

Las **redes neuronales convolucionales** representan un avance significativo en la aplicación de redes neuronales al procesamiento de imágenes y otras tareas que involucran datos estructurados en cuadrículas. Al aprovechar operaciones como la convolución y el pooling, las CNNs son capaces de extraer características significativas de las imágenes y manejar la alta dimensionalidad de los datos visuales.

Puntos Clave

- **Maldición de la Dimensionalidad:** Las imágenes de alta resolución presentan desafíos que las CNNs abordan eficientemente mediante convoluciones y pooling.
- **Aprendizaje de Filtros:** Las CNNs aprenden automáticamente los filtros óptimos para la tarea, evitando la necesidad de diseñar manualmente filtros específicos.
- **Arquitectura Eficiente:** Combinando capas convolucionales, funciones de activación, pooling y capas totalmente conectadas, las CNNs pueden aprender representaciones jerárquicas desde características de bajo nivel hasta abstracciones de alto nivel.

Aplicaciones

- **Reconocimiento de Imágenes:** Clasificación, detección de objetos, segmentación.
- **Visión por Computador:** Análisis de videos, reconocimiento facial, conducción autónoma.
- **Procesamiento de Lenguaje Natural:** Modelado de secuencias y representaciones de texto (cuando se aplican CNNs a datos secuenciales).

Este resumen ha presentado una visión detallada de las redes neuronales convolucionales, explicando su fundamento teórico y práctico, y cómo abordan los desafíos asociados con el procesamiento de imágenes de alta dimensión. Al entender estos conceptos, podemos apreciar el poder y la versatilidad de las CNNs en el campo del aprendizaje automático y la inteligencia artificial.

Transformers: Razonamiento y Representación del Conocimiento

Introducción

Las **redes neuronales** son modelos computacionales inspirados en el cerebro humano que procesan información mediante la combinación lineal del vector de entrada con los pesos de la red. Con el avance del **deep learning**, estas redes han evolucionado para incluir muchas capas, principalmente **convolucionales**, permitiendo aprender relaciones complejas en los datos.

Limitaciones de las Redes Neuronales Convolucionales

Las redes neuronales convolucionales (CNNs) son especialmente efectivas en el procesamiento de datos donde las relaciones locales son importantes, como en imágenes y señales. Capturan relaciones entre datos que están localmente “cerca” entre sí. Sin embargo, presentan limitaciones al manejar **secuencias de datos largas**, donde las dependencias pueden abarcar largas distancias:

- **Lenguaje Natural:** Las relaciones entre palabras pueden ser a largo plazo, y palabras distantes en una oración pueden influir en el significado.
- **Secuencias de Video:** La comprensión de eventos puede requerir información de fotogramas distantes.

Transformers

Los **Transformers** surgen como una solución para abordar las limitaciones de las CNNs y otros modelos al procesar secuencias largas de datos. Fueron introducidos por Vaswani et al. en 2017 en el artículo “Attention is All You Need”.

Características Clave

- **Captura de Relaciones a Largo Plazo:** Pueden modelar dependencias entre elementos distantes en una secuencia.
- **Dependencia de la Entrada:** Las relaciones se aprenden de manera adaptativa en función de la entrada específica.

Por ejemplo, en las oraciones:

- “El gato está sobre la mesa.”
- “El gato, que es de color negro, está sobre la mesa.”

En ambas oraciones, la relación entre “gato” y “está” es importante, incluso si están separadas por varias palabras.

Procesamiento de Secuencias en Transformers

El proceso general para manejar secuencias de datos en Transformers consta de tres pasos principales:

1. **Tokenización**
2. **Embeddings**
3. **Aprendizaje mediante Autoatención**

1. Tokenización

La **tokenización** consiste en separar la secuencia de datos de entrada en unidades discretas llamadas **tokens**.

Tipos de Tokenizadores

- **Character Tokenizer:** Divide la entrada en caracteres individuales.
- **Word Tokenizer:** Separa la entrada en palabras completas.
- **Sub-word Tokenizer:** Combina los anteriores, dividiendo en unidades menores que palabras pero mayores que caracteres.

Byte-Pair Encoding (BPE)

El **GPT Tokenizer** utiliza el método de **Byte-Pair Encoding (BPE)**:

- **Inicio:** Comienza con un conjunto de n-gramas de 1 carácter.
- **Proceso Iterativo:**
 - Se identifican las parejas de caracteres adyacentes más frecuentes y se combinan en nuevos tokens.
 - Se repite hasta alcanzar el tamaño de vocabulario deseado.
- **Resultado:** Un vocabulario eficiente que equilibra entre tokens de un solo carácter y palabras completas.

Nota: GPT-4 utiliza un vocabulario de 100,256 n-gramas.

2. Embeddings

Los **embeddings** transforman los tokens en vectores numéricos que capturan características semánticas y sintácticas.

- **Proceso:**
 - A cada token se le asigna un vector en un espacio de alta dimensión.
 - Estos vectores se entrenan para que tokens con significados similares tengan representaciones cercanas.
- **Ajuste del Embedding:**
 - Utiliza una **red neuronal** (como una **CNN** o una capa densa) para aprender las representaciones vectoriales.

3. Mecanismo de Autoatención (Self-Attention)

El núcleo del Transformer es el **mecanismo de autoatención**, que permite al modelo enfocarse en diferentes partes de la secuencia al procesar cada elemento.

Problemas con Convoluciones Largas

- **Limitaciones:**
 - Las convoluciones tradicionales tienen un alcance local limitado.
 - Para capturar dependencias a larga distancia, se necesitarían convoluciones con kernels muy grandes o muchas capas, lo cual es ineficiente.

Introducción al Mecanismo de Autoatención

- **Idea Fundamental:** Permitir que cada elemento de la secuencia preste atención a todos los demás elementos, ponderando su influencia.
- **Función de Puntuación de Atención (Attention Scoring Function):**
 - Calcula una puntuación que indica la importancia de cada elemento en relación con los demás.

Definición de Matrices Entrenables

Se definen tres matrices entrenables que transforman los embeddings de entrada:

- **Query (Q):** Representa la consulta que hacemos sobre otros elementos.
- **Key (K):** Representa la clave que otros elementos usan para comparar con la consulta.
- **Value (V):** Contiene la información que se agregará a la salida.

Cálculo en la Capa de Autoatención

1. Cálculo de Q, K y V:

$$\begin{aligned} \mathbf{Q} &= \mathbf{XW}^Q \\ \mathbf{K} &= \mathbf{XW}^K \\ \mathbf{V} &= \mathbf{XW}^V \end{aligned}$$

Donde:

- \mathbf{X} es la matriz de embeddings de entrada.
- $\mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V$ son matrices de pesos entrenables.

2. Cálculo de Puntuaciones de Atención:

$$\text{Atención}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{QK}^\top}{\sqrt{d_k}}\right) \mathbf{V}$$

Donde d_k es la dimensión de los vectores de clave.

3. Salida:

- Se obtiene un nuevo conjunto de representaciones que incorporan información global de la secuencia.

Multi-Head Attention (Atención Multi-Cabeza)

En lugar de calcular una única función de atención, se utilizan múltiples cabezas de atención para capturar diferentes tipos de relaciones.

• Proceso:

- Se realizan varias operaciones de autoatención en paralelo, cada una con sus propias matrices $\mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V$.
- Las salidas de cada cabeza se concatenan y pasan a través de una capa lineal final.

• Ventajas:

- Permite que el modelo preste atención a diferentes posiciones y relaciones en simultáneo.
- Mejora la capacidad de captura de patrones complejos.

Arquitectura del Transformer

El Transformer está compuesto por la repetición de bloques que combinan mecanismos de autoatención y redes neuronales feed-forward.

Bloque Básico del Transformer

1. **Capa de Multi-Head Attention**
2. **Adición y Normalización:** Residual connection seguida de Layer Normalization.
3. **Capa Feed-Forward:** Red neuronal con una o más capas ocultas.
4. **Adición y Normalización:** Otra residual connection y layer normalization.

Pre-normalización vs. Post-normalización

- **Pre-normalización:** La normalización se aplica antes del bloque.
- **Post-normalización:** La normalización se aplica después del bloque.

Modelo Básico del Transformer

1. Tokenización y Embeddings:

- Se tokeniza y se embeben los datos de entrada.
- Se añaden **positional embeddings** para incorporar información sobre el orden de los tokens.

2. Aplicación de Bloques del Transformer:

- Se aplican uno o más bloques básicos descritos anteriormente.


3. Capa de Salida:

- Dependiendo de la tarea (e.g., clasificación, traducción), se aplica una capa de salida apropiada.

Ejemplo de Aplicación: Traducción Automática

En tareas de traducción:

- **Entrada:** Secuencia de tokens en el idioma de origen.
- **Proceso:**
 - El encoder procesa la secuencia de entrada y genera representaciones internas.
 - El decoder utiliza estas representaciones junto con mecanismos de atención para generar la secuencia traducida.
- **Salida:** Secuencia de tokens en el idioma de destino.

 Transformer en Traducción

Bibliografía Recomendada

- "Alice's Adventures in a Differentiable Wonderland" de Simone Scardapane.

Este recurso proporciona una introducción accesible y amigable a los conceptos detrás de los Transformers y el deep learning.

Conclusiones

Los **Transformers** han revolucionado el campo del procesamiento de lenguaje natural y han sido adaptados a otras áreas como visión por computador y generación de secuencias.

- **Ventajas:**
 - Capturan dependencias a largo plazo de manera efectiva.
 - Paralelización eficiente durante el entrenamiento.
 - Flexibilidad para diversas tareas de secuencias.
- **Avances Recientes:**
 - Modelos como **GPT-3** y **GPT-4** demuestran el poder de los Transformers en generación de texto coherente y tareas complejas.
- **Desafíos:**
 - Alto costo computacional y requerimientos de datos masivos para entrenar grandes modelos.

Este resumen ha explorado los fundamentos de los Transformers, su arquitectura y su importancia en la representación del conocimiento y el razonamiento en secuencias de datos. Comprender estos conceptos es esencial para avanzar en el desarrollo de sistemas de inteligencia artificial capaces de manejar información compleja y secuencial como lenguaje natural y video.