कृत्रिम बुद्धिमत्ता और जनरेटिव एआई के बीच की कड़ी: बुद्धिमत्ता के विष्य की ओर एक यात्रा

प्रोफे. डॉ. एस. मोहन कुमार
प्रो. डॉ. के. पी. यादव

कृत्रिम बुद्धिमत्ता और जनरेटिव एआई के बीच की कड़ी: बुद्धिमत्ता के विष्य की ओर एक यात्रा

प्रोफे. डॉ. एस. मोहन कुमार
प्रो. डॉ. के. पी. यादव

JUPITER PUBLICATIONS CONSORTIUM
INDIA

अनुमति प्राप्त करने के लिए संपर्क करें:
Jupiter Publications Consortium
22/102, द्वितीय स्ट्रीट,
वडापलानी, चेन्नई - 600 092
वेबसाइट: www.jpc.in.net
ईमेल: director@jpc.in.net

आभार:
यहां उल्लिखित कोई भी ट्रेडमार्क, सेवा चिह्न, उत्पाद नाम या ब्रांड उनके संबंधित स्वामियों की संपत्ति माने गए हैं और केवल संदर्भ के उद्देश्य से उपयोग किए गए हैं। यदि ऐसा कोई नाम उपयोग में लिया गया है, तो उसका कोई विशेष समर्थन या अनुमोदन अभिप्रेत नहीं है।

प्रकाशित:
Jupiter Publications Consortium.

20 जून 2025

कृत्रिम बुद्धिमत्ता और जनरेटिव एआई के बीच की कड़ी: बुद्धिमत्ता के विष्य की ओर एक यात्रा

प्रोफे. डॉ. एस. मोहन कुमार

प्रो. डॉ. के. पी. यादव

ISBN: 978-93-86388-78-0
प्रथम प्रकाशन: 25 मार्च 2025
DOI: https://www.doi.org/10.47715/978-93-86388-78-0
मूल्य: ₹375/-
पृष्ठों की संख्या: 272

प्रकाशक:
ज्यूपिटर पब्लिकेशन्स कंसोर्टियम
चेन्नई, तमिलनाडु, भारत
ईमेल: director@jpc.in.net
वेबसाइट: www.jpc.in.net

पुस्तक का नाम: *कृ*

त्रिम बुद्धिमत्ता और जनरेटिव एआई के बीच की कड़ी: बुद्धिमत्ता के विष्य की ओर एक यात्रा

लेखक:

प्रो प्रोफे. डॉ. एस. मोहन कुमार

प्रो. डॉ. के. पी. यादव

ISBN: 978-93-86388-78-0
खंड: I
संस्करण: प्रथम

मुद्रित एवं प्रकाशित:
ज्यूपिटर पब्लिकेशन्स कंसोर्टियम,
चेन्नई, भारत
ईमेल: director@jpc.in.net
वेबसाइट: www.jpc.in.net

ज्यूपिटर पब्लिकेशन्स कंसोर्टियम
22/102, सेकेंड स्ट्रीट, वडुगामबक्कम, चेन्नई – 600092
ईमेल: director@jpc.in.net. **वेबसाइट:** www.jpc.in.net

लेखक परिचय

Prof. (Dr.) S. Mohan Kumar

M.Tech.[Software Engineering]., MBA., PhD [CSE-Medical Diagnosis CAD System]., Ph.D[CSE- Medical Imaging -Machine Learning]., Post Doctorate Degree D.Sc. Engg.[Deep learning]., EPLM (IIM-Calcutta).,

Dean, Indra Ganesan College of Engineering

(Autonomous- Higher Educational Research Institution), NAAC Accredited, Indra Ganesan Group of Institutions, Trichy, Tamil Nadu, India.)

लेखक परिचय

प्रो. (डॉ.) के. पी. यादव
कुलपति, मात्स विश्वविद्यालय

संक्षिप्त प्रोफ़ाइल: प्रो. (डॉ.) के. पी. यादव (विश्व गुरु)

शैक्षणिक योग्यता: B. Tech., PGDBM, M. Tech., Ph.D., Post Doctorate (D.Sc., D.Litt.-Hon.),
तथा अंतरराष्ट्रीय विश्वविद्यालयों/संस्थानों से डॉक्टरेट उपाधियाँ (मानद) – इंजीनियरिंग, जीवन एवं स्वास्थ्य विज्ञान, शिक्षा एवं प्रबंधन, मानविकी, कृषि, विधि आदि क्षेत्रों में।

वर्तमान पद:

कुलपति, MATS विश्वविद्यालय (NAAC A+ ग्रेड), रायपुर, छत्तीसगढ़ (माननीय राज्यपाल द्वारा नियुक्त) – द्वितीय कार्यकाल।

राष्ट्रीय एवं अंतरराष्ट्रीय दायित्व:

माननीय राष्ट्रपति / विज़िटर के प्रतिनिधि के रूप में:

- भारतीय सूचना प्रौद्योगिकी संस्थान (IIIT), तिरुचिरापल्ली, तमिलनाडु
- नागालैंड केंद्रीय विश्वविद्यालय, लुमामी

माननीय छत्तीसगढ़ राज्यपाल के नामित सदस्य:

- महात्मा गांधी बागवानी एवं वानिकी विश्वविद्यालय, दुर्ग/रायपुर
- अटल बिहारी वाजपेयी विश्वविद्यालय, बिलासपुर

अन्य प्रमुख भूमिकाएँ:

- **देश प्रमुख (भारत):** Yesbud University, लुसाका, जाम्बिया (कुलपति समकक्ष)
- **सलाहकार:** KLCU University, साउथ कैरोलिना, USA

प्रमुख उपलब्धियाँ एवं योगदान:

प्रो. यादव ने उच्च शिक्षा में गुणवत्ता, नवाचार, अनुसंधान एवं तकनीकी उन्नयन को बढ़ावा देने हेतु AI, डेटा साइंस, मशीन लर्निंग इत्यादि में उन्नत प्रयोगशालाओं की स्थापना की है। उन्होंने विभिन्न संस्थानों में आधुनिक पाठ्यक्रम प्रारंभ किए तथा NAAC/NBA/ICAR जैसी मान्यताओं के माध्यम से संस्थागत गुणवत्ता में उल्लेखनीय वृद्धि की।

विशेष दायित्व एवं सदस्यता:

- **मार्गदर्शक:** AICTE, नई दिल्ली एवं राजस्थान तकनीकी विश्वविद्यालय, कोटा
- **सलाहकार:** हिंदी साहित्य भारती (राजस्थान), SSPITM, रायपुर
- **सदस्य:** इंटीग्रल यूनिवर्सिटी, लखनऊ (NAAC A+), YBN यूनिवर्सिटी, रांची
- **संरक्षक:** महात्मा गांधी इंटरनेशनल पीस सेंटर
- **पीस एम्बेसडर:** वर्ल्ड वाइड पीस ऑर्गनाइजेशन, USA
- **विशेषज्ञ:** AICTE, NAAC, NBA, ICAR, UPSC, PSC आदि के लिए पैनल सदस्य
- **प्रमुख सदस्य:**
    - राष्ट्रीय शिक्षक पुरस्कार चयन समिति (2020–2021)
    - नई शिक्षा नीति (NEP 2020) कार्यान्वयन समिति
    - PPP मॉडल उच्च शिक्षा समिति (विशेष आमंत्रित सदस्य)

अनुभव:

उच्च शिक्षा, प्रशासन, शिक्षण एवं शोध के क्षेत्र में **29+ वर्षों** का समृद्ध अनुभव। NCR क्षेत्र के प्रमुख संस्थानों में निदेशक तथा संगम विश्वविद्यालय, राजस्थान में कुलपति एवं अध्यक्ष के रूप में कार्य कर चुके हैं।

प्रकाशन एवं शोध कार्य:

- **53 पुस्तकें** प्रकाशित
- **200+ शोध पत्र** (SCI, Scopus, UGC व रेफ़री जर्नल्स में)
- **22 पीएच.डी.**, **4 पोस्ट-डॉक्टोरल** एवं **5 डी.एससी.** शोधार्थियों का निर्देशन
- कई IAS, IPS, IFS, न्यायिक अधिकारी आदि को डॉक्टरेट शोध निर्देशन प्रदान किया

उल्लेखनीय गतिविधियाँ:

- **150+ राष्ट्रीय/अंतरराष्ट्रीय कार्यशालाओं**, सेमिनारों व व्याख्यानों में वक्ता
- दीक्षांत समारोहों का संचालन
- UGC 12(B)/2(F), OBE, Design Thinking, Industry Integration जैसे विषयों में विशेषज्ञता

उद्योग एवं सरकारी परियोजनाएँ:

- MSME, DST, ICAR, ICSSR द्वारा वित्तपोषित कई परियोजनाओं में सक्रिय सहभागिता
- **19 पेटेंट/कॉपीराइट:** माट्स ऑर्थो ऑयल, ओरल गम, एंटी-डायबिटिक टैबलेट आदि नवाचारों पर

सम्मान एवं पुरस्कार:

- **Nobel Laureate Award 2024** (Yesbud University)
- **Most Influential Vice Chancellor Award 2021**
- **National Education Excellence Award**
- **IEEE International Elite Academician Award 2022**
- **ASSOCHAM Excellence in Research Innovation 2022**
- **Outstanding Vice Chancellor of India 2025**
- **Global Guru Award 2021** (USA, Nepal)
- **Life Time International Research Award 2022**
- अन्य कई **राष्ट्रीय एवं अंतरराष्ट्रीय सम्मान**

प्रो. यादव पर आधारित प्रकाशित पुस्तकें (अन्य लेखकों द्वारा):

1. **मूर्त से अमूर्त तक** – डॉ. ए.के. जौहरी
2. **The Diamond of Earth** – प्रो. (डॉ.) हरविंदर कुमार पटेल
3. **The Perfect Trailblazer** – डॉ. संदीप कुलकर्णी
4. **करुणा से प्रताप की ओर** – डॉ. प्रदीप कुमार
5. **Vision and Synergy** – प्रो. (डॉ.) ए.के. त्रिपाठी
6. **A Man of Vision and Mission** – डॉ. अंकित मिश्रा
7. **A Nation's Inspiration** – प्रो. (डॉ.) एस. मोहन कुमार
8. **राष्ट्र नायक: प्रो. डॉ. के. पी. यादव** – प्रो. (डॉ.) एस. मोहन कुमार

प्रस्तावना (Preface)

कृत्रिम बुद्धिमत्ता (AI) और जनरेटिव एआई वर्तमान तकनीकी युग के दो ऐसे स्तंभ हैं, जो मानव बुद्धि और मशीन क्षमताओं के समन्वय को एक नई दिशा प्रदान कर रहे हैं। यह पुस्तक विशेष रूप से तकनीकी छात्रों, शोधकर्ताओं, और नवाचार-प्रेमियों के लिए तैयार की गई है, ताकि वे AI की नींव से लेकर जनरेटिव एआई की उच्चतम अवधारणाओं तक की यात्रा में सक्षम मार्गदर्शन प्राप्त कर सकें।

यूनिट 1: परिचय और बुद्धिमान एजेंट्स

इस इकाई में कृत्रिम बुद्धिमत्ता की मूलभूत परिभाषा, ऐतिहासिक विकास, और बुद्धिमान एजेंट्स की अवधारणा को स्पष्ट किया गया है। साथ ही इसमें एजेंट्स और उनके कार्य पर्यावरण की गहराई से व्याख्या की गई है, जो AI प्रणालियों की आधारभूत समझ के लिए अनिवार्य है।

यूनिट 2: खोज के माध्यम से समस्या समाधान

यह खंड समस्या-सुलझाने वाले एजेंट्स और विविध खोज तकनीकों जैसे ब्रेड्थ-फर्स्ट सर्च, A*, और हीयूरिस्टिक खोज पर केंद्रित है। यह भाग छात्रों को समस्याओं का तर्कसंगत समाधान निकालने की दिशा में सशक्त बनाता है।

यूनिट 3: ज्ञान प्रतिनिधित्व और तर्क

इस इकाई में तर्कशास्त्र, प्रस्तावनात्मक और प्रथम-क्रम तर्क, तथा ज्ञान के प्रभावी प्रतिनिधित्व की विधियों का समावेश है। यह खंड मशीनों को "सोचने" के तरीके को समझने में मदद करता है।

यूनिट 4: अधिगम (Learning)

यह भाग मशीन लर्निंग, सुपरवाइज्ड लर्निंग, निर्णय वृक्ष, रैग्रेशन, और कृत्रिम न्यूरल नेटवर्क जैसे विषयों से संबंधित है। यह AI को अनुभव से सीखने और आत्म-सुधार की दिशा में अग्रसर करता है।

यूनिट 5: एआई के अनुप्रयोग

यह इकाई AI के विभिन्न अनुप्रयोगों जैसे प्राकृतिक भाषा प्रसंस्करण, वाक् पहचान, कंप्यूटर विज़न और रोबोटिक्स की व्यावहारिक समझ प्रदान करती है। यहाँ पाठक यह जान सकते हैं कि AI कैसे हमारे दैनिक जीवन को प्रभावित कर रहा है।

यूनिट 6: जनरेटिव एआई – बुद्धिमान प्रणालियों का भविष्य

यह अंतिम खंड जनरेटिव एआई की परिभाषा, इसकी तकनीकों, अनुप्रयोगों और इंडस्ट्री 5.0 में इसकी भूमिका को उजागर करता है। इसमें GPT, GANs, और अन्य मॉडल्स की सहायता से

सृजनात्मक क्षमता की विवेचना की गई है, जो एआई को केवल समस्या-समाधान से आगे बढ़ाकर कल्पना और नवाचार के स्तर पर ले जाती है।

निष्कर्षतः, यह पुस्तक केवल सैद्धांतिक ज्ञान नहीं, बल्कि व्यवहारिक दृष्टिकोण और नैतिक मंथन को भी साथ लेकर चलती है, ताकि भविष्य के एआई विशेषज्ञ तकनीकी रूप से सशक्त और मानवीय दृष्टिकोण से संतुलित हों।

– प्रो. (डॉ.) एस. मोहन कुमार

– प्रो. (डॉ.) के. पी. यादव

"Generative AI is not just a tool; it's a collaborator that expands the boundaries of human creativity."

"जनरेटिव एआई केवल एक उपकरण नहीं, बल्कि मानव रचनात्मकता की सीमाओं को विस्तार देने वाला एक सहयोगी है।"

"The future will not be created by code alone, but by those who teach machines to imagine."

"भविष्य केवल कोड से नहीं बनेगा, बल्कि उन लोगों से बनेगा जो मशीनों को कल्पना करना सिखाते हैं।"

सारांश (Abstract):

यह पुस्तक कृत्रिम बुद्धिमत्ता (AI) और जनरेटिव एआई के सिद्धांत, तकनीक और अनुप्रयोगों की एक व्यापक व्याख्या प्रस्तुत करती है। प्रारंभिक अध्यायों में AI की मूल अवधारणाएँ, एजेंट्स, खोज रणनीतियाँ और ज्ञान प्रतिनिधित्व की गहराई से चर्चा की गई है, जो बुद्धिमत्ता की नींव को स्पष्ट करती हैं। इसके बाद, मशीन लर्निंग, निर्णय वृक्ष, और न्यूरल नेटवर्क्स जैसे आधुनिक अधिगम तकनीकों को शामिल किया गया है, जो AI की आत्म-सीखने की क्षमताओं को दर्शाते हैं। अंतिम इकाइयाँ जनरेटिव एआई की परिकल्पना, तकनीकी ढाँचा और रचनात्मक अनुप्रयोगों पर केंद्रित हैं, जैसे GPT, DALL·E, और GANs, जो मशीनों को कल्पनाशक्ति की दिशा में सक्षम बनाते हैं। यह पुस्तक छात्रों, शोधकर्ताओं और नवाचार के क्षेत्र में काम करने वालों के लिए एक संपूर्ण मार्गदर्शक है।

कीवर्ड्स **(Keywords):** *कृत्रिम बुद्धिमत्ता, जनरेटिव एआई, बुद्धिमान एजेंट्स, समस्या समाधान, मशीन लर्निंग, निर्णय वृक्ष, न्यूरल नेटवर्क्स, प्राकृतिक भाषा प्रसंस्करण, कंप्यूटर विज़न, GPT, GANs, DALL·E, एजेंट-आधारित प्रणालियाँ, इंडस्ट्री 5.0, स्वचालन, रचनात्मकता, तर्कशक्ति, अधिगम एल्गोरिद्म*

"Innovation begins when machines start understanding context, not just commands."

"नवाचार वहीं से शुरू होता है जहाँ मशीनें केवल आदेश नहीं, संदर्भ समझने लगती हैं।"

"Generative AI doesn't replace creativity — it redefines what's possible with it."

"जनरेटिव एआई रचनात्मकता का स्थान नहीं लेता, बल्कि उसकी संभावनाओं को फिर से परिभाषित करता है।"

Table of Contents

"Generative AI is not just about automation — it's about amplifying human imagination with machine intelligence."

"जनरेटिव एआई केवल स्वचालन के बारे में नहीं है — यह मानव कल्पना को मशीन बुद्धिमत्ता से सशक्त बनाने का माध्यम है।"

"In the age of Generative AI, creativity is no longer limited by tools, but expanded by them."

"जनरेटिव एआई के युग में, रचनात्मकता अब उपकरणों से सीमित नहीं, बल्कि उनके माध्यम से विस्तारित होती है।"

काई-1: परिचय

1.1 कृत्रिम बुद्धिमत्ता (Artificial Intelligence) क्या है?

चित्र 1.1 कृत्रिम बुद्धिमत्ता (एआई) की अवधारणा को दर्शाता है, जिसमें मशीन लर्निंग, न्यूरल नेटवर्क्स, रोबोटिक्स, डाटा विश्लेषण और प्राकृतिक भाषा संसाधन जैसे तत्वों को स्वच्छ और न्यूनतम शैली में सम्मिलित किया गया है।

कृत्रिम बुद्धिमत्ता (AI) कंप्यूटर विज्ञान की एक ऐसी शाखा है जिसका उद्देश्य ऐसे सिस्टम विकसित करना है जो उन कार्यों को कर सकें जिन्हें सामान्यतः मानव बुद्धि की आवश्यकता होती है। इनमें सीखना, तर्क करना, समस्या समाधान, बोध, भाषा की समझ और रचनात्मकता जैसे कार्य शामिल हैं। AI का लक्ष्य कंप्यूटर सिस्टम की मदद से मानव विचार प्रक्रियाओं की नकल करना है। यह क्षेत्र मनोविज्ञान, दर्शनशास्त्र, भाषाविज्ञान, गणित और तंत्रिका विज्ञान जैसे विषयों के साथ घनिष्ठ रूप से जुड़ा हुआ है, जिससे बुद्धिमत्ता को समझने और बुद्धिमान मशीनों के निर्माण के लिए बहु-विषयक दृष्टिकोण मिलता है।

1.1.1 AI की आधारशिलाएं

AI निम्नलिखित प्रमुख स्तंभों पर आधारित है:

- **कंप्यूटर विज्ञान**: एआई सिस्टम्स के लिए हार्डवेयर, सॉफ्टवेयर, एल्गोरिदम और डाटा संरचनाएं प्रदान करता है।
- **गणित और सांख्यिकी**: समस्याओं को मॉडल करने और AI एल्गोरिदम के प्रदर्शन का मूल्यांकन करने के लिए औपचारिक उपकरण प्रदान करते हैं, जैसे कैलकुलस, रैखिक बीजगणित, संभाव्यता और अनुकूलन।
- **संज्ञानात्मक विज्ञान (Cognitive Science)**: मानव मस्तिष्क और बुद्धि का अध्ययन करता है, जिससे मशीनों में मानवीय सोच को पुनः प्रस्तुत करने की समझ मिलती है।
- **भाषाविज्ञान (Linguistics)**: प्राकृतिक भाषा संसाधन को समझने में मदद करता है, जिससे मशीनें मानव भाषा को समझ और उत्पन्न कर सकती हैं।
- **दर्शनशास्त्र**: बुद्धिमत्ता और चेतना की प्रकृति की जांच करता है, जिससे एआई की वैचारिक नींव मजबूत होती है।

1.1.2 AI के प्रकार

AI को विभिन्न तरीकों से वर्गीकृत किया जा सकता है, लेकिन एक सामान्य विभाजन निम्नलिखित है:

- **संकुचित AI (Narrow AI या Weak AI)**: ऐसे सिस्टम जो किसी एक विशिष्ट कार्य के लिए बनाए और प्रशिक्षित किए जाते हैं। जैसे वर्चुअल असिस्टेंट्स (Siri, Alexa), इमेज रिकग्निशन सॉफ्टवेयर, और सिफारिशी प्रणालियाँ। ये सीमित और पूर्वनिर्धारित संदर्भों में कार्य करते हैं।
- **सामान्य AI (General AI या Strong AI)**: सैद्धांतिक सिस्टम जो मानवों की तरह विभिन्न संदर्भों में समझने, सीखने और ज्ञान को लागू करने में सक्षम हों। ये सिस्टम सामान्य बुद्धिमत्ता वाले कार्यों को कर सकते हैं और भविष्य के अनुसंधान का लक्ष्य हैं।

1.1.3 AI के मुख्य क्षेत्र

AI में कई तकनीकों और पद्धतियों को शामिल किया गया है, जिनमें प्रमुख हैं:

- **मशीन लर्निंग (Machine Learning)**: यह सिस्टम्स को डाटा से सीखने और समय के साथ अपने प्रदर्शन को बेहतर बनाने की क्षमता देता है, बिना स्पष्ट रूप से प्रोग्राम किए।
- **डीप लर्निंग**: मशीन लर्निंग की एक उपशाखा जो बहुस्तरीय (गहरे) न्यूरल नेटवर्क्स का उपयोग कर इमेज, आवाज आदि जैसे जटिल डाटा का विश्लेषण करती है।

- **प्राकृतिक भाषा संसाधन (NLP)**: मशीनों को मानव भाषा को समझने और उपयोग करने की क्षमता देता है, जिससे अनुवाद, चैटबॉट्स, और भावनात्मक विश्लेषण जैसे अनुप्रयोग संभव होते हैं।
- **रोबोटिक्स**: यांत्रिक इंजीनियरिंग के साथ AI को जोड़कर ऐसे रोबोट्स बनाए जाते हैं जो कार्य कर सकते हैं या मानवीय क्रियाओं की नकल कर सकते हैं।
- **कंप्यूटर विज़न**: मशीनों को दृश्य संसार को समझने और उसके आधार पर निर्णय या क्रिया करने में सक्षम बनाता है।

1.1.4 AI के अनुप्रयोग

AI के अनुप्रयोग व्यापक और तीव्र गति से विकसित हो रहे हैं, जो अनेक क्षेत्रों में क्रांतिकारी परिवर्तन ला रहे हैं:

- **पूर्वानुमानात्मक विश्लेषण**: डाटा का विश्लेषण करके भविष्य की प्रवृत्तियों और पैटर्न का अनुमान लगाना।
- **स्वचालित वाहन (Autonomous Vehicles)**: ऐसे वाहन जो सेंसर डाटा का विश्लेषण करके स्वयं निर्णय लेते हुए चल सकते हैं।
- **स्वास्थ्य सेवा**: बीमारियों की पहचान, उपचार की सिफारिश और रोगी के परिणामों की भविष्यवाणी में AI की मदद ली जाती है।
- **स्मार्ट असिस्टेंट्स**: ऐसे उपकरण और एप्लिकेशन जो आवाज़ पहचान कर कार्य करते हैं।

1.1.5 नैतिक पहलू और भविष्य की दिशा

AI के निरंतर विकास के साथ कुछ महत्वपूर्ण नैतिक प्रश्न भी उठते हैं, जैसे गोपनीयता, सुरक्षा, रोजगार पर प्रभाव, और निर्णय-प्रक्रियाओं में पारदर्शिता की आवश्यकता। भविष्य में AI का विकास केवल तकनीकी नहीं बल्कि इन नैतिक चुनौतियों का समाधान करके संतुलित और लाभकारी दिशा में होना चाहिए, जिससे यह मानवता के लिए सुरक्षित और उपयोगी बना रहे।

1.1.6 निष्कर्ष

कृत्रिम बुद्धिमत्ता (AI) मशीनों को सीखने, तर्क करने, बोध करने, अनुमान लगाने, संवाद करने और कार्य करने की क्षमता प्रदान कर एक क्रांतिकारी परिवर्तन का प्रतीक है। इसका मुख्य उद्देश्य ऐसे एल्गोरिद्म और सिस्टम विकसित करना है जो मानवीय संज्ञानात्मक क्रियाओं की नकल कर सकें। मशीन लर्निंग, न्यूरल नेटवर्क्स, रोबोटिक्स और NLP जैसी तकनीकों के माध्यम से AI ने लगभग हर क्षेत्र में अपना प्रभाव दिखाया है, जिससे न केवल उद्योगों में दक्षता बढ़ी है बल्कि नवाचार के नए द्वार भी खुले हैं।

AI अब केवल स्वचालन का उपकरण नहीं रहा, बल्कि यह जटिल समस्याओं के समाधान, व्यक्तिगत अनुभव प्रदान करने और ऐसी चुनौतियों का समाधान करने में सक्षम है जो पूर्व में संभव नहीं थीं। हालांकि, इसके साथ गोपनीयता, नैतिकता और सुरक्षा जैसे विषयों पर ध्यान देना अनिवार्य है।

अतः, AI न केवल एक तकनीकी उपलब्धि है, बल्कि यह मानवीय प्रयासों की परिभाषा को पुनः गढ़ने वाला उत्प्रेरक है, जो भविष्य में मनुष्यों के साथ मिलकर एक अधिक जुड़ा हुआ, कुशल और बुद्धिमान संसार रच सकता है।

1.2 AI की आधारशिलाएं

कृत्रिम बुद्धिमत्ता (AI) की नींव कई शैक्षणिक क्षेत्रों में गहराई से जुड़ी हुई है, जिन्होंने इसके विकास और प्रगति में महत्वपूर्ण योगदान दिया है। इन आधारों को समझना यह जानने में मदद करता है कि AI कैसे कार्य करता है, इसकी क्षमताएं क्या हैं, और इसकी सीमाएं क्या हैं। नीचे AI की प्रगति में योगदान देने वाले प्रमुख विषयों का विस्तृत विवरण दिया गया है:

चित्र 1.2: वे विषय जिनकी नींव पर कृत्रिम बुद्धिमत्ता (AI) का विकास आधारित है

1.2.1 गणित (Mathematics)

AI का मूल आधार गणित है, जो एल्गोरिदम, तर्क, संभाव्यता और सांख्यिकीय विश्लेषण के लिए सैद्धांतिक ढांचा प्रदान करता है। प्रमुख क्षेत्र हैं:

- **रेखीय बीजगणित (Linear Algebra)**: डाटा का प्रतिनिधित्व, रूपांतरण और न्यूरल नेटवर्क्स में संचालन हेतु।
- **कलन (Calculus)**: एल्गोरिद्म के व्यवहार में परिवर्तन को समझने और मशीन लर्निंग मॉडलों का अनुकूलन करने के लिए।
- **संभाव्यता और सांख्यिकी (Probability and Statistics)**: अपूर्ण या अनिश्चित जानकारी के आधार पर पूर्वानुमान और निर्णय लेने के लिए।
- **सांतत्य गणित (Discrete Mathematics)**: तर्क, ग्राफ सिद्धांत और एल्गोरिद्मिक विश्लेषण के लिए।

1.2.2 कंप्यूटर विज्ञान और अभियांत्रिकी (Computer Science and Engineering)

AI प्रणालियों के व्यावहारिक क्रियान्वयन में कंप्यूटर विज्ञान और इंजीनियरिंग का गहरा योगदान है:

- **एल्गोरिद्म डिज़ाइन**: बड़े डाटा सेट्स का विश्लेषण और निर्णय लेने हेतु कुशल एल्गोरिद्म विकसित करना।
- **सॉफ्टवेयर विकास**: AI अनुप्रयोगों का समर्थन करने वाले सॉफ्टवेयर उपकरण और प्लेटफॉर्म बनाना।
- **हार्डवेयर इंजीनियरिंग**: GPU, TPU जैसे हार्डवेयर का निर्माण जो AI गणनाओं को कुशलता से संचालित कर सकें।

1.2.3 संज्ञानात्मक विज्ञान (Cognitive Science)

यह मानव मस्तिष्क और उसकी प्रक्रियाओं (सोचना, सीखना, समझना, संवाद करना) का अध्ययन करता है, जिससे मानवीय बुद्धिमत्ता की नकल करने वाली AI प्रणालियाँ विकसित होती हैं:

- **प्राकृतिक भाषा संसाधन (NLP)**: कंप्यूटर को मानव भाषा समझने, व्याख्या करने और उत्पन्न करने की क्षमता देना।
- **कंप्यूटर दृष्टि (Computer Vision)**: मशीनों को दृश्य जानकारी की व्याख्या करने में सक्षम बनाना।

1.2.4 दर्शनशास्त्र (Philosophy)

AI में दर्शनशास्त्र बौद्धिक, नैतिक और चेतना से जुड़े मौलिक प्रश्नों को संबोधित करता है:

- **मन-शरीर समस्या (Mind-Body Problem)**: भौतिक अवस्था और चेतना के संबंध को समझना।
- **AI की नैतिकता (Ethics of AI)**: AI के निर्णयों और कार्यों के नैतिक परिणामों पर विचार करना।

1.2.5 मनोविज्ञान (Psychology)

मानव व्यवहार, सीखने और स्मृति की मनोवैज्ञानिक समझ से AI को प्रेरणा मिलती है:

- **सीखने के एल्गोरिद्म (Learning Algorithms)**: मानव और पशु सीखने के सिद्धांतों पर आधारित।
- **मानव-कंप्यूटर अंतःक्रिया (Human-Computer Interaction)**: सहज और उपयोगकर्ता-अनुकूल AI सिस्टम बनाना।

1.2.6 भाषाविज्ञान (Linguistics)

AI में भाषाविज्ञान की भूमिका कंप्यूटर को मानव भाषा संसाधित करने में सक्षम बनाना है:

- **वाक्य रचना (Syntax)**: वाक्य की संरचना को समझना।
- **ार्थविज्ञान (Semantics)**: शब्दों और वाक्यों के अर्थ को समझना।
- **प्रयोजनोन्मुखता (Pragmatics)**: संदर्भ के अनुसार भाषा की व्याख्या।

1.2.7 तर्कशास्त्र (Logic)

AI प्रणालियों के लिए निर्णय, निष्कर्ष और समस्या समाधान हेतु तर्कशास्त्र आवश्यक है:

- **उपपत्ति और गणनात्मक तर्क (Propositional and Predicate Logic)**: औपचारिक तर्क के लिए।
- **फ़ज़ी लॉजिक (Fuzzy Logic)**: अपरिभाषित या अनुमानात्मक परिस्थितियों में तर्क करने के लिए।

1.2.8 अर्थशास्त्र (Economics)

AI में निर्णय-निर्माण और अनिश्चितता में अनुकूलन हेतु अर्थशास्त्र विशेष रूप से सहायक है:

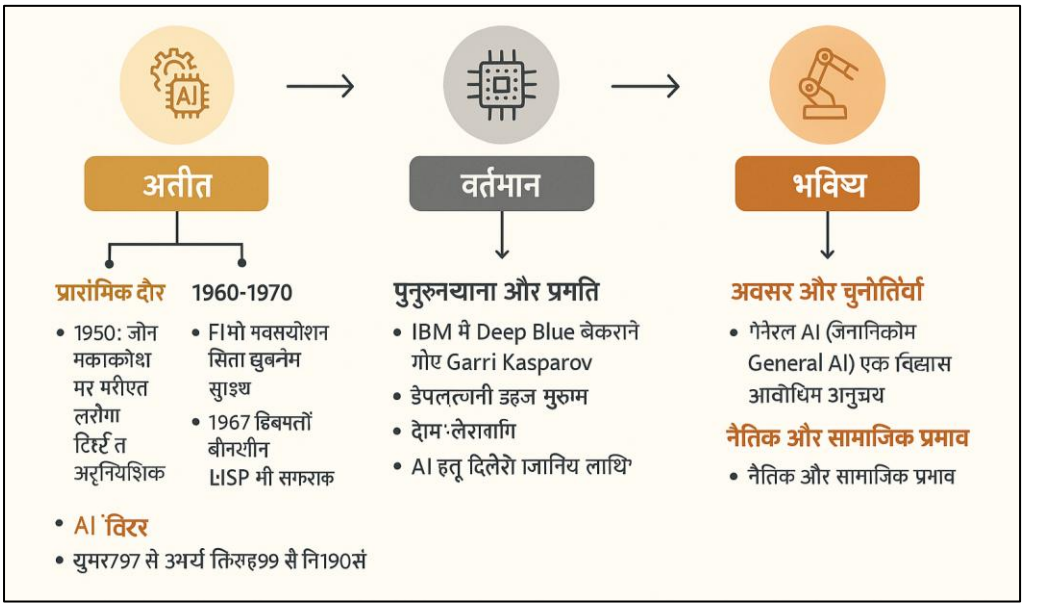
- **खेल सिद्धांत (Game Theory)**: प्रतिस्पर्धी परिस्थितियों में रणनीतिक निर्णय के लिए।
- **यांत्रिक संरचना (Mechanism Design)**: ऐसी प्रणालियाँ बनाना जो व्यक्तिगत हितों को सामूहिक उद्देश्यों से जोड़ सकें।

1.2.9 निष्कर्ष

AI की अंतर्विषयक आधारशिलाएं इसकी जटिलता और व्यापक संभावनाओं को दर्शाती हैं। गणित, कंप्यूटर विज्ञान, संज्ञानात्मक विज्ञान और अन्य विषयों से प्रेरणा लेकर AI निरंतर विकसित हो रहा है, और यह जानना कि इसकी नींव कहाँ-कहाँ से आती है, AI की क्षमताओं और सीमाओं को समझने के लिए अत्यंत आवश्यक है।

1.3 इतिहास: कृत्रिम बुद्धिमत्ता – अतीत, वर्तमान और भविष्य

कृत्रिम बुद्धिमत्ता (AI) का इतिहास एक रोमांचक यात्रा है, जो इसके सैद्धांतिक आरंभ से लेकर आज की अत्याधुनिक प्रगति और संभावनाओं तक फैली हुई है। इस इतिहास को समझना न केवल AI के विकास को रेखांकित करता है, बल्कि इसके भविष्य की दिशा का पूर्वानुमान लगाने में भी मदद करता है।



चित्र 1.3: कृत्रिम बुद्धिमत्ता का ऐतिहासिक विकास

चित्र 1.3 का विवरण

यह चित्र कृत्रिम बुद्धिमत्ता (AI) के ऐतिहासिक विकास को तीन कालखंडों में दर्शाता है: **अतीत**, **वर्तमान**, और **भविष्य**।

अतीत (1950–1990)

- **प्रारंभिक युग**:
  - 1950 में एलन ट्यूरिंग ने ट्यूरिंग टेस्ट प्रस्तुत किया।
  - 1956 में डार्टमाउथ सम्मेलन में AI शब्द की उत्पत्ति हुई (जॉन मैकार्थी)।
- **1960-1970**:
  - LISP और Prolog जैसी AI प्रोग्रामिंग भाषाओं का विकास।

    - विशेषज्ञ प्रणालियाँ विकसित हुईं।
- **AI विंटर**:
    - अत्यधिक अपेक्षाओं और तकनीकी सीमाओं के कारण 1970 के उत्तरार्ध से 1990 के दशक तक रुचि और निवेश में गिरावट आई।

वर्तमान (1990 के बाद से)

- **पुनरुत्थान और सफलताएँ**:
    - 1997 में IBM के Deep Blue ने Garry Kasparov को हराया।
    - Deep Learning और NLP ने AI को मुख्यधारा में पहुंचाया।
    - AlphaGo ने 2016 में विश्व Go चैंपियन को हराया।
- **AI का दैनिक जीवन में समावेश**:
    - वर्चुअल असिस्टेंट्स, सिफारिश प्रणालियाँ, स्वास्थ्य सेवाओं में उपयोग।

भविष्य

- **अवसर**:
    - जनरल AI का विकास, जो मानव जैसी सामान्य बुद्धिमत्ता प्रदर्शित कर सके।
- **चुनौतियाँ**:
    - नैतिकता, गोपनीयता, और रोजगार पर प्रभाव।
    - AI के सामाजिक परिणामों को संतुलित करना।

यह चित्र AI की ऐतिहासिक यात्रा को सरल, क्रमबद्ध और दृश्य रूप में प्रस्तुत करता है — शिक्षा और प्रस्तुति दोनों के लिए उपयुक्त।

1.3.1 अतीत (The Past)

- **प्रारंभिक दौर**
    - **1950 का दशक**: "कृत्रिम बुद्धिमत्ता" शब्द का पहली बार प्रयोग *जॉन मैकार्थी* द्वारा 1956 में *डार्टमाउथ सम्मेलन* (Dartmouth Conference) में किया गया था। इस दौर में एलन ट्यूरिंग ने **"ट्यूरिंग टेस्ट"** प्रस्तुत किया, जिसका उद्देश्य यह जांचना था कि क्या कोई मशीन ऐसी बुद्धिमत्ता प्रदर्शित कर सकती है जो मनुष्य जैसी हो या उससे अलग न लगती हो।

- **1960–1970 का दशक**:

  इस समय AI शोध में तीव्र प्रगति हुई। AI की पहली भाषाएँ जैसे **LISP** और **Prolog** विकसित की गईं।

  विशेषज्ञ प्रणाली (Expert Systems) भी बनाई गईं जो मानव विशेषज्ञों की निर्णय लेने की क्षमता की नकल करती थीं।

  इस अवधि में सरकारी संस्थानों से भारी मात्रा में धन प्राप्त हुआ और भविष्य को लेकर अत्यधिक आशावाद था।

AI विंटर (AI Winters)

- **1970 के उत्तरार्ध से 1990 के प्रारंभ तक**:

  यह समय **"AI विंटर"** के नाम से जाना जाता है — ऐसी अवधि जब फंडिंग में कटौती, सार्वजनिक रुचि में कमी और तकनीकी सीमाओं ने क्षेत्र की गति को धीमा कर दिया। यह एहसास हुआ कि मानव-स्तरीय बुद्धिमत्ता की नकल करना जितना सोचा गया था, उससे कहीं अधिक जटिल है।

1.3.2 वर्तमान (The Present)

पुनरुत्थान और प्रगति

- **1990 के दशक के अंत से वर्तमान तक**:

  कंप्यूटर हार्डवेयर में सुधार, बड़े डाटा की उपलब्धता और मशीन लर्निंग में उन्नति के चलते AI का पुनरुत्थान हुआ। कुछ प्रमुख मील के पत्थर:

  - **1997**: IBM का **Deep Blue** विश्व चेस चैंपियन **गैरी कास्पारोव** को हराने वाला पहला कंप्यूटर बना।
  - **डीप लर्निंग**: इस तकनीक ने छवि और ध्वनि पहचान, प्राकृतिक भाषा प्रसंस्करण और स्वचालित वाहनों में क्रांति ला दी।
  - **2016**: Google DeepMind का **AlphaGo** विश्व गो चैंपियन को हराकर एक ऐतिहासिक उपलब्धि हासिल करता है।

AI का दैनिक जीवन में समावेशन

- आज AI हमारे जीवन का अभिन्न हिस्सा बन चुका है — **Siri** और **Alexa** जैसे व्यक्तिगत सहायक, **Netflix** और **Amazon** की सिफारिश प्रणाली, साथ ही स्वास्थ्य सेवा, वित्त, लॉजिस्टिक्स और अन्य उद्योगों में इसका प्रभाव स्पष्ट है।

1.3.3 भविष्य (The Future)

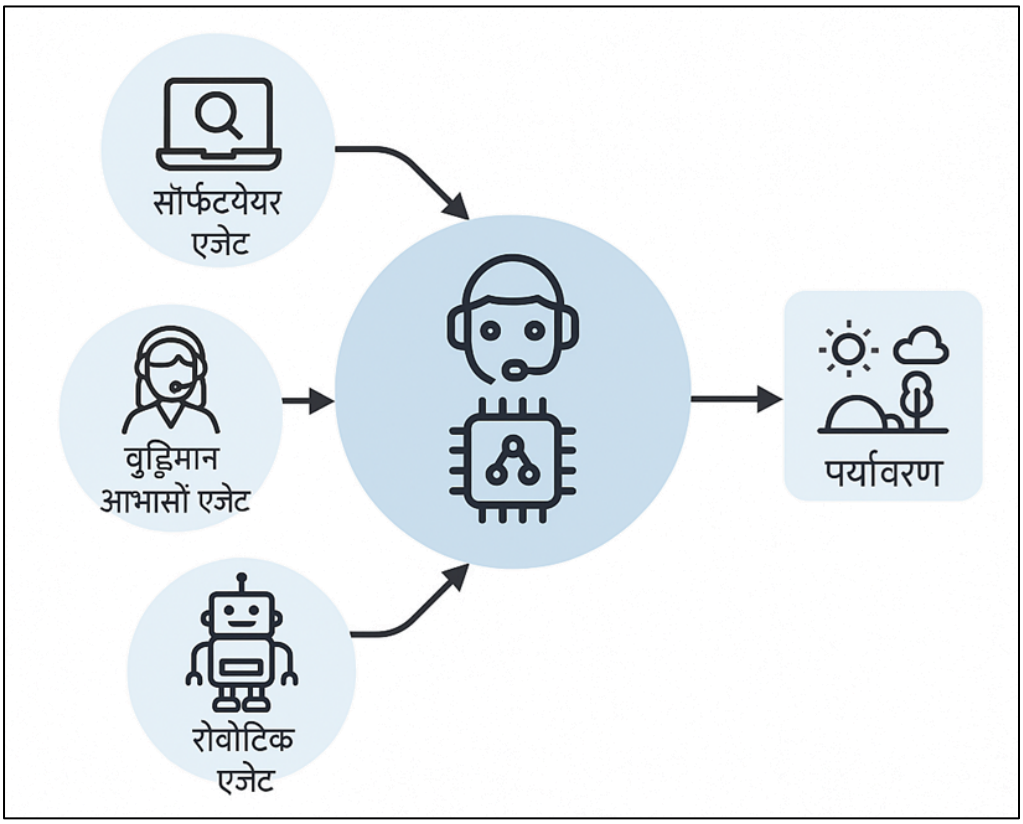
- **अवसर और चुनौतियाँ**
  - **जनरल AI (AGI)**: ऐसे AI सिस्टम्स का विकास जो विभिन्न कार्यों में मानव-स्तर की समझ, सीखने की क्षमता और प्रदर्शन कर सकें।
  - **नैतिक और सामाजिक प्रभाव**:
    जैसे-जैसे AI समाज में गहराई से प्रवेश करता है, गोपनीयता, नैतिकता, और रोजगार की स्थिति जैसे मुद्दों पर विचार करना अनिवार्य होगा।
- **तकनीकी प्रगति**
  - AI के भविष्य को **उन्नत एल्गोरिद्म**, **कंप्यूटिंग शक्ति**, और संभवतः **क्वांटम कंप्यूटिंग** आकार देंगे।
  - **मानव-AI सहयोग** बढ़ेगा, जिससे नई किस्म की सहभागिता और क्षमताओं का जन्म होगा।

निष्कर्ष (Conclusion)

कृत्रिम बुद्धिमत्ता का इतिहास मानवीय नवाचार और सतत प्रयास का प्रमाण है। सैद्धांतिक अवधारणाओं से लेकर आज के दैनिक जीवन में व्याप्त उपयोग तक, AI ने कई बाधाओं को पार कर अपनी क्षमताएं बढ़ाई हैं।

भविष्य में AI की संभावनाएं असीम हैं — बशर्ते हम इसे नैतिक रूप से, विवेकपूर्ण दिशा में आगे बढ़ाएं। AI का यह भविष्य हमारे जीवन को उस रूप में बदल सकता है, जिसकी हम आज केवल कल्पना ही कर सकते हैं।

1.4 बुद्धिमान एजेंट्स (Intelligent Agents)



चित्र 1.4.1: AI में एजेंट्स (Agents in AI)

चित्र 1.4.1: एआई में एजेंट्स इस दृश्य चित्रण में कृत्रिम बुद्धिमत्ता के तीन प्रमुख प्रकार के एजेंट्स दिखाए गए हैं:

1. **सॉफ़्टवेयर एजेंट** – डिजिटल सिस्टम में कार्यरत जैसे सर्च बॉट्स, रिकमेंडेशन सिस्टम।
2. **बुद्धिमान आभासी एजेंट** – जैसे वॉयस असिस्टेंट्स या वर्चुअल अवतार्स।
3. **रोबोटिक एजेंट** – भौतिक दुनिया में कार्यरत स्वायत्त रोबोट्स।

चित्र में इन एजेंट्स को एक केंद्रीय इकाई के रूप में दिखाया गया है जो **"पर्यावरण"** के साथ संवाद करता है – यानी एजेंट वातावरण को समझता है और उसमें कार्य करता है। यह चित्र एजेंट्स की कार्यप्रणाली और उनकी विविधता को सरलता से समझाता है।

कृत्रिम बुद्धिमत्ता (AI) में **"एजेंट"** उस **स्वायत्त इकाई** को कहा जाता है जो:

- अपने **वातावरण** को देख और समझ सकती है,
- **डाटा की व्याख्या** कर सकती है,
- और निर्धारित **लक्ष्यों** को पूरा करने के लिए उपयुक्त **क्रियाएं** कर सकती है।

एजेंट्स दो रूपों में हो सकते हैं:

- **भौतिक (Physical)** — जैसे रोबोट्स, ड्रोन
- **अमूर्त (Software-based)** — जैसे वर्चुअल असिस्टेंट्स, चैटबॉट्स

AI एजेंट का मूल सिद्धांत यह है कि वह **स्वतंत्र रूप से तर्कपूर्ण निर्णय** लेकर वातावरण में प्रभावी तरीके से कार्य कर सकता है।

1.4.1 AI में एजेंट्स और वातावरण (Agents and Environments in AI)

एजेंट (Agent)

एजेंट वह इकाई है जो:

- अपने वातावरण को **संवेदित** (Sense) करता है,
- प्राप्त डाटा के आधार पर **निर्णय** लेता है,
- और निर्धारित उद्देश्यों की पूर्ति हेतु **क्रियान्वयन** (Act) करता है।

एजेंट्स के प्रकार:

- **सरल प्रतिक्रियाशील एजेंट्स (Simple Reflex Agents)**
- **मॉडल-आधारित एजेंट्स (Model-Based Agents)**
- **लक्ष्य-आधारित एजेंट्स (Goal-Based Agents)**
- **उपयोगिता-आधारित एजेंट्स (Utility-Based Agents)**
- **सीखने वाले एजेंट्स (Learning Agents)**
- **वातावरण (Environment)**

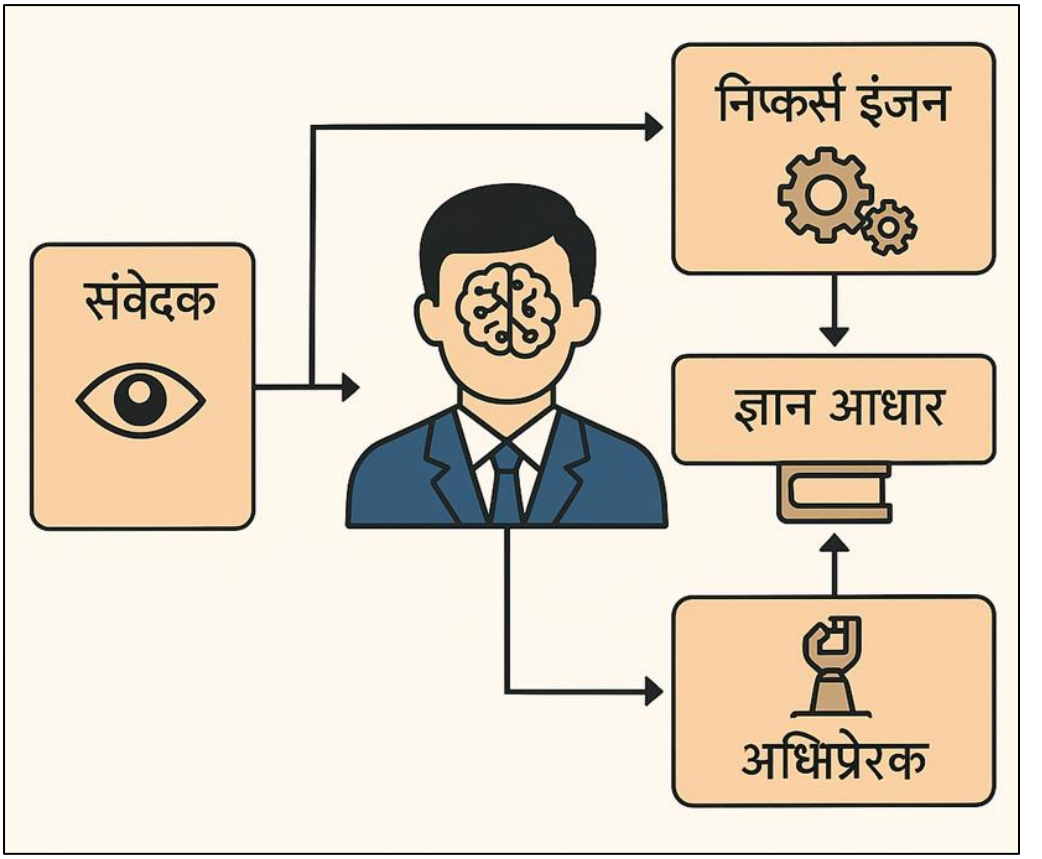
वातावरण वह **बाहरी स्थिति और तत्व** होते हैं जिनसे एजेंट संवाद करता है। यह:

- रोबोट्स के लिए भौतिक संसार हो सकता है

- या सॉफ़्टवेयर एजेंट्स के लिए **वर्चुअल वातावरण**

एजेंट की क्रियाओं या बाहरी प्रभावों से वातावरण बदल सकता है।

1.4.2 एजेंट के प्रमुख घटक (Key Components of an Agent)



चित्र 1.4.2: ज्ञान-आधारित एजेंट की संरचना

1. **सेंसर (Sensors)**:
    - एजेंट का बाहरी दुनिया को देखने-सुनने का साधन।
    - उदाहरण: कैमरा, माइक, या डिजिटल डाटा प्राप्त करने वाले सॉफ्टवेयर।
2. **एक्चुएटर्स (Actuators)**:
    - वातावरण में कार्य करने हेतु एजेंट के क्रिया उपकरण।
    - उदाहरण: रोबोट के मोटर, सॉफ्टवेयर कमांड्स।
3. **आंतरिक निर्णय प्रणाली (Internal Function)**:

- निर्णय लेने वाला इंजन जो डाटा का विश्लेषण करता है और उपयुक्त कार्य तय करता है।
- यहीं पर AI एल्गोरिद्म कार्य करते हैं।

1.4.3 AI एजेंट्स के प्रकार (Types of AI Agents)

चित्र 1.4.3: समस्या-सुलझाने वाले एजेंट का कार्यचक्र

1. सॉफ़्टवेयर एजेंट्स

- **सर्च एजेंट्स** – वेब पेजों को इंडेक्स और रैंक करने वाले बॉट्स (जैसे Googlebot)।

- **सिफारिश प्रणालियाँ (Recommender Systems)** – Netflix, Amazon पर व्यक्तिगत सुझाव।
- **चैटबॉट्स** – ग्राहक सेवा, व्यक्तिगत सहायक जैसे संवादात्मक टूल्स।

2. रोबोटिक एजेंट्स

- **स्वायत्त रोबोट्स** – ड्रोन, सेल्फ-ड्राइविंग कारें जो सेंसर से नेविगेट कर कार्य करती हैं।
- **औद्योगिक रोबोट्स** – फैक्ट्री में असेंबली, वेल्डिंग, पैकेजिंग कार्य।

3. बुद्धिमान वर्चुअल एजेंट्स (Intelligent Virtual Agents)

- **वर्चुअल अवतार्स** – गेमिंग में गतिशील अनुभव देने वाले पात्र।
- **वॉयस असिस्टेंट्स** – Siri, Alexa, Cortana जैसे वॉइस-आधारित सहायक।

एजेंट्स की भूमिका

AI एजेंट्स आज:

- स्मार्टफ़ोन से लेकर स्मार्ट फैक्ट्रियों तक
- हेल्थकेयर, लॉजिस्टिक्स, ग्राहक सेवा, ऑटोमेशन तक **अत्यंत महत्वपूर्ण और विविध भूमिका** निभा रहे हैं।

वे सेंसर, एक्चुएटर्स और इंटेलिजेंट निर्णय क्षमता के मेल से **आधुनिक समाज में AI की शक्ति** को साकार करते हैं।

1.4.4 ज्ञान-आधारित एजेंट्स (Knowledge-Based Agents)

चित्र 1.4.2: ज्ञान-आधारित एजेंट्स

ज्ञान-आधारित एजेंट्स (Knowledge-Based Agents) AI की एक परिष्कृत शाखा हैं, जो विशिष्ट क्षेत्रों में **मानव विशेषज्ञता और निर्णय-प्रक्रिया** की **नकल** करते हैं। ये एजेंट्स एक **संरचित ज्ञान भंडार (Knowledge Base)** के आधार पर कार्य करते हैं और **समस्या-समाधान, निर्णय** और **सलाह** देने में सक्षम होते हैं।

🔍 **मुख्य घटक और विशेषताएँ:**

1. **ज्ञान भंडार (Knowledge Base):**
   एजेंट का मूल घटक, जिसमें तथ्यों, नियमों, और अनुभवजन्य ज्ञान (heuristics) का संग्रह होता है।

2. **तर्क इंजन (Inference Engine):**
   एजेंट का 'मस्तिष्क' जो ज्ञान को प्रोसेस करता है, तर्क करता है, और निर्णय लेता है।
3. **ज्ञान अभ्यावेदन (Knowledge Representation):**
   जानकारी को ऐसा स्वरूप देना जिसे सिस्टम समझे और उपयोग कर सके। तकनीकें: Predicate Logic, Semantic Networks, Rule-Based Systems आदि।
4. **नियम आधारित तर्क (Rule-Based Reasoning):**
   "यदि–तो" जैसे स्पष्ट नियमों पर आधारित निर्णय प्रणाली।
5. **डोमेन विशेषज्ञता (Domain Expertise):**
   मानव विशेषज्ञों द्वारा दी गई विशिष्ट क्षेत्रीय जानकारी को शामिल करना।
6. **समस्या समाधान और निर्णय लेना:**
   एजेंट अपने ज्ञान और तर्कशक्ति से जटिल प्रश्नों को हल करता है और निर्णय देता है।
7. **स्पष्टीकरण और पारदर्शिता:**
   ये एजेंट अपने निर्णयों के पीछे का तर्क स्पष्ट कर सकते हैं – उपयोगकर्ता विश्वास हेतु आवश्यक।
8. **विशेषीकृत क्षेत्र (Specialised Domains):**
   जैसे – स्वास्थ्य (डायग्नोसिस), वित्त (निवेश रणनीतियाँ), तकनीकी सहायता (ट्रबलशूटिंग)।
9. **सीमित अधिगम (Limited Learning):**
   मशीन लर्निंग की तरह आत्म-सीखने में सक्षम नहीं; ज्ञान भंडार को विशेषज्ञों द्वारा अद्यतन किया जाता है।
10. **उपयोगकर्ता संवाद (User Interaction):**
    उपयोगकर्ता से सीधे संवाद कर सकते हैं और उत्तर, सलाह या समाधान दे सकते हैं।

📌 **ज्ञान-आधारित एजेंट्स क्लासिकल AI और मानवीय विशेषज्ञता का संयोजन हैं**, जो पारदर्शिता, विश्वसनीयता और क्षेत्रीय बुद्धिमत्ता प्रदान करते हैं।

1.4.5 समस्या-सुलझाने वाले एजेंट्स (Problem-Solving Agents in AI)

समस्या-सुलझाने वाले एजेंट्स उन AI प्रणालियों को कहते हैं जो **जटिल समस्याओं को हल करने** के लिए रणनीतिक **सोच**, **तर्क**, और **क्रियान्वयन** करते हैं। ये एजेंट्स विभिन्न डोमेनों में स्वतंत्र रूप से कार्य करते हैं।

🔧 **मुख्य लक्षण और क्रियाविधियाँ:**

1. **लक्ष्य अभिमुखता (Goal Orientation):**
   ये एजेंट स्पष्ट लक्ष्यों के लिए डिज़ाइन किए जाते हैं।
2. **वातावरण की धारणा (Environmental Perception):**
   सेंसर या डाटा से स्थिति का आकलन करते हैं।
3. **समस्या और ज्ञान का अभ्यावेदन:**
   तथ्यों, नियमों, लक्ष्यों, और सीमाओं को संरचित रूप में प्रस्तुत करते हैं।
4. **खोज और योजना (Search and Planning):**
   लक्ष्य प्राप्ति हेतु विभिन्न स्थितियों और कार्यों के संभावित पथों की खोज करते हैं।
5. **तर्क और निर्णय:**
   तर्क आधारित निर्णय लेते हैं जो वातावरण और लक्ष्य के अनुसार उपयुक्त हो।
6. **क्रिया निष्पादन (Action Execution):**
   योजना के अनुसार वातावरण में बदलाव लाते हैं।
7. **अनुकूली अधिगम (Adaptive Learning):**
   अनुभव से सीखते हैं और भविष्य के कार्यों में सुधार करते हैं।
8. **प्रदर्शन आकलन और प्रतिक्रिया उपयोग:**
   फीडबैक के आधार पर अपनी रणनीति में सुधार करते हैं।
9. **सर्वोत्तम समाधान की खोज (Optimisation Focus):**
   सबसे अच्छा, कुशल और प्रभावी समाधान प्राप्त करने के लिए अनुकूलन विधियाँ अपनाते हैं।
10. **डोमेन बहु-उपयोगिता (Domain Versatility):**
    विभिन्न क्षेत्रों जैसे – नेविगेशन, भाषा प्रसंस्करण, सिफारिश प्रणाली, गेमिंग आदि में अनुकूल।

उदाहरण (Examples):

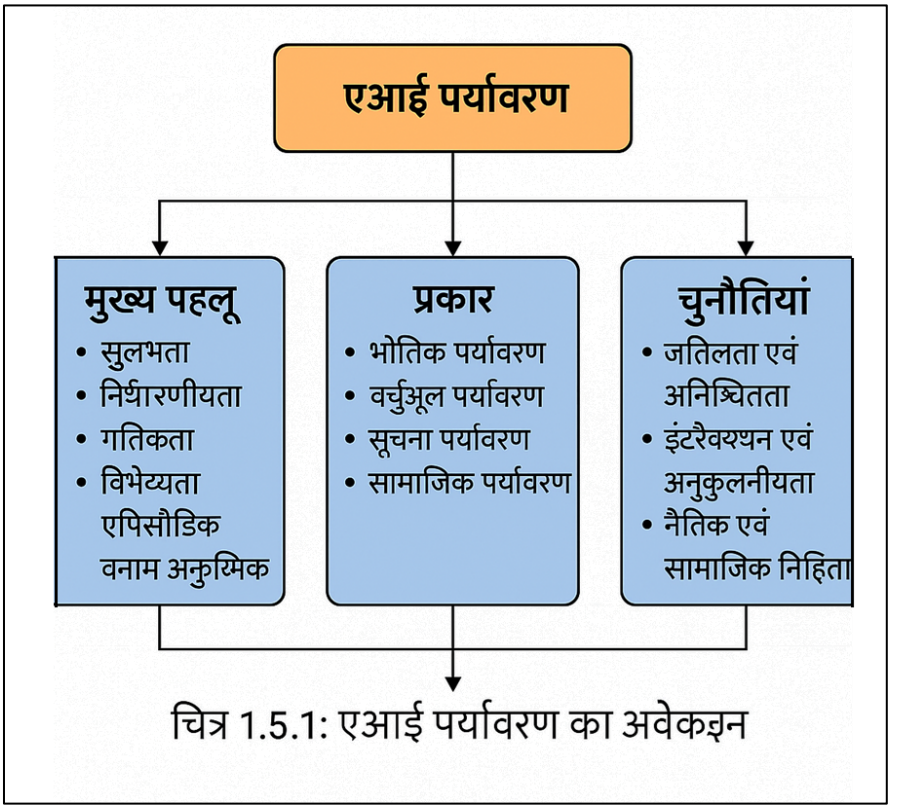
- **शतरंज खेलने वाले एजेंट्स** – चालों का मूल्यांकन कर सर्वोत्तम चाल चुनते हैं।
- **पथ खोजने वाले एजेंट्स (Pathfinding)** – सबसे कुशल मार्ग का निर्धारण करते हैं।
- **योजना निर्माण एजेंट्स (Planning Agents)** – फैक्ट्री या प्रोजेक्ट कार्यों की योजना बनाते हैं।

- **विशेषज्ञ प्रणाली (Expert Systems)** – चिकित्सा, तकनीकी सहायता आदि में निर्णय समर्थन देते हैं।
- **रीइन्फोर्समेंट लर्निंग एजेंट्स** – पुरस्कार अधिकतम करने हेतु लगातार सीखते हैं।

निष्कर्ष:

Knowledge-Based Agents विशिष्ट ज्ञान और नियमों से विशेषज्ञता प्रदान करते हैं। **Problem-Solving Agents** अनिश्चित और गतिशील समस्याओं को हल करने की क्षमता रखते हैं। दोनों ही प्रकार AI की बुद्धिमत्ता और आत्मनिर्णय को अलग-अलग रूपों में दर्शाते हैं।

1.5 एआई में पर्यावरण (Environments in AI)



चित्र 1.5.1: एआई पर्यावरण का अवलोकन

(ब्लॉक आरेख — परिवर्तित संरचना में, जो मुख्य पहलुओं, प्रकारों और चुनौतियों को दर्शाता है)

पर्यावरण (Environment) वह संदर्भ या स्थान होता है जिसमें कोई **AI एजेंट** कार्य करता है, निर्णय लेता है और प्रतिक्रिया करता है। ये वातावरण वास्तविक (भौतिक) से लेकर पूर्णतः वर्चुअल या सिमुलेटेड हो सकते हैं। वातावरण की प्रकृति यह निर्धारित करती है कि एजेंट को कितनी जटिलता और चुनौती का सामना करना पड़ेगा।

1.5.1 पर्यावरण के प्रमुख पहलू (Key Aspects of Environments)

1. **सुलभता (Accessibility):**
    - सुलभ वातावरण में एजेंट को सभी जानकारी उपलब्ध होती है।

- असुलभ वातावरण में जानकारी आंशिक होती है, और एजेंट को अनुमान आधारित निर्णय लेने होते हैं।

2. **निर्धारितता (Determinism):**
   - निर्धारित वातावरण में हर क्रिया का पूर्वानुमेय परिणाम होता है।
   - अनिर्धारित वातावरण में परिणाम अनिश्चित हो सकते हैं।
3. **गतिकता (Dynamism):**
   - गतिशील वातावरण समय के साथ बदलता रहता है।
   - स्थैतिक वातावरण एजेंट की क्रिया के बिना नहीं बदलता।
4. **विभाज्यता (Discreteness):**
   - विभाजित वातावरण में सीमित अवस्थाएँ और क्रियाएँ होती हैं।
   - सतत वातावरण में अवस्थाएँ और क्रियाएँ लगातार परिवर्तनशील होती हैं।
5. **एपिसोडिक बनाम अनुक्रमिक (Episodic vs Sequential):**
   - एपिसोडिक कार्य स्वतंत्र होते हैं;
   - अनुक्रमिक वातावरण में पिछली क्रियाओं का प्रभाव भविष्य पर पड़ता है।
6. **एकल एजेंट बनाम बहु-एजेंट (Single-Agent vs Multi-Agent):**
   - कुछ वातावरणों में एक एजेंट होता है,
   - जबकि अन्य में कई एजेंट होते हैं जो सहयोग या प्रतिस्पर्धा कर सकते हैं।

1.5.2 एआई में पर्यावरण के प्रकार (Types of Environments)

1. **भौतिक वातावरण (Physical Environments):**
   - वास्तविक दुनिया में एजेंट्स जैसे ड्रोन, रोबोट्स, या सेल्फ-ड्राइविंग कारों के लिए।
   - उदाहरण: सड़क पर चलती स्वायत्त कारें, निगरानी ड्रोन।
2. **वर्चुअल वातावरण (Virtual Environments):**
   - पूर्णतः सॉफ्टवेयर-आधारित डिजिटल जगत।
   - जैसे गेमिंग वर्ल्ड, मशीन लर्निंग प्रशिक्षण सिमुलेशन।
3. **सूचनात्मक वातावरण (Information Environments):**

- इंटरनेट, डाटाबेस जैसे डाटा-केंद्रित स्थान।
- कार्य: सूचना खोज, डाटा विश्लेषण, साइबर सुरक्षा।

4. **सामाजिक वातावरण (Social Environments):**
    - मानव या अन्य एजेंट्स के साथ संवाद।
    - उदाहरण: चैटबॉट्स, वॉयस असिस्टेंट्स, सोशल मीडिया विश्लेषण।

1.5.3 चुनौतियाँ और विचार (Challenges and Considerations)

- **जटिलता और अनिश्चितता:**
    - एजेंट को विभिन्न और अप्रत्याशित परिस्थितियों में काम करने के लिए अनुकूल और मजबूत एल्गोरिद्म चाहिए।
- **संवाद और अनुकूलनशीलता (Interactivity & Adaptability):**
    - एजेंट को वातावरण के साथ प्रभावी संवाद करना और बदलती स्थितियों के अनुसार अनुकूलन करना आना चाहिए।
- **नैतिक और सामाजिक प्रभाव:**
    - गोपनीयता, निष्पक्षता और मानव मूल्यों के प्रति सम्मान AI एजेंट्स के लिए अनिवार्य है।

1.5.4 निष्कर्ष (Conclusion)

कृत्रिम बुद्धिमत्ता (AI) में पर्यावरण की अवधारणा AI एजेंट्स के विकास, कार्यप्रणाली और विभिन्न क्षेत्रों में उनके अनुप्रयोग को आकार देने में एक मूलभूत भूमिका निभाती है।

ये वातावरण — भौतिक जगत की वास्तविकताओं से लेकर वर्चुअल और सूचना आधारित जटिल संरचनाओं तक — AI एजेंट्स को वह संदर्भ प्रदान करते हैं जिसमें वे जानकारी ग्रहण करते हैं, संवाद करते हैं और निर्णय लेते हैं।

इन पर्यावरणों की विविधता — जैसे सुलभता, निश्चितता, गतिकता, सततता या विभाज्यता, तथा एकल या बहु-एजेंट की उपस्थिति — AI विकास के लिए कई चुनौतियाँ और संभावनाएँ उत्पन्न करती है।

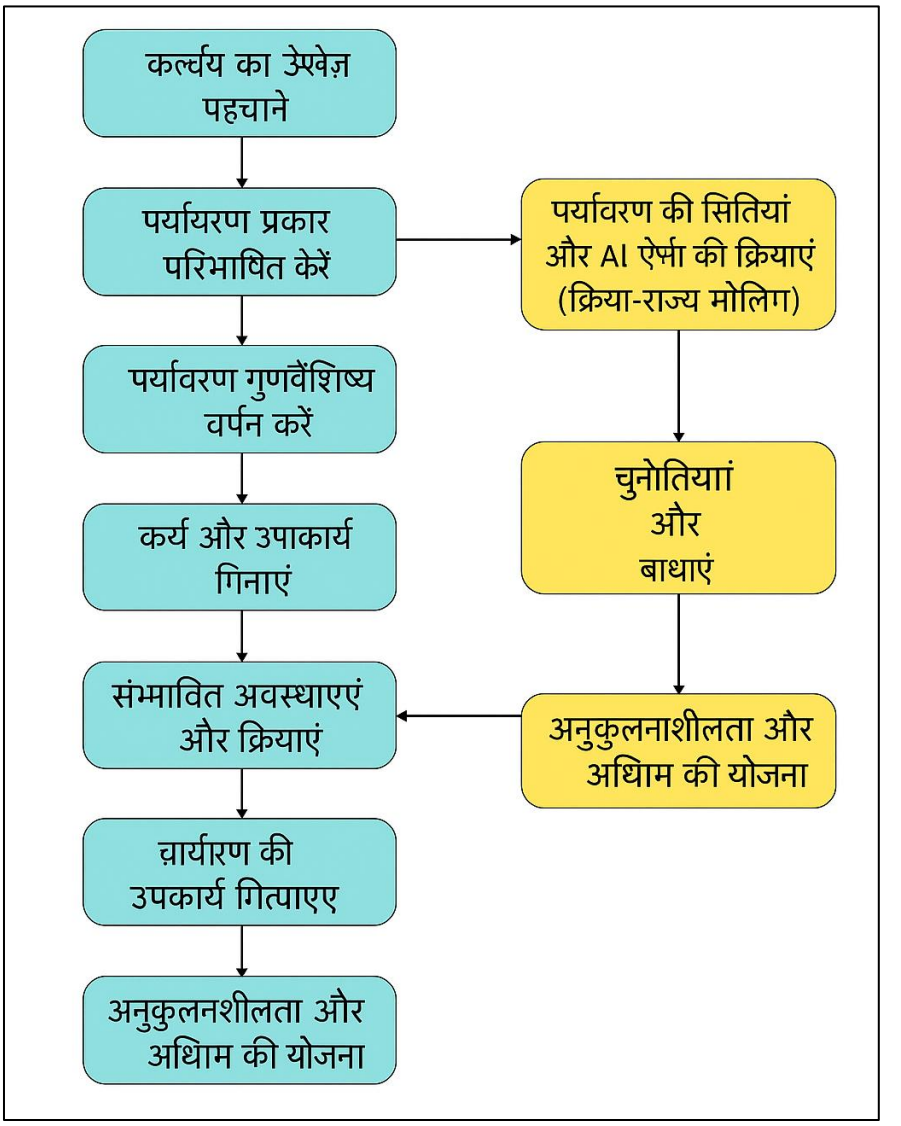
इन जटिलताओं को सफलतापूर्वक समझने और संभालने के लिए:

- उन्नत एल्गोरिद्म और मॉडल,
- तथा नैतिक, सामाजिक और व्यावहारिक प्रभावों की गहन समझ आवश्यक है।

जैसे-जैसे AI हमारे जीवन के हर पहलू में समाहित होता जा रहा है, एजेंट्स की सीखने, अनुकूलन करने और नैतिक रूप से बातचीत करने की क्षमता अत्यधिक महत्वपूर्ण बनती जा रही है।

AI वातावरण में अन्वेषण और नवाचार इस क्षेत्र की गतिशील प्रकृति और प्रभावशाली संभावनाओं को दर्शाते हैं — जो यह दर्शाते हैं कि यह तकनीक, समाज और बुद्धिमत्ता की हमारी समझ को पुनः परिभाषित कर सकती है।

1.6 कार्य पर्यावरण का निर्धारण



चित्र 1.6.1: कार्य पर्यावरण निर्दिष्ट करने की संरचित प्रक्रिया का आरेख

कृत्रिम बुद्धिमत्ता (AI) सिस्टम को डिज़ाइन करते समय **कार्य पर्यावरण का स्पष्ट निर्धारण** अत्यंत आवश्यक होता है। यह वह संदर्भ है जिसमें एजेंट कार्य करता है — जिसमें वस्तुएं, अन्य एजेंट्स, चुनौतियाँ,

और नियम शामिल होते हैं। एक स्पष्ट परिभाषित कार्य पर्यावरण से ऐसा एजेंट विकसित करना संभव होता है जो **प्रभावी और कुशलतापूर्वक अपने उद्देश्य पूरे** कर सके।

1.6.1 AI एजेंट के उद्देश्य की पहचान (Identify the Purpose of the AI Agent)

AI एजेंट का उद्देश्य स्पष्ट रूप से परिभाषित करना सबसे पहला और बुनियादी कदम होता है। यह तय करता है कि एजेंट को **क्या हल करना है** या **कौन-सा कार्य करना है**। उद्देश्य के आधार पर एजेंट की संरचना, एल्गोरिद्म और व्यवहार तय होता है।

उदाहरणों के माध्यम से उद्देश्य:

1. **भौतिक क्षेत्र में संचालन (Piloting Physical Spaces)**
    - जैसे: रोबोट वैक्यूम या डिलीवरी ड्रोन
    - कार्य: स्थानिक जानकारी का विश्लेषण करना, रास्ता तय करना, बाधाओं से बचना, और वस्तुओं से संवाद करना।
2. **शेयर ट्रेडिंग (Trading Stocks)**
    - कार्य: वित्तीय बाज़ार का विश्लेषण, स्टॉक की भविष्यवाणी, और लाभ अधिकतम करना।
    - विशेषताएँ: रीयल-टाइम डाटा प्रोसेसिंग, त्वरित निर्णय क्षमता।
3. **ग्राहक सेवा प्रदान करना (Providing Customer Service)**
    - जैसे: चैटबॉट्स, वर्चुअल असिस्टेंट्स
    - कार्य: ग्राहकों की भाषा को समझना, उपयुक्त उत्तर देना, और अनुभव को बेहतर बनाना।

1.6.2 पर्यावरण का प्रकार निर्धारित करना (Define the Environment Type)

AI एजेंट जिस वातावरण में कार्य करता है, उसकी प्रकृति को समझना बेहद ज़रूरी है। इससे यह तय होता है कि उसे **किस प्रकार के सेंसर, एक्चुएटर्स, और निर्णय लेने की प्रणाली** चाहिए।

मुख्य प्रकार के पर्यावरण:

1. **भौतिक वातावरण (Physical Environment)**
    - वास्तविक संसार — जैसे सेल्फ-ड्राइविंग कार, मैन्युफैक्चरिंग रोबोट

- आवश्यकताएँ: कैमरा, LIDAR, स्पर्श सेंसर जैसे फिजिकल इनपुट्स और मोटर्स, आर्म्स जैसे आउटपुट्स।
- चुनौतियाँ: रीयल-टाइम प्रतिक्रिया, सुरक्षा, सेंसर नॉइज़।

2. **वर्चुअल वातावरण (Virtual Environment)**
   - पूरी तरह कंप्यूटर सिमुलेशन पर आधारित — जैसे वीडियो गेम्स, सॉफ़्टवेयर परीक्षण
   - इनपुट: गेम की स्थिति, छवियाँ आदि
   - आउटपुट: सिमुलेशन में की गई डिजिटल क्रियाएं
   - लाभ: नियंत्रित परीक्षण और जटिल इंटरैक्शन की सुविधा।
3. **सूचनात्मक वातावरण (Informational Environment)**
   - मुख्य रूप से डाटा और सूचना पर केंद्रित — जैसे सिफारिश प्रणाली, डाटा विश्लेषण टूल्स
   - इनपुट: उपयोगकर्ता व्यवहार, दस्तावेज़
   - आउटपुट: सिफारिशें, रिपोर्ट, अंतर्दृष्टियाँ
   - तकनीकें: मशीन लर्निंग, पैटर्न पहचान, भाषा संसाधन।

निष्कर्ष:

प्रत्येक प्रकार का कार्य पर्यावरण AI एजेंट के लिए **अलग-अलग चुनौतियाँ और आवश्यकताएँ** प्रस्तुत करता है।

- किस प्रकार के सेंसर-एक्चुएटर की ज़रूरत है,
- किस प्रकार का डाटा प्रोसेसिंग और निर्णय प्रणाली चाहिए — यह सब **पर्यावरण की प्रकृति** पर निर्भर करता है।

इसलिए, कार्य पर्यावरण को **संगठित, स्पष्ट और संदर्भ-उन्मुख रूप से परिभाषित करना** AI डिज़ाइन का एक अनिवार्य हिस्सा है।

1.6.3 पर्यावरण गुणों का विश्लेषण (Characterise the Environment Properties)

AI एजेंट के सुचारु संचालन के लिए उसके वातावरण के महत्वपूर्ण गुण स्पष्ट करना अत्यावश्यक है:

पर्यावरण गुण	पूछे जाने वाले मुख्य प्रश्न	निहित अर्थ
सुलभता (Accessibility)	क्या एजेंट वातावरण की पूर्ण जानकारी देख सकता है, या केवल आंशिक?	पूर्णतः सुलभ = पूरे राज्य का ज्ञान; आंशिक = अनिश्चितता व अनुमानित निर्णय
निर्धारितता (Determinism)	प्रत्येक क्रिया का परिणाम निश्चित है या यादृच्छिक (Stochastic)?	निर्धारित वातावरण = पूर्वानुमेय; अनिर्धारित = सम्भावनात्मक परिणाम
एपिसोडिक बनाम अनुक्रमिक	क्या कार्य अलग-अलग एपिसोड में हैं या पूर्व क्रियाएँ भविष्य को प्रभावित करती हैं?	एपिसोडिक = स्वतंत्र कृत्य; अनुक्रमिक = दीर्घकालिक रणनीति आवश्यक
स्थिर बनाम गतिशील (Static vs Dynamic)	क्या एजेंट के विचार करने के दौरान वातावरण बदलता है?	गतिशील = रीयल-टाइम अनुकूलन जरूरी
विभाजित बनाम सतत (Discrete vs Continuous)	राज्य/क्रिया स्थान गिनती-योग्य है या निरंतर?	सतत = गणितीय मॉडलों व अप्रोक्सिमेशन की जरूरत
एकल-एजेंट बनाम बहु-एजेंट	क्या एजेंट अकेला कार्य कर रहा है या अन्य एजेंटों के साथ सहयोग/प्रतिस्पर्धा में?	बहु-एजेंट = गेम-थ्योरी, समन्वय या विरोधी रणनीति

1.6.4 प्रदर्शन मापदण्ड निर्दिष्ट करना (Specify Performance Measures)

AI एजेंट की सफलता आँकने हेतु स्पष्ट, मात्रात्मक मापदण्ड निर्धारित करें:

मापदण्ड	वर्णन व उपयोग-क्षेत्र
शुद्धता (Accuracy)	सत्य/गलत निर्णयों का अनुपात – चिकित्सा निदान, धोखाधड़ी पता लगाना
दक्षता (Efficiency)	समय, ऊर्जा या कम्प्यूटेशनल संसाधनों का किफ़ायती उपयोग – रूट अनुकूलन, ऊर्जा-कुशल रोबोटिक्स

गति (Speed)	कार्य पूर्ण करने की तीव्रता – हाई-फ़्रीक्वेंसी ट्रेडिंग, आपात-प्रतिक्रिया
अनुकूलनशीलता (Adaptability)	नए अथवा बदले परिवेश में सीखने-समायोजित करने की क्षमता
रोबस्टनेस/विश्वसनीयता	विविध या अप्रत्याशित स्थितियों में त्रुटिरहित कार्य करने की स्थिरता – स्वायत्त वाहन, उद्योग-स्वचालन

इन मापदण्डों का प्रारम्भ से निर्धारण एजेण्ट डिज़ाइन को वांछित लक्ष्य के अनुरूप रखता है।

1.6.5 कार्यों एवं उप-कार्यों का सूचीकरण (Enumerate Tasks and Subtasks)

क्रमबद्ध (Hierarchical) विभाजन विकास-प्रक्रिया को व्यवस्थित व प्रबंधनीय बनाता है:

1. **मुख्य कार्य पहचानें (Task Identification)**
    - उदाहरण: घरेलू रोबोट के लिए साफ़-सफ़ाई, नेविगेशन, बैटरी चार्ज इत्यादि।
2. **उप-कार्य विभाजन (Subtask Breakdown)**
    - साफ़-सफ़ाई हेतु: गंदगी पहचानना → गंदगी क्षेत्र तक पहुँचना → सफ़ाई-मोड सक्रिय करना।
3. **प्राथमिकता व क्रम निर्धारण (Prioritisation & Sequencing)**
    - कौन-सा उप-कार्य पहले? किन कार्यों पर अन्य कार्य निर्भर?
4. **निर्णय-प्रक्रिया रूपरेखा (Decision-Making Processes)**
    - उदाहरण: बैटरी स्तर कम हो तो पहले चार्जिंग डॉक पर लौटना।
5. **क्रिया-चयन तंत्र (Action-Selection Mechanisms)**
    - नियम-आधारित लॉजिक, निर्णय वृक्ष, या पिछले अनुभव से सीखते हुए अनुकूलित एल्गोरिद्म।

इस प्रकार की संरचित विश्लेषण पद्धति यह सुनिश्चित करती है कि AI एजेंट का व्यवहार सुव्यवस्थित, कुशल और उसके समग्र उद्देश्य के अनुरूप रहे।

1.6.6 संभावित अवस्थाओं और क्रियाओं का वर्णन (Describe Possible States and Actions)

AI एजेंट के व्यवहार और निर्णय प्रणाली को डिज़ाइन करने के लिए यह जानना आवश्यक है कि:

1. स्थिति स्थान (State Space Definition):

- पर्यावरण की वे सभी संभावित स्थितियाँ जिनमें एजेंट कार्य कर सकता है।
- उदाहरण: रोबोट वैक्यूम के लिए यह हो सकता है — फर्श पर गंदगी की स्थिति, बाधाएं, और बैटरी का स्तर।

2. क्रिया स्थान (Action Space Definition):

- एजेंट द्वारा की जा सकने वाली सभी क्रियाएं।
- उदाहरण: उत्तर की ओर बढ़ना, सफ़ाई शुरू करना, चार्जिंग डॉक पर लौटना।

3. स्थिति-क्रिया मैपिंग (State-Action Mapping):

- यह दर्शाता है कि कोई विशेष क्रिया कैसे स्थिति को बदलती है।
- उदाहरण: उत्तर दिशा में एक कदम चलना = ग्रिड की एक इकाई ऊपर स्थानांतरण।

4. बाधाएं और नियम (Constraints & Rules):

- कुछ क्रियाएं केवल कुछ स्थितियों में ही संभव होती हैं।
- उदाहरण: दीवार के कारण आगे बढ़ना असंभव; या बैटरी कम होने पर सफ़ाई नहीं कर पाना।

इन सभी पहलुओं को स्पष्ट करके, एजेंट को बेहतर ढंग से पर्यावरण में नेविगेट करने और लक्ष्य प्राप्त करने में मदद मिलती है।

1.6.7 चुनौतियाँ और सीमाएँ पहचानें (Identify Challenges and Constraints)

AI एजेंट को विश्वसनीय और नैतिक रूप से सक्षम बनाने हेतु संभावित बाधाओं का पूर्वानुमान आवश्यक है:

1. बोध संबंधी सीमाएँ (Perceptual Limitations):

- सेंसर की सीमा, सटीकता और विश्वसनीयता प्रभावित करती है कि एजेंट दुनिया को कितना सही समझता है।

2. क्रिया प्रतिबंध (Action Constraints):

- भौतिक सीमाएँ, संसाधन सीमाएँ या पर्यावरणीय बाधाएं कुछ क्रियाओं को असंभव बना सकती हैं।

3. सुरक्षा विचार (Safety Considerations):

- विशेषकर मानव संपर्क वाले एजेंट्स के लिए — आपातकालीन रुकावट, फेल-सेफ तंत्र आवश्यक।

4. नैतिक और सामाजिक प्रभाव (Ethical and Social Implications):

- गोपनीयता, पूर्वग्रह, स्वायत्तता, रोजगार पर प्रभाव — इन सब पर एजेंट को डिज़ाइन करते समय विचार करना चाहिए।

5. पर्यावरणीय गतिकता (Environmental Dynamics):

- परिवर्तनीय वातावरण में कार्य करते समय एजेंट को लगातार अनुकूलन करना आना चाहिए।

1.6.8 अनुकूलनशीलता और अधिगम की योजना (Plan for Adaptability and Learning)

एक उन्नत AI एजेंट की विशेषता है उसकी **सीखने और स्वयं को सुधारने की क्षमता**।

1. अधिगम तंत्र (Learning Mechanisms):

- रिइन्फोर्समेंट लर्निंग, सुपरवाइज़्ड या अनसुपरवाइज़्ड लर्निंग जैसी तकनीकों का उपयोग।

2. फीडबैक तंत्र (Feedback Loops):

- एजेंट प्रदर्शन मापदंडों, पर्यावरणीय संकेतों, या मानव इनपुट के आधार पर रणनीति सुधारता है।

3. पर्यावरण से संवाद और प्रयोग (Interaction for Learning):

- वातावरण में इंटरैक्शन के ज़रिए प्रयोगात्मक अधिगम संभव होता है।
- उदाहरण: वर्चुअल सिमुलेशन में रणनीति आज़माना।

🔧 **4. निरंतर अद्यतन और सुधार चक्र (Continuous Update & Improvement):**

- मॉडल्स का दोबारा प्रशिक्षण, ज्ञान का अद्यतन और नए डेटा से एल्गोरिद्म का परिष्करण एजेंट को अद्यतन बनाए रखता है।

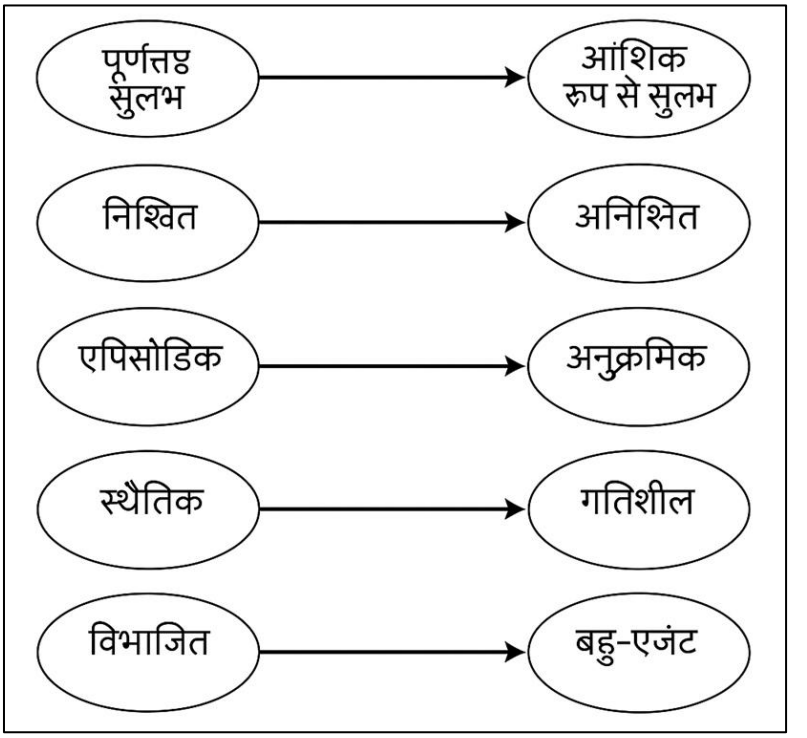
समग्र निष्कर्ष:

इन अंतिम चरणों को लागू करके हम ऐसे AI एजेंट बना सकते हैं जो:

- **जटिल वातावरण में आत्मनिर्भर रूप से काम कर सकें**,
- **बाधाओं का समाधान कर सकें**,
- और **समय के साथ सीखकर लगातार बेहतर होते जाएं**।

यही प्रक्रिया AI सिस्टम्स को **व्यावहारिक, विश्वसनीय और भविष्य-प्रूफ** बनाती है।

1.7 कार्य पर्यावरण के गुण



चित्र 1.7.1: एआई में कार्य पर्यावरण के गुणों का अवस्थांतर आरेख

(यह चित्र पर्यावरण गुणों के दो चरम सिरों के बीच अवस्थांतर को दर्शाता है — जैसे पूर्णतः सुलभ से आंशिक सुलभ, निश्चित से अनिश्चित, आदि।)

AI एजेंट की डिज़ाइन और कार्य-क्षमता को कार्य पर्यावरण की विशेषताएँ गहराई से प्रभावित करती हैं। ये गुण यह तय करते हैं कि एजेंट **कैसे निर्णय ले**, **कैसे कार्य करे**, और **कैसे अनिश्चितताओं का सामना करे**।

1.7.1 पूर्णतः सुलभ बनाम आंशिक रूप से सुलभ (Fully Observable vs. Partially Observable)

- 🔹 **पूर्णतः सुलभ:**

  एजेंट को पर्यावरण की संपूर्ण जानकारी हर समय उपलब्ध होती है।

  निर्णय पूरी जानकारी के आधार पर होते हैं।

- 🔹 **आंशिक रूप से सुलभ:**

  एजेंट को सभी स्थितियाँ नहीं दिखतीं — या तो सेंसर की सीमाएँ होती हैं या जानकारी छिपी होती है।

  निर्णय **अनिश्चितता** के साथ लिए जाते हैं।

1.7.2 निश्चित बनाम अनिश्चित (Deterministic vs. Stochastic)

- 🔹 **निश्चित (Deterministic):**

  वर्तमान स्थिति + क्रिया = अगली स्थिति स्पष्ट रूप से ज्ञात होती है।

  पूर्वानुमेय वातावरण।

- 🔹 **अनिश्चित (Stochastic):**

  परिणाम में **रैन्डमनेस** होती है।

  वही क्रिया अलग-अलग परिणाम दे सकती है।

1.7.3 एपिसोडिक बनाम अनुक्रमिक (Episodic vs. Sequential)

- 🔹 **एपिसोडिक:**

  हर निर्णय स्वतंत्र होता है।

  जैसे — छवियों की पहचान में हर छवि अलग एपिसोड।

- 🔹 **अनुक्रमिक:**

  प्रत्येक निर्णय भविष्य को प्रभावित करता है।

  जैसे — नेविगेशन, गेम, रणनीति निर्धारण।

1.7.4 स्थैतिक बनाम गतिशील (Static vs. Dynamic)

- 🔹 **स्थैतिक:**

  निर्णय लेते समय वातावरण स्थिर रहता है।

  समय नहीं बदलता।

- 🔹 **गतिशील:**

  वातावरण समय के साथ या अन्य एजेंट्स की क्रियाओं से बदल सकता है।

  एजेंट को समय-आधारित निर्णय लेने होते हैं।

1.7.5 विभाजित बनाम सतत (Discrete vs. Continuous)

- 🔹 **विभाजित (Discrete):**

  सीमित और गिनती योग्य अवस्थाएँ/क्रियाएँ।

  जैसे – चेस बोर्ड, ग्रिड आधारित नक्शा।

- 🔹 **सतत (Continuous):**

  स्थिति/क्रिया निरंतर बदल सकती है।

  जैसे – रोबोट की गति, ड्रोन की उड़ान।

1.7.6 एकल एजेंट बनाम बहु-एजेंट (Single Agent vs. Multi-Agent)

- 🔹 **एकल एजेंट:**

  एजेंट अकेला निर्णय लेता है, बाहरी हस्तक्षेप नहीं होता।

- 🔹 **बहु-एजेंट:**

  अन्य एजेंट्स भी मौजूद होते हैं — सहयोगी, प्रतिस्पर्धी, या दोनों।

  गेम-थ्योरी, समन्वय और प्रतिक्रिया अनिवार्य हो जाते हैं।

निष्कर्ष (Conclusion):

इन गुणों को समझकर AI एजेंट को उसके पर्यावरण के अनुसार डिज़ाइन किया जा सकता है:

- क्या वह संपूर्ण जानकारी के आधार पर निर्णय ले सकता है या अनुमान करना होगा?
- क्या निर्णयों के दीर्घकालिक परिणाम होंगे?
- क्या पर्यावरण पूर्वानुमेय है या बदलता रहता है?

📌 **AI सिस्टम को प्रभावी, अनुकूलनशील और मजबूत बनाने के लिए यह विश्लेषण आवश्यक है।**

1.8 एजेंट-आधारित प्रोग्राम्स (Agent-Based Programs)

एजेंट-आधारित प्रोग्राम्स ऐसे सॉफ़्टवेयर तंत्र होते हैं जो किसी वातावरण में **स्वायत्त रूप से कार्य करते हैं** ताकि निर्दिष्ट लक्ष्यों की पूर्ति की जा सके। ये प्रोग्राम व्यक्तिगत या समूह एजेंट्स की क्रियाओं और परस्पर संवाद का अनुकरण (simulation) करते हैं ताकि उनके प्रणाली पर प्रभावों का मूल्यांकन किया जा सके।

इनकी **लचीलापन और अनुकूलन क्षमता** उन्हें अनेक क्षेत्रों में उपयोगी बनाती है, जैसे:

- अर्थशास्त्र,
- जीवविज्ञान,
- सामाजिक विज्ञान,
- कंप्यूटर विज्ञान।

1.8.1 एजेंट-आधारित प्रोग्राम की संरचना (Structure of Agent-Based Programs)

एक एजेंट की मूल संरचना में मुख्यतः निम्नलिखित घटक होते हैं:

- **सेंसर (Sensors):** वातावरण को देखने और समझने के लिए।
- **एक्चुएटर्स (Actuators):** वातावरण में कार्य करने के लिए।
- **आंतरिक मॉडल (Internal Model):** ज्ञान भंडार, निर्णय लेने की क्षमता, नियम और तर्क प्रणाली शामिल होती है।

इस संरचना को समझना प्रभावी, अनुकूल और बुद्धिमान एजेंट्स बनाने के लिए आवश्यक है।

1.8.2 एजेंट्स के प्रकार (Types of Agents)

1. **साधारण प्रतिक्रिया एजेंट (Simple Reflex Agents):**
    - "स्थिति → क्रिया" नियम पर कार्य करते हैं।
2. **मॉडल-आधारित प्रतिक्रिया एजेंट (Model-Based Reflex Agents):**
    - दुनिया की आंतरिक स्थिति बनाए रखते हैं ताकि बेहतर निर्णय लिया जा सके।
3. **लक्ष्य-आधारित एजेंट (Goal-Based Agents):**
    - अपने लक्ष्यों की प्राप्ति हेतु कार्य करते हैं। निर्णय लेते समय भविष्य का प्रभाव विचार में रखते हैं।

4. **उपयोगिता-आधारित एजेंट (Utility-Based Agents):**
    - ऐसे विकल्प चुनते हैं जो उन्हें अधिकतम संतुष्टि (utility) प्रदान करें।
5. **सीखने वाले एजेंट (Learning Agents):**
    - समय के साथ अपने अनुभवों से सीखते हैं और प्रदर्शन में सुधार करते हैं।

1.8.3 एजेंट-आधारित सिस्टम का डिज़ाइन (Designing Agent-Based Systems)

- उपयुक्त एजेंट प्रकार का चयन
- परस्पर संवाद और सहयोग के लिए प्रोटोकॉल बनाना
- पर्यावरण में स्वतंत्र संचालन की क्षमता
- मापनीयता (Scalability), मजबूती (Robustness), और अनुकूलनशीलता की योजना

1.8.4 एजेंट-आधारित प्रोग्राम के अनुप्रयोग (Applications of Agent-Based Programs)

क्षेत्र	उपयोग
सामाजिक विज्ञान	मानव व्यवहार, सामाजिक नेटवर्क और समाज में परिवर्तन का मॉडलिंग
अर्थशास्त्र	बाज़ार मॉडलिंग, उपभोक्ता व्यवहार विश्लेषण
पारिस्थितिकी और जीवविज्ञान	पारिस्थितिकी तंत्र, जनसंख्या गतिशीलता
यातायात और परिवहन	ट्रैफ़िक प्रवाह सिमुलेशन, कुशल रूट डिज़ाइन
वितरित नियंत्रण प्रणाली	लॉजिस्टिक्स, आपूर्ति श्रृंखला प्रबंधन

1.8.5 एजेंट-आधारित प्रोग्रामिंग में चुनौतियाँ (Challenges in Agent-Based Programming)

- **1. गणनात्मक जटिलता (Computational Complexity):**
    - अनेक एजेंट्स व उनकी बातचीत = अधिक संसाधन उपयोग।
    - **रणनीति:** एल्गोरिद्म को अनुकूलित करना, समानांतर गणना, एजेंट निर्णय सरल बनाना।
- **2. यथार्थवादी व्यवहार सुनिश्चित करना (Ensuring Realistic Agent Behavior):**
    - मानव या जीवों जैसे व्यवहार का अनुकरण कठिन।
    - **रणनीति:** मशीन लर्निंग, विविध डाटा सेट, व्यवहारिक मॉडलिंग।
- **3. बहु-एजेंट संवाद प्रबंधन (Managing Interactions in Multi-Agent Systems):**

- सहयोग, प्रतिस्पर्धा, संवाद में सामंजस्य लाना चुनौतीपूर्ण।
- **रणनीति:** संचार प्रोटोकॉल, संघर्ष समाधान फ्रेमवर्क, साझा लक्ष्य आधारित डिज़ाइन।

निष्कर्ष:

एजेंट-आधारित प्रोग्राम्स जटिल प्रणालियों के अनुकरण और समाधान में अत्यंत उपयोगी हैं।

यदि उपयुक्त डिज़ाइन, यथार्थ व्यवहार और अनुकूल रणनीतियाँ अपनाई जाएं, तो ये कार्यक्रम **AI के विकास में क्रांतिकारी भूमिका** निभा सकते हैं।

एजेंट-आधारित प्रोग्राम की मूल संरचना (Basic Structure of an Agent-Based Program)

एजेंट-आधारित प्रोग्रामिंग में, एक **एजेंट क्लास** परिभाषित की जाती है, जिसमें निम्नलिखित मुख्य कार्य होते हैं:

- **मुख्य कार्यात्मकता:**
  - **पर्यावरण को देखना (perceive)**
  - **स्थिति के अनुसार निर्णय लेना (decide)**
  - **कोई क्रिया करना (act)**

बेस क्लास (Python कोड):

```
class Agent:
    def __init__(self, environment):
        self.environment = environment
        self.internal_state = {}
    def perceive(self):
        raise NotImplementedError  # वातावरण से स्थिति प्राप्त करें
    def decide(self):
        raise NotImplementedError  # निर्णय लें
    def act(self, action):
        raise NotImplementedError  # क्रिया निष्पादित करें
```

विभिन्न प्रकार के एजेंट्स का कार्यान्वयन (Implementing Different Types of Agents)

1. Simple Reflex Agent (सरल प्रतिक्रिया एजेंट)

```
class SimpleReflexAgent(Agent):
    def decide(self):
        current_state = self.perceive()
        if 'condition' in current_state:
            return 'action'
        else:
            return 'default_action'
```

यह एजेंट केवल वर्तमान स्थिति के आधार पर तत्काल प्रतिक्रिया देता है।

2. Model-Based Reflex Agent (मॉडल आधारित प्रतिक्रिया एजेंट)

```
class ModelBasedReflexAgent(Agent):
    def perceive(self):
        # आंतरिक स्थिति अपडेट करें
        pass
    def decide(self):
        # आंतरिक मॉडल के आधार पर निर्णय लें
        pass
```

यह एजेंट आंतरिक मॉडल बनाए रखता है ताकि अतीत की जानकारी के साथ निर्णय लिया जा सके।

3. Goal-Based Agent (लक्ष्य-आधारित एजेंट)

```
class GoalBasedAgent(Agent):
    def __init__(self, environment, goal):
        super().__init__(environment)
        self.goal = goal
    def decide(self):
        # लक्ष्य की प्राप्ति हेतु निर्णय करें
```

```
    pass
```

यह एजेंट कार्यों का चयन अपने लक्ष्य की प्राप्ति को ध्यान में रखकर करता है।

पर्यावरण अनुकरण (Environment Simulation):

AI एजेंट्स को किसी **सिमुलेटेड वातावरण** में रखने की आवश्यकता होती है, ताकि वे वस्तुओं, नियमों और अन्य एजेंट्स के साथ संवाद कर सकें। पर्यावरण का प्रकार और जटिलता एप्लिकेशन के अनुसार बदलती है।

1.8.6 भविष्य की दिशा (Future Directions)

AI और Machine Learning में उन्नति

- एजेंट्स अब जटिल व्यवहार प्रदर्शित कर सकते हैं।
- **Reinforcement Learning**, **Deep Learning**, आदि तकनीकें एजेंट्स को **स्वायत्त रूप से सीखने और निर्णय लेने** में सक्षम बना रही हैं।

पूर्वानुमानात्मक विश्लेषण और जटिल प्रणालियों में उपयोग

- स्मार्ट सिटी प्लानिंग, पर्यावरण प्रबंधन, सार्वजनिक स्वास्थ्य आदि क्षेत्रों में उपयोग बढ़ रहा है।
- **Big Data + IoT** के साथ मिलकर ABMs अधिक प्रभावी और सटीक बन रहे हैं।

सुधारित स्वायत्तता और निर्णय प्रणाली

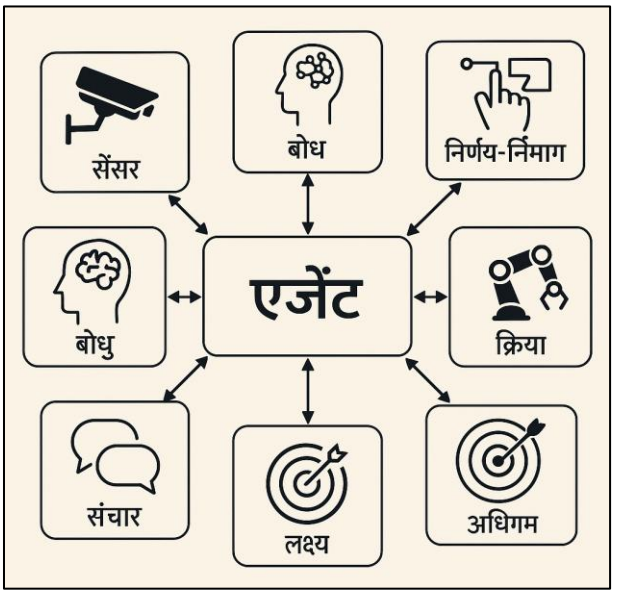
- एजेंट्स भविष्य में और अधिक **स्वतंत्र, बुद्धिमान और जोखिम-संवेदनशील निर्णय लेने** में सक्षम होंगे।
- **Cognitive Modeling** और **Decision Theory** इसमें सहायक बनेंगे।

निष्कर्ष:

एजेंट-आधारित प्रोग्रामिंग:

- **जटिल प्रणालियों को समझने और अनुकरण करने** का शक्तिशाली उपकरण है।
- इसका भविष्य मशीन लर्निंग, डेटा विश्लेषण और AI प्रगति से और भी **सशक्त और प्रभावशाली** बनने जा रहा है।

1.9 एजेंट की संरचना



चित्र 1.9.1: कृत्रिम बुद्धिमत्ता (AI) में एजेंट की संरचना का आरेख

कृत्रिम बुद्धिमत्ता (AI) के संदर्भ में एजेंट की संरचना यह परिभाषित करती है कि एजेंट अपने पर्यावरण को कैसे समझता है, निर्णय कैसे लेता है और लक्ष्यों की प्राप्ति के लिए कार्य कैसे करता है। यह संरचना सरल से लेकर जटिल AI सिस्टमों के निर्माण की नींव होती है।

1.9.1 सेंसर और बोध (Sensors and Perception)

सेंसर एजेंट को पर्यावरण से जानकारी प्राप्त करने की क्षमता देते हैं। बोध वह प्रक्रिया है जिससे एजेंट इन संकेतों की व्याख्या करता है और अपनी आंतरिक स्थिति को अपडेट करता है। यह सरल स्थिति-जांच से लेकर जटिल डेटा प्रोसेसिंग तक हो सकता है।

1.9.2 एक्टुएटर और क्रिया (Actuators and Action)

एक्टुएटर वह माध्यम होते हैं जिनके ज़रिए एजेंट अपने पर्यावरण में हस्तक्षेप करता है। क्रिया वह कार्य है जिसे एजेंट निर्णय लेने के बाद करता है — जैसे किसी बटन को दबाना, दिशा बदलना, या सॉफ़्टवेयर में कोई कमांड देना।

1.9.3 आंतरिक स्थिति (Internal State)

एजेंट की आंतरिक स्थिति उसके पर्यावरण की एक संज्ञानात्मक छवि होती है, जो सेंसरों द्वारा प्राप्त जानकारी पर आधारित होती है। इसमें विश्वास (beliefs), इच्छाएं (desires), और इरादे (intentions) शामिल हो सकते हैं जो निर्णय प्रक्रिया को प्रभावित करते हैं।

1.9.4 निर्णय-निर्माण की क्षमताएँ (Decision-Making Capabilities)

एजेंट द्वारा कौन-सी क्रिया चुनी जाए, यह उसकी निर्णय प्रक्रिया तय करती है। यह नियम-आधारित हो सकता है, या जटिल योजना, तर्क या अधिगम तकनीकों का उपयोग कर सकता है ताकि लक्ष्यों की प्राप्ति हो सके।

1.9.5 अधिगम (Learning)

अधिगम एजेंट को अनुभव के आधार पर अपने प्रदर्शन को बेहतर बनाने में सक्षम बनाता है। यह रीनफोर्समेंट लर्निंग, सुपरवाइज़्ड या अनसुपरवाइज़्ड लर्निंग जैसे तरीकों से हो सकता है।

1.9.6 लक्ष्य और उद्देश्य कार्य (Goals and Objective Functions)

लक्ष्य वे वांछनीय स्थितियाँ होती हैं जिन्हें एजेंट प्राप्त करना चाहता है। उद्देश्य कार्य (Objective Function) यह मापता है कि एजेंट के निर्णय और परिणाम कितनी अच्छी तरह से लक्ष्यों के अनुरूप हैं।

1.9.7 संचार (Communication)

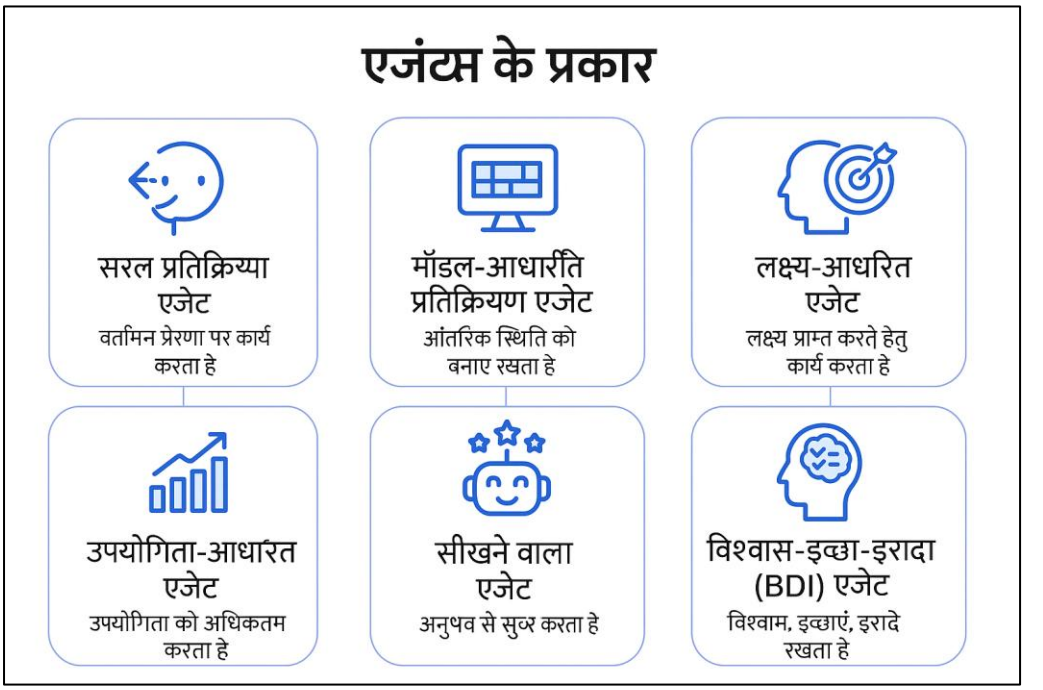
मल्टी-एजेंट सिस्टम में, संचार एजेंटों को जानकारी साझा करने, कार्यों का समन्वय करने और समझौता करने में मदद करता है। यह सामूहिक दक्षता को बढ़ाता है।

1.9.8 स्वायत्तता और नियंत्रण (Autonomy and Control)

स्वायत्तता दर्शाती है कि एजेंट कितनी स्वतंत्रता से निर्णय ले सकता है। नियंत्रण तंत्र (Control Mechanisms) तब आवश्यक हो जाते हैं जब जटिल या सहयोगात्मक वातावरण में एजेंट को निर्देशित करना हो।

यदि आप इस आरेख को किसी रिपोर्ट या प्रस्तुति में सम्मिलित करना चाहते हैं, तो मैं इसे एक स्पष्ट हिंदी लेबल सहित संशोधित भी कर सकता हूँ। बताइए यदि ऐसा करना हो।

1.10 एजेंट्स के प्रकार (Types of Agents in AI)



चित्र 1.10: कृत्रिम बुद्धिमत्ता में पाए जाने वाले मुख्य एजेंट्स के प्रकारों का आरेख
(यह चित्र सरल प्रतिक्रिया एजेंट से लेकर BDI एजेंट तक के विभिन्न प्रकारों और उनके लक्षणों को प्रदर्शित करता है, जिससे यह स्पष्ट होता है कि ये एजेंट्स अपने वातावरण में कैसे कार्य करते हैं।)

कृत्रिम बुद्धिमत्ता (AI) में एजेंट्स को उनके **संचालन कौशल**, **निर्णय-प्रक्रिया**, और **पर्यावरण से संवाद** के आधार पर विभिन्न प्रकारों में वर्गीकृत किया जाता है। इन वर्गों को समझने से यह निर्धारित करना आसान हो जाता है कि कौन-सा एजेंट किस कार्य के लिए उपयुक्त है।

1.10.1 सरल प्रतिक्रिया एजेंट (Simple Reflex Agents)

- केवल **मौजूदा इनपुट (percept)** के आधार पर कार्य करते हैं।
- इतिहास की कोई जानकारी नहीं रखते।
- उदाहरण: वैक्यूम क्लीनर जो दीवार से टकराने पर दिशा बदलता है।

लाभ: तेज़ प्रतिक्रिया, सरलता

सीमा: सीमित समझ, केवल वर्तमान पर निर्भर

1.10.2 मॉडल-आधारित प्रतिक्रिया एजेंट (Model-Based Reflex Agents)

- **आंतरिक स्थिति** बनाए रखते हैं — यानी वे दुनिया का एक मानसिक मॉडल रखते हैं।
- आंशिक रूप से सुलभ वातावरण में भी बेहतर निर्णय ले सकते हैं।

लाभ: जटिलता के साथ भी निर्णय लेने में सक्षम

सीमा: ज़्यादा गणनात्मक संसाधन की आवश्यकता

1.10.3 लक्ष्य-आधारित एजेंट (Goal-Based Agents)

- एक या अधिक **लक्ष्यों** की प्राप्ति हेतु कार्य करते हैं।
- वर्तमान स्थिति, संभावित क्रियाएं और उनका परिणाम मूल्यांकन कर निर्णय लेते हैं।

लाभ: लचीलापन, स्मार्ट निर्णय

सीमा: लक्ष्य निर्धारण और योजना बनाना आवश्यक

1.10.4 उपयोगिता-आधारित एजेंट (Utility-Based Agents)

- प्रत्येक स्थिति के लिए **संतुष्टि मापने वाला फ़ंक्शन (Utility Function)** होता है।
- केवल लक्ष्य प्राप्ति नहीं, बल्कि **सर्वोत्तम विकल्प का चुनाव** करते हैं।

लाभ: बेहतर विकल्प विश्लेषण

सीमा: उपयोगिता फ़ंक्शन पर निर्भरता

1.10.5 सीखने वाले एजेंट (Learning Agents)

- समय के साथ **अपने अनुभवों से प्रदर्शन में सुधार** करते हैं।
- चार घटकों से मिलकर बने होते हैं:
    1. **Learning Element** – सुधार करता है
    2. **Performance Element** – निर्णय करता है
    3. **Critic** – प्रतिक्रिया देता है
    4. **Problem Generator** – नए अनुभव उत्पन्न करता है

लाभ: अनुकूलनशीलता, स्वतः सुधार

सीमा: अधिगम में समय और डाटा की आवश्यकता

1.10.6 विश्वास-इच्छा-इरादा (BDI) एजेंट्स (Belief-Desire-Intention Agents)

- **मानव जैसी व्यावहारिक सोच** का अनुकरण करते हैं।
- तीन घटक:
    - **Beliefs (विश्वास):** पर्यावरण की जानकारी
    - **Desires (इच्छाएं):** प्राप्त की जाने वाली स्थितियाँ
    - **Intentions (इरादे):** कार्य योजनाएँ

लाभ: अत्यधिक जटिल निर्णय-क्षमता

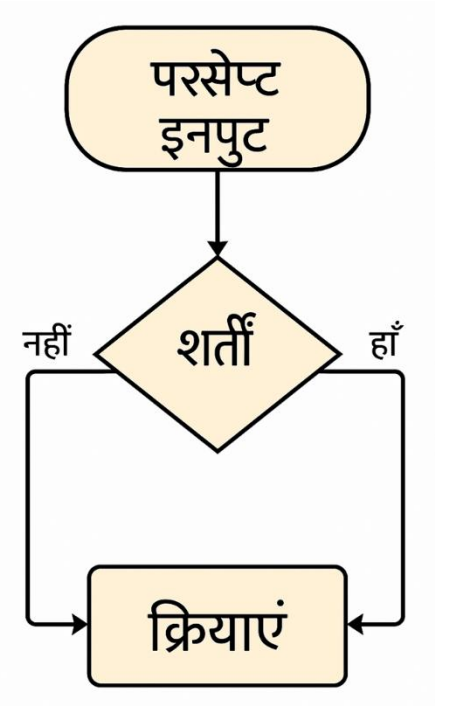
सीमा: डिज़ाइन और कार्यान्वयन में जटिलता

✅ **निष्कर्ष:**

हर एजेंट प्रकार की अपनी विशेषताएँ होती हैं, और वे **अलग-अलग समस्याओं** और **वातावरणों** के लिए उपयुक्त होते हैं।

सही प्रकार का चयन करके, हम ऐसे AI सिस्टम बना सकते हैं जो **प्रभावी, अनुकूलनशील और उद्देश्य-सक्षम** हों।

1.11 सरल प्रतिक्रिया एजेंट्स (Simple Reflex Agents)



चित्र 1.11.1: कृत्रिम बुद्धिमत्ता में सरल प्रतिक्रिया एजेंट्स के कार्यप्रणाली का अवस्थांतर आरेख

(यह चित्र दर्शाता है कि एजेंट कैसे वर्तमान इनपुट (Percept) के आधार पर एक निश्चित क्रिया (Action) करता है, बिना पूर्व इतिहास को ध्यान में रखे।)

1.11.1 कार्य तंत्र (Operation Mechanism)

सरल प्रतिक्रिया एजेंट एक **सीधे नियम आधारित प्रणाली** पर कार्य करता है जिसे "अगर–तो (If-Then)" नियम कहा जाता है।

उदाहरण:

अगर तापमान 25°C से अधिक है

तो एसी चालू करें।

यह एजेंट केवल वर्तमान इनपुट के आधार पर निर्णय लेता है और पिछली स्थितियों या अनुभवों को ध्यान में नहीं रखता।

1.11.2 लाभ (Advantages)

1. **सरलता (Simplicity):**

   डिज़ाइन करना आसान, समझने योग्य।

2. **दक्षता (Efficiency):**

   स्थिति बदलते ही तेज़ी से प्रतिक्रिया करता है।

3. **विश्वसनीयता (Reliability):**

   स्थिर वातावरण में पूर्वनिर्धारित नियमों के आधार पर विश्वसनीय परिणाम देता है।

1.11.3 सीमाएँ (Limitations)

1. **संदर्भ की कमी (Lack of Context):**

   यह एजेंट अपने कार्यों का कोई इतिहास नहीं रखता, जिससे यह जटिल या गतिशील वातावरण में कमजोर पड़ता है।

2. **सीमित लचीलापन (Limited Flexibility):**

   केवल ऐसे कार्य कर सकता है जिनमें वर्तमान इनपुट से सीधा क्रिया संबंध हो।

3. 📈 **स्केलेबिलिटी की समस्या (Scalability Issues):**

   जैसे-जैसे नियमों की संख्या बढ़ती है, सिस्टम को बनाए रखना और अपडेट करना कठिन हो जाता है।

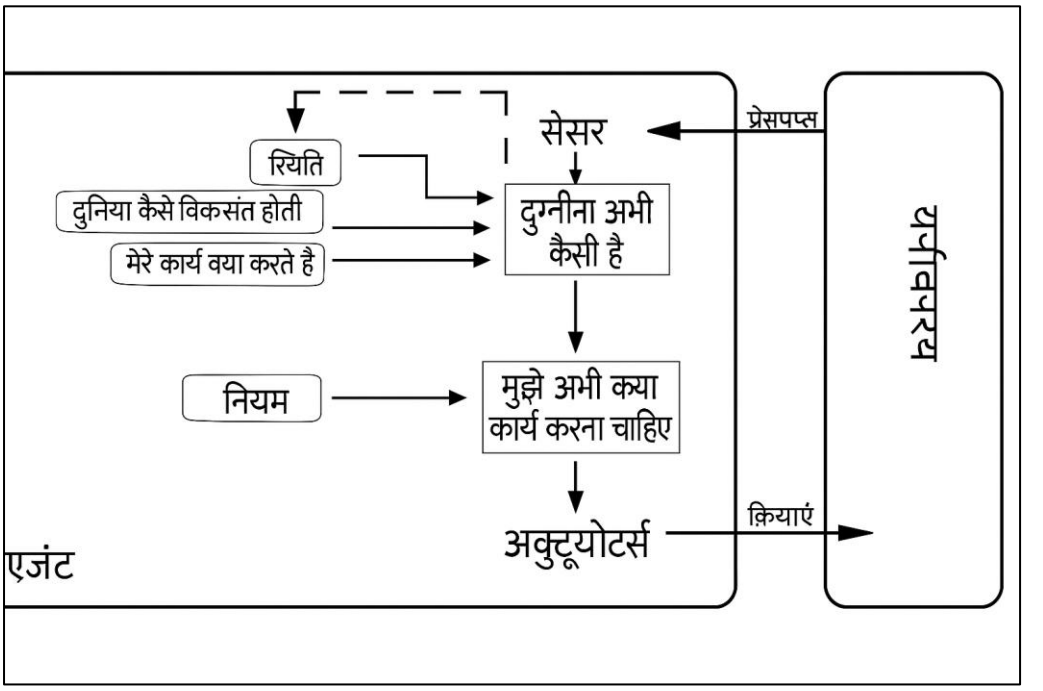
निष्कर्ष (Conclusion):

सरल प्रतिक्रिया एजेंट्स कृत्रिम बुद्धिमत्ता में **आधारभूत अवधारणा** हैं।

वे यह दर्शाते हैं कि कैसे **बिना जटिल एल्गोरिद्म के, सीधे-सरल नियमों से** भी कुछ कार्य स्वायत्त रूप से किए जा सकते हैं।

हालाँकि, जटिल समस्याओं के समाधान के लिए इन्हीं सीमाओं के कारण **अधिक उन्नत एजेंट्स** की आवश्यकता होती है।

1.12 मॉडल-आधारित प्रतिक्रिया एजेंट्स (Model-Based Reflex Agents)



चित्र 1.12.1: मॉडल-आधारित प्रतिक्रिया एजेंट का आरेख

(यह चित्र दर्शाता है कि एजेंट कैसे आंतरिक स्थिति और विश्व मॉडल के आधार पर आंशिक रूप से सुलभ वातावरण में निर्णय लेता है।)

मॉडल-आधारित प्रतिक्रिया एजेंट्स, **सरल प्रतिक्रिया एजेंट्स की तुलना में अधिक उन्नत** होते हैं। ये एजेंट अपने वातावरण की **आंशिक जानकारी के बावजूद प्रभावी निर्णय** लेने में सक्षम होते हैं क्योंकि ये **आंतरिक स्थिति (internal state)** बनाए रखते हैं।

1.12.1 आंतरिक स्थिति और विश्व मॉडल (Internal State and World Model)

आंतरिक स्थिति (Internal State):

एजेंट का वर्तमान समझ — यानी वह पर्यावरण को कैसे "देखता" है।

यह स्थिति लगातार निम्नलिखित से अपडेट होती है:

- **सेंसरी इनपुट्स (Sensory Inputs):**
  एजेंट के सेंसर से मिलने वाली तत्काल जानकारी।
- **ऐतिहासिक डेटा (Historical Data):**
  पिछले इनपुट्स और एजेंट की स्वयं की क्रियाओं का रिकॉर्ड।
- **निष्कर्ष (Inferences):**
  अनुमान या तर्क जो एजेंट अपने विश्व मॉडल के आधार पर करता है।

विश्व मॉडल (World Model):

पर्यावरण को समझने और अनुमान लगाने के लिए उपयोग किए जाने वाले नियमों और ज्ञान का सेट:

- **परिवर्तन के नियम (Rules of Change):**
  वातावरण कैसे बदलता है, यह बताने वाले सिद्धांत।
- **कारण-प्रभाव संबंध (Causality Relations):**
  एजेंट की क्रियाओं और परिणामों के बीच का संबंध।
- **स्थिति संक्रमण (State Transition Dynamics):**
  एक स्थिति से दूसरी स्थिति तक कैसे पहुँचा जाता है।

निर्णय-निर्माण में महत्व (Significance in Decision-Making)

1. **आंशिक अवलोकन की पूर्ति (Compensate for Partial Observability):**
   एजेंट उन पहलुओं का अनुमान लगा सकता है जो सीधे दिखाई नहीं दे रहे।
2. **भविष्य की स्थिति का पूर्वानुमान (Predict Future States):**
   एजेंट अनुमान लगा सकता है कि वर्तमान क्रियाओं से भविष्य में क्या होगा।
3. **परिवर्तन के प्रति अनुकूलता (Adapt to Environmental Changes):**
   वातावरण में बदलाव होने पर अपनी रणनीति को समायोजित कर सकता है।

1.12.2 लाभ और अनुप्रयोग (Advantages and Applications)

लाभ:

- **सुधारित निर्णय-निर्माण (Improved Decision-Making):**
  एजेंट पर्यावरण के परिणामों को ध्यान में रखकर निर्णय लेता है।

- **आंशिक जानकारी में दक्षता (Partial Observability Handling):**
  अधूरी जानकारी के बावजूद सटीक निर्णय लेने की क्षमता।
- **जटिल वातावरण में प्रभावशीलता (Efficiency in Complex Environments):**
  सरल नियमों की बजाय पूरे मॉडल पर आधारित कार्य।

अनुप्रयोग क्षेत्र:

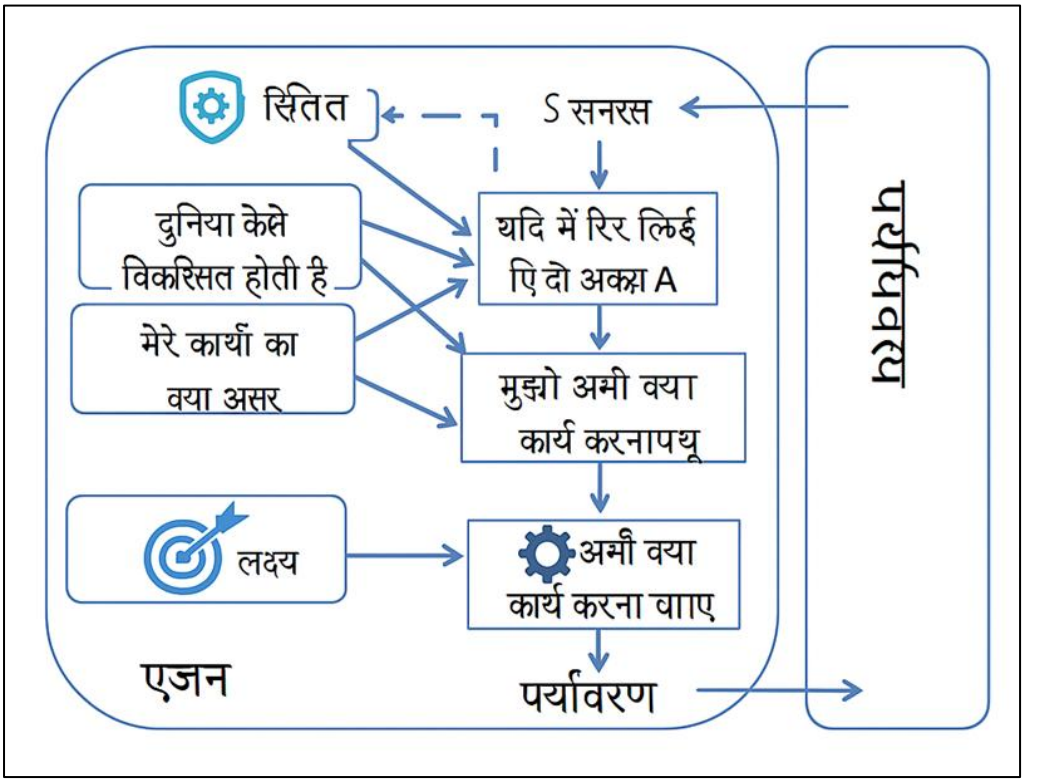
- **स्वायत्त वाहन (Autonomous Vehicles):**
  ट्रैफ़िक और मौसम जैसे बदलते परिवेश में संचालन।
- **घरेलू रोबोट (Household Robots):**
  अपरिभाषित और अधूरी जानकारी के बीच कार्य करना।
- **नेटवर्क एजेंट्स (Software Agents):**
  नेटवर्क संसाधनों का स्मार्ट प्रबंधन, जैसे — क्लाउड लोड बैलेंसिंग।

निष्कर्ष:

मॉडल-आधारित प्रतिक्रिया एजेंट्स, **सरल एजेंट्स और लक्ष्य-आधारित एजेंट्स के बीच की कड़ी** हैं।

ये एजेंट बेहतर समझ, भविष्यवाणी, और अनुकूलन के माध्यम से अधिक जटिल कार्यों में प्रभावी होते हैं।

1.13 लक्ष्य-आधारित एजेंट्स (Goal-Based Agents)



चित्र 1.13: मॉडल और लक्ष्य-आधारित एजेंट का आरेख

(यह आरेख दर्शाता है कि कैसे एजेंट वर्तमान स्थिति का मूल्यांकन कर, विभिन्न संभावित क्रियाओं के परिणामों का आकलन करता है और उन विकल्पों को चुनता है जो पूर्व निर्धारित लक्ष्यों की ओर ले जाते हैं।)

लक्ष्य-आधारित एजेंट कृत्रिम बुद्धिमत्ता में एक महत्वपूर्ण प्रगति को दर्शाते हैं। ये एजेंट न केवल तत्काल स्थितियों पर प्रतिक्रिया करते हैं, बल्कि **लक्ष्य की प्राप्ति हेतु योजनाबद्ध निर्णय** भी लेते हैं।

1.13.1 अवधारणा और कार्यप्रणाली (Concept and Operation)

लक्ष्य-आधारित एजेंट निम्नलिखित कदमों पर कार्य करते हैं:

लक्ष्य निर्माण (Goal Formulation):

एजेंट को पहले यह जानना होता है कि उसे **क्या प्राप्त करना है**।

लक्ष्य ऐसी भाषा में परिभाषित होते हैं जिसे एजेंट समझ और तुलना कर सके।

स्थिति मूल्यांकन (State Evaluation):

एजेंट संभावित क्रियाओं के परिणामस्वरूप आने वाली **भविष्य की स्थितियों का मूल्यांकन** करता है।

निर्णय-निर्माण (Decision Making):

एजेंट वह क्रिया चुनता है जो लक्ष्य प्राप्ति की **सबसे अधिक संभावना** रखती हो।

1.13.2 लाभ (Advantages)

1. **लचीलापन (Flexibility):**
   एजेंट पर्यावरण के अनुसार अपना व्यवहार बदल सकता है।
2. **समस्या समाधान में दक्षता (Efficiency in Problem Solving):**
   कई संभावित रास्तों का मूल्यांकन कर **सबसे उपयुक्त रणनीति** चुनता है।
3. **योजना बनाने की क्षमता (Capability for Planning):**
   ये एजेंट **आगे की कई क्रियाओं की योजना** बना सकते हैं — जटिल समस्याओं में विशेष रूप से उपयोगी।

1.13.3 कार्यान्वयन चुनौतियाँ (Implementation Challenges)

1. **लक्ष्य निर्माण की जटिलता (Complexity in Goal Formulation):**
   जटिल या बहुपरतीय लक्ष्यों को परिभाषित करना कठिन हो सकता है।
2. **गणनात्मक आवश्यकताएँ (Computational Demands):**
   सभी संभावित भविष्य स्थितियों का आकलन भारी प्रोसेसिंग संसाधनों की मांग करता है।
3. **गतिशील वातावरण (Dynamic Environments):**
   वातावरण में तेजी से होने वाले परिवर्तनों को ट्रैक करना और योजना को अद्यतन रखना चुनौतीपूर्ण हो सकता है।

1.13.4 अनुप्रयोग (Applications)

- **स्वायत्त वाहन:**
  रास्ता तय करना, ट्रैफ़िक के अनुसार रणनीति बदलना।

- **व्यक्तिगत सहायक:**

  उपयोगकर्ता के लक्ष्यों के अनुसार कार्यों की योजना बनाना।

- **रणनीतिक गेम AI:**

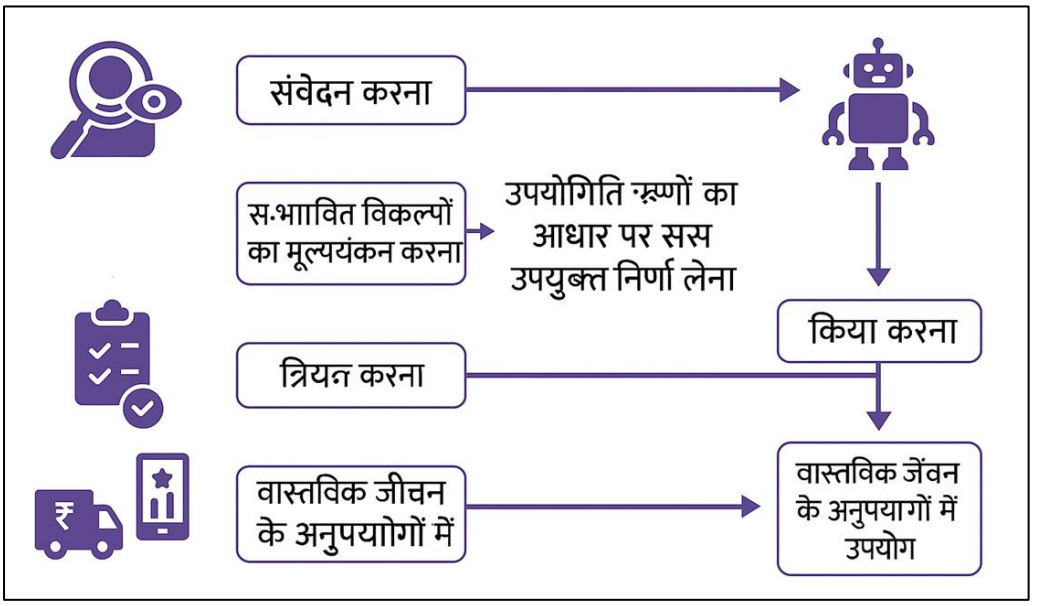
  जीत की स्थिति प्राप्त करने के लिए चालों की योजना बनाना।

निष्कर्ष (Conclusion)

लक्ष्य-आधारित एजेंट, **AI को जटिल, गतिशील समस्याओं से निपटने में अधिक स्वायत्त और प्रभावी** बनाते हैं।

ये एजेंट भविष्य की योजना, आकलन और अनुकूलन जैसे गुणों से युक्त होते हैं, जो उन्हें सरल एजेंट्स से कहीं अधिक शक्तिशाली बनाते हैं।

1.14 अवधारणा और कार्यप्रणाली (Concept and Operation)



चित्र 1.14.1: उपयोगिता-आधारित एजेंट की अवधारणा को दर्शाने वाला आरेख (बाएं से दाएं प्रवाह में)

उपयोगिता-आधारित एजेंट्स, लक्ष्य-आधारित एजेंट्स से एक कदम आगे बढ़कर, निर्णय लेने में **"संतोष" या "लाभ"** को परिमाणित (quantify) करते हैं।

उपयोगिता फ़ंक्शन (Utility Function):

- एक गणितीय मॉडल जो विभिन्न स्थितियों को एक मान देता है।
- यह मान उस स्थिति के प्रति एजेंट की पसंद/संतोष को दर्शाता है।

निर्णय प्रक्रिया (Decision Making):

- संभावित क्रियाओं के **अपेक्षित उपयोगिता (Expected Utility)** का मूल्यांकन किया जाता है।
- एजेंट वह क्रिया चुनता है जो **उच्चतम उपयोगिता** प्रदान करती हो।

1.14.2 लाभ (Advantages)

1. **लचीलापन और अनुकूलनशीलता (Flexibility and Adaptability):**
   एजेंट वातावरण और लक्ष्यों में बदलाव के अनुसार व्यवहार समायोजित कर सकता है।

2. **अनिश्चितता से निपटना (Handling Uncertainty):**
   उपयोगिता-आधारित मूल्यांकन के माध्यम से जोखिम और लाभों का संतुलन कर सकता है।
3. **संतोष का अनुकूलन (Optimizing Satisfaction):**
   केवल लक्ष्य प्राप्त करना नहीं, बल्कि **सबसे संतोषजनक तरीका** अपनाना।

1.14.3 कार्यान्वयन की चुनौतियाँ (Implementation Challenges)

1. **उपयोगिता फ़ंक्शन की डिज़ाइन (Designing the Utility Function):**
   एजेंट की प्राथमिकताओं का सटीक प्रतिनिधित्व करना कठिन हो सकता है।
2. **गणनात्मक जटिलता (Computational Complexity):**
   सभी संभावित स्थितियों की उपयोगिता का मूल्यांकन करना भारी संसाधन ले सकता है।
3. **गतिशील प्राथमिकताएँ (Dynamic Preferences):**
   यदि उपयोगकर्ता की पसंद समय के साथ बदलती है, तो एजेंट को अपने उपयोगिता फ़ंक्शन को भी अपडेट करना पड़ता है।

1.14.4 अनुप्रयोग (Applications)

क्षेत्र	उदाहरण
वित्तीय ट्रेडिंग सिस्टम	जोखिम-संतुलित लाभ अधिकतम करने हेतु निर्णय लेना
संसाधन आवंटन (Resource Allocation)	जैसे – क्लाउड कंप्यूटिंग या लॉजिस्टिक्स में लागत-प्रभाविता
अनुशंसा प्रणाली (Recommendation Systems)	उपयोगकर्ता की पसंद के आधार पर व्यक्तिगत सुझाव देना

निष्कर्ष (Conclusion)

उपयोगिता-आधारित एजेंट्स **AI का एक परिष्कृत रूप** हैं, जो न केवल लक्ष्य की प्राप्ति करते हैं, बल्कि उसे इस प्रकार करते हैं कि **एजेंट या उपयोगकर्ता की संतुष्टि अधिकतम** हो सके।
यह दृष्टिकोण ऐसे **स्मार्ट, अनुकूलनीय और कुशल AI सिस्टम** के निर्माण की अनुमति देता है जो जटिल निर्णय-परिस्थितियों में बेहतर प्रदर्शन कर सकें।

"Man needs his difficulties because they are necessary to enjoy success. But true progress lies in using knowledge to build intelligent systems that help humanity."

*"मनुष्य को कठिनाइयों की आवश्यकता होती है, क्योंकि वही सफलता का आनंद लेना सिखाती हैं। लेकिन सच्ची प्रगति तब है जब हम ज्ञान का उपयोग मानवता की सहायता करने वाले बुद्धिमान प्रणालियाँ बनाने में करें।"

— डॉ. ए. पी. जे. अब्दुल कलाम

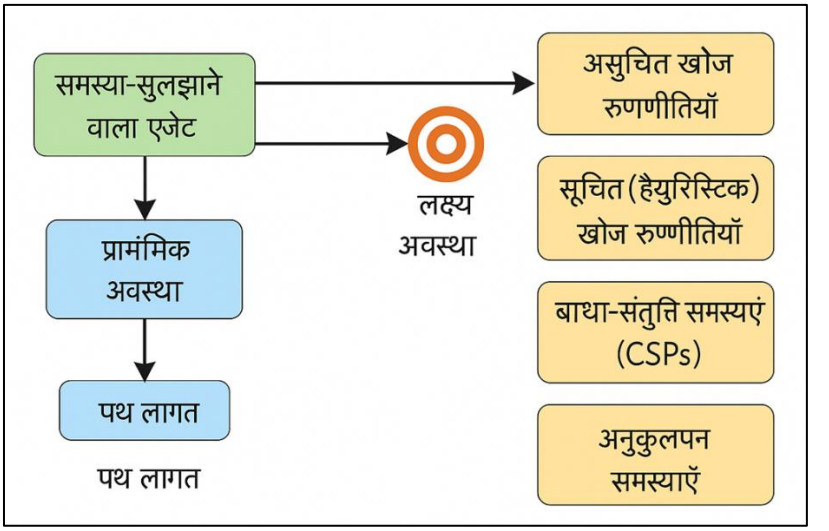
"The true sign of intelligence is not knowledge but imagination. In AI, we imagine machines that think – and that is the beginning of wisdom."

*"बुद्धिमत्ता की असली पहचान ज्ञान नहीं, बल्कि कल्पना है। कृत्रिम बुद्धिमत्ता में हम ऐसी मशीनों की कल्पना करते हैं जो सोच सकती हैं — और वहीं से समझदारी की शुरुआत होती है।"

— अल्बर्ट आइंस्टीन

यूनिट – 2: खोज के माध्यम से समस्या समाधान

2.1 समस्या-सुलझाने वाले एजेंट



चित्र 2.1: एक आरेख जो कृत्रिम बुद्धिमत्ता में समस्या-सुलझाने वाले एजेंट की वैचारिक रूपरेखा और रणनीतियों को दर्शाता है।

समस्या-सुलझाने वाले एजेंट कृत्रिम बुद्धिमत्ता की ऐसी प्रणालियाँ होती हैं जिन्हें प्रारंभिक अवस्था से लक्ष्य अवस्था तक पहुँचने के लिए क्रियाओं की एक अनुक्रमिक श्रृंखला उत्पन्न करने और उसे निष्पादित करने हेतु डिज़ाइन किया गया है। ये एजेंट विभिन्न रणनीतियों और एल्गोरिदम का उपयोग करके सबसे कुशल मार्ग की पहचान करते हैं, जिससे वे कई एआई अनुप्रयोगों में अत्यंत उपयोगी सिद्ध होते हैं।

2.1.1 संकल्पना और रूपरेखा (Concept and Framework)

समस्या-सुलझाने वाले एजेंट का आधार लक्ष्य-उन्मुख व्यवहार (goal-oriented behaviour) होता है, जहाँ एजेंट को स्पष्ट रूप से एक समस्या सौंपी जाती है, जिसे निम्नलिखित तत्वों के माध्यम से परिभाषित किया जाता है:

- **प्रारंभिक अवस्था (Initial State):** वह प्रारंभिक स्थिति जहाँ से एजेंट अपनी क्रियाओं की शुरुआत करता है।
- **क्रियाएँ (Actions):** वे सभी संभावित कदम या गतिविधियाँ जिन्हें एजेंट वर्तमान अवस्था से दूसरी अवस्था में जाने के लिए उठा सकता है।
- **लक्ष्य अवस्था (Goal State):** वह इच्छित अंतिम स्थिति जो समस्या के समाधान का प्रतिनिधित्व करती है।
- **पथ लागत (Path Cost):** प्रारंभिक अवस्था से लक्ष्य अवस्था तक पहुँचने वाले प्रत्येक मार्ग की संख्यात्मक लागत, जिससे एजेंट विभिन्न समाधानों की कुशलता की तुलना कर सकता है।

इस रूपरेखा के अंतर्गत एजेंट पहले समस्या का सूत्रीकरण करता है, फिर समाधान खोजने के लिए समस्या-स्थान (problem space) में खोज करता है, और अंततः लक्ष्य अवस्था तक पहुँचने के लिए एक क्रियात्मक श्रृंखला को निष्पादित करता है।

2.1.2 समस्या समाधान की रणनीतियाँ (Strategies for Problem Solving)

समस्या-सुलझाने वाले एजेंट विभिन्न रणनीतियों का उपयोग करते हैं ताकि वे खोज स्थान में नेविगेट कर सकें और लक्ष्य तक पहुँचने का इष्टतम मार्ग खोज सकें:

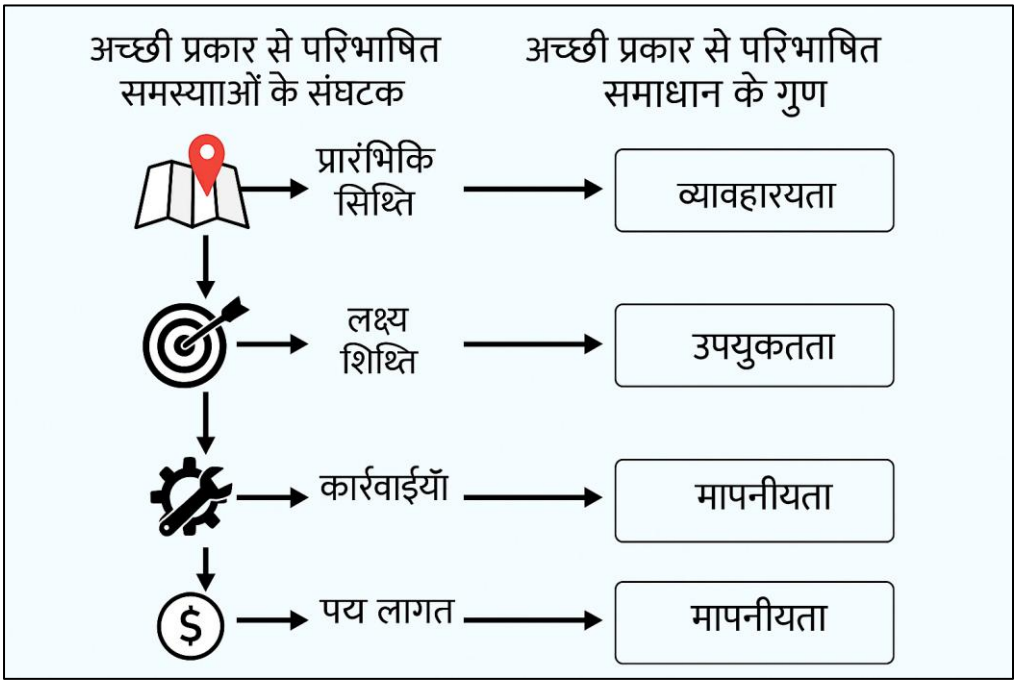
- **असूचित खोज रणनीतियाँ (Uninformed Search Strategies):**
  जैसे कि ब्रेड्थ-फर्स्ट सर्च, डेप्थ-फर्स्ट सर्च और यूनिफॉर्म-कॉस्ट सर्च। ये केवल वही जानकारी उपयोग में लाती हैं जो समस्या-परिभाषा में उपलब्ध होती है, और खोज स्थान को व्यवस्थित ढंग से खोजती हैं।
- **सूचित (हेयुरिस्टिक) खोज रणनीतियाँ (Informed/Heuristic Search Strategies):**
  ये रणनीतियाँ अतिरिक्त ज्ञान (जैसे हेयुरिस्टिक फ़ंक्शन) का उपयोग करती हैं, जो एजेंट को लक्ष्य की दिशा में अधिक कुशलता से मार्गदर्शन करती हैं। उदाहरण: A* सर्च, ग्रीडी बेस्ट-फर्स्ट सर्च।

- **बाधा-संतुष्टि समस्याएँ (Constraint Satisfaction Problems - CSPs):**
  जब कोई समस्या विभिन्न बाधाओं के रूप में परिभाषित होती है, तब एजेंट ऐसे मान ढूँढता है जो सभी बाधाओं को संतुष्ट करते हैं।
- **अनुकूलन समस्याएँ (Optimization Problems):**
  जब किसी समस्या में अनेक संभावित समाधानों में से सर्वोत्तम समाधान खोजने की आवश्यकता होती है, तो जेनेटिक एल्गोरिद्म, सिम्युलेटेड एनीलिंग जैसे एल्गोरिद्म उपयोग किए जाते हैं।

निष्कर्ष:

प्रत्येक रणनीति के अपने लाभ और विशिष्ट अनुप्रयोग होते हैं। उपयुक्त रणनीति का चयन समस्या की प्रकृति, लक्ष्य की जटिलता, और समाधान की सीमाओं पर निर्भर करता है।

2.2 सुव्यवस्थित समस्याएँ और उनके समाधान



चित्र 2.2.1: कृत्रिम बुद्धिमत्ता (AI) में सुव्यवस्थित समस्याओं के घटक और उनके समाधानों की विशेषताओं को दर्शाने वाला आरेख (बाएं से दाएं प्रवाह में)।

(**Well-defined Problems and Solutions**)

कृत्रिम बुद्धिमत्ता (AI) और संगणकीय समस्या समाधान के संदर्भ में, एक **सुव्यवस्थित समस्या** वह होती है जिसमें प्रारंभिक अवस्था, लक्ष्य अवस्था, और अवस्थाओं के बीच संक्रमण हेतु उपलब्ध क्रियाओं की स्पष्ट रूप से परिभाषा होती है। इस प्रकार की समस्याएँ AI एजेंट्स और एल्गोरिद्म के लिए उपयुक्त होती हैं, जिससे वे संरचित और कुशल ढंग से समाधान खोज सकें।

2.2.1 सुव्यवस्थित समस्याओं के घटक (Components of Well-defined Problems)

एक सुव्यवस्थित समस्या में निम्नलिखित प्रमुख घटक होते हैं:

- **प्रारंभिक अवस्था (Initial State):**
  समस्या हल करने की प्रक्रिया की प्रारंभिक स्थिति। यही वह बिंदु है जहाँ से एजेंट समाधान की खोज प्रारंभ करता है।

- **लक्ष्य अवस्था (Goal State):**
  स्पष्ट रूप से परिभाषित वह स्थिति जिसे एजेंट को प्राप्त करना है। यह समस्या समाधान की पूर्णता का संकेत देती है।
- **क्रियाएँ या संचालक (Actions or Operators):**
  वे सभी संभावित कदम या क्रियाएँ जो एजेंट विभिन्न अवस्थाओं के बीच संक्रमण हेतु कर सकता है।
  ये क्रियाएँ विशेष नियमों द्वारा नियंत्रित होती हैं जो उनकी वैधता और प्रभाव को परिभाषित करती हैं।
- **पथ लागत (Path Cost):**
  एक लागत फ़ंक्शन जो प्रारंभिक अवस्था से लक्ष्य अवस्था तक जाने वाले प्रत्येक मार्ग को एक संख्यात्मक लागत प्रदान करता है। यह विभिन्न समाधानों की कुशलता की तुलना हेतु उपयोगी होता है।

2.2.2 सुव्यवस्थित समाधानों की विशेषताएँ (Characteristics of Well-defined Solutions)

सुव्यवस्थित समस्याओं के समाधान में कुछ विशेष विशेषताएँ पाई जाती हैं जो उन्हें तुलनात्मक और मूल्यांकन योग्य बनाती हैं:

- **व्यवहार्यता (Feasibility):**
  समाधान केवल तभी मान्य है जब वह परिभाषित क्रियाओं और नियमों के अंतर्गत प्रारंभिक अवस्था से लक्ष्य अवस्था तक पहुँचे।
- **सर्वोत्तमता (Optimality):**
  यदि समाधान सबसे कम लागत या संसाधन में लक्ष्य को प्राप्त करता है, तो वह सर्वोत्तम माना जाता है। यह AI में एक प्रमुख उद्देश्य होता है।
- **मापनयोग्यता (Measurability):**
  समाधान की गुणवत्ता या दक्षता को **पथ लागत** या अन्य मात्रात्मक मानकों द्वारा मापा जा सकता है, जिससे विभिन्न समाधानों की तुलना की जा सकती है।

निष्कर्ष (Conclusion):

सुव्यवस्थित समस्याओं और उनके समाधानों की स्पष्ट समझ, प्रभावी AI एजेंट और एल्गोरिद्म डिजाइन करने के लिए अनिवार्य है। यह एक संरचित ढांचा प्रदान करता है, जिसके अंतर्गत एजेंट समस्या-स्थान को प्रभावी रूप से नेविगेट कर सकते हैं, संभावित क्रियाओं का मूल्यांकन कर सकते हैं, और सबसे उपयुक्त मार्ग का चयन कर लक्ष्य को प्राप्त कर सकते हैं।

2.3 उदाहरण समस्याएँ (Examples Problems)

कृत्रिम बुद्धिमत्ता (AI) में, विभिन्न **सुव्यवस्थित समस्याएँ** ऐसे मानक (benchmarks) के रूप में कार्य करती हैं जिनका उपयोग **समस्या-सुलझाने वाले एल्गोरिद्म और एजेंटों के विकास, परीक्षण और सुधार** हेतु किया जाता है। ये समस्याएँ सरल पज़ल्स से लेकर जटिल वास्तविक जीवन की चुनौतियों तक फैली होती हैं और AI तकनीकों की क्षमताओं और सीमाओं को समझने में सहायक होती हैं।

2.3.1 पज़ल समस्याएँ (Puzzle Problems)

8-पज़ल (8-Puzzle)

(और इसके विविध रूप जैसे 15-पज़ल)

इस पज़ल में, एक खिलाड़ी को टाइल्स को एक ग्रिड पर एक विशेष विन्यास (configuration) में व्यवस्थित करना होता है, जो एक अव्यवस्थित प्रारंभिक स्थिति से शुरू होता है। चुनौती यह है कि खाली स्थान सीमित होता है और लक्ष्य न्यूनतम चालों में टाइल्स को लक्ष्य क्रम में व्यवस्थित करना होता है।

🔧 **कार्यशील उदाहरण: 8-पज़ल (Working Example: The 8-Puzzle)**

संरचना:

- यह एक 3x3 ग्रिड है जिसमें **8 संख्यात्मक टाइल्स** होती हैं और एक **रिक्त स्थान** (blank space) होता है।
- टाइल्स को क्षैतिज या ऊर्ध्वाधर रूप से खाली स्थान की ओर खिसकाया जा सकता है।
- उद्देश्य है कि प्रारंभिक स्थिति से टाइल्स को लक्षित क्रम में लाया जाए।

प्रारंभिक स्थिति (Initial Configuration):

| 2 | 8 | 3 |

| 1 | 6 | 4 |

| 7 | | 5 |

लक्ष्य स्थिति (Goal Configuration):

| 1 | 2 | 3 |

| 4 | 5 | 6 |

| 7 | 8 | |

मुख्य घटक (Key Components):

- **प्रारंभिक अवस्था (Initial State):**
  टाइल्स का प्रारंभिक विन्यास
- **क्रियाएँ (Actions):**
  टाइल को खाली स्थान की ओर ऊपर, नीचे, दाएं या बाएं खिसकाना
- **लक्ष्य अवस्था (Goal State):**
  निर्दिष्ट अनुक्रम में टाइल्स की व्यवस्था
- **पथ लागत (Path Cost):**
  प्रत्येक चाल की लागत सामान्यतः 1 मानी जाती है, इस प्रकार कुल चालों की संख्या = कुल लागत

समाधान के लिए खोज एल्गोरिद्म (Problem-Solving with Search Algorithms)

इस समस्या को हल करने के लिए हम *A सर्च एल्गोरिद्म** का उपयोग कर सकते हैं, जो कि हेयुरिस्टिक आधारित खोज तकनीक है। यह वर्तमान अवस्था से लक्ष्य तक पहुँचने की अनुमानित लागत का मूल्यांकन कर **सबसे कुशल मार्ग** खोजता है।

हेयुरिस्टिक फंक्शन: Manhattan Distance

- प्रत्येक टाइल के वर्तमान और लक्ष्य स्थिति के बीच की कुल क्षैतिज और ऊर्ध्वाधर दूरी का योग
- उदाहरण के लिए: टाइल '2' की दूरी = 1 (एक स्थान बाएं), टाइल '8' = 1 नीचे + 1 दाएं = 2, आदि
- सभी टाइल्स के लिए इन दूरीयों का योग = अनुमानित दूरी

समाधान प्रक्रिया (Solution Process):

1. **संभावित चालें उत्पन्न करें (Generate Possible Moves):**
   उदाहरण में, टाइल्स 6, 7 और 5 खाली स्थान की ओर जा सकती हैं।
2. **हेयुरिस्टिक लागू करें (Apply Heuristic):**
   हर चाल के लिए Manhattan Distance की गणना करें।
3. **सर्वोत्तम चाल चुनें (Choose Move):**
   वह चाल चुनें जो लक्ष्य से सबसे कम दूरी पर ले जाती है।

4. **पुनरावृत्ति करें (Repeat):**
   अगली अवस्था से नई संभावनाएँ उत्पन्न करें और यही प्रक्रिया दोहराएँ जब तक लक्ष्य प्राप्त न हो।

उदाहरण समाधान पथ (Example Solution Path):

- टाइल 5 → टाइल 4 → टाइल 1 ... इस तरह से चालें लक्ष्य स्थिति तक ले जाती हैं।
- **वास्तविक पथ** प्रयुक्त एल्गोरिद्म और हेयुरिस्टिक पर निर्भर करता है।

टावर्स ऑफ हनोई: कार्य उदाहरण (Towers of Hanoi: Working Example)

टावर्स ऑफ हनोई एक प्रसिद्ध समस्या है जो **संगणकीय सिद्धांत** और **कृत्रिम बुद्धिमत्ता (AI)** में जटिल समस्याओं को हल करने के लिए **पुनरावृत्ति (recursion)** और **कलनात्मक रणनीति (algorithmic strategy)** के उपयोग को दर्शाती है। यह पहेली तीन खंभों (pegs) और विभिन्न आकारों के डिस्कों से बनी होती है जिन्हें किसी भी खंभे पर खिसकाया जा सकता है।

उद्देश्य (Objective):

सभी डिस्कों को एक खंभे (जैसे A) से किसी अन्य खंभे (जैसे C) पर ले जाना, निम्नलिखित नियमों का पालन करते हुए:

1. एक बार में केवल एक डिस्क ही स्थानांतरित की जा सकती है।
2. प्रत्येक चाल में केवल सबसे ऊपरी डिस्क को ही हटाया जा सकता है।
3. कोई बड़ा डिस्क कभी भी छोटे डिस्क के ऊपर नहीं रखा जा सकता।

प्रारंभिक विन्यास (Initial Configuration):

मान लीजिए कि हमारे पास 3 डिस्क हैं:

yaml

CopyEdit

खंभा A: |3| |2| |1|

खंभा B: | || || |

खंभा C: | || || |

(**|1|** सबसे छोटा डिस्क और **|3|** सबसे बड़ा डिस्क को दर्शाता है।)

समाधान रणनीति (Solution Strategy):

इस समस्या का समाधान **रिकर्सन** द्वारा बड़े ही सहज रूप से किया जा सकता है:

1. **आधार मामला (Base Case):** यदि केवल एक डिस्क है, तो उसे सीधे लक्ष्य खंभे (C) पर स्थानांतरित करें।
2. **पुनरावृत्ति चरण (Recursive Step):** यदि n डिस्क हैं, तो पहले n-1 डिस्कों को सहायक खंभे (B) पर स्थानांतरित करें, फिर nवीं डिस्क को लक्ष्य खंभे (C) पर रखें, और अंत में n-1 डिस्कों को सहायक खंभे (B) से लक्ष्य खंभे (C) पर ले जाएँ।

3-डिस्क उदाहरण के लिए समाधान प्रक्रिया:

चरण 1: डिस्क 1 और 2 को खंभा B पर ले जाएँ (C सहायक के रूप में):

- A → C (डिस्क 1)
- A → B (डिस्क 2)
- C → B (डिस्क 1)

चरण 2: डिस्क 3 को खंभा C पर ले जाएँ:

- A → C (डिस्क 3)

चरण 3: डिस्क 1 और 2 को B से C पर ले जाएँ (A सहायक के रूप में):

- B → A (डिस्क 1)
- B → C (डिस्क 2)
- A → C (डिस्क 1)

अंतिम विन्यास (Final Configuration):

खंभा A: | || || |

खंभा B: | || || |

खंभा C: |3| |2| |1|

रिकर्सिव एल्गोरिद्म (Recursive Algorithm):

```
def TowersOfHanoi(n , source, target, auxiliary):
    if n == 1:
        print(f"डिस्क 1 को खंभा {source} से खंभा {target} पर ले जाएँ")
```

```
    return
TowersOfHanoi(n-1, source, auxiliary, target)
print(f"डिस्क {n} को खंभा {source} से खंभा {target} पर ले जाएँ")
TowersOfHanoi(n-1, auxiliary, target, source)
```

AI में प्रासंगिकता:

यह पहेली **रिकर्सन की शक्ति**, **समस्या विभाजन** और **योजना निर्माण** के सिद्धांतों को दर्शाती है, जो कि **AI समस्या-सुलझाने** की जड़ों में अत्यंत महत्वपूर्ण है।

2.3.2 पथ-खोज समस्याएँ (Pathfinding Problems)

कृत्रिम बुद्धिमत्ता (AI) में कई जानी-मानी समस्याएँ हैं जिनका उपयोग समस्या-सुलझाने के एल्गोरिद्म को विकसित करने, परीक्षण करने और सुधारने के लिए किया जाता है। इनमें से दो प्रमुख उदाहरण हैं:

1. ट्रैवलिंग सेल्समैन समस्या (TSP)

परिभाषा:

ट्रैवलिंग सेल्समैन समस्या में, एक विक्रेता को कुछ शहरों की यात्रा करनी होती है और प्रत्येक शहर में एक बार जाकर वापस मूल शहर में लौटना होता है, वह भी सबसे कम दूरी में। यह एक **NP-कठिन अनुकूलन समस्या (optimization problem)** है।

उदाहरण परिदृश्य (Example Scenario):

चार शहर: A, B, C, D

दूरीयाँ (Distance Table):

- A से B: 10
- A से C: 15
- A से D: 20
- B से C: 35
- B से D: 25
- C से D: 30

(ध्यान दें: दूरी सिमेट्रिक है, जैसे A से B = B से A)

उद्देश्य (Goal):

A से शुरू होकर, प्रत्येक शहर में एक बार जाकर A पर वापस लौटना, कुल दूरी न्यूनतम हो।

ब्रूट-फोर्स दृष्टिकोण (Brute-Force Approach):

1. **सभी संभावित मार्गों की सूची बनाएँ:**
    - A → B → C → D → A
    - A → B → D → C → A
    - A → C → B → D → A
    - A → C → D → B → A
    - A → D → B → C → A
    - A → D → C → B → A
2. **प्रत्येक मार्ग की कुल दूरी निकालें:**

    उदाहरण:

    A → B → C → D → A

    दूरी = 10 (A-B) + 35 (B-C) + 30 (C-D) + 20 (D-A) = **95**
3. **न्यूनतम दूरी वाले मार्ग की पहचान करें।**

Python कोड स्निपेट (Code Snippet):

```
from itertools import permutations
distances = {('A', 'B'): 10, ('A', 'C'): 15, ('A', 'D'): 20,
        ('B', 'C'): 35, ('B', 'D'): 25, ('C', 'D'): 30}
# दूरी को सिमेट्रिक बनाएं
for k, v in list(distances.items()):
    distances[(k[1], k[0])] = v
cities = ['B', 'C', 'D']
city_permutations = permutations(cities)
def calculate_distance(route):
    total = 0
    prev = 'A'
```

```
    for city in route:
        total += distances[(prev, city)]
        prev = city
    total += distances[(prev, 'A')]
    return total
shortest_distance = float('inf')
shortest_route = None
for route in city_permutations:
    d = calculate_distance(route)
    if d < shortest_distance:
        shortest_distance = d
        shortest_route = route
print(f"Shortest route: A -> {' -> '.join(shortest_route)} -> A with distance {shortest_distance}")
```

प्रयोग और निष्कर्ष:

यह विधि छोटी संख्या में शहरों के लिए उपयुक्त है। बड़े डेटा सेट में, हम ह्यूरिस्टिक विधियाँ जैसे **Nearest Neighbor**, **Genetic Algorithm**, या **Ant Colony Optimization** का प्रयोग करते हैं, जो समाधान को करीब लाती हैं लेकिन हमेशा इष्टतम (optimal) नहीं होतीं।

2. भूलभुलैया नेविगेशन (Maze Navigation)

विवरण:

एक एजेंट को एक भूलभुलैया के आरंभिक बिंदु से लक्ष्य बिंदु तक पहुँचने के लिए रास्ता खोजना होता है।

विशेषताएँ:

- यह समस्या जटिल वातावरण में नेविगेशन, निर्णय-निर्माण, और गतिशील रणनीतियों को दर्शाती है।

- एजेंट को **डेड एंड्स (dead ends)** से सीखना पड़ता है और फिर बेहतर रास्ते खोजना होता है।
- **सर्च एल्गोरिद्म** जैसे **DFS**, **BFS**, **A*** आदि का प्रयोग इस कार्य में किया जाता है।

2.3.3 भूलभुलैया नेविगेशन समस्या (Maze Navigation Problem: कार्य उदाहरण)

भूलभुलैया नेविगेशन समस्या में एक एजेंट को एक जटिल भूलभुलैया में प्रारंभिक बिंदु से लक्ष्य तक पहुँचने का रास्ता खोजने का कार्य सौंपा जाता है। यह समस्या कृत्रिम बुद्धिमत्ता में सर्च एल्गोरिद्म की शक्ति को परखती है।

भूलभुलैया विन्यास (Maze Configuration):

S . . X . .

. X . X . G

. . . . X .

. X . . . .

X . X X . .

- **S**: प्रारंभिक बिंदु (Start)
- **G**: लक्ष्य बिंदु (Goal)
- **X**: दीवारें (Walls)
- **.**: खुली जगहें (Open Spaces)

उद्देश्य (Objective):

'S' से 'G' तक का सबसे छोटा रास्ता खोजें।

समाधान रणनीति: ब्रेड्थ-फर्स्ट सर्च (Breadth-First Search - BFS)

BFS एक उपयुक्त सर्च एल्गोरिद्म है जो स्तर दर स्तर (level-wise) भूलभुलैया का अन्वेषण करता है।

चरण:

1. **प्रारंभ करें:** S बिंदु से खोज शुरू करें और एक कतार (queue) बनाएँ।
2. **अन्वेषण:**
    - कतार से एक बिंदु निकालें।
    - क्या वह G है? हाँ, तो पथ पुनर्निर्मित करें।

- नहीं, तो उसके सभी वैध पड़ोसी (adjacent open spaces) को कतार में जोड़ें।

3. **पथ पुनर्निर्माण:**
   - प्रत्येक कदम पर पिछले बिंदु (predecessor) को रिकॉर्ड करें।
   - G तक पहुँचने के बाद, पीछे की ओर जाकर पूर्ण रास्ता पुनर्निर्मित करें।

समाधान पथ उदाहरण:

S P P X . .

. X . X . G

. . . . X .

. X . . . .

X . X X . .

Python कोड स्निपेट:

```
from collections import deque
def bfs_maze(maze, start, goal):
    directions = [(-1, 0), (1, 0), (0, -1), (0, 1)]
    queue = deque([start])
    visited = set([start])
    predecessor = {start: None}
    while queue:
        current = queue.popleft()
        if current == goal:
            path = []
            while current:
                path.append(current)
                current = predecessor[current]
            return path[::-1]
        for d in directions:
```

```
            neighbor = (current[0] + d[0], current[1] + d[1])
            if (0 <= neighbor[0] < len(maze) and 0 <= neighbor[1] < len(maze[0]) and
                maze[neighbor[0]][neighbor[1]] != 'X' and neighbor not in visited):
                queue.append(neighbor)
                visited.add(neighbor)
                predecessor[neighbor] = current
    return None
maze = [
    ['S', '.', '.', 'X', '.', '.'],
    ['.', 'X', '.', 'X', '.', 'G'],
    ['.', '.', '.', '.', 'X', '.'],
    ['.', 'X', '.', '.', '.', '.'],
    ['X', '.', 'X', 'X', '.', '.']
]
start = (0, 0)
goal = (1, 5)
solution_path = bfs_maze(maze, start, goal)
print("Solution Path:", solution_path)
```

2.3.4 रणनीतिक खेल समस्याएँ (Strategic Game Problems)

♟ **शतरंज (Chess) और गो (Go):**

ये दो-खिलाड़ियों वाले रणनीतिक खेल हैं जिनके नियम सरल हैं, लेकिन रणनीति बहुत गहराई लिए होती है।

- **Deep Blue** (IBM द्वारा विकसित) ने शतरंज में विश्व चैंपियन को हराया।
- **AlphaGo** (Google DeepMind द्वारा विकसित) ने Go में इतिहास रचा।

विशेषताएँ:

- विशाल संभावित चालें

- पूर्वानुमानात्मक एल्गोरिद्म
- मूल्यांकन कार्य (evaluation function)
- मशीन लर्निंग का उपयोग

पोकर (Poker):

पोकर एक अपूर्ण जानकारी (imperfect information) वाला खेल है, जिसमें ब्लफिंग और संभाव्य तर्क (probabilistic reasoning) आवश्यक होता है।

AI को ऐसे निर्णय लेने होते हैं जहाँ सभी जानकारी उपलब्ध नहीं होती।

2.3.5 वास्तविक जीवन की समस्याएँ (Real-world Problems)

🏭 **स्वचालित योजना और अनुसूची (Automated Planning and Scheduling):**

उद्योग, लॉजिस्टिक्स, और अंतरिक्ष अभियानों में AI का उपयोग योजनाओं को बनाने और कार्यों को कुशलता से समयबद्ध करने के लिए किया जाता है।

- संसाधन अनुकूलन (resource optimization)
- समय और लागत नियंत्रण
- बाधा (constraints) प्रबंधन

चिकित्सा निदान (Medical Diagnosis):

AI सिस्टम लक्षणों, परीक्षणों, और चिकित्सा इतिहास के आधार पर रोगों की पहचान करने के लिए प्रशिक्षित होते हैं।

- विशेषज्ञ प्रणाली (Expert Systems)
- मशीन लर्निंग के मॉडल
- निर्णय समर्थन प्रणाली (Decision Support Systems)

निष्कर्ष (Conclusion):

ये सभी उदाहरण दर्शाते हैं कि AI कितने विविध क्षेत्रों में काम करता है —

सरल पहेलियों से लेकर **उन्नत रणनीतिक खेलों** और **वास्तविक जीवन की जटिल समस्याओं** तक। हर समस्या के लिए एक उपयुक्त समाधान रणनीति और एल्गोरिद्म की आवश्यकता होती है। AI की इन चुनौतियों से जूझने की क्षमता ही इसे आधुनिक तकनीकी विकास का स्तंभ बनाती है।

2.4 समाधान की खोज (Searching for Solutions)

कृत्रिम बुद्धिमत्ता (AI) में, समाधान की खोज का तात्पर्य संभावित अवस्थाओं (states) या विन्यासों (configurations) की एक विशाल जगह में सही मार्ग तलाशने से है, जो प्रारंभिक अवस्था से लक्ष्य अवस्था तक पहुँचाता है। यह प्रक्रिया उन जटिल समस्याओं को हल करने की मूलभूत विधि है, जहाँ समाधान सीधे स्पष्ट नहीं होता।

AI में **सर्च एल्गोरिद्म** का उपयोग समस्याओं की जगह (problem space) को व्यवस्थित रूप से खोजने और उन समाधानों की पहचान के लिए किया जाता है, जो दिए गए प्रतिबंधों (constraints) और उद्देश्यों (objectives) को सर्वोत्तम रूप से संतुष्ट करते हैं।

2.4.1 खोज एल्गोरिद्म के प्रकार (Types of Search Algorithms)

बिना जानकारी वाली खोज (Uninformed Search):

- जिसे ब्लाइंड सर्च (Blind Search) भी कहा जाता है।
- इसमें लक्ष्य की स्थिति या उस तक पहुँचने की लागत की जानकारी नहीं होती।
- ये एल्गोरिद्म समस्या स्थान को क्रमबद्ध ढंग से खोजते हैं बिना किसी अतिरिक्त जानकारी के।
- प्रमुख उदाहरण:
    - **ब्रेड्थ-फर्स्ट सर्च (Breadth-First Search)**
    - **डेप्थ-फर्स्ट सर्च (Depth-First Search)**
    - **यूनिफॉर्म कॉस्ट सर्च (Uniform Cost Search)**

सूचित खोज (Informed Search):

- इसे **हीयुरिस्टिक सर्च (Heuristic Search)** भी कहते हैं।
- इसमें अतिरिक्त जानकारी (heuristics) का उपयोग किया जाता है, जो खोज को अधिक कुशल बनाता है।
- ये एल्गोरिद्म ऐसे रास्तों को प्राथमिकता देते हैं, जो लक्ष्य तक जल्दी पहुँच सकते हैं।
- प्रमुख उदाहरण:
    - *A सर्च एल्गोरिद्म (A-star Search)**
    - **ग्रीडी बेस्ट-फर्स्ट सर्च (Greedy Best-First Search)**

2.4.2 खोज स्थान का अन्वेषण (Exploring the Search Space)

खोज स्थान (Search Space) वह सैद्धांतिक संरचना है जिसमें सभी संभावित अवस्थाएँ (states) या विन्यास (configurations) मौजूद होते हैं।

एक कुशल सर्च एल्गोरिद्म का उद्देश्य इस स्थान का व्यवस्थित ढंग से अन्वेषण कर **प्रारंभिक अवस्था से लक्ष्य अवस्था तक पहुँचने वाला मार्ग** खोजना होता है।

🔑 **प्रभावशीलता** और **कुशलता (Efficiency)** इस बात पर निर्भर करती है कि:

- खोज स्थान कितना जटिल है
- कौन-सी रणनीति अपनाई गई है

2.4.3 मूल्यांकन कार्य (Evaluation Functions)

सूचित खोज (Informed Search) के संदर्भ में, मूल्यांकन कार्य (Evaluation Function) महत्वपूर्ण भूमिका निभाते हैं।

ये यह तय करने में मदद करते हैं कि सर्च स्पेस में किस नोड (node) या पथ को अगला विस्तारित किया जाए।

एक सामान्य मूल्यांकन फ़ंक्शन इस प्रकार कार्य करता है:

$f(n)=g(n)+h(n)f(n) = g(n) + h(n)f(n)=g(n)+h(n)$

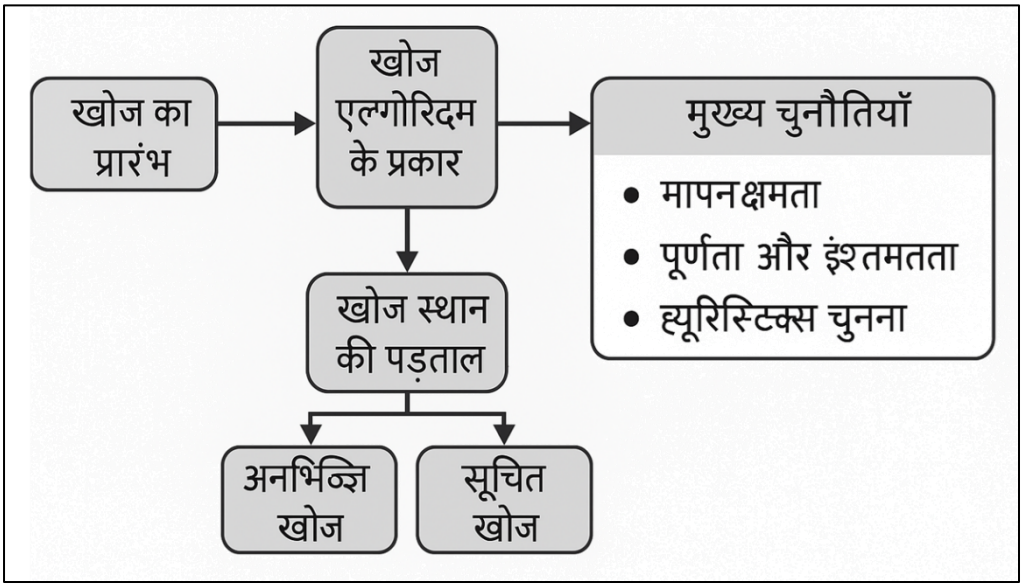
- **g(n)**: प्रारंभिक नोड से वर्तमान नोड तक पहुँचने की वास्तविक लागत
- **h(n)**: वर्तमान नोड से लक्ष्य तक पहुँचने की अनुमानित लागत (हीयुरिस्टिक)

इस कार्य के माध्यम से एल्गोरिद्म यह तय करता है कि कौन-सा नोड आगे अन्वेषण के लिए सबसे उपयुक्त है।

यदि आप चाहें तो मैं इस अनुभाग का **आरेख** या **फ्लोचार्ट** भी बना सकता हूँ — जिसमें Uninformed और Informed Search, Search Space, तथा Evaluation Functions को सरल हिंदी में दर्शाया जाए।

यह आरेख कृत्रिम बुद्धिमत्ता (AI) में समाधान खोजने की प्रक्रिया को दर्शाता है, जिसमें खोज प्रयासों की शुरुआत से लेकर विभिन्न प्रकार की खोज एल्गोरिदम — जैसे कि अनभिज्ञ (Uninformed) और सूचित (Informed) खोज — के उपयोग, खोज स्थान की पड़ताल, और रास्ते में आने वाली प्रमुख चुनौतियों

जैसे मापनक्षमता (Scalability), पूर्णता (Completeness), इष्टता (Optimality) और प्रभावी ह्यूरिस्टिक्स के चयन को दर्शाया गया है।



चित्र 2.4.1: समाधान खोजने की प्रक्रिया का आरेख

2.4.4 अनुमानी विधियाँ (Heuristics)

अनुमानी विधियाँ (Heuristics) ऐसे नियम या विधियाँ होती हैं जो खोज स्थान (Search Space) में किसी दिए गए नोड से लक्ष्य तक पहुँचने की लागत का अनुमान लगाने में मदद करती हैं।
एक अच्छी अनुमानी विधि किसी खोज एल्गोरिद्म की दक्षता को काफी हद तक बढ़ा सकती है, क्योंकि इससे उन नोड्स की संख्या घट जाती है जिनकी जांच करनी पड़ती है।

❗ हालांकि, प्रभावी ह्यूरिस्टिक बनाना एक चुनौतीपूर्ण कार्य है और यह पूरी तरह उस विशेष समस्या पर निर्भर करता है जिसे हल किया जा रहा है।

2.4.5 अनुकूलन और प्रतिबंध (Optimization and Constraints)

कई **AI समस्याएँ** केवल कोई भी समाधान खोजने तक सीमित नहीं होतीं, बल्कि किसी विशेष **मानदंड (Criteria)** के अनुसार **सबसे अच्छा समाधान (Best Solution)** ढूँढने की आवश्यकता होती है।

साथ ही, समाधान को **कुछ सीमाओं (Constraints)** का पालन भी करना होता है।

इस तरह की समस्याओं को हल करने के लिए सर्च एल्गोरिद्म में इस प्रकार के **तंत्र (Mechanisms)** शामिल किए जाते हैं जो विभिन्न समाधानों की तुलना कर सकें और एक **उद्देश्य फ़ंक्शन (Objective Function)** के आधार पर सबसे उपयुक्त समाधान चुन सकें।

2.4.6 समाधान खोजने में चुनौतियाँ (Challenges in Searching for Solutions)

- **स्केलेबिलिटी (Scalability):**

जैसे-जैसे समस्या का आकार और जटिलता बढ़ती है, सर्च स्पेस भी तेजी से विशाल होता जाता है। इससे एल्गोरिद्म के लिए सीमित समय में समाधान ढूँढना कठिन हो सकता है।

- **पूर्णता और इष्टता (Completeness and Optimality):**

कई खोज रणनीतियों में यह कोई गारंटी नहीं होती कि एल्गोरिद्म समाधान ढूँढ ही लेगा (**पूर्णता**) या फिर सबसे उत्तम समाधान ही मिलेगा (**इष्टता**)।

- **अनुमानी विधियों का चयन (Choosing Heuristics):**

सटीक अनुमानी विधि बनाना जटिल कार्य है। एक **उपयुक्त ह्यूरिस्टिक**:

- **Admissible** होनी चाहिए (यानी लक्ष्य तक पहुँचने की लागत को कभी ज़्यादा नहीं आँकना चाहिए)
- **Consistent** होनी चाहिए (यानी नोड्स के बीच स्थानांतरण की वास्तविक लागत को ध्यान में रखना चाहिए)

निष्कर्ष

समाधान की खोज AI की आधारशिला है। यह पज़ल-सुलझाने, योजना बनाने और जटिल निर्णय लेने जैसी समस्याओं के समाधान में अहम भूमिका निभाती है।

- सही **खोज एल्गोरिद्म** का चयन,
- अच्छी तरह से डिज़ाइन की गई **ह्यूरिस्टिक**,

AI सिस्टम को जटिल समस्याओं को **प्रभावी और कुशल** रूप से हल करने योग्य बनाते हैं।

2.5 बिना सूचना आधारित खोज रणनीतियाँ (Uninformed Search Strategies)

बिना सूचना आधारित खोज रणनीतियाँ, जिन्हें अक्सर **ब्लाइंड सर्च (Blind Search)** कहा जाता है, कृत्रिम बुद्धिमत्ता (AI) के क्षेत्र में उन समस्याओं के समाधान हेतु अत्यंत महत्वपूर्ण हैं जहाँ लक्ष्य तक पहुँचने का मार्ग पहले से स्पष्ट नहीं होता।

इन एल्गोरिद्म की विशेषता यह है कि ये खोज की प्रक्रिया के दौरान **किसी भी प्रकार की अनुमानी जानकारी (Heuristic Information)** या **लक्ष्य की स्थिति** के बारे में अतिरिक्त जानकारी पर निर्भर नहीं होते। ये केवल निम्नलिखित सूचनाओं पर आधारित होते हैं:

- प्रारंभिक अवस्था (Initial State)
- उपलब्ध क्रियाएँ (Actions)
- लक्ष्य स्थिति की परिभाषा (Goal State)

2.5.1 विशेषताएँ और उपयोग (Characteristics and Application)

- **सुनियोजित अन्वेषण (Systematic Exploration):**

बिना सूचना वाली खोज रणनीतियाँ समस्या के संभावित हलों की व्यवस्थित रूप से जांच करती हैं। वे एक निश्चित अनुक्रम में उपलब्ध अवस्थाओं का मूल्यांकन करती हैं, जिससे कोई भी संभावित समाधान छूटता नहीं है।

2.5.2 बहु-प्रयोगशीलता (Versatility):

इन रणनीतियों की ताकत उनकी **सामान्य प्रकृति (Generality)** में है। इन्हें विभिन्न प्रकार की समस्याओं जैसे:

- मार्ग खोज (Pathfinding)
- पज़ल समाधान (Puzzle Solving)
- नियोजन और अनुसूची (Planning & Scheduling)

में उपयोग किया जा सकता है, और इसके लिए किसी डोमेन-विशिष्ट ज्ञान की आवश्यकता नहीं होती।

2.5.3 समाधान खोज (Solution Discovery):

इन रणनीतियों का मुख्य उद्देश्य एक ऐसा पथ खोजना है जो प्रारंभिक स्थिति से लक्ष्य स्थिति तक ले जाए। यह कार्य वे क्रियाओं के परिणामों की पूरी तरह जांच करके करते हैं।

2.5.4 समस्याएँ जहाँ इनका प्रयोग उपयोगी होता है (Problem-Solving Scenarios):

यह रणनीतियाँ विशेष रूप से उपयोगी हैं जब:

- लक्ष्य की स्थिति के बारे में कोई अतिरिक्त जानकारी उपलब्ध न हो।
- लागत (Cost) का मूल्यांकन संभव न हो।

ऐसे में ये एल्गोरिद्म सुनिश्चित करते हैं कि अगर कोई समाधान मौजूद है, तो उसे खोजा जा सके।

2.5.5 सीमाएँ (Limitations):

❗ हालाँकि ये रणनीतियाँ व्यापक रूप से उपयोग की जा सकती हैं, लेकिन इनमें कुछ कमियाँ होती हैं:

- **गाइडेंस का अभाव (Lack of Guidance):**
  बिना अनुमानी जानकारी के यह एल्गोरिद्म ज़रूरत से ज्यादा अवस्थाएँ खोज सकते हैं।
- **प्रदर्शन में कमी (Performance Lag):**
  ये एल्गोरिद्म अक्सर समय और संसाधनों की अत्यधिक खपत करते हैं।
- **महंगे पथ का चयन (Cost Inefficiency):**
  ये रणनीतियाँ उन पथों को प्राथमिकता नहीं देतीं जो लक्ष्य तक जल्दी पहुँच सकते हैं या कम लागत वाले हों।

निष्कर्ष:

बिना सूचना आधारित खोज रणनीतियाँ, AI समस्या समाधान के उपकरणों में एक **आधारशिला** मानी जाती हैं।

ये न केवल एक मजबूत प्रारंभिक ढाँचा प्रदान करती हैं, बल्कि अधिक परिष्कृत **सूचित खोज रणनीतियों (Informed Search Strategies)** के लिए **बेसलाइन** के रूप में कार्य करती हैं।

2.6 ब्रेड्थ-फर्स्ट सर्च (BFS)

ब्रेड्थ-फर्स्ट सर्च (BFS) एक महत्वपूर्ण **बिना सूचना आधारित खोज रणनीति (uninformed search strategy)** है, जो विशेष रूप से **ग्राफ ट्रैवर्सल** और **खोज एल्गोरिद्म** के संदर्भ में उपयोग की जाती है। यह एक साधारण लेकिन प्रभावशाली सिद्धांत पर कार्य करती है: वर्तमान गहराई के सभी पड़ोसी नोड्स की खोज पहले की जाती है, उसके बाद अगली गहराई के नोड्स की। यह क्रमिक (लेवल-बाय-लेवल) खोज **सबसे छोटा मार्ग (shortest path)** खोजने के लिए विशेष रूप से उपयोगी है, विशेषकर बिना वेट वाले ग्राफ्स में।

2.6.1 मुख्य तंत्र (Core Mechanism)

BFS एक **queue (कतार)** का उपयोग करता है ताकि उन नोड्स को ट्रैक किया जा सके जिन्हें अभी खोजा जाना है। प्रारंभिक नोड से शुरू करके, यह सभी आस-पास के नोड्स की जांच करता है, उन्हें कतार में जोड़ता है, और उन्हें "visited" के रूप में चिह्नित करता है ताकि दोबारा न जाया जाए। यह प्रक्रिया तब तक दोहराई जाती है जब तक या तो लक्ष्य न मिल जाए या कतार खाली हो जाए।

2.6.2 विशेषताएँ और गुण (Characteristics and Features)

- **पूर्णता (Completeness):** यदि समाधान मौजूद है, तो BFS निश्चित रूप से उसे खोज लेगा।
- **उत्कृष्टता (Optimality):** यदि सभी चालों की लागत समान है, तो BFS सबसे छोटा मार्ग देगा।
- **समय जटिलता (Time Complexity):** O(b^d), जहाँ b ब्रांचिंग फैक्टर और d सबसे छिछली गहराई है जहाँ लक्ष्य स्थित है।
- **स्थान जटिलता (Space Complexity):** O(b^d), क्योंकि यह सभी नोड्स को एक गहराई स्तर पर स्टोर करता है।

2.6.3 अनुप्रयोग (Applications)

BFS का उपयोग कई क्षेत्रों में किया जाता है:

- **सबसे छोटा मार्ग (Shortest Path)**
- **पज़ल गेम्स (Puzzle Games)**
- **नेटवर्क विश्लेषण (Network Analysis)**

- **AI में पथ खोज (Pathfinding in AI)**

2.6.4 BFS एल्गोरिद्म (BFS Algorithm)

1. प्रारंभ करें: एक खाली queue बनाएं और प्रारंभिक नोड को उसमें डालें, तथा उसे visited चिह्नित करें।
2. लूप करें: जब तक queue खाली न हो:
    - queue से अगला नोड निकालें।
    - यदि वह लक्ष्य नोड है, तो सफलता लौटाएं।
    - अन्यथा, उसके सभी पड़ोसी नोड्स को queue में डालें, यदि वे पहले visit नहीं किए गए हों।
3. यदि लक्ष्य नहीं मिला, तो विफलता लौटाएं।

2.6.5 BFS का पायथन प्रोग्राम (Python Program for BFS)

python

CopyEdit

```
def bfs(graph, start, goal):
    queue = [(start, [start])]
    visited = set()
    while queue:
        (current_node, path) = queue.pop(0)
        if current_node not in visited:
            visited.add(current_node)
            if current_node == goal:
                return path
            for neighbor in graph[current_node]:
                queue.append((neighbor, path + [neighbor]))
    return None
```

```
graph = {
    'A': ['B', 'C'],
    'B': ['D', 'E'],
    'C': ['F', 'G'],
    'D': [],
    'E': ['H'],
    'F': [],
    'G': ['H'],
    'H': []
}
path = bfs(graph, 'A', 'H')
print("Path from A to H:", path)
```

यह प्रोग्राम A से H तक का सबसे छोटा मार्ग खोजता है।

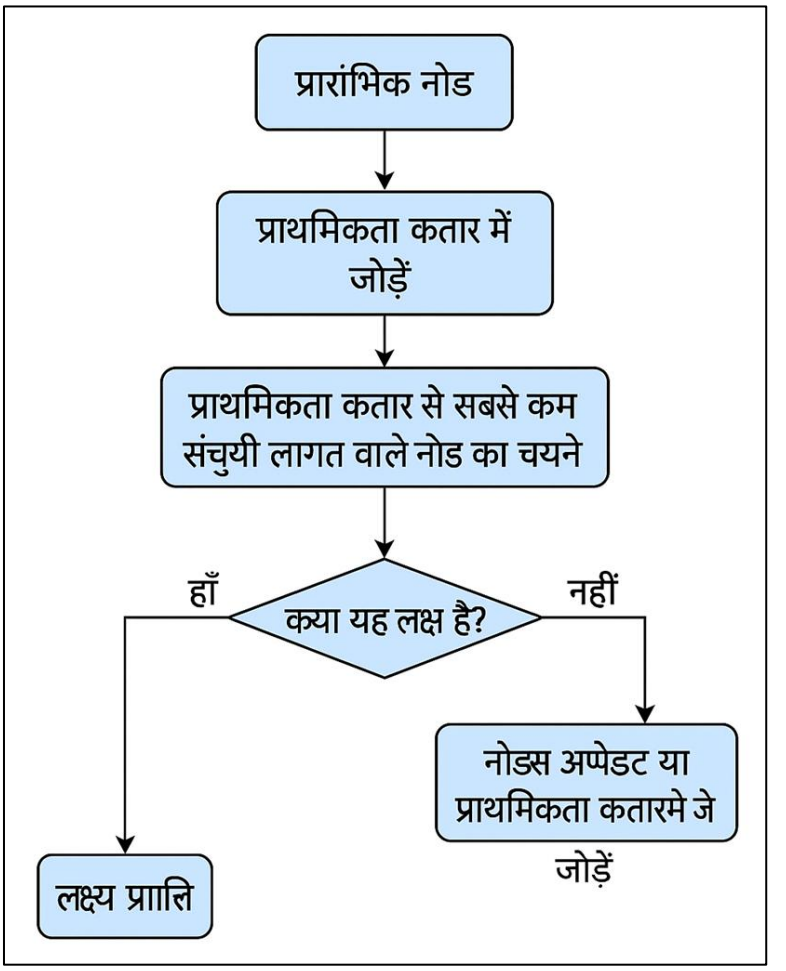
2.6.6 सीमाएँ (Limitations)

- **स्मृति की अधिक आवश्यकता:** एक गहराई पर सभी नोड्स को संग्रहित करना पड़ता है।
- **बड़े ग्राफ के लिए अनुपयुक्त:** बहुत बड़े ग्राफ्स में BFS बहुत अधिक स्मृति का उपयोग कर सकता है।
- **गहराई-प्रबल समस्याओं के लिए अक्षम:** जहाँ लक्ष्य बहुत गहराई में स्थित हो, वहाँ यह समय और संसाधन दोनों की खपत करता है।

निष्कर्ष:

BFS एक बुनियादी लेकिन अत्यंत शक्तिशाली खोज तकनीक है। इसकी **सरलता**, **पूर्णता**, और **लघुत्तम पथ की गारंटी** इसे AI और कंप्यूटर विज्ञान में उपयोगी बनाती है।

2.7 यूनिफॉर्म-कॉस्ट सर्च (Uniform-Cost Search)



चित्र 2.7.1: यह ग्राफ आरेख यूनिफॉर्म-कॉस्ट सर्च (UCS) एल्गोरिद्म की प्रक्रिया को दर्शाता है — प्रारंभिक नोड से शुरू करते हुए, प्राथमिकता कतार में सबसे कम संचयी लागत वाले नोड का विस्तार करना, लक्ष्य की प्राप्ति की जाँच करना, और लागत के आधार पर नोड्स को अपडेट या जोड़ना।

2.7 यूनिफॉर्म-कॉस्ट सर्च का परिचय

यूनिफॉर्म-कॉस्ट सर्च (UCS), जिसे **सस्ती-पहले खोज (Cheapest First Search)** भी कहा जाता है, एक **बिना सूचना (uninformed)** आधारित खोज विधि है, जो उस स्थिति में बहुत प्रभावी होती है

जहाँ पथ की लागत समान नहीं होती। UCS का उद्देश्य यह है कि वह सबसे कम लागत वाले पथ को पहले खोजे, बजाय केवल चरणों या नोड्स की संख्या पर आधारित चयन के।

2.7.1 मुख्य सिद्धांत (Core Principle)

UCS का मूल सिद्धांत है कि वह हमेशा उस नोड का विस्तार करता है जिसकी **संचयी लागत (cumulative path cost)** सबसे कम होती है। जब लक्ष्य नोड का विस्तार किया जाता है, तब यह सुनिश्चित करता है कि उस तक पहुँचने का मार्ग सबसे कम लागत वाला है। इसलिए, UCS को **इष्टतम (optimal)** खोज रणनीति माना जाता है।

2.7.2 एल्गोरिद्म (Algorithm)

1. एक **प्राथमिकता कतार (priority queue)** बनाएं और उसमें प्रारंभिक नोड को डालें। प्राथमिकता, पथ लागत पर आधारित होती है।
2. जब तक प्राथमिकता कतार खाली न हो:
    - सबसे कम लागत वाले नोड को हटाएं।
    - यदि वह लक्ष्य नोड है, तो पथ और लागत लौटाएं।
    - अन्यथा, उसके सभी उत्तराधिकारियों (successors) को विस्तार करें, उनकी संचयी लागत की गणना करें, और प्राथमिकता कतार में अपडेट करें।
3. यदि लक्ष्य नहीं मिला और कतार खाली हो गई, तो "विफलता" लौटाएं।

2.7.3 UCS का पायथन प्रोग्राम (Python Program for UCS)

```
import heapq
def uniform_cost_search(graph, start, goal):
    frontier = [(0, start, [start])]
    visited = set()
    while frontier:
        cost, node, path = heapq.heappop(frontier)
        if node == goal:
            return path, cost
        if node not in visited:
```

```
            visited.add(node)
            for neighbor, neighbor_cost in graph[node]:
                if neighbor not in visited:
                    heapq.heappush(frontier, (cost + neighbor_cost, neighbor, path +
[neighbor]))
    return "Goal not reachable", float("inf")
graph = {
    'A': [('B', 1), ('C', 3)],
    'B': [('D', 3), ('E', 1)],
    'C': [('D', 1), ('F', 5)],
    'D': [('G', 2)],
    'E': [('G', 4)],
    'F': [],
    'G': []
}
path, cost = uniform_cost_search(graph, 'A', 'G')
print("Least cost path:", path)
print("Total cost:", cost)
```

मुख्य बिंदु (Key Points)

- ग्राफ को **डिक्शनरी (dictionary)** के रूप में दर्शाया गया है जहाँ प्रत्येक कुंजी (node) के साथ उसके पड़ोसी और लागत की सूची है।
- heapq मॉड्यूल का उपयोग **प्राथमिकता कतार (priority queue)** के लिए किया गया है।
- कतार का प्रत्येक तत्व तीन भागों में होता है: **कुल लागत**, **वर्तमान नोड**, और **अब तक का मार्ग**।
- एल्गोरिद्म हर बार उस नोड को विस्तार करता है जिसकी संचयी लागत सबसे कम हो।
- जब लक्ष्य नोड को हटाया जाता है, तब वह इष्टतम पथ लौटाता है।

- यदि लक्ष्य पहुँचा नहीं जा सकता, तो वह संदेश और अनंत लागत (inf) लौटाता है।

2.7.4 विशेषताएँ (Characteristics)

- **इष्टतमता (Optimality)**: यदि लागत फलन (cost function) में कोई नकारात्मक लागत नहीं है, तो UCS हमेशा लक्ष्य तक पहुँचने वाला **सबसे कम लागत वाला पथ** खोजने की गारंटी देता है।
- **सम्पूर्णता (Completeness)**: यदि कोई समाधान मौजूद है, तो UCS उसे अवश्य खोज लेगा, क्योंकि यह तब तक नहीं रुकता जब तक कि लक्ष्य प्राप्त न हो जाए या संपूर्ण खोज स्थान समाप्त न हो जाए।
- **समय और स्थान जटिलता (Time and Space Complexity)**: इसकी जटिलता समस्या की संरचना पर अत्यधिक निर्भर करती है। सबसे खराब स्थिति में, UCS को सभी संभावित पथों की जांच करनी पड़ सकती है, जो बड़े खोज स्थानों के लिए अव्यवहारिक हो सकता है।

2.7.5 अनुप्रयोग (Applications)

UCS विशेष रूप से उन परिस्थितियों में उपयोगी होता है जहाँ **लक्ष्य तक पहुँचने की लागत में काफी अंतर** होता है और सबसे सस्ते पथ की आवश्यकता होती है। इसके सामान्य उपयोग निम्नलिखित हैं:

- **रूटिंग एल्गोरिद्म (Routing Algorithms)**: सड़कों के नेटवर्क या नेटवर्क नोड्स के बीच सबसे छोटा या सबसे तेज़ मार्ग खोजने के लिए।
- **अनुसूची और योजना (Scheduling and Planning)**: किसी लक्ष्य को प्राप्त करने के लिए **सबसे लागत-प्रभावी क्रियाओं के अनुक्रम** को निर्धारित करने में।

2.7.6 चुनौतियाँ (Challenges)

UCS की सबसे बड़ी चुनौती है **संसाधनों की अधिक आवश्यकता**। यह सभी खोजे गए नोड्स को प्राथमिकता कतार (priority queue) में संग्रहीत करता है और लगातार उनके पथ और लागत को अपडेट करता है, जिससे यह **स्मृति (memory)** की दृष्टि से भारी हो सकता है।
इसके अतिरिक्त, यदि लक्ष्य प्रारंभिक नोड से बहुत दूर है या बहुत सारे संभावित पथ हैं, तो UCS की **विस्तृत प्रकृति** के कारण यह **धीमा** हो सकता है।

2.7.7 उदाहरण (Example)

कल्पना करें कि UCS का उपयोग किसी शहर के सड़क नेटवर्क का प्रतिनिधित्व करने वाले **वज़नदार ग्राफ** (weighted graph) में **सबसे कम लागत वाला मार्ग** खोजने के लिए किया जा रहा है। इस ग्राफ में:

- नोड्स (nodes) सड़क चौराहों को दर्शाते हैं
- किनारे (edges) सड़कों को दर्शाते हैं और लागत के रूप में दूरी दी गई होती है।

UCS प्रारंभिक चौराहे से पथों का क्रमबद्ध रूप से अन्वेषण करता है — **उन पथों का पहले विस्तार करता है जिनकी कुल दूरी कम होती है**, जिससे यह सुनिश्चित करता है कि किसी भी चौराहे तक पहुँचने का सबसे छोटा मार्ग पहले खोजा जाए।

UCS की **इष्टतमता और सम्पूर्णता** इसे AI और रोबोटिक्स में वज़नदार ग्राफों की समस्याओं को हल करने का एक **मूल्यवान उपकरण** बनाती है। हालांकि, इसका व्यवहारिक उपयोग उपलब्ध **गणनात्मक संसाधनों** और कार्य की **विशिष्ट आवश्यकताओं** के अनुसार संतुलित किया जाना चाहिए।

2.8 गहराई-प्रथम खोज (Depth-first search - DFS)

आकृति 2.9.1: गहनता-सीमित खोज की प्रक्रिया

गहराई-प्रथम खोज (DFS) कृत्रिम बुद्धिमत्ता (AI) में एक मौलिक एल्गोरिद्म है जिसका उपयोग ग्राफ़ या वृक्ष (tree) को स्रोत नोड से शुरू करते हुए शाखाओं के सबसे अंतिम नोड तक खोजने में किया जाता है, उसके बाद पीछे हटकर (backtracking) अन्य शाखाओं का अन्वेषण किया जाता है। यह रणनीति उन समस्याओं के लिए विशेष रूप से उपयोगी है जिनमें समाधान प्राप्त करने के लिए सभी संभावित मार्गों का परीक्षण आवश्यक होता है। DFS को पुनरावृत्ति (recursion) या स्टैक (stack) के माध्यम से लागू किया जा सकता है।

2.8.1 एल्गोरिद्म का वर्णन

DFS एल्गोरिद्म जड़ नोड (root node) से शुरू होता है (ग्राफ़ के मामले में कोई भी नोड चुना जा सकता है) और प्रत्येक शाखा के अंत तक जितना संभव हो अन्वेषण करता है, उसके बाद पीछे हटकर अन्य विकल्पों की जाँच करता है।

चरण-दर-चरण प्रक्रिया:

1. वर्तमान नोड को विज़िट किया गया चिह्नित करें और उसे प्रिंट करें या समाधान पथ के रूप में रिकॉर्ड करें।
2. प्रत्येक आसन्न (adjacent) नोड के लिए, यदि वह पहले विज़िट नहीं किया गया है, तो DFS एल्गोरिद्म को पुनरावर्ती रूप से उस नोड के साथ कॉल करें।

2.8.2 विशेषताएँ

- **पूर्णता (Completeness):** DFS पूर्ण नहीं है; यदि वृक्ष अनंत गहराई वाला हो, तो यह समाधान खोजने में असफल हो सकता है।
- **समय जटिलता (Time Complexity):** O(V + E), जहाँ V = नोड्स की संख्या और E = किनारों (edges) की संख्या (adjacency list वाले ग्राफ़ के लिए)।
- **स्थान जटिलता (Space Complexity):** O(V), क्योंकि इसमें स्टैक और विज़िटेड नोड्स को संग्रहीत करना पड़ता है।

2.8.3 अनुप्रयोग (Applications)

DFS का उपयोग विभिन्न परिदृश्यों में किया जाता है, जैसे:

- **पहेली और गेम समाधान:** प्रारंभिक स्थिति से सभी संभावित अवस्थाओं का अन्वेषण करके समाधान ढूंढना।
- **टोपोलॉजिकल क्रमबद्धता:** निर्देशित ग्राफ़ में नोड्स को उनके पदानुक्रम के अनुसार क्रम में लाने के लिए।
- **चक्रीयता का पता लगाना:** ग्राफ़ में सायकल (cycle) का पता लगाने के लिए, जैसे कि डेडलॉक डिटेक्शन में।

2.8.4 पायथन में कार्यान्वयन

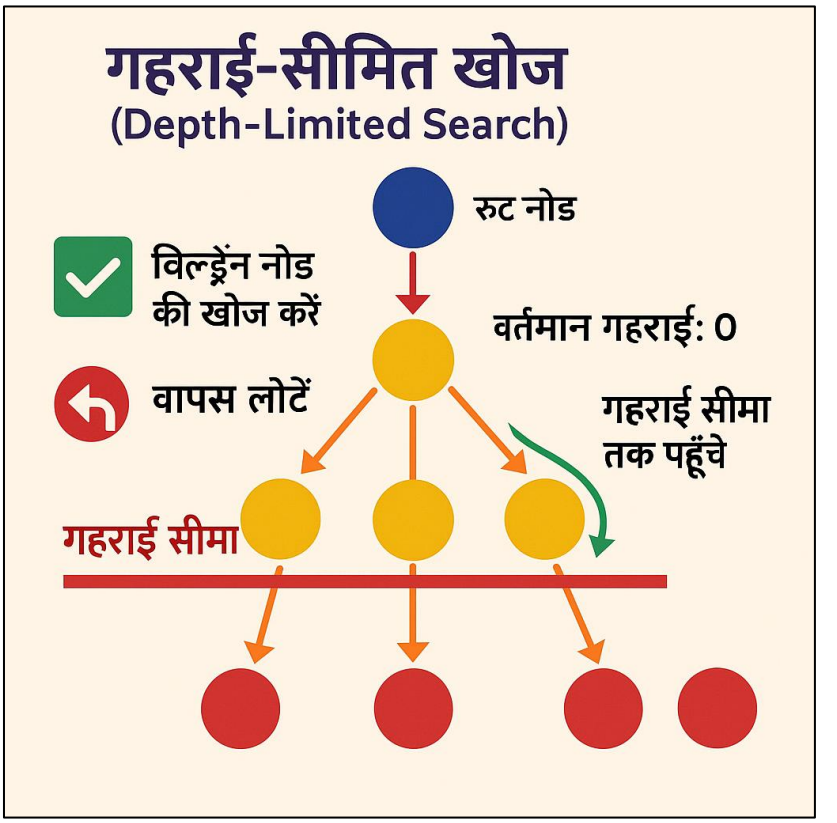
```
def dfs(graph, node, visited):
    if node not in visited:
        print(node, end=' ')
        visited.add(node)
        for neighbour in graph[node]:
            dfs(graph, neighbour, visited)
# उदाहरण के लिए ग्राफ़
graph = {
    'A': ['B', 'C'],
    'B': ['D', 'E'],
```

```
    'C': ['F'],
    'D': [],
    'E': ['F'],
    'F': []
}
visited = set()  # विज़िटेड नोड्स को ट्रैक करने के लिए सेट
dfs(graph, 'A', visited)
```

इस उदाहरण में, ग्राफ़ को एक डिक्शनरी के रूप में दर्शाया गया है जहाँ प्रत्येक कुंजी एक नोड है और उसका मान उन नोड्स की सूची है जो उससे जुड़े हैं। dfs फ़ंक्शन ग्राफ़, वर्तमान नोड, और विज़िटेड नोड्स का सेट लेता है और प्रत्येक शाखा के अंत तक अन्वेषण करता है।

DFS की सरलता और ग्राफ़ या वृक्ष में गहराई तक जाने की क्षमता इसे AI में उपयोगी बनाती है, विशेषकर उन समस्याओं में जहाँ सभी संभावनाओं का पूर्ण अन्वेषण आवश्यक होता है।

2.9 गहराई-सीमित खोज (Depth-limited search)



चित्र 2.9.1: यह आरेख डैप्थ-लिमिटेड सर्च (DLS) की प्रक्रिया को दर्शाता है, जिसमें निर्धारित गहराई सीमा तक नोड्स का अन्वेषण किया जाता है और आवश्यकता पड़ने पर बैकट्रैकिंग की जाती है, ताकि अनावश्यक गहराई में जाने से बचा जा सके।

गहराई-सीमित खोज (Depth-limited Search - DLS) पारंपरिक गहराई-प्रथम खोज (Depth-first Search - DFS) एल्गोरिद्म का एक विस्तार है, जिसे बहुत गहराई तक या अनंत पथों में जाने से रोकने के लिए डिज़ाइन किया गया है। DLS खोज की गहराई पर एक सीमा लगाकर DFS की मुख्य समस्या को हल करता है, जिसमें एजेंट बिना समाधान पाए बहुत गहराई तक चला जाता है। यह DLS को उन खोज स्थानों के लिए उपयोगी बनाता है जहाँ गहराई बहुत अधिक या असीमित हो सकती है।

2.9.1 एल्गोरिद्म का विवरण

DLS एल्गोरिद्म DFS की तरह ही कार्य करता है, लेकिन इसमें एक अतिरिक्त पैरामीटर होता है: **गहराई सीमा (Depth Limit)**। यह सीमा यह निर्धारित करती है कि रूट नोड से कितने स्तर तक खोज की जा सकती है।

कार्यप्रणाली:

1. रूट नोड से प्रारंभ करें और वर्तमान गहराई को 0 पर सेट करें।
2. प्रत्येक चाइल्ड नोड का अन्वेषण करें और गहराई को +1 करें।
3. यदि गहराई सीमा तक पहुँच गए हैं, तो उस शाखा की खोज रोक दें।
4. जब शाखा पूरी हो जाए या सीमा आ जाए, तो बैकट्रैक करें।
5. तब तक दोहराएँ जब तक समाधान मिल न जाए या सभी संभावित नोड्स को खंगाल न लिया जाए।

2.9.2 विशेषताएँ

- **पूर्णता (Completeness):** यह पूर्ण नहीं है। यदि समाधान गहराई सीमा से नीचे है, तो वह नहीं मिलेगा।
- **समय जटिलता (Time Complexity):** O(bl), जहाँ b शाखा कारक (branching factor) और l गहराई सीमा है।
- **स्थान जटिलता (Space Complexity):** O(bl), क्योंकि इसे स्टैक में नोड्स को गहराई सीमा तक संग्रहीत करना होता है।

2.9.3 अनुप्रयोग (Applications)

DLS उन समस्याओं के लिए उपयोगी है जहाँ:

- समाधान एक सीमित गहराई में है,
- या जहाँ अनंत लूप्स या साइकल्स मौजूद हो सकते हैं।

उदाहरण:

- पहेली सुलझाना (Puzzle solving)
- अनंत पथों से बचाव (Avoiding infinite paths)

2.9.4 Python में कार्यान्वयन

```
def depth_limited_search(graph, start, goal, limit):
    stack = [(start, 0)]
    while stack:
        (node, depth) = stack.pop()
        if depth > limit:
            continue
        if node == goal:
            return True
        for neighbor in graph.get(node, []):
            stack.append((neighbor, depth + 1))
    return False
# उदाहरण ग्राफ
graph = {
    'A': ['B', 'C', 'D'],
    'B': ['E'],
    'C': ['F', 'G'],
    'D': ['H'],
    'E': [], 'F': [], 'G': [], 'H': []
}
limit = 3
print(depth_limited_search(graph, 'A', 'H', limit))  # True
```

इस कोड में, ग्राफ को एक डिक्शनरी के रूप में दर्शाया गया है और एल्गोरिद्म तब तक खोज करता है जब तक लक्ष्य नोड पाया न जाए या गहराई सीमा पार न कर ली जाए।

2.9.5 सीमाएँ (Limitations)

- सही गहराई सीमा निर्धारित करना कठिन होता है।
- यदि सीमा बहुत कम है, तो समाधान नहीं मिलेगा।
- बहुत अधिक होने पर यह DFS जैसा व्यवहार करता है और उसके नुकसान झेलता है (जैसे लूप्स में फँसना)।

2.10 इटरेटिव डीपनिंग डेप्थ-फ़र्स्ट सर्च (Iterative Deepening Depth-First Search – IDDFS)

IDDFS गहराई-प्रथम खोज (DFS) की स्मृति-कुशलता और ब्रेड्थ-फ़र्स्ट खोज (BFS) की सम्पूर्णता/इष्टता को एक साथ जोड़ता है।

यह रूट से शुरू होकर बढ़ती हुई *गहराई सीमा* (depth limit) पर बार-बार **गहराई-सीमित खोज** (DLS) चलाता है। बड़े या अनंत खोज-स्थान, तथा अज्ञात लक्ष्य-गहराई वाले परिदृश्यों में यह विधि अत्यन्त उपयोगी है।

आकृति 2.10.1: इटरेटिव डीपनिंग डेप्थ-फ़र्स्ट सर्च (IDDFS) प्रक्रिया को दर्शाने वाला आरेख

2.10.1 एल्गोरिद्म विवरण

1. गहराई सीमा 0 (या 1) से आरम्भ करें।
2. उसी सीमा तक DLS चलाएँ।
3. यदि लक्ष्य नहीं मिला, सीमा +1 करें।
4. लक्ष्य मिलने तक (या सम्पूर्ण खोज-स्थान समाप्त होने तक) दोहराएँ।

2.10.2 विशेषताएँ

गुण	विवरण
सम्पूर्णता	हाँ — गहराई सीमा क्रमशः बढ़ने से समाधान अवश्य मिलेगा।
इष्टता	समान-लागत वातावरण में लक्ष्य की सबसे कम गहराई खोजेगा।
समय जटिलता	$O(b^d)$ (b = branching factor, d = लक्ष्य गहराई) — पुनः-भ्रमण का ओवरहेड नगण्य।
स्थान जटिलता	$O(b^d)$ (DFS-जैसी) — एक ही पथ व उसके अविज़िट सिब्लिंग नोड्स स्टोर करता है।

2.10.3 अनुप्रयोग

- **पहेलियाँ / समस्या-समाधान** — जहाँ समाधान-गहराई ज्ञात नहीं।
- **गेम AI** — चालों की गहराई अज्ञात होने पर गेम ट्री का कुशल अन्वेषण।

2.10.4 Python कार्यान्वयन (सरल रूप)

```
def iterative_deepening_dfs(graph, start, goal):
    depth = 0
    while True:
        result = depth_limited_search(graph, start, goal, depth)
        if result is not None:
            return result          # लक्ष्य पथ मिला
        depth += 1                 # गहराई सीमा बढ़ाएँ
def depth_limited_search(graph, node, goal, limit, path=None, depth=0):
    if path is None:               # प्रारंभिक पथ
        path = []
    if depth > limit:              # सीमा पार, वापस लौटें
        return None
    if node == goal:               # लक्ष्य मिला
        return path + [node]
    for neighbor in graph.get(node, []):
```

```
        if neighbor not in path:  # साइकल से बचाव
            res = depth_limited_search(graph, neighbor, goal,
                                       limit, path + [node], depth + 1)
            if res is not None:
                return res
    return None                   # इस शाखा में लक्ष्य नहीं
# उदाहरण ग्राफ़ (Adjacency-List)
graph = {
    'A': ['B', 'C', 'D'],
    'B': ['E'],
    'C': ['F', 'G'],
    'D': ['H'],
    'E': [],
    'F': [],
    'G': [],
    'H': []
}
print(iterative_deepening_dfs(graph, 'A', 'H'))
```

यह प्रोग्राम आयरन-स्ट्रेटेजी IDDFS द्वारा 'A' से 'H' तक का पथ खोजता है,

बढ़ती हुई गहराई (0, 1, 2, ...) पर बार-बार DLS चलाकर।

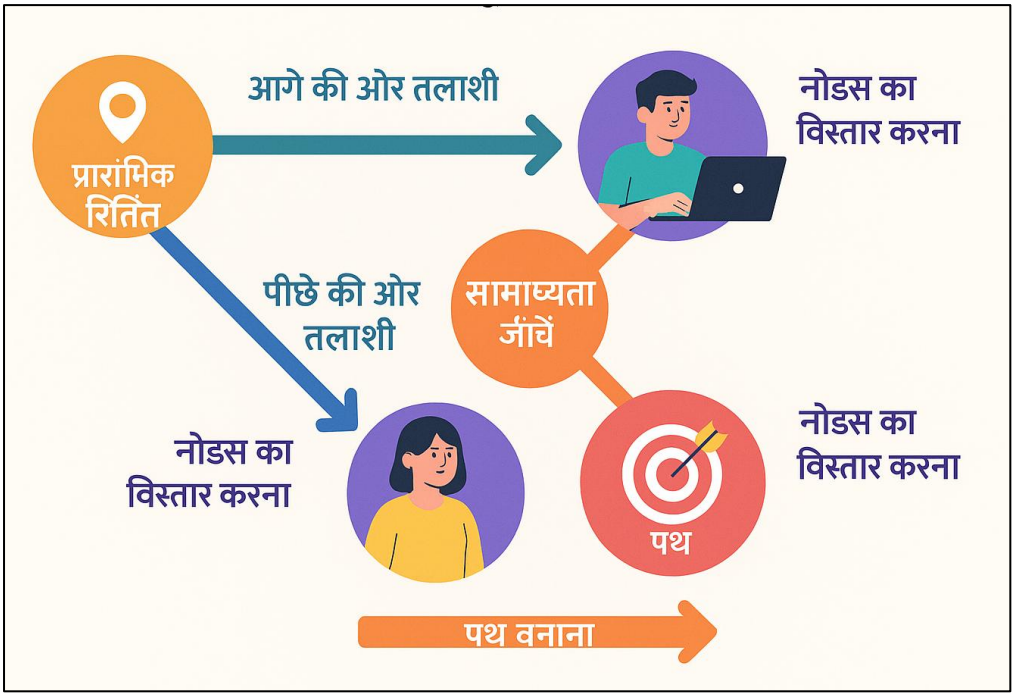
2.10.5 सीमाएँ

- **उचित गहराई सीमा का अनुमान** कठिन; बहुत कम होने पर समाधान छूट सकता है।
- सीमा बहुत अधिक होने पर यह साधारण DFS जैसा ही व्यवहार करेगा, जिसकेव्यय (साइकल, अनावश्यक गहराई) उसी तरह होंगे।

IDDFS की **स्मृति-कुशलता + सम्पूर्णता/इष्टता** का संयोजन इसे AI खोज-एल्गोरिद्म के शस्त्रागार में अत्यन्त मूल्यवान बनाता है।

2.11 द्विदिशीय खोज (Bidirectional Search)

द्विदिशीय खोज कृत्रिम बुद्धिमत्ता (AI) में प्रयुक्त एक कुशल खोज रणनीति है, जिसका उद्देश्य प्रारंभिक स्थिति से लक्ष्य स्थिति तक का सबसे छोटा मार्ग खोजना होता है। यह एक साथ दो दिशाओं में खोज करता है—एक प्रारंभिक स्थिति से लक्ष्य की ओर और दूसरी लक्ष्य से प्रारंभिक स्थिति की ओर—जब तक कि दोनों खोजें किसी सामान्य नोड पर नहीं मिल जातीं। यह विधि विशेष रूप से सघन ग्राफों में समाधान खोजने के समय और स्थान दोनों को काफी हद तक कम कर सकती है।



चित्र 2.11.1: द्विदिश खोज प्रक्रिया को दर्शाने वाला आरेख

यह आरेख द्विदिश खोज (Bidirectional Search) की प्रक्रिया को दर्शाता है, जिसमें प्रारंभिक स्थिति से आगे की ओर और लक्ष्य स्थिति से पीछे की ओर एक साथ खोज की जाती है। जैसे ही दोनों दिशाओं से की गई खोज एक सामान्य नोड (मिलन बिंदु) पर मिलती हैं, समस्या का समाधान प्राप्त हो जाता है। यह विधि खोज स्थान को महत्वपूर्ण रूप से घटाकर समाधान तक पहुँचने का समय कम करती है, विशेष रूप से जब ग्राफ बहुत घना हो।

2.11.1 एल्गोरिद्म विवरण (Algorithm Description)

द्विदिशीय खोज में दो ब्रेड्थ-फर्स्ट सर्च (BFS) एक साथ चलाए जाते हैं—एक प्रारंभिक स्थिति से लक्ष्य की ओर और दूसरी लक्ष्य से प्रारंभिक स्थिति की ओर। जब दोनों खोजें एक सामान्य नोड पर मिलती हैं, तो इसका अर्थ है कि समाधान मिल गया है।

प्रमुख चरण:

1. प्रारंभिक स्थिति और लक्ष्य स्थिति से दो खोज फ्रंट प्रारंभ करें।
2. दोनों फ्रंट से बारी-बारी से नोड्स का विस्तार करें और उनके आस-पास के नोड्स जोड़ें।
3. जांचें कि क्या दोनों फ्रंट किसी सामान्य नोड पर मिले हैं। यदि हाँ, तो समाधान मिल गया है।
4. उस सामान्य नोड के माध्यम से दोनों दिशाओं के पथ को जोड़कर पूर्ण पथ बनाएँ।

2.11.2 प्रोग्राम (Program)

नीचे दिया गया **Python प्रोग्राम** एक द्विदिशीय खोज को प्रदर्शित करता है जो एक **undirected graph** पर काम करता है। यह दोनों दिशाओं में BFS चलाता है और यदि दोनों खोजें किसी साझा नोड पर मिलती हैं, तो समाधान का पूरा पथ निकालता है।

```
from collections import deque
def bidirectional_search(graph, start, goal):
    if start == goal:
        return [start]
    frontier_start = deque([(start, [start])])
    frontier_goal = deque([(goal, [goal])])
    explored_start = set([start])
    explored_goal = set([goal])
    while frontier_start and frontier_goal:
        if frontier_start:
            current_node, path = frontier_start.popleft()
            for neighbor in graph[current_node]:
                if neighbor not in explored_start:
```

```
                if neighbor in explored_goal:
                    return path + bidirectional_path(frontier_goal, neighbor)[::-1]
                explored_start.add(neighbor)
                frontier_start.append((neighbor, path + [neighbor]))
        if frontier_goal:
            current_node, path = frontier_goal.popleft()
            for neighbor in graph[current_node]:
                if neighbor not in explored_goal:
                    if neighbor in explored_start:
                        return bidirectional_path(frontier_start, neighbor) + path[::-1]
                    explored_goal.add(neighbor)
                    frontier_goal.append((neighbor, path + [neighbor]))
    return []
def bidirectional_path(frontier, meeting_point):
    for node, path in frontier:
        if node == meeting_point:
            return path
    return []
# उदाहरण ग्राफ (Example usage)
graph = {
    'A': ['B', 'C'],
    'B': ['A', 'D', 'E'],
    'C': ['A', 'F'],
    'D': ['B'],
    'E': ['B', 'F', 'G'],
    'F': ['C', 'E'],
```

```
    'G': ['E']
}
start = 'A'
goal = 'G'
path = bidirectional_search(graph, start, goal)
print(f"Path from {start} to {goal}: {path}")
```

महत्वपूर्ण बातें (Key Points)

- यह प्रोग्राम एक **adjacency list** के रूप में ग्राफ को दर्शाता है।
- दो फ्रंटियर्स का उपयोग किया गया है—एक प्रारंभिक नोड से और दूसरी लक्ष्य से।
- यदि कोई सामान्य नोड दोनों दिशाओं में पाया जाता है, तो दोनों दिशाओं के पथ को जोड़कर समाधान निकाला जाता है।
- यह कार्य केवल **undirected graphs** में सीधे लागू किया जा सकता है।

2.11.3 विशेषताएँ (Characteristics)

- **पूर्णता (Completeness):** द्विदिशीय खोज पूर्ण होती है यदि खोज स्थान सीमित (finite) है।
- **उत्तमत्व (Optimality):** यह खोज विधि तभी उत्तम परिणाम देती है जब दोनों दिशाओं में ब्रेड्थ-फर्स्ट सर्च का उपयोग किया जाए।
- **समय जटिलता (Time Complexity):** यह एकल-दिशा खोज (unidirectional search) की तुलना में तेज़ होती है। सामान्यतः इसकी समय जटिलता **O(b^{d/2})** होती है, जहाँ **b** शाखा गुणांक (branching factor) और **d** प्रारंभ और लक्ष्य के बीच की गहराई है।
- **स्थान जटिलता (Space Complexity):** समय जटिलता की तरह, स्थान जटिलता भी कम होती है। इसमें आमतौर पर कम मेमोरी की आवश्यकता होती है।

2.11.4 अनुप्रयोग (Applications)

द्विदिशीय खोज निम्नलिखित परिदृश्यों में विशेष रूप से उपयोगी है:

- **मार्ग खोज (Route Finding):** नक्शे में दो स्थानों के बीच सबसे छोटा रास्ता खोजने के लिए।

- **पहेली समाधान (Puzzle Solving):** जहाँ प्रारंभिक और लक्ष्य स्थिति ज्ञात हो, वहाँ समाधान ढूँढने के लिए।
- **ग्राफ विश्लेषण (Graph Analysis):** सामाजिक नेटवर्क या वेब लिंक संरचनाओं जैसे घने ग्राफों में कनेक्शन का विश्लेषण करने के लिए।

2.11.5 कार्यान्वयन संबंधी विचार (Implementation Considerations)

- द्विदिशीय खोज को लागू करना चुनौतीपूर्ण हो सकता है क्योंकि इसमें दो खोज सीमाओं (search fronts) का प्रबंधन करना पड़ता है और उनके मिलान की कुशलता से जांच करनी होती है।
- यदि कार्यों की लागतें अलग-अलग हैं या लक्ष्य स्थिति स्पष्ट नहीं है, तो इस खोज विधि को अनुकूल बनाने के लिए अतिरिक्त रणनीतियों (जैसे कि heuristic मार्गदर्शन) की आवश्यकता हो सकती है।

2.11.6 सीमाएँ (Limitations)

- यद्यपि द्विदिशीय खोज एकल-दिशा खोज की तुलना में काफी तेज़ हो सकती है, इसकी **प्रभावशीलता समस्या की संरचना** पर निर्भर करती है।
- जब लक्ष्य स्थिति अस्पष्ट हो या कई संभावित लक्ष्य हों, तो पीछे की दिशा से खोज शुरू करना कठिन हो सकता है।
- दो खोज सीमाओं को बनाए रखना और उनका मिलान करना संसाधनों पर अतिरिक्त दबाव डाल सकता है, जिससे समग्र दक्षता पर असर पड़ सकता है।

निष्कर्ष: द्विदिशीय खोज का मुख्य लाभ यह है कि यह खोज स्थान को काफी हद तक कम कर सकती है। यह तब विशेष रूप से प्रभावी होती है जब प्रारंभिक और लक्ष्य स्थितियाँ स्पष्ट रूप से परिभाषित हों और उनके बीच का मार्ग घना या जटिल हो।

2.12 ग्रीडी बेस्ट-फर्स्ट सर्च (Greedy Best-First Search)

ग्रीडी बेस्ट-फर्स्ट सर्च (GBFS) कृत्रिम बुद्धिमत्ता (AI) में उपयोग किया जाने वाला एक खोज एल्गोरिद्म है जो किसी समस्या स्थान (problem space) में लक्ष्य स्थिति (goal state) तक पहुँचने के लिए सबसे निकटतम प्रतीत होने वाले पथ का चयन करता है। यह एल्गोरिद्म एक ह्यूरिस्टिक (heuristic) फ़ंक्शन का उपयोग करता है, जो किसी नोड से लक्ष्य तक की अनुमानित लागत का आकलन करता है, और उसी के आधार पर खोज को निर्देशित करता है।

2.12.1 एल्गोरिद्म विवरण (Algorithm Description)

ग्रीडी बेस्ट-फर्स्ट सर्च निम्नलिखित चरणों में कार्य करता है:

1. एक प्राथमिकता कतार (priority queue) में प्रारंभिक स्थिति (initial state) को जोड़ें। प्राथमिकता ह्यूरिस्टिक फ़ंक्शन द्वारा निर्धारित की जाती है।
2. जब तक लक्ष्य नहीं मिल जाता या कतार खाली नहीं हो जाती:
    - सबसे कम ह्यूरिस्टिक लागत वाले नोड को निकालें।
    - उसके उत्तराधिकारियों (successors) को प्राथमिकता कतार में जोड़ें। उनकी प्राथमिकता उनके ह्यूरिस्टिक मान के आधार पर तय होती है।
3. यदि लक्ष्य मिल गया है, तो समाधान पथ लौटाएँ; अन्यथा विफलता रिपोर्ट करें।

2.12.2 कार्यक्रम (Python प्रोग्राम)

```
import heapq
def greedy_best_first_search(graph, start, goal, heuristic):
    frontier = [(heuristic[start], start)]
    visited = set()
    parent = {start: None}
    while frontier:
        current_cost, current_node = heapq.heappop(frontier)
        if current_node == goal:
            return reconstruct_path(parent, start, goal)
```

```
        visited.add(current_node)
        for neighbor, _ in graph[current_node]:
            if neighbor not in visited:
                heapq.heappush(frontier, (heuristic[neighbor], neighbor))
                visited.add(neighbor)
                parent[neighbor] = current_node
    return []
def reconstruct_path(parent, start, goal):
    path = [goal]
    while path[-1] != start:
        path.append(parent[path[-1]])
    path.reverse()
    return path
graph = {
    'A': [('B', 2), ('C', 3)],
    'B': [('D', 1), ('E', 4)],
    'C': [('F', 5), ('G', 6)],
    'D': [],
    'E': [('H', 1)],
    'F': [],
    'G': [('H', 1)],
    'H': []
}
heuristic = {
    'A': 4, 'B': 2, 'C': 4, 'D': 4.5, 'E': 2,
```

```
    'F': 3.5, 'G': 1, 'H': 0
}
start = 'A'
goal = 'H'
path = greedy_best_first_search(graph, start, goal, heuristic)
print(f"Path from {start} to {goal}: {path}")
```

मुख्य विशेषताएँ (Key Characteristics):

- **प्राथमिकता:** GBFS हमेशा उस नोड को पहले चुनता है जो ह्यूरिस्टिक के अनुसार लक्ष्य के सबसे करीब लगता है।
- **तेज़ खोज:** यह एल्गोरिद्म अनावश्यक नोड्स को टालने की कोशिश करता है।
- **ह्यूरिस्टिक पर निर्भरता:** एल्गोरिद्म की सफलता पूरी तरह उस ह्यूरिस्टिक फ़ंक्शन की गुणवत्ता पर निर्भर करती है।

नोट:

यदि ह्यूरिस्टिक फ़ंक्शन गलत या अप्रभावी है, तो GBFS गलत दिशा में भी खोज कर सकता है। इसलिए, इसे सावधानी से उपयोग किया जाना चाहिए, खासकर तब जब आपको समाधान की गारंटी और कुशलता दोनों की आवश्यकता हो।

2.12.3 विशेषताएँ (Characteristics):

- **ह्यूरिस्टिक-आधारित (Heuristic-Driven):**
  ग्रीडी बेस्ट-फर्स्ट सर्च (GBFS) की दक्षता इस बात पर निर्भर करती है कि उपयोग की गई ह्यूरिस्टिक फंक्शन कितनी प्रभावी है। एक अच्छी ह्यूरिस्टिक खोज को तेज़ी से लक्ष्य तक पहुँचा सकती है।
- **गैर-इष्टतम (Non-Optimal):**
  यह एल्गोरिद्म संचयी लागत (cumulative cost) पर ध्यान नहीं देता, इसलिए यह हमेशा सबसे सस्ती या सबसे अच्छी राह नहीं खोजता।
- **अपूर्णता (Incomplete):**
  यदि ह्यूरिस्टिक फंक्शन किसी चक्र (loop) की ओर मार्गदर्शन कर दे, तो GBFS लक्ष्य तक

नहीं पहुँच सकता। यदि नोड्स को दोहराने से रोकने का तंत्र न हो, तो यह एल्गोरिद्म अपूर्ण रह सकता है।

2.12.4 अनुप्रयोग (Applications):

GBFS उन समस्याओं के लिए उपयुक्त है जहाँ एक सटीक समाधान आवश्यक नहीं है, बल्कि समाधान शीघ्रता से प्राप्त करना प्राथमिकता है:

- **मानचित्रों में मार्ग ढूँढना (Pathfinding in Maps):**
  जैसे कि दो स्थानों के बीच पहुँचने हेतु सीधी दूरी जैसे ह्यूरिस्टिक का उपयोग।
- **पज़ल गेम्स (Puzzle Games):**
  लक्ष्य स्थिति की ओर बढ़ने के लिए निकटता पर आधारित ह्यूरिस्टिक का प्रयोग।
- **खेलों में AI (AI in Games):**
  ऐसे निर्णय लेना जो जीत की दिशा में अधिक लाभकारी प्रतीत होते हैं।

2.12.5 कार्यान्वयन के विचार (Implementation Considerations):

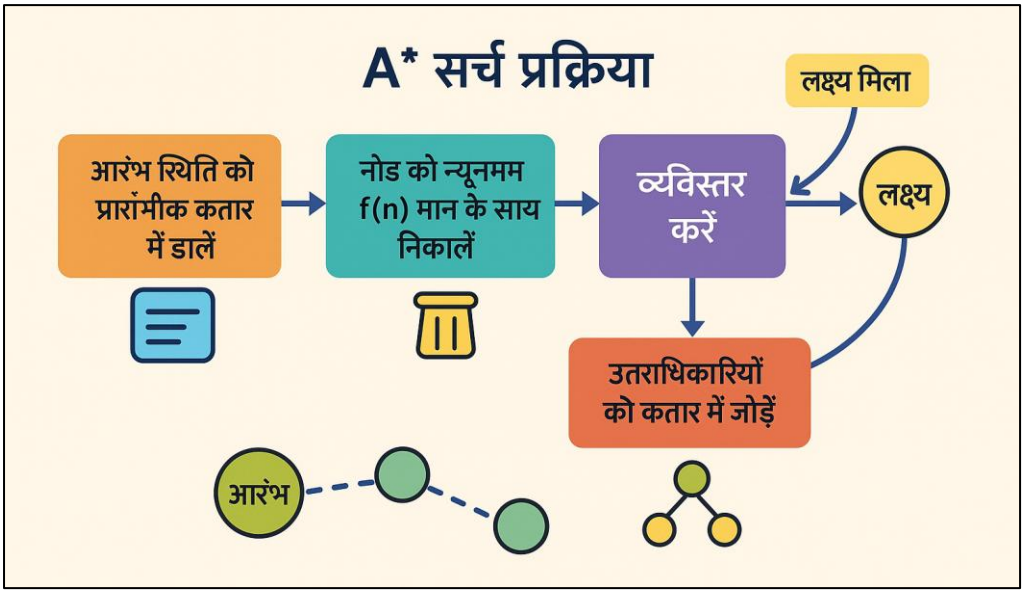
- एक उपयुक्त **ह्यूरिस्टिक फ़ंक्शन** का चयन बहुत महत्वपूर्ण है।
- यदि ह्यूरिस्टिक admissible हो (यानी यह लक्ष्य तक पहुँचने की लागत को कभी अधिक नहीं आंकता), तो खोज अधिक कुशल हो सकती है।
- GBFS गति को प्राथमिकता देता है, इसलिए यह पथ लागत की तुलना में लक्ष्य की निकटता को अधिक महत्व देता है।

2.12.6 सीमाएँ (Limitations):

- यदि ह्यूरिस्टिक विशेष समस्या के लिए उपयुक्त नहीं है, तो GBFS उस दिशा में खोज कर सकता है जो दिखने में अच्छी हो लेकिन वास्तव में अनुपयुक्त या महँगी हो।
- बिना विज़िटेड नोड्स को ट्रैक किए, एल्गोरिद्म चक्रों में फँस सकता है।
- यह एल्गोरिद्म अनुकूलित समाधान की गारंटी नहीं देता, इसलिए इसे केवल तभी प्रयोग करें जब तेजी से समाधान अपेक्षित हो, न कि सर्वोत्तम समाधान।

2.13 A सर्च (A-Star खोज)*

A* खोज एक शक्तिशाली और व्यापक रूप से प्रयुक्त एल्गोरिद्म है जो कृत्रिम बुद्धिमत्ता (AI) और रोबोटिक्स में मार्ग खोजने (pathfinding) और ग्राफ ट्रैवर्सल के लिए प्रयोग होता है। यह Greedy Best-First Search की ह्यूरिस्टिक रणनीति और Uniform-Cost Search की न्यूनतम लागत खोजने की रणनीति को जोड़कर कार्य करता है। A* एल्गोरिद्म कुशलतापूर्वक प्रारंभिक स्थिति से लक्ष्य तक सबसे कम लागत वाला मार्ग खोजता है।



चित्र 2.13.1: आरेख जो A* खोज प्रक्रिया को दर्शाता है।

2.13.1 एल्गोरिद्म विवरण (Algorithm Description)

A* सर्च प्रत्येक नोड के लिए निम्नलिखित लागत फ़ंक्शन का उपयोग करता है:

$f(n)=g(n)+h(n)f(n) = g(n) + h(n)f(n)=g(n)+h(n)$

जहाँ:

- **g(n)**: स्टार्ट नोड से नोड nnn तक की वास्तविक लागत।
- **h(n)**: नोड nnn से लक्ष्य तक अनुमानित लागत (heuristic)।

A* उस नोड को प्राथमिकता देता है जिसका f(n)f(n)f(n) सबसे कम होता है। इसके चरण निम्नलिखित हैं:

1. एक प्राथमिकता कतार (priority queue) प्रारंभ करें जिसमें स्टार्ट नोड हो, जिसकी प्राथमिकता f(n)f(n)f(n) हो।
2. जब तक प्राथमिकता कतार खाली न हो:
    - सबसे कम f(n)f(n)f(n) वाले नोड को हटाएँ।
    - यदि यह लक्ष्य है, तो पथ लौटाएँ।
    - अन्यथा, इसके सभी पड़ोसी नोड्स के लिए g(n)g(n)g(n) और f(n)f(n)f(n) की गणना करें, और उन्हें कतार में जोड़ें।
3. यदि लक्ष्य नहीं मिला, तो "पथ नहीं मिला" लौटाएँ।

2.13.2 प्रोग्राम (Python Code)

```
import heapq

def a_star_search(graph, start, goal, h):
    open_set = [(h[start], start, [start])]
    g_scores = {start: 0}
    while open_set:
        _, current, path = heapq.heappop(open_set)
        if current == goal:
            return path
        for neighbor, cost in graph[current]:
            tentative_g_score = g_scores[current] + cost
            if neighbor not in g_scores or tentative_g_score < g_scores[neighbor]:
                g_scores[neighbor] = tentative_g_score
                f_score = tentative_g_score + h[neighbor]
                heapq.heappush(open_set, (f_score, neighbor, path + [neighbor]))
    return "Path not found"
```

```
# ग्राफ और ह्यूरिस्टिक
graph = {
    'A': [('B', 1), ('E', 1)],
    'B': [('A', 1), ('C', 1), ('F', 1)],
    'C': [('B', 1), ('G', 1)],
    'D': [('E', 1), ('I', 1)],
    'E': [('A', 1), ('D', 1), ('F', 1), ('J', 1)],
    'F': [('B', 1), ('E', 1), ('G', 1), ('K', 1)],
    'G': [('C', 1), ('F', 1), ('L', 1)],
    'H': [('I', 1), ('M', 1)],
    'I': [('D', 1), ('H', 1), ('J', 1), ('N', 1)],
    'J': [('E', 1), ('I', 1), ('K', 1), ('O', 1)],
    'K': [('F', 1), ('J', 1), ('L', 1), ('P', 1)],
    'L': [('G', 1), ('K', 1)],
    'M': [('H', 1), ('N', 1)],
    'N': [('I', 1), ('M', 1), ('O', 1)],
    'O': [('J', 1), ('N', 1), ('P', 1)],
    'P': [('K', 1), ('O', 1)]
}
h = {
    'A': 4, 'B': 3, 'C': 2, 'D': 5, 'E': 3, 'F': 2, 'G': 1,
    'H': 6, 'I': 4, 'J': 2, 'K': 1, 'L': 0, 'M': 5, 'N': 3,
    'O': 1, 'P': 1
}
start, goal = 'A', 'L'
path = a_star_search(graph, start, goal, h)
```

```
print(f"Path from {start} to {goal}: {path}")
```

महत्वपूर्ण विशेषताएँ (Key Characteristics)

- **संपूर्णता (Completeness):** यदि समाधान मौजूद है, तो A* उसे खोज लेता है।
- **इष्टतमता (Optimality):** यदि heuristic admissible और consistent है, तो A* सबसे कम लागत वाला पथ देगा।
- **समय जटिलता (Time Complexity):** O(bd)O(b^d)O(bd) — जहाँ bbb है branching factor और ddd है लक्ष्य की गहराई।
- **ह्यूरिस्टिक आधारित मार्गदर्शन:** A* heuristic और actual cost दोनों को संतुलित करता है।

A का प्रयोग कहाँ करें? (Applications of A)**

- **नक्शों में रास्ता ढूँढना (Pathfinding in Maps)**
- **रोबोटिक्स में नेविगेशन**
- **गेम AI (Game Pathfinding)**
- **शेड्यूलिंग और ऑप्टिमाइज़ेशन समस्याएँ**

2.13.3 विशेषताएँ (Characteristics)

• **सम्पूर्णता (Completeness):** A* एक सम्पूर्ण एल्गोरिद्म है; यदि समाधान मौजूद है, तो यह अवश्य उसे खोज लेता है।

• **इष्टतमता (Optimality):** यदि प्रयुक्त heuristic h(n)h(n)h(n) admissible है (अर्थात यह लक्ष्य तक पहुँचने की वास्तविक लागत को कभी अधिक नहीं आंकता), तो A* हमेशा सबसे कम लागत वाला रास्ता खोजता है।

• **कुशलता (Efficiency):** यदि heuristic सटीक है, तो A* खोज समय और मेमोरी के दृष्टिकोण से अत्यधिक कुशल होता है।

2.13.4 अनुप्रयोग (Applications)

A* सर्च का उपयोग कई क्षेत्रों में किया जाता है, जैसे:

• **नेविगेशन और पाथफाइंडिंग (Navigation and Pathfinding):** गेम्स और रोबोटिक्स में दो बिंदुओं के बीच सबसे छोटा रास्ता खोजने के लिए।

• **पहेली हल करना (Puzzle Solving):** AI में 8-पज़ल, रूबिक क्यूब जैसी पहेलियों को हल करने के लिए।

• **नेटवर्क रूटिंग (Network Routing):** टेलीकम्युनिकेशन में नेटवर्क के भीतर सबसे कम दूरी वाला रास्ता खोजने हेतु।

2.13.5 कार्यान्वयन विचार (Implementation Considerations)

A* को प्रभावी ढंग से लागू करने के लिए आवश्यक है:

• **एक उत्कृष्ट heuristic फ़ंक्शन h(n)h(n)h(n):** एल्गोरिद्म की कार्यक्षमता इस पर निर्भर करती है। सामान्यतः प्रयुक्त heuristic में Euclidean distance, Manhattan distance या समस्या-विशिष्ट heuristic शामिल हैं।

• **कुशल डेटा स्ट्रक्चर:** प्राथमिकता कतार (priority queue) का उपयोग किया जाता है ताकि सबसे कम f(n)f(n)f(n) वाले नोड को शीघ्रता से एक्सेस किया जा सके।

2.13.6 सीमाएँ (Limitations)

• **heuristic पर निर्भरता:** यदि heuristic सटीक नहीं है, तो A* गैर-प्रभावी दिशा में बहुत अधिक नोड्स का अन्वेषण कर सकता है।

• **मेमोरी की खपत:** A* को सभी खुले नोड्स (open set) और प्रत्येक नोड तक पहुँचने के पथ को ट्रैक करना पड़ता है, जिससे मेमोरी का उपयोग बहुत अधिक हो सकता है।

निष्कर्ष:

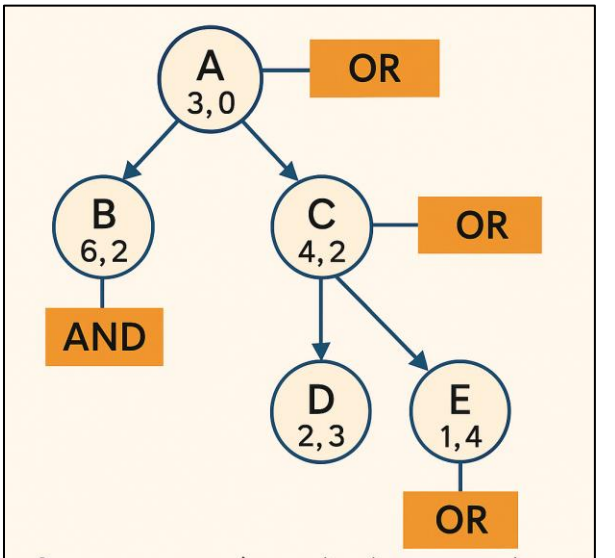
A* सर्च अपनी **सम्पूर्णता**, **इष्टतमता**, और **प्रभावशीलता** के संतुलन के कारण जटिल समस्याओं के लिए अत्यधिक उपयुक्त है, जहाँ सबसे लागत-कुशल मार्ग खोजना आवश्यक होता है।

2.14 AO* खोज: सूचित (हीयूरिस्टिक) खोज रणनीतियाँ

AO* खोज या **And-Or ग्राफ खोज** एक सूचित खोज रणनीति है, जिसका उपयोग कृत्रिम बुद्धिमत्ता (AI) में उन समस्याओं को हल करने के लिए किया जाता है जिन्हें And-Or ग्राफ के रूप में दर्शाया जा सकता है। इन ग्राफों में:

- **नोड्स (Nodes)** किसी स्थिति (state) का प्रतिनिधित्व करते हैं।
- **एजेस (Edges)** कार्यों (actions) को दर्शाते हैं जो एक स्थिति को दूसरी में बदलते हैं।
- **"OR" नोड** – किसी एक बच्चे को हल करना पर्याप्त है।
- **"AND" नोड** – सभी बच्चों को हल करना आवश्यक है।

AO* खोज का उद्देश्य एक समाधान ग्राफ (solution graph) खोजना होता है जो प्रारंभिक स्थिति से लक्ष्य तक न्यूनतम लागत में पहुँचे।



चित्र 2.14.1: AND और OR नोड्स के साथ AO* खेज का

2.14.1 एल्गोरिद्म विवरण (Algorithm Description)

AO* खोज निम्नलिखित चरणों में कार्य करती है:

1. **प्रारंभिक नोड** को वर्तमान नोड के रूप में सेट करें।
2. **हीयूरिस्टिक मान** का मूल्यांकन करें।

3. सबसे **कम लागत** वाले नोड का विस्तार करें:
    - OR नोड के लिए – सबसे कम हीयूरिस्टिक वाले बच्चे को चुनें।
    - AND नोड के लिए – सभी बच्चों को शामिल करें।
4. यह प्रक्रिया तब तक दोहराएं जब तक लक्ष्य नोड समाधान ग्राफ में शामिल न हो जाए और कोई अन्य विस्तार आवश्यक न हो।

2.14.2 Python कार्यक्रम (Program)

```
class Node:
    def __init__(self, name, node_type, children=[], cost=0):
        self.name = name
        self.node_type = node_type  # 'AND' या 'OR'
        self.children = children
        self.cost = cost
        self.heuristic = 0
        self.total_cost = self.cost + self.heuristic
        self.visited = False
def ao_star(graph, start, heuristic):
    open_list = [start]
    while open_list:
        node = open_list.pop(0)
        if node.visited:
            continue
        if node.node_type == 'OR':
            node.total_cost = min(child.total_cost for child in node.children) + node.cost
        else:
            node.total_cost = sum(child.total_cost for child in node.children) + node.cost
        node.heuristic = heuristic[node.name]
```

```
        node.visited = True
        print(f"Visiting Node {node.name} with total cost: {node.total_cost}")
        for child in node.children:
            if not child.visited:
                open_list.append(child)
    return graph[start.name].total_cost
# ग्राफ उदाहरण
graph = {
    'A': Node('A', 'OR', [], 0),
    'B': Node('B', 'AND', [], 2),
    'C': Node('C', 'OR', [], 2),
    'D': Node('D', 'AND', [], 3),
    'E': Node('E', 'OR', [], 4)
}
# बच्चों को परिभाषित करें
graph['A'].children = [graph['B'], graph['C']]
graph['B'].children = [graph['D'], graph['E']]
graph['C'].children = [graph['D']]
graph['D'].children = [graph['E']]
graph['E'].children = []
# हीयूरिस्टिक मान
heuristic = {
    'A': 3,
    'B': 6,
    'C': 4,
    'D': 2,
```

```
    'E': 1
}
for node in graph.values():
    node.heuristic = heuristic[node.name]
# AO* खोज निष्पादन
total_cost = ao_star(graph, graph['A'], heuristic)
print(f"Total cost to reach the goal: {total_cost}")
```

मुख्य बिंदु:

- AO* **हीयूरिस्टिक** गाइडेंस का उपयोग करता है।
- यह **AND** और **OR** दोनों प्रकार के निर्णयों को संभाल सकता है।
- इसका उद्देश्य **कम लागत वाला समाधान ग्राफ** बनाना है।
- इसका उपयोग **डायग्नोस्टिक सिस्टम**, **विकल्प विश्लेषण**, **योजना निर्माण**, आदि क्षेत्रों में होता है।

2.14.2 विशेषताएँ

· हीयूरिस्टिक-आधारित: AO* एल्गोरिथ्म की कार्यक्षमता मुख्य रूप से उस हीयूरिस्टिक फ़ंक्शन पर निर्भर करती है जो किसी नोड से लक्ष्य तक पहुँचने की लागत का अनुमान लगाता है।

· ऑप्टिमलिटी (सर्वोत्तमता): यदि हीयूरिस्टिक फ़ंक्शन एडमिसिबल (admissible) है, तो AO* एक इष्टतम समाधान प्रदान करने की गारंटी देता है।

· दक्षता: AO* जटिल समस्याओं को हल करने में कुशल हो सकता है, क्योंकि यह सबसे संभावित पथों पर पहले ध्यान केंद्रित करता है, हालांकि इसकी दक्षता हीयूरिस्टिक फ़ंक्शन की सटीकता पर निर्भर करती है।

2.14.3 अनुप्रयोग

AO* खोज उन अनुप्रयोगों के लिए उपयुक्त है जहाँ निर्णयों से अनेक संभावित परिणाम निकलते हैं और लक्ष्य प्राप्त करने के लिए कुछ निर्णयों के अंतर्गत सभी स्थितियों की पूर्ति आवश्यक होती है, जैसे:

· योजना और अनुसूची (Planning and Scheduling): जहाँ कार्यों की एक श्रृंखला (AND स्थितियाँ) और विकल्प (OR स्थितियाँ) को हल करना होता है।

· कृत्रिम बुद्धिमत्ता में समस्या समाधान: विशेष रूप से उन क्षेत्रों में जहाँ समस्याओं को उपसमस्याओं में विभाजित किया जा सकता है जिनके बीच परस्पर निर्भरता होती है।

2.14.4 कार्यान्वयन विचार

AO* को लागू करने के लिए आवश्यक हैं:

· एंड-ऑर ग्राफ का एक उपयुक्त प्रतिनिधित्व।

· एक हीयूरिस्टिक मूल्यांकन फ़ंक्शन जो किसी भी नोड से लक्ष्य तक पहुँचने की लागत का अनुमान लगाता है।

· खोज प्रक्रिया के दौरान AND और OR दोनों प्रकार के नोड्स को उपयुक्त रूप से संभालने के लिए तंत्र।

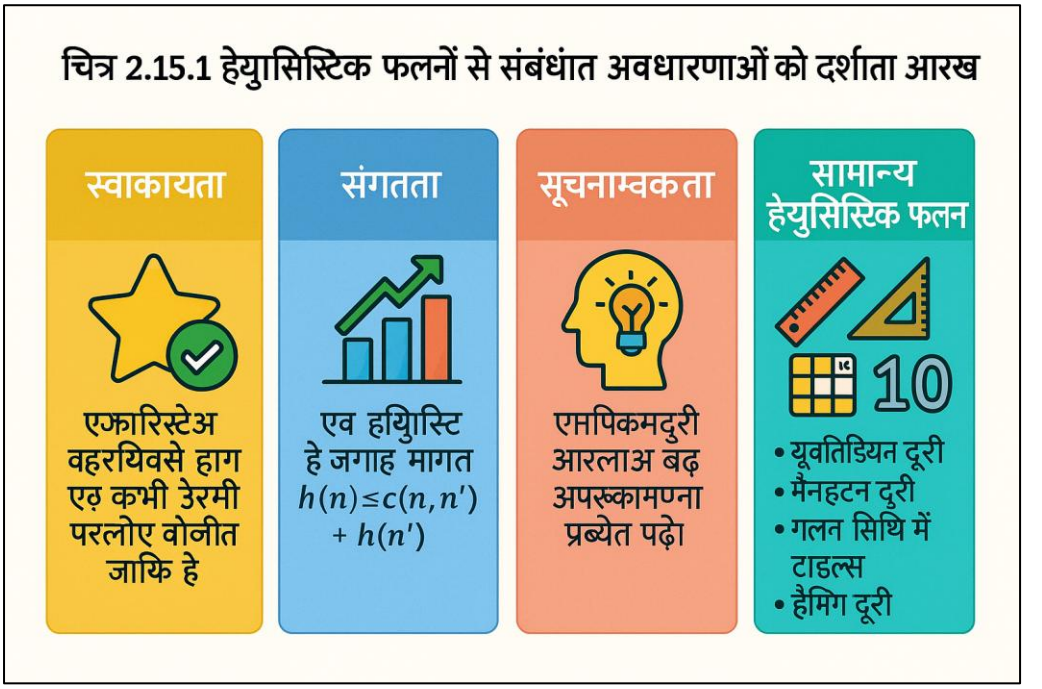
2.14.5 सीमाएँ

AO* खोज की प्रभावशीलता इस बात पर अत्यधिक निर्भर करती है कि हीयूरिस्टिक फ़ंक्शन लागत का कितना सटीक अनुमान लगा पाता है। साथ ही, AO* को लागू करना जटिल हो सकता है क्योंकि इसमें AND और OR दोनों प्रकार के नोड्स को संभालना होता है और खोज के दौरान समाधान ग्राफ को लगातार अपडेट करना होता है।

सारांश रूप में, AO* खोज एंड-ऑर ग्राफ़ द्वारा निरूपित समस्याओं को हल करने के लिए एक सशक्त रूपरेखा प्रदान करता है, जो हीयूरिस्टिक जानकारी का उपयोग कर जटिल निर्णय प्रक्रियाओं को कुशलतापूर्वक नेविगेट करने में मदद करता है।

2.15 हेयुरिस्टिक फलन (Heuristic Functions)

हेयुरिस्टिक फलन कृत्रिम बुद्धिमत्ता (AI) में विशेष रूप से खोज एल्गोरिदम के संदर्भ में एक अत्यंत महत्वपूर्ण घटक हैं। हेयुरिस्टिक फलन, जिसे **h(n)** द्वारा दर्शाया जाता है, किसी दिए गए नोड **n** से लक्ष्य नोड तक न्यूनतम लागत या दूरी का अनुमान प्रदान करता है। इसका उपयोग खोज एल्गोरिदम जैसे **A*** और **Greedy Best-First Search** में सबसे संभावित मार्गों को प्राथमिकता देने के लिए किया जाता है जो लक्ष्य की दिशा में सीधे ले जाते हैं। इन एल्गोरिदम की प्रभावशीलता काफी हद तक प्रयुक्त हेयुरिस्टिक फलन की गुणवत्ता पर निर्भर करती है।



2.15.1 हेयुरिस्टिक फलनों की विशेषताएँ

- **स्वीकार्यता (Admissibility):** यदि कोई हेयुरिस्टिक कभी भी लक्ष्य तक पहुँचने की लागत को अधिक नहीं आंकता (यानी यह आशावादी होता है), तो वह स्वीकार्य कहलाता है। A* खोज में, एक स्वीकार्य हेयुरिस्टिक एक इष्टतम मार्ग सुनिश्चित करता है।

- **संगतता (Consistency या Monotonicity):** यदि हर नोड **n** और उसके उत्तराधिकारी **n'** के लिए, $h(n) \leq c(n, n') + h(n')$ होता है, तो हेयुरिस्टिक संगत है। संगतता यह सुनिश्चित करती है कि एक बार किसी नोड को विस्तारित करने के बाद उसकी इष्टतम लागत नहीं बदलेगी।
- **सूचनात्मकता (Informativeness):** एक अधिक सटीक हेयुरिस्टिक वास्तविक लागत के करीब का अनुमान देता है, जिससे खोज प्रक्रिया कुशल बनती है और कम नोड्स का विस्तार करना पड़ता है।

2.15.2 सामान्य हेयुरिस्टिक फलन

- **यूक्लिडियन दूरी (Euclidean Distance):** जब लक्ष्य दो बिंदुओं के बीच न्यूनतम सीधी दूरी हो, तब प्रयुक्त होता है।
- **मैनहट्टन दूरी (Manhattan Distance):** ग्रिड आधारित समस्याओं के लिए, जहाँ केवल ऊपर, नीचे, बाएँ और दाएँ दिशा में गति संभव हो।
- **गलत स्थिति में टाइल्स (Misplaced Tiles):** पहेली समस्याओं में, यह गिनती करता है कि कितनी टाइल्स अपने लक्ष्य स्थान पर नहीं हैं।
- **हैमिंग दूरी (Hamming Distance):** दो अनुक्रमों के बीच वे स्थितियाँ जहाँ तत्व अलग होते हैं।

2.15.3 हेयुरिस्टिक फलनों की डिज़ाइनिंग

- **समस्या क्षेत्र को समझना:** समस्या की संरचना और बाधाओं का विश्लेषण करना ताकि उपयोगी गुण पहचाने जा सकें।
- **सटीकता बनाम गणना लागत:** अधिक सटीक हेयुरिस्टिक खोज स्थान को घटाता है, लेकिन उसकी गणना की लागत अधिक न हो इसका भी ध्यान रखना चाहिए।
- **व्यावहारिक परीक्षण (Empirical Testing):** विभिन्न हेयुरिस्टिक और पैरामीटर का प्रयोग करके सबसे प्रभावी संयोजन की पहचान करना।

2.15.4 सीमाएँ

- **सटीकता बनाम लागत:** सटीक हेयुरिस्टिक की गणना महंगी हो सकती है, जिससे कुल प्रदर्शन प्रभावित हो सकता है।

- **डोमेन विशेष:** प्रभावी हेयुरिस्टिक आमतौर पर किसी विशेष समस्या या डोमेन के लिए ही उपयुक्त होती हैं।
- **गुमराह करने का जोखिम:** यदि हेयुरिस्टिक खराब ढंग से डिज़ाइन की गई हो, तो यह खोज प्रक्रिया को गलत दिशा में ले जा सकती है।

सारांश:

हेयुरिस्टिक फलन सूचित खोज रणनीतियों में एक आधारशिला की तरह हैं, जो जटिल समस्याओं के स्थान में कुशलता से नेविगेट करने में सहायता करते हैं। इनका डिज़ाइन और चयन इन एल्गोरिदम की सफलता के लिए अत्यंत महत्वपूर्ण है।

"We cannot solve our problems with the same thinking we used when we created them."

"हम अपनी समस्याओं को उसी सोच से हल नहीं कर सकते जिससे हमने उन्हें पैदा किया था।"

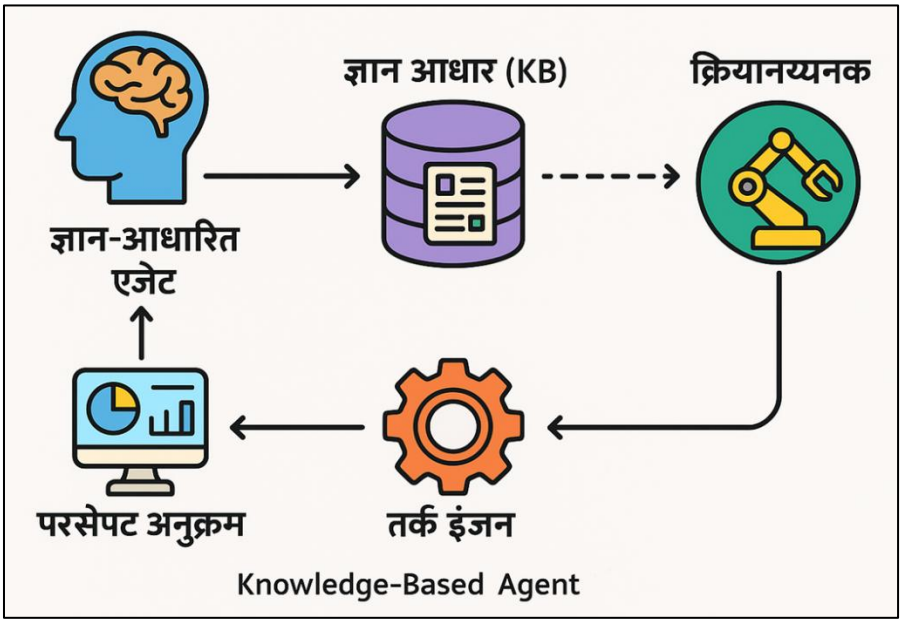
"All of us do not have equal talent. But, all of us have an equal opportunity to develop our talents."

"हम सभी में समान प्रतिभा नहीं होती, लेकिन हम सभी को अपनी प्रतिभा को विकसित करने का समान अवसर मिलता है।"

UNIT-3: ज्ञान प्रतिनिधित्व

3.1 परिचय

ज्ञान-आधारित एजेंट (Knowledge-Based Agents) कृत्रिम बुद्धिमत्ता (AI) प्रणाली का एक प्रकार हैं, जो अपने पास मौजूद ज्ञान के आधार पर कार्य करते हैं और निर्णय लेते हैं। ये एजेंट **सरल रिफ्लेक्स एजेंट्स** से अलग होते हैं, जो केवल पहले से तय नियमों के अनुसार वातावरण पर प्रतिक्रिया करते हैं। ज्ञान-आधारित एजेंट आंतरिक रूप से दुनिया का एक प्रतिनिधित्व रखते हैं और तार्किक तर्क के माध्यम से निर्णय लेते हैं। वे नए हालातों के अनुरूप ढल सकते हैं और जटिल समस्याओं को हल कर सकते हैं।



चित्र 3.1.1: ज्ञान प्रतिनिधित्व की संरचना

एक ज्ञान-आधारित एजेंट में निम्नलिखित मुख्य घटक होते हैं:

- **ज्ञान आधार (Knowledge Base - KB):** दुनिया से संबंधित तथ्यों, नियमों और संबंधों का भंडार।

- **तर्क इंजन (Inference Engine):** ज्ञान आधार की जानकारी का उपयोग करके निर्णय लेने या नई जानकारी निष्कर्षित करने का तंत्र।
- **परसेप्ट अनुक्रम (Percept Sequence):** वातावरण से प्राप्त जानकारी जिसे एजेंट ज्ञान आधार को अद्यतन करने के लिए उपयोग करता है।
- **क्रियान्वयनक (Actuators):** तर्क इंजन द्वारा लिए गए निर्णयों के आधार पर वातावरण में कार्य करते हैं।

एजेंट का संचालन एक चक्र में होता है:

परसेप्ट अनुक्रम द्वारा वातावरण की जानकारी प्राप्त करना → **ज्ञान आधार को अद्यतन करना** → **तर्क इंजन द्वारा निर्णय लेना** → **क्रियान्वयनक के माध्यम से कार्य करना**।

यह चक्र एजेंट को अधिक जानकारीपूर्ण और अनुकूल निर्णय लेने की क्षमता प्रदान करता है।

3.1.1 ज्ञान-आधारित एजेंट के घटक

1. **ज्ञान आधार (Knowledge Base):** तथ्यों, नियमों और संबंधों का संग्रह जो एजेंट की समझ और विश्वासों का प्रतिनिधित्व करता है।
2. **तर्क इंजन (Inference Engine):** ज्ञान आधार पर तार्किक नियम लागू करके नई जानकारी उत्पन्न करता है या निर्णय लेता है।
3. **परसेप्ट अनुक्रम:** वातावरण से प्राप्त जानकारी जिसे एजेंट अपने ज्ञान को अद्यतन करने हेतु उपयोग करता है।
4. **क्रियान्वयनक (Actuators):** एजेंट द्वारा लिए गए निर्णयों को वातावरण में क्रियान्वित करते हैं।

3.1.2 ज्ञान-आधारित एजेंट के कार्य

- **व्याख्या (Interpretation):** नए परसेप्ट को समझना और ज्ञान आधार को अद्यतन करना।
- **पूर्वानुमान (Prediction):** वर्तमान ज्ञान के आधार पर भविष्य की स्थितियों की भविष्यवाणी करना।
- **निदान (Diagnosis):** अपेक्षित और वास्तविक स्थिति के बीच विसंगतियों के कारणों की पहचान करना।
- **योजना बनाना (Planning):** विशेष लक्ष्यों को प्राप्त करने हेतु कार्यों की श्रृंखला बनाना।

- **सीखना (Learning):** अनुभवों से नया ज्ञान प्राप्त करके ज्ञान आधार को अद्यतन करना।

3.1.3 ज्ञान-आधारित एजेंट के लाभ

- **लचीलापन (Flexibility):** जटिल और परिवर्तनीय वातावरण से निपटने की क्षमता।
- **अनुकूलता (Adaptability):** अनुभवों से सीखकर निर्णय लेने की क्षमता में सुधार।
- **व्याख्यायता (Explainability):** निर्णय तार्किक तर्क और स्पष्ट ज्ञान पर आधारित होते हैं, जिससे उन्हें समझना और समझाना आसान होता है।

3.1.4 चुनौतियाँ और विचारणीय बिंदु

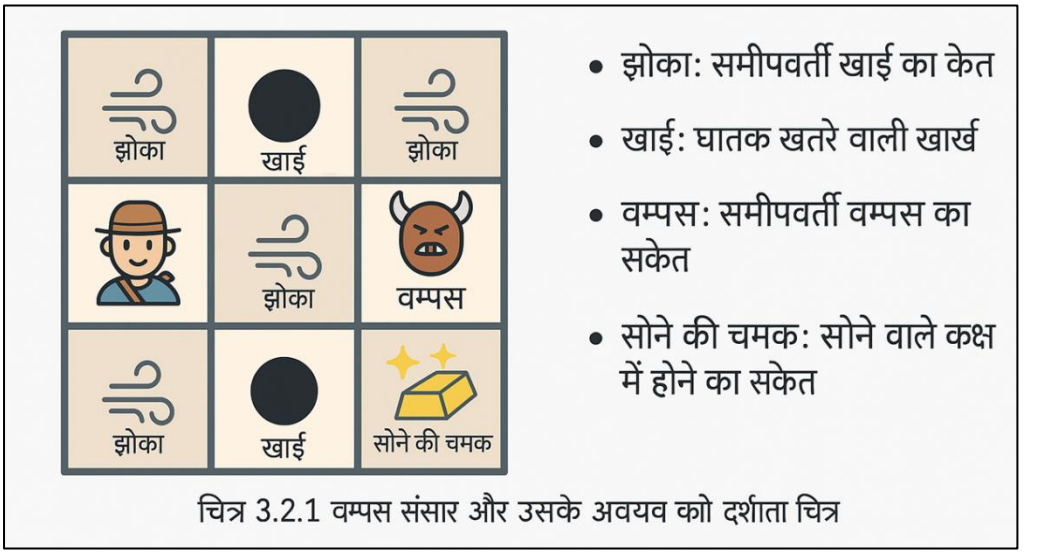
- **ज्ञान प्रतिनिधित्व:** ऐसा प्रतिनिधित्व तैयार करना जो कुशलता से तर्क और अद्यतन की अनुमति दे।
- **तर्क की दक्षता:** जैसे-जैसे ज्ञान बढ़ता है, तर्क इंजन की गति और शुद्धता बनाए रखना।
- **सुसंगतता और पूर्णता:** ज्ञान आधार में विरोधाभास न हों (सुसंगतता) और आवश्यक सभी जानकारी शामिल हो (पूर्णता)।

निष्कर्ष:

ज्ञान-आधारित एजेंट AI का एक उन्नत रूप हैं, जो निर्णय लेने और समस्याएं हल करने के लिए परिष्कृत विधियों का उपयोग करते हैं। ये विशेषज्ञ प्रणालियों, प्राकृतिक भाषा प्रसंस्करण, और स्वायत्त रोबोटिक्स जैसे क्षेत्रों में महत्वपूर्ण भूमिका निभाते हैं, जहाँ जटिल तर्क और अनुकूलन आवश्यक होते हैं।

3.2 वम्पस वर्ल्ड

वम्पस वर्ल्ड कृत्रिम बुद्धिमत्ता (AI) में उपयोग किया जाने वाला एक क्लासिक समस्या परिदृश्य है, जो ज्ञान-आधारित एजेंटों की कार्यप्रणाली को एक सरल, नियम-आधारित वातावरण में प्रदर्शित करता है। यह तर्कसंगत विचार, निर्णय-निर्माण और AI तकनीकों को सीमित स्थान में दर्शाने का आदर्श माध्यम है। वम्पस वर्ल्ड एक ग्रिड-आधारित गुफा में सेट की गई समस्या है, जिसमें एजेंट को खतरनाक बाधाओं जैसे गड्ढों और वम्पस नामक प्राणी से बचते हुए सोना खोजना होता है। एजेंट को अपने इंद्रियबोध (percepts) जैसे हवा (breeze) और दुर्गंध (stench) के आधार पर सूचित निर्णय लेने होते हैं। यह समस्या ज्ञान प्रतिनिधित्व, तार्किक निष्कर्ष और अनिश्चितता में निर्णय लेने की AI अवधारणाओं को दर्शाती है।



चित्र 3.2.1 वम्पस संसार और उसके अवयव को दर्शाता चित्र

3.2.1 वम्पस वर्ल्ड का विवरण

वम्पस वर्ल्ड का वातावरण आमतौर पर 4x4 ग्रिड के रूप में होता है। एजेंट का लक्ष्य इस ग्रिड में सोना खोजना होता है और साथ ही घातक बाधाओं से बचना होता है — जैसे कि गड्ढे और वम्पस। एजेंट ग्रिड के एक कोने से शुरुआत करता है और उसे सोने तक पहुंचना होता है।

मुख्य विशेषताएँ:

- **कक्ष (Rooms):** एजेंट एक कक्ष से उसके ऊपरी, निचले, दाएं या बाएं कक्ष में जा सकता है। तिरछी चालें नहीं हो सकतीं।
- **गड्ढे (Pits):** यदि एजेंट गड्ढे में गिरता है, तो उसकी मृत्यु हो जाती है।
- **वम्पस (Wumpus):** वम्पस से टकराने पर एजेंट मर जाता है। हालांकि, एजेंट के पास वम्पस को मारने के लिए एक तीर होता है।
- **सोना (Gold):** एजेंट का अंतिम लक्ष्य सोना खोजना और वापस सुरक्षित लौटना होता है।
- **संवेदी अनुभव (Percepts):**
    - **हवा (Breeze):** संकेत देता है कि आस-पास गड्ढा है।
    - **दुर्गंध (Stench):** दर्शाता है कि वम्पस पास के किसी कक्ष में है।
    - **चमक (Glitter):** उसी कक्ष में सोने की उपस्थिति का संकेत देता है।
    - **टक्कर (Bump):** दर्शाता है कि एजेंट दीवार से टकराया है।
    - **चीख (Scream):** संकेत देता है कि वम्पस मारा गया है।

3.2.2 रणनीति और तर्क

एजेंट निम्नलिखित रणनीतियों का प्रयोग करता है:

- **इंद्रिय व्याख्या (Percept Interpretation):** एजेंट सीमित संवेदी संकेतों से आसपास के खतरे और अवसरों का अनुमान लगाता है।
- **तार्किक निष्कर्ष (Logical Deduction):** ज्ञात नियमों और संवेदी संकेतों के आधार पर, एजेंट यह निर्धारित करता है कि कौन-से कक्ष सुरक्षित हैं।
- **सुरक्षित मार्ग योजना (Safe Path Mapping):** एजेंट एक मानसिक नक्शा बनाता है और जोखिम वाले क्षेत्रों को चिह्नित करता है।
- **वम्पस को समाप्त करना (Wumpus Elimination):** यदि एजेंट वम्पस की स्थिति का सही अनुमान लगाता है, तो वह तीर चलाकर उसे मार सकता है।
- **सोना प्राप्त करना और बाहर निकलना (Gold Retrieval and Exit):** एजेंट कम से कम जोखिम वाले रास्ते से सोना उठाता है और सुरक्षित रूप से वापस आता है।

3.2.3 AI में अनुप्रयोग

वम्पस वर्ल्ड AI की कई मूलभूत अवधारणाओं को दर्शाता है:

- **ज्ञान प्रतिनिधित्व (Knowledge Representation):** एजेंट का अपने वातावरण को मॉडल करना और विश्वासों को प्रबंधित करना।
- **तार्किक निष्कर्ष (Logical Inference):** अधूरे डेटा से दुनिया की स्थिति का अनुमान लगाना।
- **खोज और योजना (Search and Planning):** खतरे से बचते हुए लक्ष्य तक पहुंचने की योजना बनाना।
- **निर्णय-निर्माण (Decision Making):** एजेंट प्रत्येक कदम पर तर्कसंगत निर्णय लेता है।

निष्कर्ष:

वम्पस वर्ल्ड एक उत्कृष्ट उदाहरण है, जो यह दर्शाता है कि कैसे एक बुद्धिमान एजेंट सीमित जानकारी, अनिश्चितता और खतरों के बीच निर्णय लेकर अपने लक्ष्य तक पहुंच सकता है। यह AI में ज्ञान आधारित एजेंटों के डिज़ाइन और परीक्षण के लिए एक प्रभावी मॉडल है।

3.3 तर्क (Logic)

कृत्रिम बुद्धिमत्ता (AI) में तर्क एक मूलभूत घटक है जिसका उपयोग ज्ञान को दर्शाने, निष्कर्ष निकालने और उसके आधार पर तर्क करने के लिए किया जाता है ताकि कंप्यूटर उसे समझ सके। यह दुनिया के बारे में जानकारी को संहिताबद्ध करने, नियमों को परिभाषित करने, और उन नियमों के आधार पर निष्कर्ष निकालने के लिए एक व्यवस्थित ढांचा प्रदान करता है। तर्क AI सिस्टम को निर्णय लेने, समस्याओं को हल करने और डेटा में जटिल संबंधों को समझने में सहायता करता है।

चित्र 3.3.1: एक मानसिक मानचित्र आरेख जो कृत्रिम बुद्धिमत्ता (AI) में प्रयुक्त तर्क के प्रकारों और उनके अनुप्रयोगों को दर्शाता है।

3.3.1 AI में प्रयुक्त तर्क के प्रकार

प्रपोज़िशनल तर्क (Propositional Logic):

जिसे बूलियन तर्क या कथन तर्क भी कहा जाता है, यह ऐसे वाक्यों से संबंधित होता है जो या तो सत्य (True) या असत्य (False) हो सकते हैं। इसमें AND (और), OR (या), NOT (नहीं), IMPLIES (यदि...तो...), और EQUIVALENT (समतुल्य) जैसे लॉजिक ऑपरेटर का उपयोग किया जाता है। यह तर्क सरल निर्णय लेने की स्थितियों में उपयोगी होता है।

प्रेडिकेट तर्क (Predicate Logic):

इसे फर्स्ट-ऑर्डर तर्क भी कहा जाता है। यह प्रपोज़िशनल तर्क को विस्तार देता है, जिसमें वस्तुओं और उनके गुणों को दर्शाने के लिए वैरिएबल्स और प्रेडिकेट्स का उपयोग होता है। यह "सभी के लिए" (∀) और "कुछ के लिए" (∃) जैसे क्वांटिफ़ायर्स का उपयोग करता है और जटिल AI कार्यों के लिए उपयुक्त होता है।

फज़ी तर्क (Fuzzy Logic):

यह तर्क अनिश्चितता और अस्पष्टता से निपटने के लिए उपयोग किया जाता है। इसमें सत्य मान 0 और 1 के बीच किसी भी मान में हो सकता है। यह मानव जैसे निर्णय लेने की प्रक्रिया को मॉडल करता है और नियंत्रण प्रणालियों व पैटर्न पहचान में बहुत उपयोगी होता है।

टेम्पोरल तर्क (Temporal Logic):

यह समय के साथ तर्क करने पर केंद्रित होता है। इसमें यह बताया जाता है कि वाक्य का सत्य मान समय के साथ कैसे बदलता है। यह वास्तविक समय प्रणालियों, सॉफ़्टवेयर व योजनाबद्धन जैसे अनुप्रयोगों में विशेष रूप से उपयोगी होता है।

3.3.2 AI में तर्क का उपयोग

विशेषज्ञ प्रणालियाँ (Expert Systems):

डोमेन-विशिष्ट ज्ञान और नियमों के आधार पर समस्या हल करने और निर्णय लेने के लिए उपयोग की जाती हैं। इसमें ज्ञान भंडार और एक तर्क इंजन होता है।

प्राकृतिक भाषा प्रसंस्करण (Natural Language Processing - NLP):

तर्क मानव भाषाओं को समझने, व्याख्या करने, अनुवाद करने और प्रतिक्रिया देने में सहायक होता है। यह वाक्य संरचना और अर्थ को समझने के लिए आवश्यक होता है।

स्वचालित तर्क (Automated Reasoning):

कंप्यूटर की सहायता से प्रमेय साबित करना, पहेली हल करना या निष्कर्ष निकालना शामिल होता है। यह गणित, लॉजिक प्रोग्रामिंग आदि में उपयोगी है।

ज्ञान प्रतिनिधित्व (Knowledge Representation):

दुनिया के ज्ञान को इस तरह से संरचित करना कि AI सिस्टम उसे उपयोग करके समस्याएं हल कर सकें। तर्क का उपयोग इस ज्ञान को संगठित करने और उपयोग में लाने के लिए किया जाता है।

3.3.3 चुनौतियाँ

अनिश्चितता का प्रबंधन:

क्लासिकल तर्क बाइनरी (सत्य या असत्य) होता है, जिससे यह अधूरी या अनिश्चित जानकारी को संभालने में कठिन होता है। फज़ी लॉजिक और संभाव्यता आधारित विधियाँ इसे हल करती हैं।

गणनात्मक जटिलता:

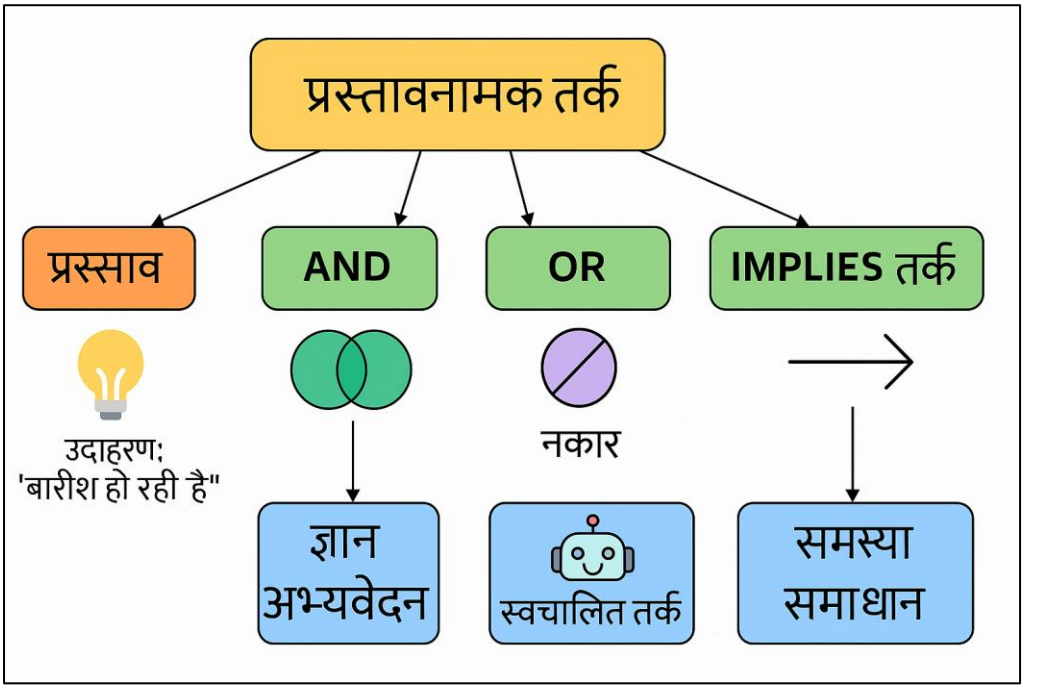
बड़े नियमों और जटिल तर्क के सेट के साथ काम करने में अधिक संसाधनों और समय की आवश्यकता होती है।

निष्कर्ष:

तर्क AI की नींव है जो जटिल तर्क और समझ की आवश्यकता वाले कार्यों को करने में सक्षम बनाता है। इसकी सीमाओं के बावजूद, यह आज भी AI अनुसंधान और अनुप्रयोगों में प्रमुख भूमिका निभा रहा है।

3.4 प्रस्तावनात्मक तर्क (Propositional Logic)

प्रस्तावनात्मक तर्क, जिसे प्रस्तावनात्मक गणना (Propositional Calculus) या कथन तर्क (Statement Logic) भी कहा जाता है, तर्कशास्त्र की वह शाखा है जो ऐसे कथनों (Propositions) से संबंधित है जो या तो सत्य (True) हो सकते हैं या असत्य (False)। यह सरल कथनों को तार्किक संयोजकों (Logical Connectives) के माध्यम से जोड़कर जटिल अभिव्यक्तियाँ बनाने की अनुमति देता है। यह तर्कशास्त्र गणित, कंप्यूटर विज्ञान और कृत्रिम बुद्धिमत्ता (AI) में बुनियादी भूमिका निभाता है, खासकर कथनों की सत्यता का विश्लेषण करने और स्वचालित तर्क के एल्गोरिद्म विकसित करने में।



चित्र 3.4.1 क्षैतिज दिशा में प्रस्तुत प्रस्ताविक तर्क के प्रमुख अवधारणाओं और उनके आपसी संबंधों को दर्शाने वाला आरेख।

3.4.1 प्रस्तावनात्मक तर्क की मुख्य अवधारणाएँ

- **प्रस्ताव (Propositions):** ऐसे मूल कथन जो या तो सत्य होते हैं या असत्य। उदाहरण: "बारिश हो रही है" एक प्रस्ताव है।

- **तार्किक संयोजक (Logical Connectives):** प्रस्तावों को जोड़ने के लिए उपयोग किए जाने वाले ऑपरेटर:
    - **AND (और):** जब दोनों प्रस्ताव सत्य हों तभी संयुक्त प्रस्ताव सत्य होगा।
    - **OR (या):** जब कम से कम एक प्रस्ताव सत्य हो, तब संयुक्त प्रस्ताव सत्य होगा।
    - **NOT (नकार):** किसी प्रस्ताव का नकार तब सत्य होता है जब वह प्रस्ताव असत्य हो।
    - **IMPLIES (निष्कर्ष):** यदि एक प्रस्ताव सत्य है तो दूसरा भी होना चाहिए (A ⇒ B).
    - **IFF (यदि और केवल यदि):** दोनों प्रस्ताव एक-दूसरे के समतुल्य हों (A ⇔ B)।

3.4.2 एआई में अनुप्रयोग (Applications in AI)

ज्ञान अभ्यवेदन (Knowledge Representation):

एआई में ज्ञान को एक संगठित रूप में निरूपित करने के लिए प्रस्तावनात्मक तर्क का उपयोग किया जाता है। इसके माध्यम से तथ्यों, नियमों और उनके संबंधों को मशीन पठन योग्य प्रारूप में संजोया जाता है, जैसा कि विशेषज्ञ प्रणालियों और सेमांटिक वेब में होता है।

स्वचालित तर्क (Automated Reasoning):

कंप्यूटर के माध्यम से तार्किक निष्कर्ष निकालना, प्रमेय सिद्ध करना आदि कार्यों में प्रस्तावनात्मक तर्क का प्रयोग होता है। यह विधि फॉर्मल वेरिफिकेशन, योजना निर्माण, और निर्णय लेने जैसे कार्यों में उपयोगी है।

समस्या समाधान (Problem Solving):

एआई में किसी समस्या का समाधान निकालने के लिए प्रस्तावनात्मक तर्क द्वारा नियमों, परिस्थितियों और लक्ष्य की परिभाषा दी जाती है। इससे सिस्टम विभिन्न संभावित कदमों का तार्किक विश्लेषण कर सकता है, जैसे कि रोबोटिक्स में योजना निर्माण, या सुडोकू हल करना।

3.4.3 चुनौतियाँ (Challenges):

- **अभिव्यक्तिता की सीमा:**
  प्रस्तावनात्मक तर्क "कुछ", "सभी" जैसे क्वांटीफायर को व्यक्त नहीं कर सकता — इसके लिए प्रत्यय तर्क (Predicate Logic) की आवश्यकता होती है।

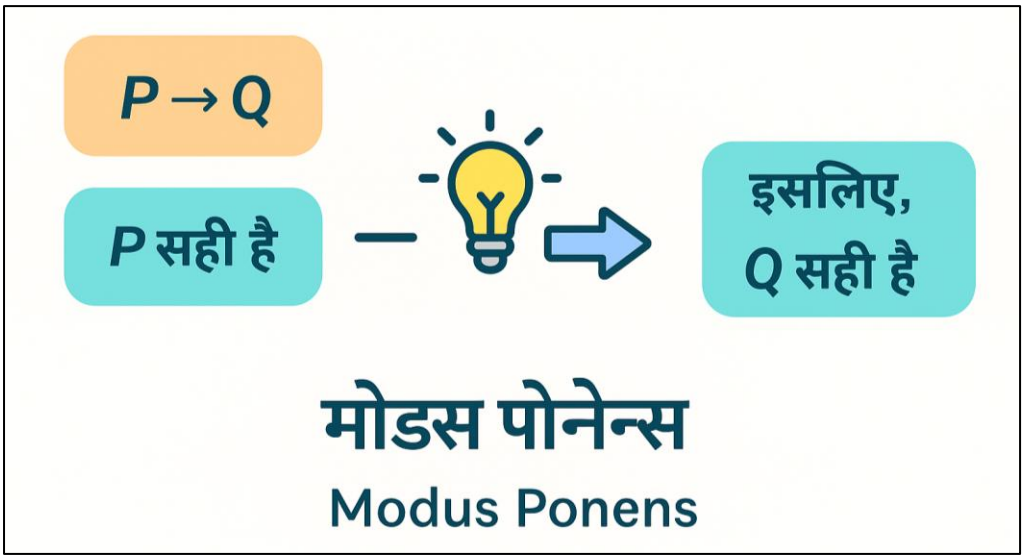
- **गणनात्मक जटिलता (Computational Complexity):**
  बड़े और जटिल प्रस्तावों के लिए सत्यता निर्धारण (Satisfiability) की प्रक्रिया संसाधन-गहन हो सकती है।

निष्कर्ष:

प्रस्तावनात्मक तर्क एआई के लिए एक मजबूत और स्पष्ट ढांचा प्रदान करता है, जिससे मशीनें तर्कसंगत निर्णय लेने, समस्या हल करने और स्वचालित निष्कर्ष निकालने में सक्षम बनती हैं।

3.5 प्रस्तावित प्रमेय सिद्ध (Propositional Theorem Proving)

प्रस्तावित प्रमेय सिद्ध एक प्रक्रिया है जिसमें किसी प्रस्तावित तर्क सूत्र को सत्य सिद्ध करने के लिए विभिन्न एल्गोरिद्म और तकनीकों का उपयोग किया जाता है। यह लॉजिक, गणित और कंप्यूटर साइंस का एक मूलभूत हिस्सा है, विशेष रूप से कृत्रिम बुद्धिमत्ता (AI) में सॉफ़्टवेयर सत्यापन, हार्डवेयर डिज़ाइन और लॉजिकल रीजनिंग सिस्टम के लिए अत्यंत उपयोगी है।



चित्र 3.5.1: प्रस्तावनात्मक प्रमेय सिद्ध करने में मोडस पोनेन्स नियम का चित्रण

3.5.1 प्रक्रिया:

इस प्रक्रिया में सामान्यतः मूल सूत्र को एक कैनॉनिकल रूप में परिवर्तित किया जाता है और फिर किसी निर्णय प्रक्रिया का उपयोग करके उसकी संतुष्टता (satisfiability) का मूल्यांकन किया जाता है। सबसे सामान्य रूप में इसका उपयोग संयोजक सामान्य रूप (CNF - Conjunctive Normal Form) में किया जाता है, जहाँ सूत्र को लिटेरल (परिवर्ती या उनका निषेध) के विस्यंदों (disjunctions) के संयोजन (conjunction) के रूप में लिखा जाता है।

3.5.2 तकनीकें:

1. **सत्य सारणी (Truth Table):** एक बुनियादी विधि जो सभी संभावित ट्रू/फॉल्स संयोजनों के लिए सूत्र का मूल्यांकन करती है। हालांकि सरल, यह बड़े सूत्रों के लिए अप्रभावी हो सकती है क्योंकि संभावनाओं की संख्या अत्यधिक हो जाती है।
2. **रिज़ोल्यूशन (Resolution):** एक अनुमिति नियम (inference rule) है जो क्लॉज़ के सेट से विरोधाभास सिद्ध करके संतुष्टता को प्रमाणित करता है। यह स्वचालित प्रमेय सिद्ध प्रणालियों का आधार बनता है।
3. **DPLL एल्गोरिद्म:** Davis-Putnam-Logemann-Loveland एल्गोरिद्म एक बैकट्रैकिंग विधि है जो यूनिट प्रोपेगेशन और प्योर लिटेरल एलिमिनेशन जैसी तकनीकों से समस्या को सरल बनाता है।
4. **SAT सॉल्वर:** यह उन्नत एल्गोरिद्म होते हैं जो बड़ी संख्या में वेरिएबल और क्लॉज़ के साथ भी तेजी से समाधान प्रदान कर सकते हैं।

3.5.3 एआई में अनुप्रयोग:

- **स्वचालित तर्क (Automated Reasoning):** तर्क करने वाले सिस्टम बनाना जो जानकारी से निष्कर्ष निकाल सकें या प्रमेय सिद्ध कर सकें।
- **औपचारिक सत्यापन (Formal Verification):** सॉफ़्टवेयर और हार्डवेयर डिज़ाइन की शुद्धता की पुष्टि।
- **ज्ञान अभ्यावेदन और तर्क (Knowledge Representation and Reasoning):** ज्ञात तथ्यों और नियमों से नया ज्ञान निकालना।

3.5.4 चुनौतियाँ:

- **गणनात्मक जटिलता:** प्रस्ताविक तर्क की संतुष्टता निर्धारण एक NP-Complete समस्या है, अर्थात् इसका कोई ज्ञात बहुपद समय समाधान नहीं है।
- **स्केलेबिलिटी:** जैसे-जैसे सूत्र बड़ा होता है, तकनीकों की प्रभावशीलता चुनौतीपूर्ण हो जाती है।

3.5.5 प्रमेय: मोडस पोनेन्स (Modus Ponens)

मोडस पोनेन्स एक मूलभूत अनुमिति नियम है जो एक प्रस्ताव और एक प्रभाव से निष्कर्ष निकालता है: **यदि P→Q और P सत्य है, तो Q भी सत्य होगा।**

प्रमाण:

1. दिए गए हैं: P→Q और P सत्य है।
2. हमें सिद्ध करना है: Q सत्य है।

सत्य सारणी:

P	**Q**	**P→Q**
T	T	T
T	F	F
F	T	T
F	F	T

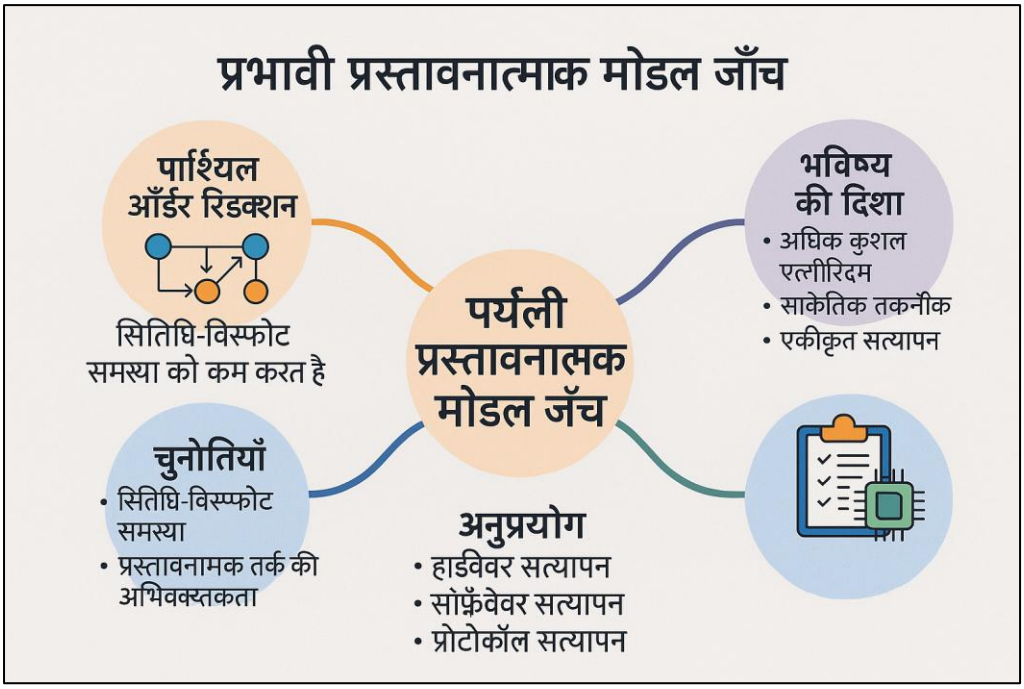
- यदि P सत्य है और P→Q भी सत्य है, तो Q असत्य नहीं हो सकता।
- इसलिए Q भी सत्य होना चाहिए।

निष्कर्ष:

P→Q और P के आधार पर Q को तर्क द्वारा सत्य सिद्ध किया जा सकता है, जो मोडस पोनेन्स की वैधता को दर्शाता है।

3.6 प्रभावी प्रस्तावनात्मक मॉडल जांच (Effective Propositional Model Checking)

मॉडल जांच (Model Checking) कंप्यूटर विज्ञान और औपचारिक सत्यापन (formal verification) में प्रयुक्त एक तकनीक है जिसका उपयोग यह स्वचालित रूप से जाँचने के लिए किया जाता है कि किसी प्रणाली (system) का मॉडल किसी निर्दिष्ट विनिर्देश (specification) को पूरा करता है या नहीं, जिसे आमतौर पर प्रस्तावनात्मक तर्क (propositional logic) या कालिक तर्क (temporal logic) में व्यक्त किया जाता है। इसमें मॉडल की स्थिति-स्थान (state space) का अन्वेषण करके यह सत्यापित किया जाता है कि सुरक्षा (safety) या जीवंतता (liveness) जैसी विशेषताएँ प्रणाली के सभी संभावित क्रियान्वयन (executions) में लागू होती हैं या नहीं।



चित्र 3.6.1: प्रभावी प्रोपोजिशनल मॉडल जांच की तकनीकों को दर्शाने वाला चित्र

3.6.1 प्रस्तावनात्मक मॉडल जांच का अवलोकन

प्रस्तावनात्मक मॉडल जांच उन मॉडलों के सत्यापन पर केंद्रित होती है जहाँ विनिर्देश प्रस्तावनात्मक तर्क में व्यक्त किए जाते हैं। यह विशेष रूप से सीमित स्थिति स्थान (finite state space) वाली प्रणालियों के

लिए प्रभावी है, जो वांछित गुणों के विरुद्ध पूर्ण सत्यापन की अनुमति देती है। इस प्रक्रिया में सिस्टम मॉडल और विनिर्देश दोनों को प्रस्तावनात्मक सूत्रों (formulas) के रूप में एन्कोड किया जाता है, और फिर यह निर्धारित करने के लिए एल्गोरिदमिक तकनीकों का उपयोग किया जाता है कि क्या मॉडल विनिर्देशों को संतुष्ट करता है।

3.6.2 प्रस्तावनात्मक मॉडल जांच की तकनीकें

प्रस्तावनात्मक मॉडल जांच की दक्षता और विस्तार क्षमता (scalability) को बेहतर बनाने के लिए कई तकनीकों का विकास किया गया है, जिनमें मुख्य रूप से निम्नलिखित शामिल हैं:

सांकेतिक मॉडल जांच (Symbolic Model Checking)

सांकेतिक मॉडल जांच ने बाइनरी निर्णय आरेख (Binary Decision Diagrams - BDDs) और अन्य सांकेतिक डेटा संरचनाओं (symbolic data structures) का उपयोग करके स्थिति-विस्फोट समस्या (state explosion problem) को हल कर औपचारिक सत्यापन के क्षेत्र में क्रांति ला दी। किसी प्रणाली की सभी संभावित स्थितियों को स्पष्ट रूप से गिनने के बजाय, यह तकनीक स्थितियों और उनके संक्रमणों (transitions) के समूहों को संक्षिप्त गणितीय अभिव्यक्तियों के रूप में दर्शाती है। विशेष रूप से, BDDs बूलियन फलनों (Boolean functions) के कुशल प्रतिनिधित्व और हेरफेर की अनुमति देते हैं, जिससे जांचकर्ता एक साथ बड़ी स्थिति-संख्या पर संचालन कर सकता है। यह विधि जाँच को अत्यधिक विस्तार योग्य बनाती है, जिससे जटिल हार्डवेयर डिज़ाइन और प्रोटोकॉल का सत्यापन संभव हो पाता है।

सीमाबद्ध मॉडल जांच (Bounded Model Checking - BMC)

सीमाबद्ध मॉडल जांच पारंपरिक मॉडल जांच विधियों की सीमाओं को संबोधित करती है, और किसी निर्दिष्ट गहराई (bound) के भीतर प्रतिवाद (counterexample) खोजने पर केंद्रित होती है। यह तकनीक सत्यापन समस्या को एक सैटिस्फायबिलिटी (SAT) समस्या में परिवर्तित करती है, और SAT सॉल्वरों की शक्ति का उपयोग करके प्रतिवाद को प्रभावी ढंग से खोजती है। इस प्रक्रिया में सिस्टम की प्रारंभिक स्थिति, संक्रमण संबंध (transition relation), और सत्यापित की जाने वाली संपत्ति का निषेध एक SAT सूत्र में एन्कोड किया जाता है। यदि SAT सॉल्वर कोई ऐसा चर असाइनमेंट ढूंढता है जो सूत्र को संतुष्ट करता है, तो इसका अर्थ है कि दी गई सीमा के भीतर संपत्ति का उल्लंघन हुआ है। यह तकनीक उथले बग्स (shallow bugs) को जल्दी पहचानने में विशेष रूप से प्रभावी है और सॉफ़्टवेयर और हार्डवेयर उद्योग में औपचारिक सत्यापन को अपनाने में सहायक रही है।

3.7 प्रोपोज़िशनल लॉजिक पर आधारित एजेंट्स

3.7.1 प्रोपोज़िशनल लॉजिक एजेंट्स का परिचय

परिभाषा और पृष्ठभूमि

प्रोपोज़िशनल लॉजिक एजेंट्स कृत्रिम बुद्धिमत्ता (AI) के क्षेत्र में ऐसे एजेंट्स होते हैं जो उन प्रस्तावों (propositions) पर आधारित निर्णय लेते हैं जो या तो सत्य (true) हो सकते हैं या असत्य (false)। प्रोपोज़िशनल लॉजिक, जिसे वाक्य तर्क भी कहते हैं, प्रतीकात्मक तर्क का एक आधारभूत भाग है और यह तर्क और निर्णय लेने की एक सरल लेकिन प्रभावी रूपरेखा प्रदान करता है।

ये एजेंट्स लॉजिक के एक औपचारिक तंत्र का उपयोग करते हैं जिसमें वेरिएबल्स (चर) होते हैं जो दुनिया से जुड़े तथ्यों को दर्शाते हैं, और लॉजिकल कनेक्टिव्स (AND, OR, NOT आदि) के माध्यम से सरल वाक्यों से जटिल वाक्य बनाए जाते हैं। ये एजेंट्स इन जटिल वाक्यों की सत्यता का मूल्यांकन करके तर्क कर सकते हैं और निर्णय ले सकते हैं।

मानव जैसी तर्क क्षमताओं को मशीनों में अनुकरण करने की खोज ने प्रोपोज़िशनल लॉजिक एजेंट्स के अध्ययन और उपयोग को जन्म दिया है। यह एजेंट्स ज्ञान को एक संरचित और संगणनात्मक रूप में संग्रहित कर, निर्णय प्रक्रिया को स्वचालित करते हैं, समस्याओं का समाधान करते हैं और दिए गए तथ्यों के आधार पर भविष्यवाणी भी कर सकते हैं।

अनुप्रयोग क्षेत्र

प्रोपोज़िशनल लॉजिक एजेंट्स कई क्षेत्रों में उपयोग किए जाते हैं, जो उनकी बहुप्रयोज्यता और जटिल निर्णयों को हल करने की क्षमता को दर्शाते हैं:

- **स्वचालित प्रमेय सिद्धि (Automated Theorem Proving):** ये एजेंट्स गणितीय प्रमेयों को सिद्ध करने में मदद करते हैं, दिए गए स्वयंसिद्धों (axioms) के आधार पर तर्क निकालकर।
- **विशेषज्ञ प्रणाली (Expert Systems):** मनुष्यों के विशेषज्ञ ज्ञान को एक लॉजिक-आधारित ज्ञान आधार में संग्रहित कर, ये एजेंट्स चिकित्सीय निदान, वित्तीय विश्लेषण जैसे क्षेत्रों में निर्णय लेते हैं।

- **खेलों में उपयोग (Game Playing):** नियमों और संभावित परिणामों वाले खेलों में ये एजेंट्स सर्वोत्तम चाल या रणनीति तय करने में सक्षम होते हैं।
- **योजना और अनुसूची निर्माण (Planning and Scheduling):** ये एजेंट्स वांछित लक्ष्यों को प्राप्त करने के लिए कार्रवाईयों का अनुक्रम निर्धारित करते हैं — जैसे कि विनिर्माण प्रक्रियाओं, लॉजिस्टिक्स और व्यक्तिगत सहायक प्रणालियों में।

3.7.2 प्रोपोज़िशनल लॉजिक एजेंट्स की संरचना

ज्ञान आधार (Knowledge Base - KB)

ज्ञान आधार एजेंट का वह घटक होता है जहाँ उसके पास मौजूद सभी तथ्यों और नियमों को संग्रहित किया जाता है। यहाँ प्रत्येक तथ्य को एक प्रस्ताव (proposition) के रूप में प्रस्तुत किया जाता है — एक साधारण कथन जो या तो सत्य होता है या असत्य।

उदाहरण के लिए: "कमरे में रोशनी है" एक प्रस्ताव हो सकता है। ज्ञान आधार लॉजिक वाक्यों द्वारा ज्ञान को संक्षिप्त और संगठित रूप में संग्रहित करता है जिससे एजेंट तर्क कर सके और बड़ी मात्रा में जानकारी को प्रभावी रूप से प्रबंधित कर सके।

इनफेरेंस इंजन (Inference Engine)

इनफेरेंस इंजन एजेंट को उसकी ज्ञान आधार से नया ज्ञान प्राप्त करने में सक्षम बनाता है। यह लॉजिक नियमों जैसे कि:

- **Modus Ponens:** यदि $P \rightarrow Q$ और P सत्य है, तो Q भी सत्य है।
- **Modus Tollens:** यदि $P \rightarrow Q$ और Q असत्य है, तो P भी असत्य है।

इन नियमों के आधार पर नए निष्कर्ष निकालता है।

यह घटक एजेंट की तर्क क्षमता का केंद्र होता है और यही तय करता है कि एजेंट कब, क्या निर्णय लेगा और किस आधार पर। इसकी दक्षता एजेंट के प्रदर्शन को सीधे प्रभावित करती है।

3.7.3 प्रोपोज़िशनल लॉजिक एजेंट्स का डिज़ाइन

सांकेतिक प्रतिनिधित्व (Symbolic Representation)

प्रोपोज़िशनल लॉजिक एजेंट्स ज्ञान को सांकेतिक रूप (symbolic representation) में दर्शाते हैं, जहाँ दुनिया के तथ्यों को वाक्यों के रूप में व्यक्त किया जाता है। ये वाक्य प्रोपोज़िशनल लॉजिक के माध्यम से बनाए जाते हैं — एक ऐसा तर्क तंत्र जो ऐसे प्रस्तावों (statements) से संबंधित होता है जो या तो सत्य हो सकते हैं या असत्य। इस तकनीक के माध्यम से एजेंट्स तर्क कर सकते हैं और प्रभावी ढंग से निर्णय ले सकते हैं। यह प्रतिनिधित्व एजेंट को बयानों की तर्क संरचना को समझने और उस पर कार्य करने में सक्षम बनाता है।

बैकवर्ड चेनिंग (Backward Chaining)

बैकवर्ड चेनिंग एक लक्ष्य-आधारित (goal-driven) निष्कर्षण तकनीक है। इसमें एजेंट पहले लक्ष्य से शुरुआत करता है और फिर पीछे की ओर काम करता है, यह पता लगाने की कोशिश करता है कि कौन से नियम उस लक्ष्य की प्राप्ति में मदद कर सकते हैं। यदि उन नियमों के उपसर्ग (premises) ज्ञात नहीं हैं, तो एजेंट recursively उन्हें सिद्ध करने का प्रयास करता है जब तक कि वह ज्ञात तथ्यों तक न पहुँच जाए या यह निर्धारित न कर ले कि लक्ष्य सिद्ध नहीं किया जा सकता। यह तकनीक तब अत्यंत प्रभावी होती है जब लक्ष्य विशेष हो और एजेंट यह जानना चाहता हो कि क्या वह लक्ष्य प्राप्त किया जा सकता है।

3.7.4 प्रोपोज़िशनल लॉजिक एजेंट्स के लिए एल्गोरिद्म

फॉरवर्ड चेनिंग (Forward Chaining)

फॉरवर्ड चेनिंग एक निष्कर्षण तकनीक है जिसमें एजेंट ज्ञात तथ्यों (facts) से शुरुआत करता है और inference rules का उपयोग करके नए तथ्यों की खोज करता है। यह एक नीचे से ऊपर (bottom-up) तर्क प्रक्रिया है, जिसमें एजेंट ज्ञान आधार पर लगातार नियम लागू करता है जब तक कि कोई लक्ष्य प्राप्त न हो जाए या और कोई नियम लागू न किया जा सके। यह विधि तब उपयोगी होती है जब ज्ञात तथ्यों के आधार पर सभी संभावित निष्कर्ष प्राप्त करने की आवश्यकता हो।

3.7.5 चुनौतियाँ और सीमाएँ

अभिव्यक्तात्मक सीमाएँ (Expressiveness Limitations)

हालाँकि प्रोपोज़िशनल लॉजिक सत्य और असत्य मूल्यों पर आधारित तर्क के लिए एक मजबूत आधार प्रदान करता है, इसमें कुछ सीमाएँ होती हैं। यह "कुछ", "सभी" जैसे क्वांटिफ़ायर्स, वस्तुओं के बीच संबंध, या समय और परिवर्तन की अवधारणाओं को सीधे अभिव्यक्त नहीं कर सकता। इसलिए, यह जटिल तर्क प्रक्रिया वाले अनुप्रयोगों के लिए उपयुक्त नहीं हो सकता।

गणनात्मक सीमाएँ (Computational Limitations)

जैसे-जैसे ज्ञान आधार बड़ा होता जाता है, तर्क करने की गणनात्मक जटिलता भी बढ़ती जाती है। किसी प्रस्ताव की सत्यता तय करने के लिए कई बार संभावित सत्यों के सभी संयोजनों की जांच करनी पड़ सकती है, जो समय और संसाधनों की दृष्टि से महंगा होता है। यद्यपि अनुकूलन (optimization) और heuristic विधियाँ इन समस्याओं को कम करने में मदद करती हैं, लेकिन वे पूरी तरह से समाधान नहीं देतीं।

3.7.6 व्यावहारिक अनुप्रयोग और केस स्टडीज़

स्वचालित प्रमेय सिद्धि (Automated Theorem Proving)

प्रोपोज़िशनल लॉजिक एजेंट्स का उपयोग गणितीय प्रमेयों को बिना मानवीय हस्तक्षेप के स्वचालित रूप से सिद्ध करने के लिए किया जाता है। ये एजेंट लॉजिक के औपचारिक नियमों का उपयोग करके संभावित प्रमाणों की खोज करते हैं। उदाहरण के लिए, सॉफ़्टवेयर के शुद्धता सत्यापन में, एजेंट यह सिद्ध कर सकते हैं कि किसी सॉफ़्टवेयर की कार्यप्रणाली सभी परिस्थितियों में उसकी विशिष्टताओं के अनुरूप है — जैसे कि एयरोस्पेस और मेडिकल उपकरणों में।

विशेषज्ञ प्रणाली (Expert Systems)

विशेषज्ञ प्रणाली ऐसे कंप्यूटर प्रोग्राम होते हैं जो किसी विशेषज्ञ के निर्णय और व्यवहार का अनुकरण करते हैं। प्रोपोज़िशनल लॉजिक एजेंट्स इन प्रणालियों में डोमेन-विशिष्ट ज्ञान को संग्रहीत और उपयोग करने की सुविधा प्रदान करते हैं। जैसे — एक चिकित्सीय विशेषज्ञ प्रणाली रोगी के लक्षणों के आधार पर संभावित बीमारियों का अनुमान लगा सकती है, या एक वित्तीय विश्लेषण प्रणाली बाज़ार संकेतकों के आधार पर निवेश निर्णय ले सकती है। ये प्रणाली तेज़, सुसंगत और मापनीय होती हैं।

नियम और वाक्य (Rules and Sentences)

प्रोपोज़िशनल लॉजिक का सिंटैक्स (syntax) एक निश्चित नियमों की व्यवस्था है जो प्रस्तावों से वाक्य बनाता है। ये वाक्य AND, OR, NOT, IMPLIES, और EQUIVALENT जैसे लॉजिकल कनेक्टिव्स द्वारा आपस में जुड़े होते हैं। प्रत्येक प्रस्ताव एक ऐसा कथन होता है जो या तो सत्य होता है या असत्य। फिर, इनके सेमांटिक्स (semantics) यह परिभाषित करते हैं कि इन वाक्यों के सत्य मूल्य कैसे निर्धारित होते हैं। इसके लिए अक्सर ट्रुथ टेबल का उपयोग किया जाता है।

3.7.7 प्रगति और भविष्य की दिशा

प्रदर्शन की दक्षता बढ़ाना (Enhancing Efficiency)

प्रोपोज़िशनल लॉजिक एजेंट्स में निष्कर्षण (inference) की दक्षता एक प्रमुख शोध क्षेत्र है। जैसे-जैसे ज्ञान आधार (Knowledge Base) बड़ा होता जाता है, प्रोपोज़िशनल लॉजिक से संबंधित गणनात्मक चुनौतियाँ भी बढ़ती जाती हैं। इन चुनौतियों को ध्यान में रखते हुए शोधकर्ता ऐसे एल्गोरिद्म और हीयूरिस्टिक (heuristics) विकसित कर रहे हैं जो निष्कर्षण की गति को तेज़ करें और इसके लिए आवश्यक संसाधनों की मात्रा को कम करें।

इसके लिए पैरेलल प्रोसेसिंग, अनुकूलन एल्गोरिद्म (optimization algorithms), और मशीन लर्निंग जैसी तकनीकों का भी प्रयोग किया जा रहा है, जिससे प्रोपोज़िशनल लॉजिक एजेंट्स को अधिक प्रभावशाली और बड़े तथा जटिल डेटा सेट को संभालने में सक्षम बनाया जा सके।

अन्य लॉजिक प्रणालियों के साथ एकीकरण (Integration with Other Forms of Logic)

प्रोपोज़िशनल लॉजिक की सीमाओं को दूर करने के लिए इसे अन्य लॉजिक प्रणालियों जैसे फर्स्ट-ऑर्डर लॉजिक (First-Order Logic - FOL) और डिस्क्रिप्शन लॉजिक (Description Logics) के साथ एकीकृत करने पर भी ज़ोर दिया जा रहा है।

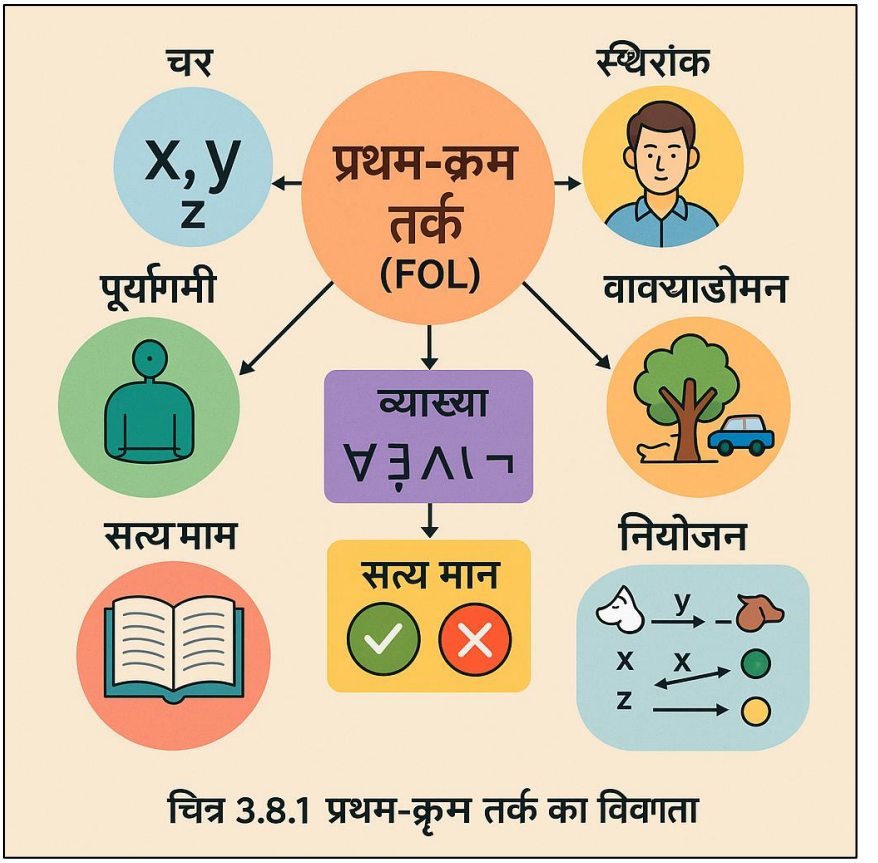
FOL प्रोपोज़िशनल लॉजिक का विस्तार करता है जिसमें क्वांटिफ़ायर्स (जैसे "सभी", "कुछ") और वेरिएबल्स को शामिल किया जाता है, जो वस्तुओं और उनके गुणों तथा आपसी संबंधों को दर्शा सकते हैं। इस एकीकरण के माध्यम से ज्ञान को अधिक अभिव्यक्तिपूर्ण ढंग से प्रस्तुत किया जा सकता है, जिससे एजेंट्स अधिक सूक्ष्म और व्यापक दृष्टिकोण से दुनिया के बारे में तर्क कर सकते हैं। यह उन्नति कृत्रिम बुद्धिमत्ता (AI) के उन क्षेत्रों में क्रांति ला रही है जहाँ जटिल निर्णय-निर्माण प्रणालियों और प्राकृतिक भाषा की अधिक सटीक समझ की आवश्यकता होती है।

निष्कर्ष (Conclusion)

ये प्रगतियाँ और अनुप्रयोग यह दर्शाते हैं कि प्रोपोज़िशनल लॉजिक एजेंट्स का शोध क्षेत्र जीवंत और सतत प्रगतिशील है। वर्तमान सीमाओं को संबोधित करने और नए एकीकरण व अनुप्रयोगों की खोज के माध्यम से यह क्षेत्र निरंतर विकसित हो रहा है, जिससे ये एजेंट वास्तविक दुनिया की समस्याओं को सुलझाने में अधिक प्रभावशाली और उपयोगी बनते जा रहे हैं।

3.8 प्रथम-क्रम तर्क – प्रथम-क्रम तर्क का वाक्यविन्यास और अर्थवत्ता (Syntax and Semantics)

प्रथम-क्रम तर्क (First-Order Logic - FOL), जिसे **पूर्वगामी तर्क** या **प्रथम-क्रम उपपत्ति कलन** भी कहा जाता है, प्रस्तावनात्मक तर्क (Propositional Logic) को विस्तार देता है। इसमें गणकों (Quantifiers) और वस्तुओं के बीच संबंधों तथा गुणों को व्यक्त करने की क्षमता होती है, जिससे यह और अधिक अभिव्यक्तिपूर्ण और जटिल बयानों को दर्शाने में सक्षम होता है।



चित्र 3.8.1 प्रथम-क्रम तर्क का विवगता

3.8.1 प्रथम-क्रम तर्क का वाक्यविन्यास (Syntax of First-Order Logic)

FOL के वाक्यविन्यास में निम्नलिखित तत्व होते हैं:

- **चर (Variables)**: ऐसे प्रतीक जो डोमेन (domain) के किसी भी वस्तु को दर्शाते हैं।

  उदा: x, y, z

- **स्थिरांक (Constants)**: डोमेन के विशिष्ट वस्तुओं को दर्शाते हैं।

  उदा: John, a, b
- **पूर्वगामी (Predicates)**: वस्तुओं के गुणों या उनके आपसी संबंधों को दर्शाते हैं।

  उदा: Father(John, x) - "John is the father of x"
- **संज्ञाएँ (Functions)**: डोमेन की वस्तुओं को अन्य वस्तुओं से जोड़ने का कार्य करते हैं।

  उदा: Mother(x) = x की माता
- **गणक (Quantifiers)**:
  - **सार्वत्रिक गणक (∀)**: "सभी के लिए" (for all)

    उदा: ∀x Human(x) → Mortal(x)
  - **अस्तित्व गणक (∃)**: "किसी एक के लिए" (there exists)

    उदा: ∃x Cat(x) ∧ Black(x)
- **तार्किक संयोजक (Logical Connectives)**:
  - AND (∧), OR (∨), NOT (¬), IMPLIES (→), EQUIVALENT (↔)
- **कोष्ठक (Parentheses)**: वाक्य रचना की प्राथमिकता को स्पष्ट करने के लिए

इन तत्वों का उपयोग करके अच्छी तरह से निर्मित सूत्र (Well-formed formulas - WFFs) बनाए जाते हैं।

3.8.2 प्रथम-क्रम तर्क की अर्थवत्ता (Semantics of First-Order Logic)

FOL की अर्थवत्ता सूत्रों के सही या गलत होने के अर्थ को निर्धारित करती है:

- **व्याख्या (Interpretation)**: यह यह तय करती है कि कौन से स्थिरांक किस वस्तु का प्रतिनिधित्व करते हैं, कौन से पूर्वगामी किस संबंध को दर्शाते हैं, और कौन से संज्ञाएँ कौन से क्रियाओं को।
- **वाक्यडोमेन (Domain of Discourse)**: उन सभी वस्तुओं का समूह जिनका वर्णन चर करते हैं।
- **नियोजन (Assignment)**: यह हर चर को डोमेन की किसी वस्तु के साथ जोड़ता है।

- **सत्य मान (Truth Value)**: किसी सूत्र का सत्य या असत्य होना इस बात पर निर्भर करता है कि वह डोमेन, व्याख्या, और नियोजन के संदर्भ में सही है या नहीं।

Universal Quantifier (∀) का अर्थ है कि सूत्र डोमेन की हर वस्तु के लिए सही होना चाहिए। Existential Quantifier (∃) का अर्थ है कि डोमेन में कम से कम एक वस्तु के लिए सूत्र सही होना चाहिए।

उपयोग (Applications)

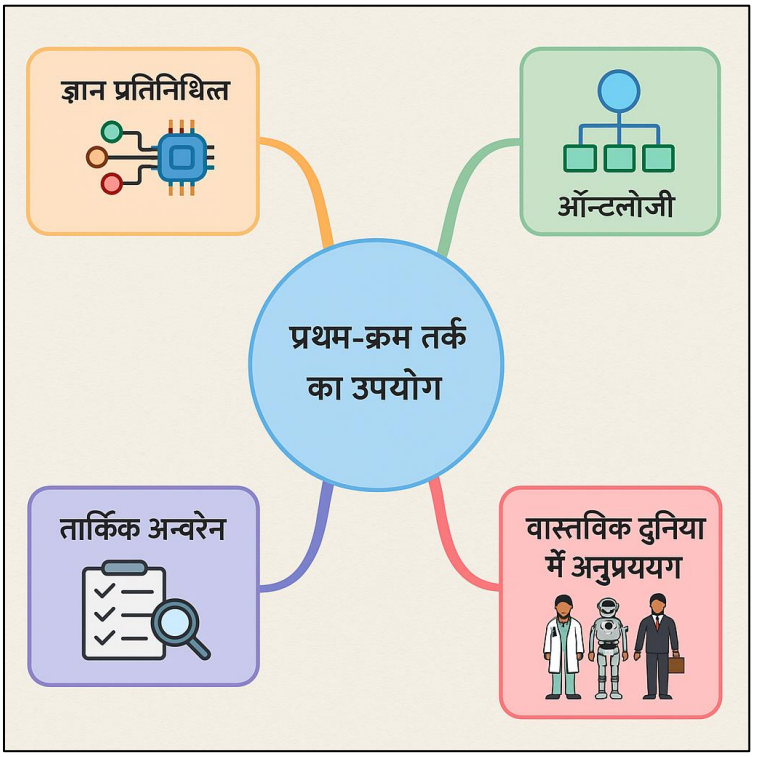
FOL का वाक्यविन्यास और अर्थवत्ता निम्न क्षेत्रों में प्रमुखता से उपयोग होता है:

- **स्वचालित तर्क (Automated Reasoning)**
- **ज्ञान प्रतिनिधित्व (Knowledge Representation)**
- **प्राकृतिक भाषा समझ (Natural Language Understanding)**
- **तार्किक निष्कर्षण (Logical Inference)**

FOL एक मजबूत तर्क प्रणाली है जो कृत्रिम बुद्धिमत्ता को वस्तुओं, उनके गुणों और उनके आपसी संबंधों के बारे में सटीक निष्कर्ष निकालने की क्षमता प्रदान करती है।

3.9 प्रथम-क्रम तर्क (First-Order Logic) का उपयोग

3.9.1 प्रथम-क्रम तर्क के साथ ज्ञान प्रतिनिधित्व की मूल बातें



चित्र 3.9.1 प्रथम-क्रम तर्क (FOL) के उपयोग के प्रमुख पहलुओं को दर्शाने वाला मस्तिष्क-मानचित्र आरेख

परिभाषा और महत्त्व

ज्ञान प्रतिनिधित्व कृत्रिम बुद्धिमत्ता (AI) का एक आधारभूत क्षेत्र है जो दुनिया की जानकारी को औपचारिक रूप से इस प्रकार संरचित करता है कि कंप्यूटर उसे समझ सके, संसाधित कर सके और मानव जैसी प्रतिक्रिया दे सके। यह इकाइयों, अवधारणाओं, घटनाओं और उनके आपसी संबंधों के बारे में ज्ञान को एक ऐसे रूप में संरचित करता है जिसे मशीन तर्क करके उपयोग में ला सके। प्रथम-क्रम तर्क (FOL) इस प्रक्रिया में एक केंद्रीय भूमिका निभाता है क्योंकि यह इकाइयों, उनके गुणों और संबंधों को सटीक रूप से दर्शाने में सक्षम होता है।

प्रथम-क्रम तर्क में क्वांटीफ़ायर (quantifiers) और वेरिएबल्स (variables) का समावेश होता है जो इसे प्रोपोज़िशनल लॉजिक की तुलना में कहीं अधिक अभिव्यक्तिपूर्ण बनाते हैं, जिससे प्राकृतिक भाषा की समझ, तर्क-स्वचालन, और जटिल निर्णय-निर्माण जैसी कार्यों को करने में सहायक होता है।

FOL की अभिव्यक्तिपूर्णता

FOL की अभिव्यक्ति क्षमता प्रोपोज़िशनल लॉजिक से काफी अधिक है। जहाँ प्रोपोज़िशनल लॉजिक केवल सत्य या असत्य वाक्यों तक सीमित है, वहीं FOL में हम "सभी मनुष्य नश्वर हैं" या "कोई ऐसा व्यक्ति है जो पाँच से अधिक भाषाएँ बोल सकता है" जैसे जटिल कथनों को भी दर्शा सकते हैं। यह वास्तविक दुनिया के मॉडलिंग में अत्यधिक सहायक है।

3.9.2 FOL में ज्ञान संरचना

वस्तुओं और संबंधों का प्रतिनिधित्व

FOL में वस्तुओं, उनके गुणों और आपसी संबंधों को दर्शाने के लिए कॉन्स्टेंट्स (constants), वेरिएबल्स (variables), प्रेडिकेट्स (predicates) और फंक्शन्स (functions) का प्रयोग किया जाता है। उदाहरण: "Paris" एक कॉन्स्टेंट है, "is a city" एक प्रेडिकेट है। "capital of" एक फंक्शन हो सकता है जो देश को उसकी राजधानी से जोड़ता है।

क्वांटीफ़ायर और उनकी भूमिका

क्वांटीफ़ायर FOL की शक्ति को कई गुना बढ़ाते हैं:

- **सर्वसमिक क्वांटीफ़ायर (∀)**: "सभी के लिए" — जैसे "सभी पक्षी उड़ सकते हैं"।
- **अस्तित्व क्वांटीफ़ायर (∃)**: "कम से कम एक के लिए" — जैसे "एक ऐसा पक्षी है जो उड़ नहीं सकता"।

ये क्वांटीफ़ायर AI को सामान्य सिद्धांत और विशिष्ट घटनाओं दोनों के आधार पर सोचने में सक्षम बनाते हैं।

3.9.3 FOL में तर्क (Inference)

तर्क तंत्र

FOL में तर्क तंत्र जैसे:

- **Modus Ponens**: यदि $p \rightarrow q$ और p सत्य है, तो q भी सत्य होगा।
- **Universal Instantiation**: सार्वत्रिक कथन को किसी विशेष मामले पर लागू करना।

- **Existential Generalization**: किसी विशेष उदाहरण से अस्तित्व वाक्य निष्कर्ष करना।

ये युक्तियाँ AI को पूर्व जानकारी के आधार पर निष्कर्ष निकालने, समस्या हल करने और भविष्यवाणी करने में मदद करती हैं।

तर्क से जुड़ी चुनौतियाँ

FOL में तर्क करना जटिल और संसाधन-गहन प्रक्रिया हो सकती है। ज्ञान आधार बड़ा होने पर निष्कर्ष निकालना धीमा और कठिन हो जाता है। इसके अलावा, कुछ मामलों में यह तय करना कठिन होता है कि कोई कथन सत्य है या नहीं। इन समस्याओं से निपटने के लिए कुशल एल्गोरिदम और अनुकूलन विधियों की आवश्यकता होती है।

3.9.4 ऑण्टोलॉजी और FOL

ऑण्टोलॉजी विकास

FOL का प्रयोग डोमेन-विशिष्ट ज्ञान के औपचारिक मॉडल (ऑण्टोलॉजी) के विकास में किया जाता है। ये मॉडल किसी क्षेत्र के अवधारणाओं और उनके बीच के संबंधों को स्पष्ट रूप से परिभाषित करते हैं। FOL की मदद से ऑण्टोलॉजी में जटिल संबंधों और प्रतिबंधों को भी सटीकता से व्यक्त किया जा सकता है, जिससे विभिन्न प्रणालियों और उपयोगकर्ताओं के बीच साझा समझ और इंटरऑपरेबिलिटी सुनिश्चित की जाती है।

3.9.5 प्रथम-क्रम तर्क (FOL) के साथ ज्ञान प्रतिनिधित्व में केस स्टडीज़

ऑण्टोलॉजी के अनुप्रयोग

प्रथम-क्रम तर्क (FOL) द्वारा समर्थित ऑण्टोलॉजी का व्यापक उपयोग सेमांटिक वेब तकनीकों, डेटा एकीकरण, और सूचना पुनःप्राप्ति प्रणालियों में किया जाता है।

- **सेमांटिक वेब** में, ऑण्टोलॉजी मशीनों को मानव-स्तरीय जटिल अनुरोधों को समझने और उस पर प्रतिक्रिया देने में मदद करती है, क्योंकि यह एक संरचित और सार्वभाषित शब्दावली प्रदान करती है।
- **डेटा एकीकरण** में, यह विविध स्रोतों से आए डेटा को एक सामान्य संदर्भ मॉडल में संरेखित करने में सहायता करती है।
- **सूचना पुनःप्राप्ति प्रणाली** ऑण्टोलॉजी का उपयोग शब्दों के बीच संबंधों और संदर्भ को समझकर खोज की सटीकता और प्रासंगिकता को बढ़ाने में करती है।

वास्तविक जीवन में अनुप्रयोग (Real-World Applications)

स्वास्थ्य सेवा (Healthcare):

स्वास्थ्य क्षेत्र में, FOL-आधारित प्रणालियाँ जटिल चिकित्सीय ज्ञान को मॉडल करने के लिए महत्वपूर्ण हैं। ये सिस्टम लक्षण, परीक्षण परिणाम, रोगी इतिहास और उपचार परिणाम जैसे विशाल डेटा को एकीकृत करके निदान और उपचार योजना बनाने में सहायता करते हैं। उदाहरण के लिए, एक FOL-आधारित निदान प्रणाली "बुखार" और "खाँसी" जैसे लक्षणों के साथ "इन्फ्लुएंज़ा टेस्ट पॉजिटिव" होने पर "फ्लू" के संभावित निदान का अनुमान लगा सकती है।

वित्तीय क्षेत्र (Finance):

वित्तीय उद्योग में, FOL का उपयोग जोखिम मूल्यांकन मॉडलों में किया जाता है जो निवेश की सफलता या जोखिम का विश्लेषण करते हैं। उदाहरण के लिए, ब्याज दरें, मुद्रास्फीति, भू-राजनीतिक स्थिरता, और कंपनी के प्रदर्शन को FOL में मॉडल करके निवेश रणनीतियों का सुझाव दिया जा सकता है।

रोबोटिक्स (Robotics):

रोबोटिक्स में, FOL का उपयोग भौतिक दुनिया को दर्शाने और उस पर तर्क करने के लिए किया जाता है — जैसे वस्तुओं के बीच स्थानिक संबंध, पथ नियोजन और वस्तुओं के साथ संपर्क। उदाहरण के लिए, एक रोबोट किसी कमरे में वस्तु लाने का कार्य कर रहा हो तो वह FOL का उपयोग कर यह तय कर सकता है कि कौन-सी वस्तु कहाँ है, उसकी विशेषताएँ क्या हैं (आकार, भार), और उसे कैसे उठाया जाए।

प्रमुख शिक्षाएँ और सर्वोत्तम प्रथाएँ (Lessons Learned and Best Practices)

- **अभिव्यक्ति और संगणनात्मक दक्षता के बीच संतुलन:**
  FOL की शक्ति इसकी अभिव्यक्तिपूर्णता में है, लेकिन इसे संगणनात्मक दक्षता के साथ संतुलित करना आवश्यक है, क्योंकि बड़े ज्ञान आधार पर काम करना भारी संसाधन ले सकता है।
- **तर्क सीमा का नियंत्रण:**
  निर्णय लेने और संसाधन के प्रभावी प्रबंधन हेतु केवल उन ज्ञान क्षेत्रों तक तर्क को सीमित रखना जिनकी वास्तव में आवश्यकता है।

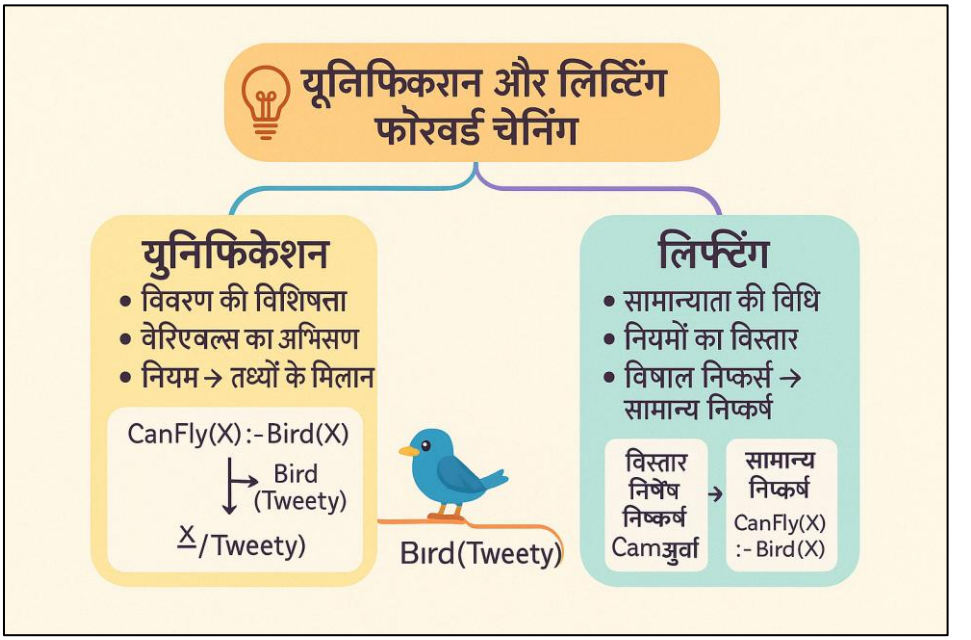
- **कुशल एल्गोरिदम का प्रयोग:**
  FOL पर आधारित तर्क प्रणालियों के प्रदर्शन को बेहतर बनाने के लिए अनुकूलित एल्गोरिदम अपनाना।
- **अनुमान आधारित तकनीकों का उपयोग:**
  जहाँ सटीक तर्क अव्यवहारिक हो, वहाँ अनुमानात्मक तरीके अपनाकर तेज़ और पर्याप्त सटीक परिणाम प्राप्त किए जा सकते हैं।
- **मॉड्यूलर ऑण्टोलॉजी का विकास:**
  ज्ञान आधार को छोटे-छोटे मॉड्यूल में विभाजित करने से रख-रखाव आसान होता है और तर्क की प्रक्रिया तेज़ हो जाती है।
- **क्वांटीफायर का विवेकपूर्ण उपयोग:**
  सर्वसमिक (∀) और अस्तित्वात्मक (∃) क्वांटीफायर की शक्ति के साथ जटिलता भी आती है। इनका विवेकपूर्ण और संरचित प्रयोग आवश्यक है।

निष्कर्ष (Conclusion)

यह संरचना न केवल प्रथम-क्रम तर्क के सैद्धांतिक आधार और व्यावहारिक अनुप्रयोगों को प्रस्तुत करती है, बल्कि इसके कार्यान्वयन में आने वाली चुनौतियों और विचारों को भी उजागर करती है। इन उपविषयों के माध्यम से पाठक यह समझ सकते हैं कि FOL कैसे जटिल ज्ञान आधारों पर आधारित बुद्धिमान प्रणालियों को विकसित करने में एक महत्वपूर्ण भूमिका निभाता है, जो उन्नत तर्क और निर्णय-निर्माण में सक्षम होती हैं।

3.10 यूनिफिकेशन और लिफ्टिंग फॉरवर्ड चेनिंग

फॉरवर्ड चेनिंग एक नियम-आधारित अनुमान विधि है जिसका उपयोग लॉजिक प्रोग्रामिंग, विशेषज्ञ प्रणालियों और विभिन्न कृत्रिम बुद्धिमत्ता (AI) अनुप्रयोगों में ज्ञात तथ्यों और नियमों के एक सेट से निष्कर्ष निकालने के लिए किया जाता है। यह विधि तब तक नियम लागू करती है जब तक संभव हो, नए ज्ञान का निर्माण करती है, जब तक कोई लक्ष्य प्राप्त नहीं हो जाता या कोई और नियम लागू नहीं हो सकता। यूनिफिकेशन और लिफ्टिंग दो महत्वपूर्ण अवधारणाएँ हैं जो विशेष रूप से फ़र्स्ट-ऑर्डर लॉजिक (FOL) में फ़ॉरवर्ड चेनिंग और ऑटोमेटेड रीजनिंग में उपयोग की जाती हैं।



चित्र 3.10.1 अग्रगामी श्रृंखला के संदर्भ में एकीकरण और उन्नयन की संकल्पनाएँ

3.10.1 यूनिफिकेशन (Unification)

यूनिफिकेशन वह प्रक्रिया है जो दो तार्किक अभिव्यक्तियों को समान बनाने के लिए उनमें मौजूद वेरिएबल्स के लिए उपयुक्त प्रतिस्थापन (substitution) खोजती है। यह FOL और फॉरवर्ड चेनिंग में तथ्यों और नियमों को लक्ष्य या क्वेरी के साथ मिलाने में उपयोग होता है।

- **परिभाषा और भूमिका:** यूनिफिकेशन लॉजिक-आधारित इंफेरेंस सिस्टम का मूल है, जो नियमों के लचीले अनुप्रयोग को संभव बनाता है। यह यह निर्धारित करता है कि कैसे नियमों में वेरिएबल्स को तथ्यों के साथ मेल कराया जा सकता है।
- **प्रक्रिया:** यूनिफिकेशन में ऐसा प्रतिस्थापन ढूंढा जाता है जो दो टर्म्स को समान बना सके। यदि ऐसा संभव हो, तो टर्म्स "यूनिफाएबल" कहलाते हैं, और उस प्रतिस्थापन का उपयोग अन्य नियमों में भी किया जा सकता है।
- **उदाहरण:**

  नियम: CanFly(X) :- Bird(X)

  तथ्य: Bird(Tweety)

  यूनिफिकेशन के बाद उपयुक्त प्रतिस्थापन {X/Tweety} से निष्कर्ष निकलेगा:

  CanFly(Tweety)

3.10.2 लिफ्टिंग (Lifting)

लिफ्टिंग एक सामान्यीकरण तकनीक है जो किसी नियम को विशिष्ट उदाहरणों से विस्तारित कर एक सामान्य रूप में लागू करने की अनुमति देती है। फॉरवर्ड चेनिंग में, यह तकनीक नियमों को व्यापक स्थितियों पर लागू करने में मदद करती है।

- **परिभाषा और महत्व:** लिफ्टिंग नियमों को केवल किसी विशेष उदाहरण तक सीमित न रखकर वेरिएबल्स और क्वांटिफायर्स के माध्यम से उन्हें किसी भी उपयुक्त उदाहरण पर लागू करने में सक्षम बनाती है।
- **प्रक्रिया:** इसमें किसी विशेष निष्कर्ष को एक सामान्य रूप में बदला जाता है जो किसी भी मिलते-जुलते उदाहरण पर लागू हो सके। यह FOL में वेरिएबल्स और क्वांटिफायर का उपयोग करके किया जाता है।
- **उदाहरण:**

  यदि CanFly(Tweety) को Bird(Tweety) से निष्कर्षित किया गया है, तो लिफ्टिंग इसे सामान्य रूप में CanFly(X) :- Bird(X) में परिवर्तित कर देती है, जिससे यह नियम किसी भी पक्षी पर लागू हो सके।

फॉरवर्ड चेनिंग में अनुप्रयोग

फॉरवर्ड चेनिंग में, **यूनिफिकेशन** और **लिफ्टिंग** मिलकर नियमों को प्रभावी ढंग से लागू करने में सहायता करते हैं:

- **यूनिफिकेशन** तथ्य और नियमों के बीच मिलान करता है।
- **लिफ्टिंग** परिणामों को सामान्यीकृत करता है ताकि वे भविष्य के लिए भी उपयोगी हो सकें।

3.11 पिछड़ी श्रृंखला (Backward Chaining)

3.11.1 पिछड़ी श्रृंखला का परिचय

पिछड़ी श्रृंखला कृत्रिम बुद्धिमत्ता (AI) में उपयोग की जाने वाली एक तर्क रणनीति है, जो किसी लक्ष्य से प्रारंभ कर तथ्यों तक पहुँचने का कार्य करती है। यह विधि विशेष रूप से नियम-आधारित प्रणालियों, विशेषज्ञ प्रणालियों, और कुछ प्रकार के लॉजिक प्रोग्रामिंग में उपयोगी होती है। इसमें ज्ञात लक्ष्य को सिद्ध करने के लिए पीछे की दिशा में कारणों और नियमों की खोज की जाती है। यह लक्ष्य-चालित (goal-driven) दृष्टिकोण होता है, जबकि अग्रगामी श्रृंखला (forward chaining) डेटा-चालित (data-driven) होती है।

उदाहरण:

यदि किसी निदान प्रणाली में लक्ष्य है: "क्या मरीज को मधुमेह है?", तो पिछड़ी श्रृंखला इस निष्कर्ष से आरंभ करेगी और नियमों के माध्यम से यह पता लगाएगी कि क्या कोई लक्षण या जांच परिणाम इस निदान का समर्थन करता है।

3.11.2 पिछड़ी श्रृंखला कैसे कार्य करती है

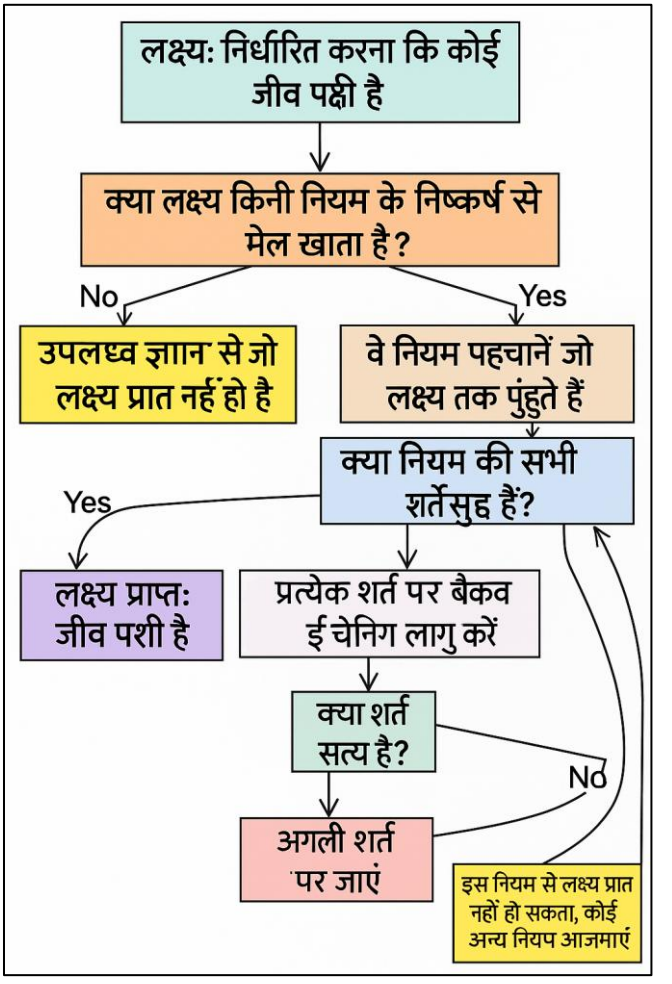
पिछड़ी श्रृंखला IF-THEN नियमों के रूप में अभिव्यक्त तर्क नियमों का उपयोग करती है। इसके चरण निम्नलिखित हैं:

1. **लक्ष्य की पहचान करें** – उस निष्कर्ष या लक्ष्य को निर्धारित करें जिसे सिद्ध करना है।
2. **लक्ष्य को नियमों से मिलाएं** – उन नियमों की खोज करें जिनका निष्कर्ष वर्तमान लक्ष्य है।
3. **पूर्वशर्तों की पहचान करें** – उन नियमों की शर्तों की जांच करें जिन्हें पूरा करना आवश्यक है।
4. **पुनरावृत्त रूप से पिछड़ी श्रृंखला लागू करें** – प्रत्येक पूर्वशर्त को एक नया लक्ष्य मानकर पुनः प्रक्रिया दोहराएं।
5. **लक्ष्य की प्राप्ति** – यदि सभी शर्तें सत्य पाई जाती हैं, तो लक्ष्य सिद्ध होता है। अन्यथा लक्ष्य को उपलब्ध ज्ञान के आधार पर सिद्ध नहीं किया जा सकता।

एल्गोरिद्म:

```
function backwardChaining(goal, rules):
  if goal पहले से ज्ञात है:
```

```
    return True
for each rule in rules:
    if rule का निष्कर्ष goal है:
        if backwardChaining(सभी शर्तें, rules) सत्य हो:
            return True
return False
```



चित्र 3.11.1: एआई में पीछे की ओर श्रृंखला प्रक्रिया को दर्शाने वाला आरेख, विशेष रूप से यह निर्धारित करने के संदर्भ में कि कोई जानवर पक्षी है या नहीं

उदाहरण:

• IF जानवर के पास पंख हैं THEN वह पक्षी है।

• IF जानवर उड़ता है AND अंडे देता है THEN वह पक्षी है।

लक्ष्य "जानवर एक पक्षी है" को सिद्ध करने के लिए, प्रणाली इस लक्ष्य से पीछे की ओर काम करेगी और देखेगी कि क्या "पंख हैं", "उड़ता है" और "अंडे देता है" जैसे तथ्य ज्ञात हैं।

3.11.3 पिछड़ी श्रृंखला के अनुप्रयोग

पिछड़ी श्रृंखला का उपयोग विभिन्न क्षेत्रों में किया जाता है:

• **चिकित्सीय निदान:** लक्षणों के आधार पर बीमारियों की पहचान करना।

• **समस्या निवारण प्रणाली:** सॉफ्टवेयर या हार्डवेयर समस्याओं के मूल कारण का पता लगाना।

• **कानूनी तर्क:** कानूनी तथ्यों और नियमों से निष्कर्ष निकालना।

3.11.4 लाभ और सीमाएँ (Advantages and Limitations)

लाभ (Advantages):

• यह उन समस्याओं के लिए प्रभावी है जहाँ लक्ष्य स्पष्ट होता है लेकिन उसे प्राप्त करने का मार्ग स्पष्ट नहीं होता।

• यह केवल लक्ष्य से संबंधित जानकारी पर ध्यान केंद्रित करके खोज स्थान (search space) को कम कर देता है।

सीमाएँ (Limitations):

• यदि लक्ष्य तक पहुँचने के लिए कई संभावित मार्ग हों या लक्ष्य स्पष्ट रूप से परिभाषित न हो, तो यह विधि कुशल नहीं होती।

• प्रभावी होने के लिए नियमों का एक व्यापक और सुस्पष्ट सेट आवश्यक होता है।

“Artificial intelligence is not a substitute for human intelligence, but a tool to amplify it.”

"कृत्रिम बुद्धिमत्ता मानव बुद्धि का विकल्प नहीं, बल्कि उसे बढ़ाने का एक उपकरण है।"

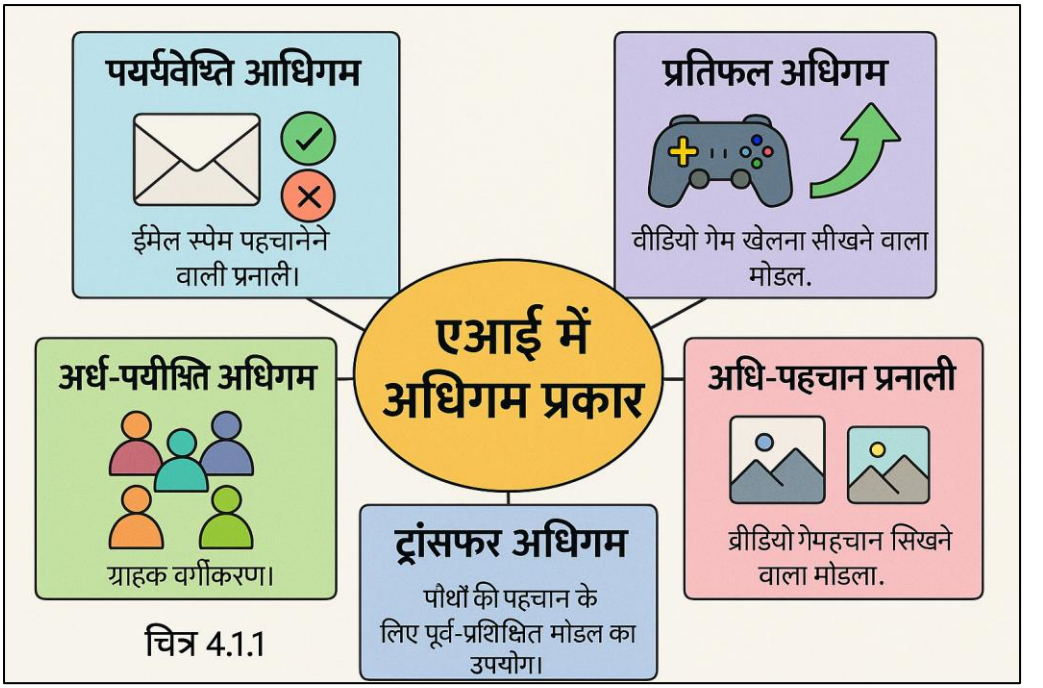
“Logic will get you from A to B. Imagination will take you everywhere.” – Albert Einstein

"तर्क आपको A से B तक ले जाएगा, लेकिन कल्पना आपको हर जगह पहुँचा सकती है।" – अल्बर्ट आइंस्टीन

एकक-4: अधिगम (Learning)

4.1 अधिगम के प्रकार

कृत्रिम बुद्धिमत्ता (AI) में अधिगम एक मौलिक अवधारणा है जो सिस्टम को डेटा के आधार पर अनुकूलन करने, सुधार करने और निर्णय लेने में सक्षम बनाती है। AI में अधिगम को कई प्रकारों में वर्गीकृत किया जा सकता है, जिनमें से प्रत्येक की अपनी विशिष्ट विशेषताएं, कार्यप्रणालियाँ और अनुप्रयोग होते हैं।



चित्र 4.1.1 कृत्रिम बुद्धिमत्ता (AI) में विभिन्न प्रकार की शिक्षण विधियों को दर्शाने वाला आरेख, जिसमें पर्यवेक्षित शिक्षण, अप्रेरित शिक्षण, प्रतिफल शिक्षण, अर्ध-पर्यवेक्षित शिक्षण और स्वयं-पर्यवेक्षित शिक्षण शामिल हैं।

4.1.1 पर्यवेक्षित अधिगम (Supervised Learning)

पर्यवेक्षित अधिगम एक प्रकार का मशीन लर्निंग है जहाँ मॉडल को एक *लेबलयुक्त डेटा सेट* पर प्रशिक्षित किया जाता है। इसका मतलब है कि प्रत्येक प्रशिक्षण उदाहरण के साथ एक आउटपुट लेबल होता है।

मॉडल इनपुट डेटा से आउटपुट की भविष्यवाणी करना सीखता है और नए डेटा पर भविष्यवाणी की सटीकता के आधार पर मूल्यांकन किया जाता है।

उदाहरण:

ईमेल स्पैम पहचान प्रणाली। मॉडल को ऐसे ईमेल्स पर प्रशिक्षित किया जाता है जिन्हें "स्पैम" या "नॉन-स्पैम" के रूप में लेबल किया गया होता है।

एल्गोरिदम:

Support Vector Machine (SVM) एक प्रसिद्ध एल्गोरिदम है, जो विभिन्न वर्गों को अलग करने के लिए सर्वोत्तम हाइपरप्लेन खोजता है।

4.1.2 अपर्यवेक्षित अधिगम (Unsupervised Learning)

अपर्यवेक्षित अधिगम में मॉडल को बिना लेबल वाले डेटा पर प्रशिक्षित किया जाता है। इसका उद्देश्य डेटा में पैटर्न और संबंधों की पहचान करना होता है।

उदाहरण:

क्लस्टरिंग, जैसे मार्केटिंग में ग्राहक विभाजन, जहाँ ग्राहक उनके खरीद व्यवहार के आधार पर समूहित किए जाते हैं।

एल्गोरिदम:

K-means एल्गोरिदम व्यापक रूप से क्लस्टरिंग के लिए उपयोग किया जाता है।

4.1.3 प्रतिफल अधिगम (Reinforcement Learning)

इस प्रकार के अधिगम में एक एजेंट वातावरण में क्रियाएँ करके निर्णय लेना सीखता है, ताकि अधिकतम पुरस्कार (reward) प्राप्त किया जा सके। सही कार्यों के लिए एजेंट को पुरस्कार मिलता है और गलत कार्यों पर दंड।

उदाहरण:

वीडियो गेम खेलना सिखाने वाला मॉडल जो स्कोर के आधार पर निर्णय लेना सीखता है।

एल्गोरिदम:

Q-learning एक प्रसिद्ध तकनीक है, जिसमें एजेंट यह सीखता है कि किस परिस्थिति में कौन-सी क्रिया करनी चाहिए।

4.1.4 अर्ध-पर्यवेक्षित अधिगम (Semi-Supervised Learning)

यह अधिगम पर्यवेक्षित और अपर्यवेक्षित दोनों का मिश्रण है। इसमें थोड़ा लेबलयुक्त डेटा और अधिक मात्रा में बिना लेबल वाला डेटा होता है।

उदाहरण:

छवि पहचान प्रणाली, जहाँ थोड़ी संख्या में लेबलयुक्त और बड़ी संख्या में बिना लेबल वाली छवियाँ होती हैं।

4.1.5 ट्रांसफर अधिगम (Transfer Learning)

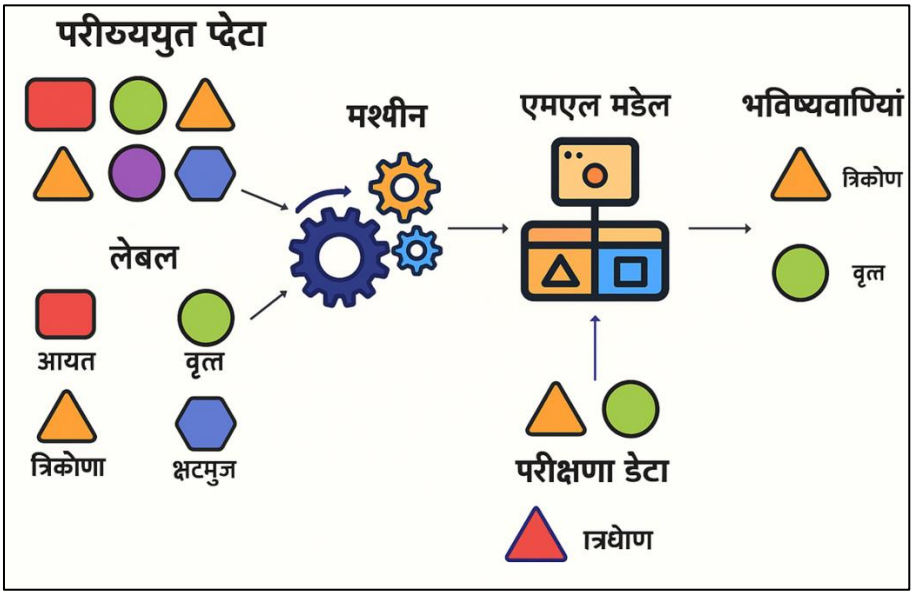
इसमें एक कार्य के लिए तैयार मॉडल का उपयोग किसी दूसरे कार्य के लिए प्रारंभिक बिंदु के रूप में किया जाता है, विशेष रूप से तब जब दूसरे कार्य के लिए प्रशिक्षण डेटा सीमित हो।

उदाहरण:

ImageNet पर प्रशिक्षित एक मॉडल का उपयोग पौधों की पहचान जैसे कार्य के लिए करना।

4.2 पर्यवेक्षित शिक्षण (Supervised Learning)

पर्यवेक्षित शिक्षण मशीन लर्निंग की एक आधारशिला है, जिसका लक्ष्य लेबल वाले प्रशिक्षण डेटा से एक फलन (function) निकालना होता है। प्रशिक्षण डेटा उदाहरणों के एक समूह से बना होता है, जिसमें प्रत्येक उदाहरण एक युग्म (pair) होता है — एक इनपुट वस्तु (आमतौर पर एक वेक्टर) और वांछित आउटपुट मान (जिसे सुपरवाइजरी सिग्नल भी कहा जाता है)। एक पर्यवेक्षित शिक्षण एल्गोरिद्म इन डेटा का विश्लेषण करता है और एक अनुमानित फलन उत्पन्न करता है, जिसे नए उदाहरणों के लिए मैपिंग करने में उपयोग किया जा सकता है।



चित्र 4.2.1 : पर्यवेक्षित शिक्षण को दर्शाने वाला आरेख

4.2.1 परिभाषा और मूल बातें

पर्यवेक्षित शिक्षण में प्रत्येक उदाहरण एक इनपुट वस्तु (आमतौर पर वेक्टर) और वांछित आउटपुट (लेबल) का एक युग्म होता है। एल्गोरिद्म ऐसा फलन सीखने का प्रयास करता है जो किसी भी दिए गए इनपुट के लिए सही आउटपुट की भविष्यवाणी कर सके। यह लक्ष्य प्रशिक्षण डेटा का विश्लेषण करके और भविष्यवाणियों तथा वास्तविक लेबल के बीच की त्रुटि को न्यूनतम करके प्राप्त किया जाता है।

उदाहरण:

एक स्पैम डिटेक्शन सिस्टम में, ईमेल को "स्पैम" या "नॉट स्पैम" के रूप में लेबल किया जाता है। यह लेबल किया गया डेटा एल्गोरिद्म को प्रशिक्षित करने के लिए उपयोग किया जाता है। एक बार प्रशिक्षित हो जाने पर, एल्गोरिद्म नए ईमेल को स्पैम या नॉट स्पैम के रूप में वर्गीकृत कर सकता है।

4.2.2 पर्यवेक्षित शिक्षण के प्रकार

आउटपुट प्रकार के आधार पर पर्यवेक्षित शिक्षण को दो श्रेणियों में विभाजित किया जा सकता है:

- **वर्गीकरण (Classification):** आउटपुट एक श्रेणी होती है, जैसे स्पैम डिटेक्शन में "स्पैम" या "नॉट स्पैम"।
- **रैग्रेशन (Regression):** आउटपुट एक वास्तविक मान (real value) होता है, जैसे किसी घर की विशेषताओं के आधार पर उसकी कीमत की भविष्यवाणी।

4.2.3 शिक्षण प्रक्रिया

पर्यवेक्षित शिक्षण प्रक्रिया में मुख्यतः निम्नलिखित चरण होते हैं:

1. **डेटा एकत्र करना और तैयार करना:**
   प्रासंगिक और पर्याप्त मात्रा में डेटा एकत्र किया जाता है, फिर उसे साफ़ किया जाता है और आवश्यकता अनुसार रूपांतरित किया जाता है।
2. **मॉडल चुनना:**
   ऐसा मॉडल चुना जाता है जो इनपुट और आउटपुट के बीच संबंध को अच्छी तरह समझ सके।
3. **मॉडल को प्रशिक्षित करना:**
   मॉडल को प्रशिक्षण डेटा दिया जाता है ताकि वह आउटपुट लेबल्स की भविष्यवाणी करना सीख सके।
4. **मॉडल का मूल्यांकन:**
   मॉडल को एक अलग परीक्षण डेटा पर जांचा जाता है, जो प्रशिक्षण के दौरान नहीं देखा गया था।
5. **पैरामीटर ट्यूनिंग और अनुकूलन:**

मॉडल की सटीकता बढ़ाने और ओवरफिटिंग को रोकने के लिए इसके पैरामीटर समायोजित किए जाते हैं।

4.2.4 पर्यवेक्षित शिक्षण में एल्गोरिद्म

पर्यवेक्षित शिक्षण एल्गोरिद्म मशीन लर्निंग अनुप्रयोगों का आधार हैं। ये साधारण लीनियर रैग्रेशन से लेकर जटिल डीप न्यूरल नेटवर्क तक हो सकते हैं। प्रत्येक एल्गोरिद्म की अपनी विशेषताएं होती हैं और वे विभिन्न प्रकार की समस्याओं के लिए उपयुक्त होते हैं।

लीनियर रैग्रेशन (Linear Regression):

एक सरल और लोकप्रिय सांख्यिकीय तकनीक जो एक निर्भर वेरिएबल और कई स्वतंत्र वेरिएबल्स के बीच रैखिक संबंध को मॉडल करती है।

अनुप्रयोग: रियल एस्टेट (घर की कीमतों की भविष्यवाणी), फाइनेंस (स्टॉक की कीमतें)।

लॉजिस्टिक रैग्रेशन (Logistic Regression):

हालांकि नाम में "रैग्रेशन" है, यह एल्गोरिद्म वर्गीकरण के लिए उपयोग किया जाता है। यह किसी वर्ग से संबंधित होने की संभावना का अनुमान लगाने के लिए लॉजिस्टिक फंक्शन का उपयोग करता है।

अनुप्रयोग: स्पैम डिटेक्शन, रोग निदान, ग्राहक छूट भविष्यवाणी।

सपोर्ट वेक्टर मशीन (SVM):

एक शक्तिशाली और बहुमुखी मॉडल जो रैखिक और गैर-रैखिक वर्गीकरण, रैग्रेशन और आउट्लायर डिटेक्शन कर सकता है। यह उच्च-आयामी स्थान में एक हाइपरप्लेन बनाता है जो दो वर्गों के बीच अधिकतम मार्जिन रखता है।

अनुप्रयोग: चेहरे की पहचान, हस्तलिखित पाठ की पहचान, छवि वर्गीकरण।

डिसीजन ट्री और रैंडम फॉरेस्ट:

डिसीजन ट्री वर्गीकरण और रैग्रेशन के लिए एक गैर-पैरामीट्रिक विधि है। यह डेटा को एक ट्री संरचना में विभाजित करता है।

रैंडम फॉरेस्ट एक एन्सेम्बल विधि है, जो कई डिसीजन ट्री बनाकर अंतिम निर्णय लेती है।

अनुप्रयोग: ग्राहक विभाजन, धोखाधड़ी पहचान, बाजार विश्लेषण।

न्यूरल नेटवर्क्स (Neural Networks):

विशेष रूप से डीप लर्निंग मॉडल, मानव मस्तिष्क की संरचना पर आधारित होते हैं और पैटर्न पहचानने में सक्षम होते हैं।

अनुप्रयोग: छवि और वाणी पहचान, प्राकृतिक भाषा प्रोसेसिंग, चिकित्सा निदान।

प्रत्येक एल्गोरिद्म की अपनी विशेषताएँ हैं और उन्हें समस्या, डेटा की प्रकृति, और अपेक्षित परिणाम के अनुसार चुना जाता है। मॉडल की सफलता इस चयन पर बहुत हद तक निर्भर करती है, और अक्सर इसका निर्धारण प्रयोग और क्रॉस-वैलिडेशन द्वारा किया जाता है।

4.2.5 पर्यवेक्षित शिक्षण में चुनौतियाँ और विचारणीय बिंदु

पर्यवेक्षित शिक्षण, यद्यपि अत्यंत प्रभावशाली है, लेकिन इसके साथ कुछ महत्वपूर्ण चुनौतियाँ और विचारणीय मुद्दे भी जुड़े होते हैं, जो मॉडल की कार्यक्षमता और उपयोगिता को गहराई से प्रभावित कर सकते हैं। इन चुनौतियों को समझना इन एल्गोरिद्म को प्रभावी रूप से लागू करने के लिए आवश्यक है।

ओवरफिटिंग (Overfitting):

जब कोई मॉडल प्रशिक्षण डेटा को अत्यधिक अच्छी तरह से सीख लेता है, तो वह उसमें मौजूद शोर (noise) और अपवादों (outliers) को भी पैटर्न समझ लेता है। इससे वह प्रशिक्षण डेटा पर अच्छा प्रदर्शन करता है, लेकिन नए और अनदेखे डेटा पर खराब।

निवारण रणनीतियाँ:

- **नियमितीकरण (Regularization):** L1 और L2 जैसी तकनीकों द्वारा कोएफिशिएंट्स पर पेनल्टी लगाकर मॉडल की जटिलता घटाना।
- **क्रॉस-वैलिडेशन:** जैसे k-fold validation, यह सुनिश्चित करता है कि मॉडल सामान्यीकरण में सक्षम है।
- **प्रूनिंग (Pruning):** डिसीजन ट्री में अनावश्यक शाखाओं को हटाना।
- **अधिक डेटा से प्रशिक्षण:** अधिक डेटा मॉडल को वास्तविक पैटर्न सीखने में मदद कर सकता है (हालांकि यह हमेशा संभव नहीं होता)।

अंडरफिटिंग (Underfitting):

जब मॉडल बहुत सरल होता है और डेटा की वास्तविक संरचना को नहीं समझ पाता, तब अंडरफिटिंग होती है। इससे मॉडल प्रशिक्षण और परीक्षण दोनों सेटों पर खराब प्रदर्शन करता है।

निवारण रणनीतियाँ:

- **मॉडल की जटिलता बढ़ाना:** जैसे उच्च डिग्री वाला पोलिनोमियल रिग्रेशन।
- **फीचर इंजीनियरिंग:** नए फीचर्स जोड़ना या मौजूदा को बदलना।
- **नियमितीकरण कम करना:** यदि उपयोग में है, तो इसकी तीव्रता को घटाना।

डेटा की गुणवत्ता (Data Quality):

उच्च गुणवत्ता और प्रासंगिक प्रशिक्षण डेटा पर्यवेक्षित शिक्षण की सफलता के लिए अत्यंत आवश्यक है। यदि डेटा अधूरा, असंगत या अप्रासंगिक हो, तो यह मॉडल को गुमराह कर सकता है।

सुधार रणनीतियाँ:

- **डेटा सफाई:** गलत मान, डुप्लिकेट्स और आउट्लायर्स को हटाना।
- **गायब डेटा को संभालना:** इम्प्यूटेशन (mean, median से भरना) या ऐसे एल्गोरिद्म का प्रयोग जो गायब मानों को संभाल सकते हैं।
- **फीचर चयन:** केवल प्रासंगिक फीचर्स को चुनना ताकि प्रदर्शन सुधरे और ओवरफिटिंग कम हो।

विचारणीय बिंदु (Considerations):

- **बायस-वैरिएंस संतुलन:** मॉडल के पूर्वग्रह (bias) और परिवर्तनशीलता (variance) के बीच सही संतुलन बनाना।
- **नैतिक चिंताएं:** यह सुनिश्चित करना कि मॉडल सामाजिक या डेटा आधारित पूर्वग्रहों को आगे न बढ़ाए, खासकर तब जब निर्णय मानव जीवन को प्रभावित करें।

इन सभी चुनौतियों का समाधान करने के लिए सही मॉडल का चुनाव, डेटा की उपयुक्त प्रोसेसिंग, और पैरामीटर ट्यूनिंग अत्यावश्यक है।

4.2.6 पर्यवेक्षित शिक्षण के वास्तविक जीवन अनुप्रयोग

पर्यवेक्षित शिक्षण ने अपनी लेबल वाले डेटा से सीखने और नए डेटा पर सटीक भविष्यवाणी करने की क्षमता के बल पर विभिन्न क्षेत्रों में व्यापक उपयोग पाया है:

छवि पहचान (Image Recognition):

तस्वीरों को पूर्वनिर्धारित श्रेणियों में वर्गीकृत करना।

उदाहरण: सोशल मीडिया प्लेटफॉर्म उपयोगकर्ताओं को स्वतः टैग करने के लिए छवि पहचान का उपयोग करते हैं।

वाक् पहचान (Speech Recognition):

बोली को पाठ में परिवर्तित करना।

उदाहरण: वर्चुअल असिस्टेंट (Siri, Alexa, Google Assistant) आवाज़ पहचान के माध्यम से कमांड समझते हैं।

वित्तीय पूर्वानुमान (Financial Forecasting):

स्टॉक की कीमतें, बाजार रुझान और आर्थिक संकेतकों की भविष्यवाणी करना। **उदाहरण:** हेज फंड और बैंक स्टॉक प्रदर्शन की भविष्यवाणी के लिए मशीन लर्निंग का उपयोग करते हैं।

चिकित्सा निदान (Medical Diagnosis):

मरीज की जानकारी से रोगों की पहचान और उपचार सुझाव देना। **उदाहरण:** MRI या CT स्कैन से कैंसर का पता लगाने में मॉडल सहायता करते हैं।

4.2.7 प्रभाव और भविष्य की दिशा

पर्यवेक्षित शिक्षण के अनुप्रयोग उपरोक्त उदाहरणों तक सीमित नहीं हैं — यह हमारे जीवन के लगभग हर क्षेत्र को छू चुका है।

- **रिटेल में:** ग्राहकों के व्यवहार के आधार पर उत्पाद अनुशंसा।
- **ऑटोमोबाइल उद्योग में:** स्वचालित वाहनों में ट्रैफिक संकेत, पैदल यात्री और अन्य वाहनों की पहचान।

जैसे-जैसे डेटा की उपलब्धता बढ़ रही है और गणनात्मक संसाधन अधिक सुलभ हो रहे हैं, पर्यवेक्षित शिक्षण की पहुंच और उपयोग और भी विस्तृत होते जा रहे हैं।

भविष्य में, यह तकनीक और अधिक सटीक, कुशल और अनुकूलित मॉडल उत्पन्न करेगी, जो जटिल समस्याओं को हल करने में सक्षम होंगे।

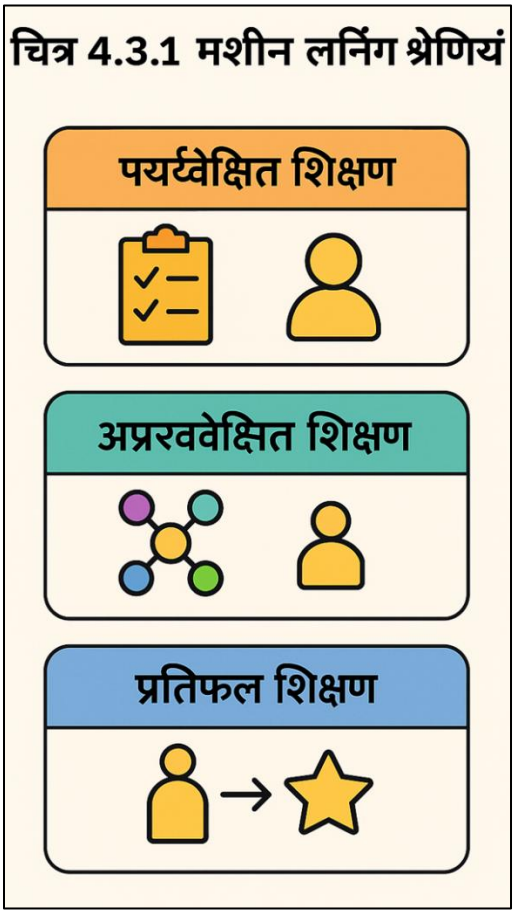
निष्कर्षतः, पर्यवेक्षित शिक्षण न केवल उद्योगों को बदल रहा है, बल्कि यह मानव जीवन को बेहतर, स्मार्ट और स्वचालित बनाने में एक केंद्रीय भूमिका निभा रहा है — और यह एआई-प्रेरित भविष्य की ओर एक निर्णायक कदम है।

4.3 मशीन लर्निंग

मशीन लर्निंग (ML) कृत्रिम बुद्धिमत्ता (AI) का एक उपवर्ग है जो प्रणालियों को स्पष्ट रूप से प्रोग्राम किए बिना अनुभव से स्वचालित रूप से सीखने और सुधार करने की क्षमता प्रदान करता है। इसका मुख्य उद्देश्य ऐसे कंप्यूटर प्रोग्राम विकसित करना है जो डेटा तक पहुँच प्राप्त कर सकें और स्वयं उससे सीख सकें।

सीखने की प्रक्रिया अवलोकनों या डेटा (जैसे उदाहरण, प्रत्यक्ष अनुभव या निर्देश) से शुरू होती है, ताकि डेटा में पैटर्न को पहचान कर भविष्य में बेहतर निर्णय लिए जा सकें। प्राथमिक उद्देश्य यह है कि कंप्यूटर स्वचालित रूप से सीख सकें, बिना मानवीय हस्तक्षेप या सहायता के, और अपने कार्यों को उसी अनुसार समायोजित कर सकें।

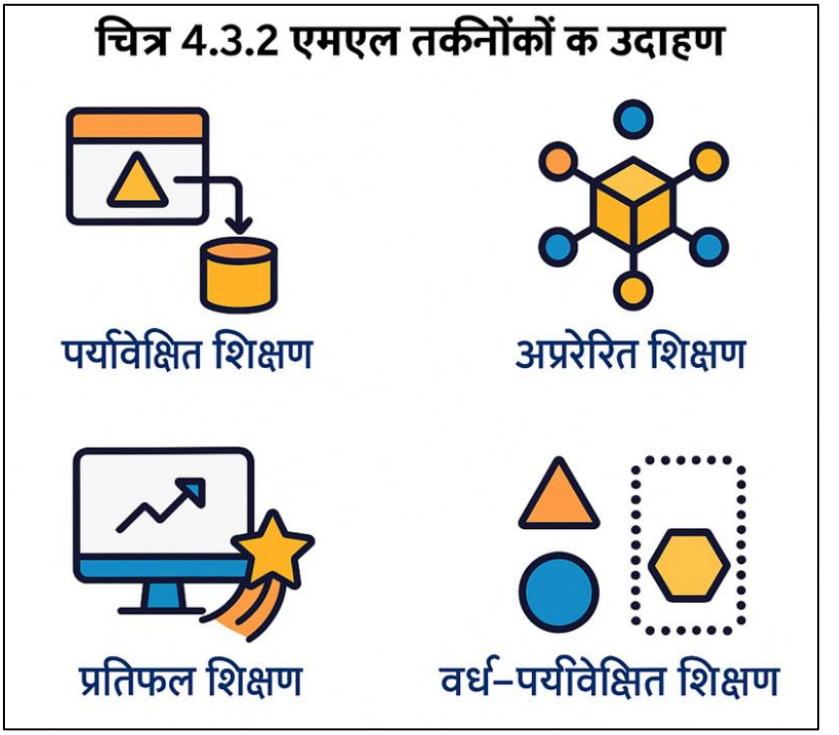
चित्र 4.3.1 मशीन लर्निंग श्रेणियं



4.3.1 परिभाषा और क्षेत्र

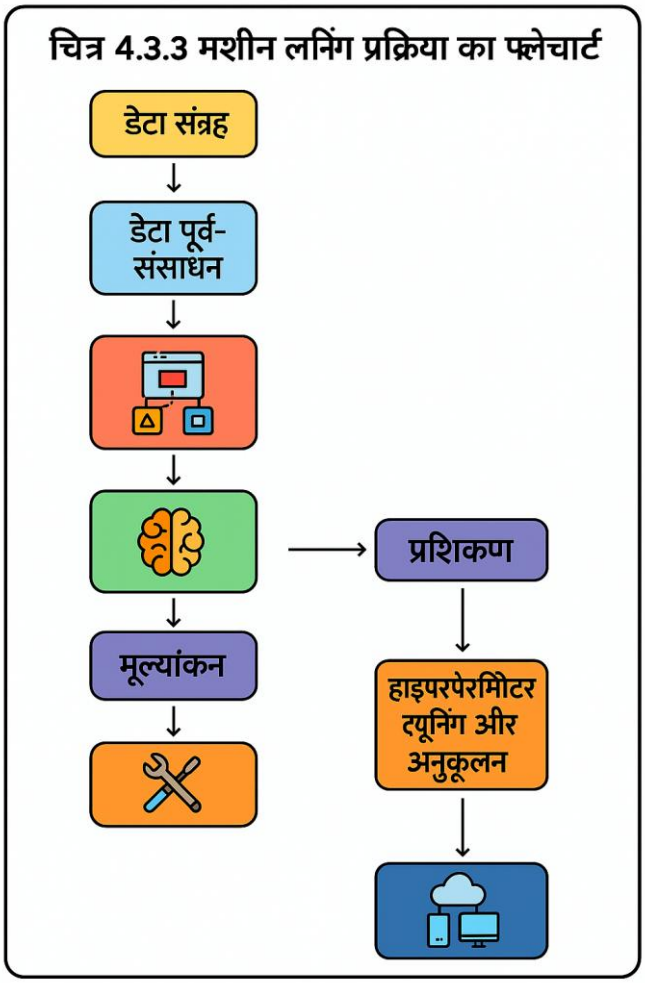
मशीन लर्निंग एल्गोरिद्म को आमतौर पर तीन प्रमुख वर्गों में विभाजित किया जाता है — पर्यवेक्षित शिक्षण, अप्रेरित शिक्षण, और प्रतिफल शिक्षण — जो इस बात पर आधारित हैं कि शिक्षण प्रणाली को किस प्रकार का फीडबैक या संकेत प्राप्त होता है।

इन एल्गोरिद्म का उपयोग ईमेल फ़िल्टरिंग से लेकर स्टॉक कीमतों की भविष्यवाणी और जेनेटिक अनुक्रमों में पैटर्न पहचानने जैसे अनेक अनुप्रयोगों में किया जाता है।



4.3.2 प्रमुख अवधारणाएं और तकनीकें

- **पर्यवेक्षित शिक्षण (Supervised Learning):** मॉडल को एक लेबलयुक्त डेटासेट पर प्रशिक्षित किया जाता है, यानी मॉडल पहले से उत्तर सहित डेटा से सीखता है।
- **अप्रेरित शिक्षण (Unsupervised Learning):** मॉडल बिना लेबल वाले डेटा पर कार्य करता है और इनपुट डेटा में पैटर्न और संबंध खोजता है।
- **प्रतिफल शिक्षण (Reinforcement Learning):** एक ऐसा शिक्षण जहाँ एजेंट पर्यावरण में कार्य करके और परिणाम देखकर सीखता है।



4.3.3 मशीन लर्निंग प्रक्रिया

मशीन लर्निंग प्रक्रिया एक व्यवस्थित ढांचा है जिसके माध्यम से मशीन लर्निंग मॉडल विकसित और परिनियोजित किए जाते हैं। यह डेटा संग्रहण से लेकर मॉडल को तैनात करने तक की कई महत्वपूर्ण चरणों से मिलकर बनी होती है:

1. डेटा संग्रहण

यह पहला और सबसे आवश्यक चरण है। इसमें उस समस्या से संबंधित पर्याप्त और प्रासंगिक डेटा इकट्ठा करना शामिल है, जिसे हल करना है। डेटा सार्वजनिक डेटासेट, कंपनी डेटाबेस, सेंसर या उपयोगकर्ताओं द्वारा उत्पन्न सामग्री से आ सकता है।

2. डेटा पूर्वप्रसंस्करण

संग्रहित डेटा को विश्लेषण योग्य बनाने के लिए उसे साफ और रूपांतरित किया जाता है। इसमें गायब मानों को संभालना, आउट्लायर हटाना, असंगतियों को ठीक करना, नॉर्मलाइज़ेशन, स्केलिंग और श्रेणीबद्ध मानों को संख्यात्मक रूप में बदलना शामिल होता है।

3. मॉडल चयन

समस्या की प्रकृति (जैसे वर्गीकरण, रिग्रेशन, क्लस्टरिंग आदि), डेटासेट का आकार और प्रकार, तथा वांछित परिणाम के आधार पर उपयुक्त एल्गोरिद्म का चयन किया जाता है। उदाहरण के लिए, रिग्रेशन के लिए लीनियर रिग्रेशन, वर्गीकरण के लिए लॉजिस्टिक रिग्रेशन और SVM, तथा बड़े और जटिल समस्याओं के लिए न्यूरल नेटवर्क।

4. प्रशिक्षण (Training)

चयनित एल्गोरिद्म को डेटा से प्रशिक्षित किया जाता है ताकि वह उसमें से पैटर्न और संबंध सीख सके। मॉडल अपने पैरामीटर को इस प्रकार समायोजित करता है कि भविष्यवाणी और वास्तविक परिणामों के बीच का अंतर न्यूनतम हो जाए।

5. मूल्यांकन (Evaluation)

प्रशिक्षण के बाद, मॉडल का मूल्यांकन एक अलग परीक्षण डेटासेट पर किया जाता है जो मॉडल ने पहले नहीं देखा होता।

मूल्यांकन मेट्रिक्स में सटीकता (Accuracy), प्रेसिजन, रिकॉल, F1 स्कोर (वर्गीकरण के लिए) तथा MSE और MAE (रिग्रेशन के लिए) शामिल हैं।

6. हाइपरपैरामीटर ट्यूनिंग और अनुकूलन

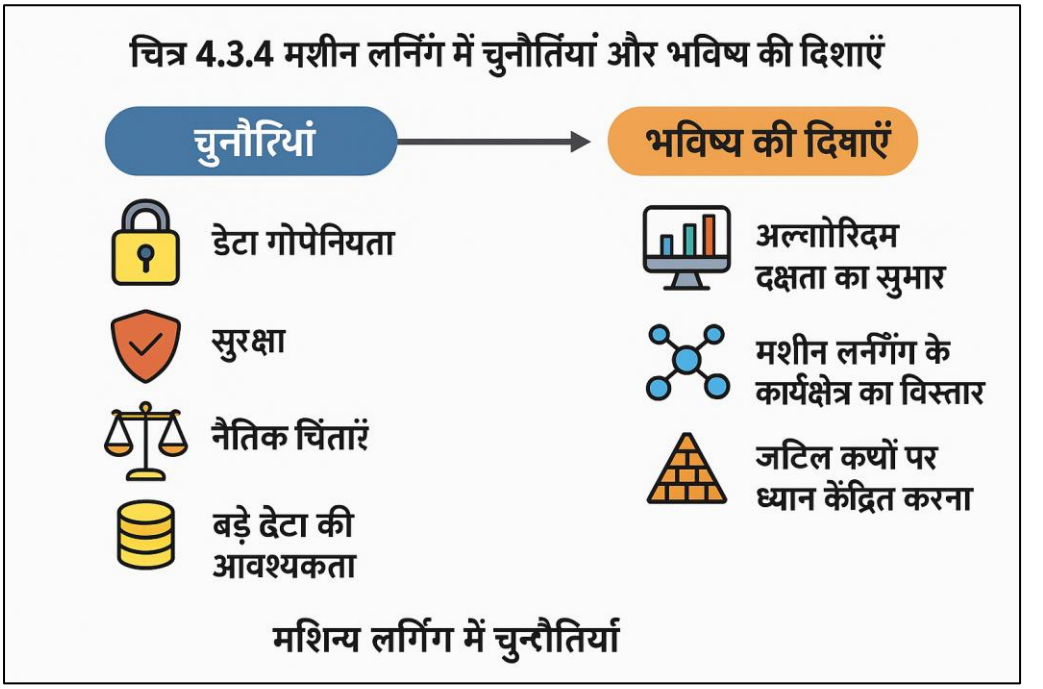
मॉडल की दक्षता को बढ़ाने के लिए उसके हाइपरपैरामीटर को ट्यून किया जाता है। इसके लिए ग्रिड सर्च, रैंडम सर्च और बेज़ियन ऑप्टिमाइज़ेशन जैसी तकनीकों का उपयोग किया जाता है।

7. परिनियोजन (Deployment)

यह अंतिम चरण है जिसमें प्रशिक्षित मॉडल को वास्तविक अनुप्रयोगों में लागू किया जाता है। इसे प्रोडक्शन वातावरण में एकीकृत किया जाता है ताकि वह नए डेटा पर निर्णय या भविष्यवाणी कर सके।

मशीन लर्निंग प्रक्रिया एक चक्रीय प्रक्रिया होती है। अंतिम चरणों से प्राप्त फीडबैक के आधार पर कई बार पूर्व के चरणों (जैसे डेटा संशोधन या मॉडल ट्यूनिंग) पर लौटना आवश्यक हो सकता है।

यह पुनरावृत्त प्रकृति समाधान की निरंतर सुधार और अनुकूलन की क्षमता सुनिश्चित करती है, जिससे वह बदलती आवश्यकताओं और चुनौतियों का सामना कर सके।

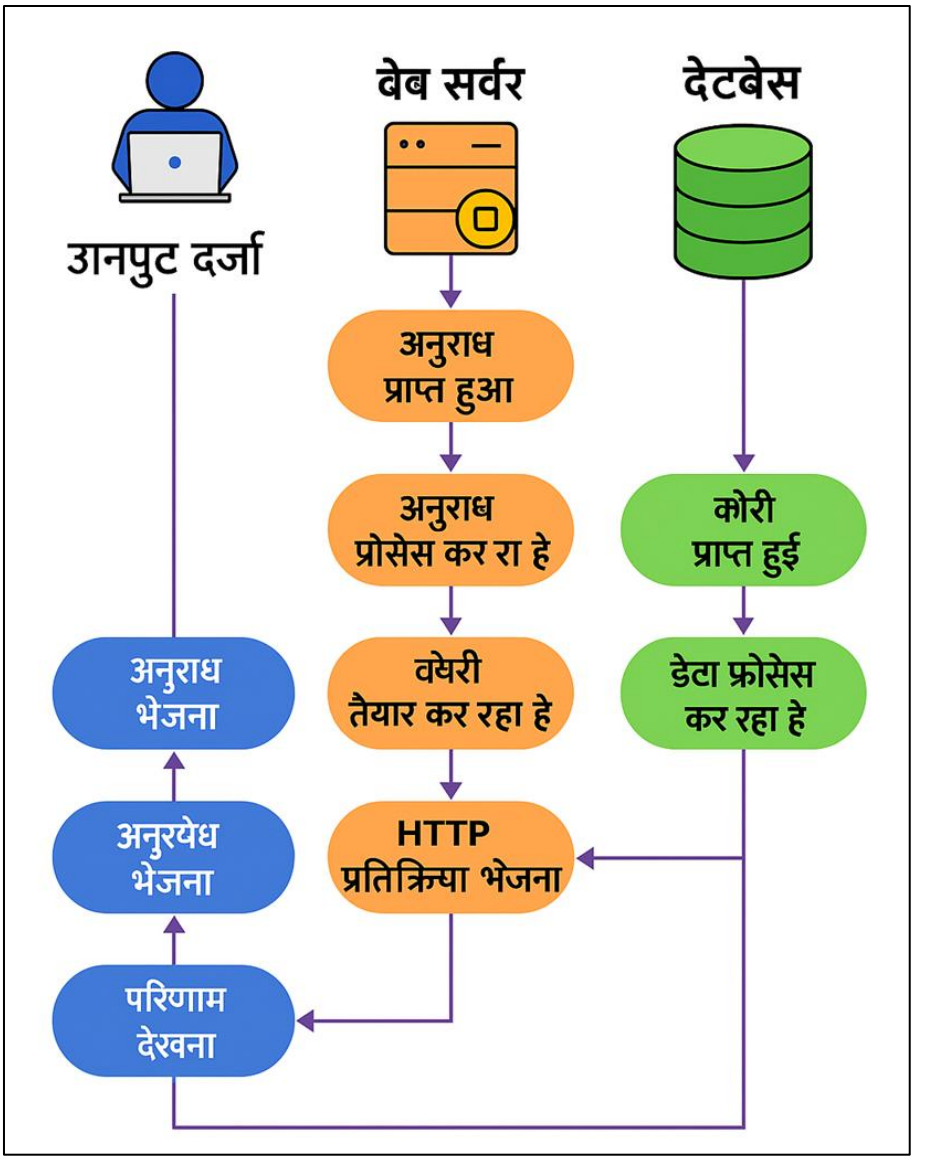


4.3.4 चुनौतियाँ और भविष्य की दिशा

हालाँकि मशीन लर्निंग अनेक लाभ प्रदान करती है, फिर भी यह कुछ प्रमुख चुनौतियों का सामना करती है, जैसे:

- **डेटा गोपनीयता (Privacy)**
- **सुरक्षा (Security)**
- **नैतिक चिंताएँ (Ethical Concerns)**
- **जटिल मॉडलों के लिए विशाल डेटा की आवश्यकता**

भविष्य में मशीन लर्निंग अनुसंधान का लक्ष्य इन चुनौतियों का समाधान करना, एल्गोरिद्म की दक्षता में सुधार करना, और मशीन लर्निंग मॉडल्स को और अधिक जटिल तथा विविध कार्यों में लागू करने की दिशा में अग्रसर होना है।



चित्र 4.3.6 उपयोगकर्ता, वेब सर्वर और डेटाबेस के बीच अंतःक्रिया में अवस्थाओं और संक्रमणों को दर्शाने वाला स्थिति आरेख

4.3.5 वास्तविक जीवन में अनुप्रयोग

मशीन लर्निंग (ML) विभिन्न क्षेत्रों में क्रांतिकारी बदलाव ला रही है, और कई ऐसे कार्यों को संभव बना रही है जिन्हें पहले विज्ञान कथा समझा जाता था। नीचे कुछ प्रमुख क्षेत्रों में इसके अनुप्रयोग दिए गए हैं:

1. वैयक्तिक अनुशंसा प्रणाली (Personalized Recommendations)

ML एल्गोरिद्म उपयोगकर्ताओं की ब्राउज़िंग हिस्ट्री, खरीदारी, और खोज गतिविधियों का विश्लेषण करके अत्यंत वैयक्तिकीकृत उत्पाद, सामग्री या सेवा सुझाव प्रदान करते हैं।

उदाहरण:

- Amazon पर व्यक्तिगत शॉपिंग अनुभव
- Netflix पर कस्टम कंटेंट फीड

2. बुद्धिमान सहायक (Intelligent Assistants)

ML-संचालित सहायक जैसे Siri, Google Assistant, और Alexa प्राकृतिक भाषा को समझते हैं, प्रश्नों का उत्तर देते हैं, और दैनिक कार्यों में सहायता करते हैं। ये समय के साथ प्रत्येक इंटरैक्शन से सीखकर अधिक अनुकूल और प्रभावी बनते हैं।

3. स्वचालित वाहन (Autonomous Vehicles)

ML एल्गोरिद्म सेंसर, कैमरा और GPS डेटा को संयोजित करके वाहन को पर्यावरण को समझने, निर्णय लेने और सुरक्षित रूप से नेविगेट करने में सक्षम बनाते हैं।

उदाहरण: Tesla और Waymo जैसी कंपनियाँ जो दुर्घटनाओं को कम करने, यातायात प्रवाह को सुधारने और परिवहन का रूपांतरण करने की क्षमता रखती हैं।

4. उन्नत चिकित्सा निदान (Advanced Medical Diagnoses)

स्वास्थ्य सेवा क्षेत्र में, ML पर आधारित मॉडल मेडिकल इमेज, जेनेटिक जानकारी और मरीज के डेटा का विश्लेषण करके बीमारियों का पहले और अधिक सटीक रूप से निदान कर सकते हैं। इससे प्रारंभिक पहचान, व्यक्तिगत उपचार और बेहतर परिणाम संभव होते हैं।

उद्योगों पर प्रभाव

मशीन लर्निंग उद्योगों को इस प्रकार नया रूप दे रही है:

- **ग्राहक अनुभव में सुधार:**
  चैटबॉट्स और व्यक्तिगत सेवाओं के माध्यम से 24/7 समर्थन और वैयक्तिकृत अनुभव प्रदान करना।
- **संचालन का अनुकूलन:**
  उत्पादन में प्रेडिक्टिव मेंटेनेंस से लेकर खुदरा क्षेत्र में डिमांड फोरकास्टिंग तक, लागत घटाना और दक्षता बढ़ाना।

- **नवाचार को प्रेरित करना:**
  नए उत्पादों और सेवाओं के लिए संभावनाओं के द्वार खोलना — जैसे वित्त, शिक्षा और मनोरंजन क्षेत्रों में।

भविष्य की संभावनाएँ

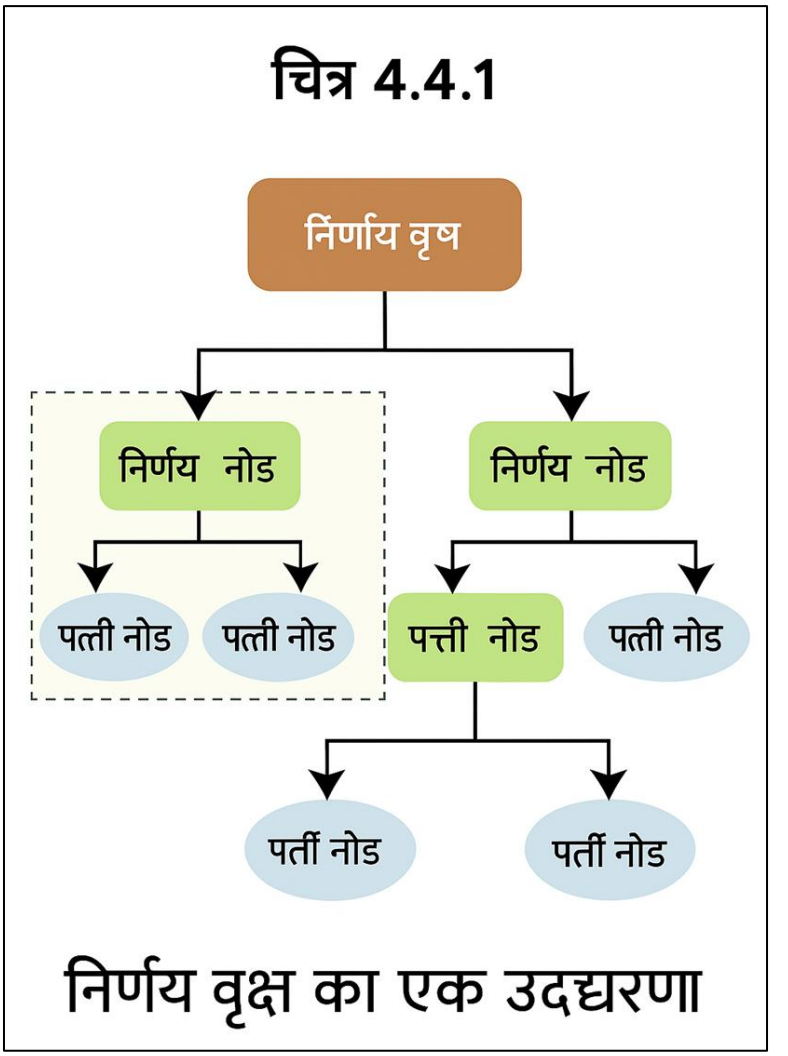
मशीन लर्निंग की संभावनाएँ विशाल हैं और अभी तक पूर्णतः दोही नहीं गई हैं। यह तकनीक भविष्य में और अधिक जटिल समस्याओं को हल करने में सहायक बनेगी, जैसे:

- क्वांटम कंप्यूटिंग
- सतत ऊर्जा समाधान
- वैश्विक स्वास्थ्य पहलों में योगदान

ML का निरंतर विकास न केवल तकनीकी क्षमताओं को बढ़ाएगा बल्कि मानवता की कुछ सबसे गंभीर समस्याओं के समाधान के लिए भी नई दिशा प्रदान करेगा।

4.4 निर्णय वृक्ष (Decision Trees)

निर्णय वृक्ष मशीन लर्निंग में एक लोकप्रिय और शक्तिशाली उपकरण है, जिसका उपयोग वर्गीकरण (Classification) और प्रतिगमन (Regression) दोनों कार्यों के लिए किया जाता है। यह मानव निर्णय-निर्माण प्रक्रिया की नकल करता है, जहाँ डेटा को निर्णय नोड्स पर शाखाओं में विभाजित किया जाता है, जो अंततः पत्तियों (leaves) पर अंतिम निर्णय या भविष्यवाणी की ओर ले जाते हैं। नीचे निर्णय वृक्षों के कार्य, लाभ और अनुप्रयोगों का विस्तृत विवरण प्रस्तुत है:



4.4.1 निर्णय वृक्ष क्या हैं?

निर्णय वृक्ष एक प्रवाहचार्ट जैसे ट्री संरचना होती है, जिसमें:

- **आंतरिक नोड (Internal Node):** कोई फीचर या गुण
- **शाखा (Branch):** कोई निर्णय नियम
- **पत्ती नोड (Leaf Node):** अंतिम परिणाम या आउटपुट होता है

ट्री का सबसे ऊपरी नोड **रूट नोड (Root Node)** कहलाता है। यह विशेषताओं के मानों के आधार पर विभाजन करना सीखता है। यह विभाजन एक पुनरावर्ती पद्धति (recursive partitioning) से किया जाता है।

4.4.2 निर्णय वृक्ष कैसे कार्य करते हैं?

1. **रूट से प्रारंभ:**

   एल्गोरिद्म रूट नोड से प्रारंभ करता है और उस फीचर के आधार पर डेटा विभाजित करता है जिससे सर्वाधिक सूचना लाभ (Information Gain) या अशुद्धता में अधिकतम कमी (Gini Impurity या Entropy) प्राप्त होती है।

2. **पुनरावर्ती विभाजन (Recursive Splitting):**

   प्रत्येक चाइल्ड नोड के लिए यही प्रक्रिया दोहराई जाती है जब तक कोई रोक मानदंड (stopping criteria) पूरा नहीं हो जाता।

3. **रोक मानदंड (Stopping Criteria):**

   विभाजन तब रुकता है जब:

   - अधिकतम गहराई (depth) तक पहुँच चुका हो,
   - नोड में न्यूनतम डेटा पॉइंट्स से कम हों,
   - या कोई और सूचना लाभ न हो।

4. **भविष्यवाणी (Prediction):**

   नए डेटा के लिए निर्णय वृक्ष रूट से पत्ती तक निर्णय का अनुसरण करता है। भविष्यवाणी उस पत्ती नोड के बहुमत वर्ग या औसत मान (regression में) पर आधारित होती है।

4.4.3 निर्णय वृक्ष के लाभ

- **स्पष्टता (Interpretability):**

निर्णय वृक्ष समझने और व्याख्या करने में आसान होते हैं, जिससे इन्हें व्यापार निर्णयों और गैर-तकनीकी हितधारकों को समझाने के लिए उपयुक्त बनाता है।

- **बहु-उपयोगिता (Versatility):**
  वर्गीकरण और रिग्रेशन दोनों कार्यों में उपयोग किए जा सकते हैं।
- **गैर-रेखीय संबंध (Non-linear Relationships):**
  विशेषताओं और लक्ष्यों के बीच जटिल संबंधों को पकड़ने में सक्षम।
- **फीचर स्केलिंग की आवश्यकता नहीं:**
  अन्य एल्गोरिद्म की तुलना में निर्णय वृक्ष को फीचर स्केलिंग की जरूरत नहीं होती।

4.4.4 सीमाएँ

- **अत्यधिक फिटिंग (Overfitting):**
  निर्णय वृक्ष बहुत जटिल बन सकते हैं जो प्रशिक्षण डेटा के अनुसार तो अच्छा कार्य करते हैं लेकिन नए डेटा पर खराब।
  इसे रोकने के लिए **प्रूनिंग (Pruning)**, **न्यूनतम सैंपल सीमा**, या **अधिकतम गहराई निर्धारित करना** आवश्यक है।
- **अस्थिरता (Instability):**
  छोटे बदलाव से पूरी तरह भिन्न वृक्ष उत्पन्न हो सकता है। इसे **एन्सेम्बल विधियों** (जैसे रैंडम फॉरेस्ट) द्वारा नियंत्रित किया जा सकता है।

4.4.5 अनुप्रयोग (Applications)

निर्णय वृक्ष का उपयोग कई क्षेत्रों में किया जाता है, जैसे:

- ग्राहक विभाजन (Customer Segmentation)
- धोखाधड़ी पहचान (Fraud Detection)
- ऋण स्वीकृति (Loan Approval)
- चिकित्सा निदान (Medical Diagnosis)
- बाजार अनुसंधान एवं विश्लेषण (Market Research and Analysis)

4.4.6 एन्सेम्बल विधियाँ: निर्णय वृक्ष की शक्ति को बढ़ाना

निर्णय वृक्ष की कुछ सीमाओं को पार करने और भविष्यवाणी की गुणवत्ता में सुधार लाने के लिए, एन्सेम्बल विधियाँ जैसे **रैंडम फॉरेस्ट**, **ग्रेडिएंट बूस्टिंग**, और **XGBoost** का उपयोग किया जाता है। ये विधियाँ कई निर्णय वृक्षों को मिलाकर अधिक मज़बूत और सटीक भविष्यवाणियाँ प्रदान करती हैं।

सारांश रूप में, निर्णय वृक्ष मशीन लर्निंग टूलकिट का एक मूलभूत घटक हैं जो सरलता और शक्ति का एक संतुलन प्रदान करते हैं। ये न केवल परिणामों की व्याख्या को आसान बनाते हैं, बल्कि एन्सेम्बल तकनीकों के साथ मिलकर जटिल समस्याओं को भी अत्यधिक सटीकता से हल करने में सक्षम होते हैं।

4.4.7 निर्णय वृक्षों को बेहतर बनाने वाली एन्सेम्बल विधियाँ

एन्सेम्बल विधियाँ कई मशीन लर्निंग मॉडल को मिलाकर कुल प्रदर्शन को बेहतर बनाती हैं, ओवरफिटिंग को कम करती हैं, और किसी एकल मॉडल की तुलना में अधिक सटीक भविष्यवाणियाँ प्रदान करती हैं। नीचे तीन प्रमुख एन्सेम्बल विधियाँ दी गई हैं जो निर्णय वृक्षों का उपयोग करती हैं:

1. **रैंडम फॉरेस्ट (Random Forests):**

   रैंडम फॉरेस्ट प्रशिक्षण के दौरान अनेक निर्णय वृक्ष बनाता है और उनके वर्गीकरण (mode) या औसत (mean) भविष्यवाणी को अंतिम परिणाम के रूप में प्रस्तुत करता है। यह ओवरफिटिंग की प्रवृत्ति को कम करता है क्योंकि यह कई वृक्षों की भविष्यवाणी का औसत लेता है, जिससे परिणाम अधिक स्थिर और सटीक होता है।

2. **ग्रेडिएंट बूस्टिंग मशीन (Gradient Boosting Machines - GBM):**

   GBM एक समय में एक वृक्ष बनाता है, जहाँ प्रत्येक नया वृक्ष पूर्ववर्ती त्रुटियों को सुधारने में सहायक होता है।

   यह कमजोर शिक्षकों (weak learners) को बूस्ट करता है, और प्रत्येक गलत वर्गीकृत ऑब्जर्वेशन का वज़न बढ़ाकर अगली पुनरावृत्ति में उस पर ध्यान केंद्रित करता है।

3. **XGBoost (Extreme Gradient Boosting):**

   ग्रेडिएंट बूस्टेड निर्णय वृक्षों का एक अनुकूलित संस्करण है जो गति और प्रदर्शन पर केंद्रित है। इसमें रेगुलराइज़ेशन टर्म शामिल होता है जो ओवरफिटिंग को नियंत्रित करता है, जिससे यह अधिक कुशल बनता है।

4.4.8 व्यावहारिक अनुप्रयोग और विचार

निर्णय वृक्ष और उनके एन्सेम्बल संस्करणों के वास्तविक जीवन में उपयोग बहुत व्यापक और प्रभावशाली हैं। पूर्व उल्लिखित उपयोगों के अतिरिक्त, इनके और भी क्षेत्र हैं:

ऊर्जा उपभोग पूर्वानुमान: ऊर्जा मांग की भविष्यवाणी कर उत्पादन और वितरण को अनुकूल बनाना।

सप्लाई चेन अनुकूलन: बाधाओं की पहचान करना और सप्लाई चेन में रुकावट की भविष्यवाणी करना।

सेंटीमेंट विश्लेषण: ग्राहक फीडबैक, सोशल मीडिया पोस्ट या उत्पाद समीक्षाओं का विश्लेषण कर जनभावना समझना।

प्रीडिक्टिव मेंटेनेंस: उपकरणों की विफलता की भविष्यवाणी कर समय पर रखरखाव की योजना बनाना।

4.4.9 सही मॉडल का चयन

यह तय करते समय कि एकल निर्णय वृक्ष का उपयोग किया जाए या एन्सेम्बल विधि का, निम्नलिखित बातों पर विचार करें:

सटीकता बनाम व्याख्या योग्यता: निर्णय वृक्ष सरल और आसानी से समझ में आने वाले होते हैं, जबकि एन्सेम्बल विधियाँ अधिक सटीकता देती हैं।

डेटा का आकार और जटिलता: बड़े और जटिल डेटासेट्स के लिए एन्सेम्बल विधियाँ बेहतर प्रदर्शन करती हैं।

गणनात्मक समय: एन्सेम्बल मॉडल का प्रशिक्षण अधिक समय लेता है; जबकि निर्णय वृक्ष शीघ्रता से प्रशिक्षित हो जाते हैं, लेकिन सटीकता सीमित हो सकती है।

4.4.10 मशीन लर्निंग में निर्णय वृक्षों का भविष्य

निर्णय वृक्षों और उनके एन्सेम्बल विधियों का विकास और अनुप्रयोग लगातार प्रगति पर है, जिसे अनुसंधान और कम्प्यूटेशनल शक्ति में प्रगति द्वारा प्रेरित किया जा रहा है। **डीप लर्निंग** और **न्यूरल डिसीजन फॉरेस्ट्स** जैसी तकनीकों का उपयोग निर्णय वृक्षों को न्यूरल नेटवर्क्स के साथ एकीकृत करने में हो रहा है — जिससे उनकी क्षमताओं और अनुप्रयोगों का दायरा और भी विस्तृत हो रहा है।

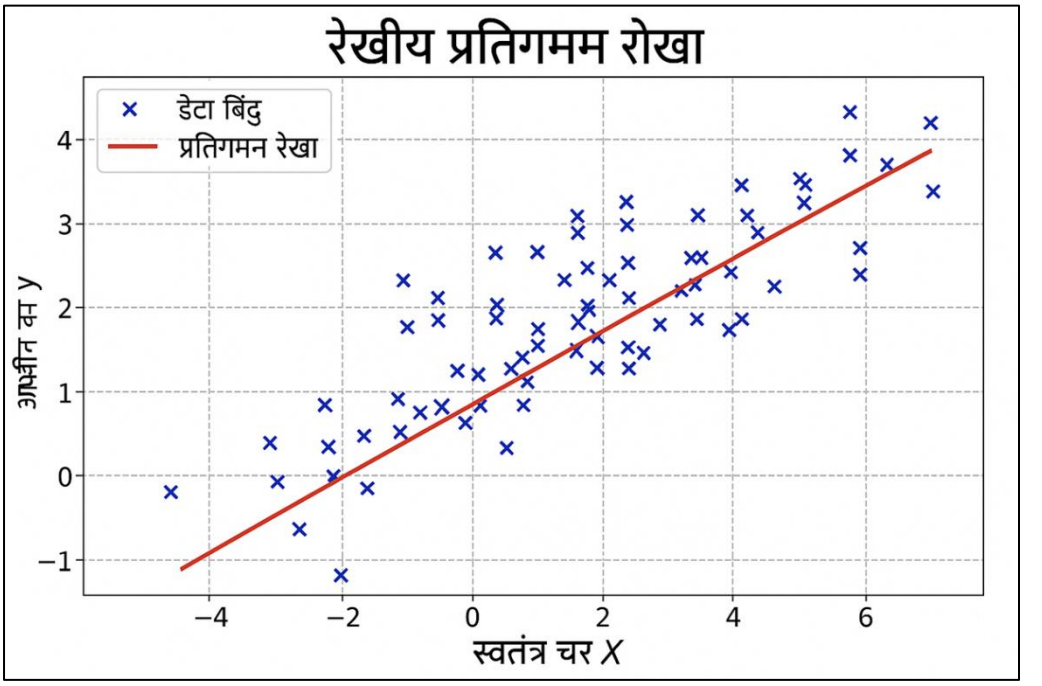
निर्णय वृक्ष अभी भी मशीन लर्निंग का एक प्रमुख आधार बने हुए हैं, जो सरलता और प्रभावशीलता के बीच संतुलन प्रदान करते हैं। उनकी अनुकूलता और एन्सेम्बल विधियों में निरंतर नवाचार यह सुनिश्चित करता है कि वे भविष्य में भी भविष्यवाणी मॉडलिंग और डेटा विश्लेषण के लिए आवश्यक रहेंगे। जैसे-जैसे मशीन लर्निंग आगे बढ़ती है, निर्णय वृक्षों का विकास और अन्य एल्गोरिद्म के साथ उनका एकीकरण नए समाधान और जटिल समस्याओं के उत्तर प्रदान करने में सक्षम होगा।

4.5 रैग्रेशन और वर्गीकरण में रैखिक मॉडल्स (Linear Models)

रैखिक मॉडल्स यह सिद्ध करते हैं कि कभी-कभी डेटा विज्ञान और विश्लेषण में **सरलता ही सर्वोच्च बुद्धिमत्ता** होती है। इनका गणितीय सूत्रीकरण अत्यंत सरल होता है, जिससे यह तेज़ी से प्रशिक्षित और भविष्यवाणी करने योग्य होते हैं। यह विशेष रूप से उन स्थितियों में उपयोगी हैं जहाँ **गति और कम्प्यूटेशनल दक्षता** महत्वपूर्ण होती है।

रैखिक मॉडल्स की पारदर्शिता यह समझने में सहायक होती है कि **इनपुट वेरिएबल्स परिणाम को कैसे प्रभावित करते हैं**, जिससे वैज्ञानिक और व्यावसायिक क्षेत्रों में इन पर विश्वास करना और सत्यापन करना आसान होता है। ये अधिक जटिल एल्गोरिद्म के लिए **बेंचमार्क** की तरह भी काम करते हैं।

नवीन तकनीकों जैसे कि **रेगुलराइज़ेशन** और **एन्सेम्बल विधियों** के साथ इनका एकीकरण यह दर्शाता है कि ये आज भी डेटा विश्लेषण की आधुनिक चुनौतियों का समाधान देने में प्रासंगिक हैं।



4.5.1 रैखिक प्रतिगमन (Linear Regression)

Linear Regression एक तकनीक है जो एक स्वतंत्र चर (X) और एक आश्रित चर (y) के बीच रैखिक संबंध को मॉडल करती है।

चित्र 4.5.1 में एक रैखिक रिग्रेशन रेखा दर्शाई गई है जिसमें नीले बिंदु डेटा पॉइंट्स हैं और लाल रेखा रिग्रेशन लाइन है, जो X और y के बीच रैखिक संबंध को दर्शाती है।

उदाहरण: किसी संपत्ति की कीमत की भविष्यवाणी करना — स्थान, आकार और बेडरूम की संख्या जैसे फीचर्स के आधार पर।

Python में Scikit-learn लाइब्रेरी का उपयोग करके Linear Regression कैसे करें:

एल्गोरिद्म अवलोकन:

1. **डेटा तैयार करना:** X और y को परिभाषित करें।
2. **डेटा विभाजित करना:** प्रशिक्षण और परीक्षण सेट में विभाजित करें।
3. **मॉडल बनाएँ:** Scikit-learn का उपयोग करें।
4. **मॉडल प्रशिक्षित करें:** .fit() से।
5. **भविष्यवाणी करें:** .predict() से।
6. **मूल्यांकन करें:** MSE और R^2 स्कोर से।

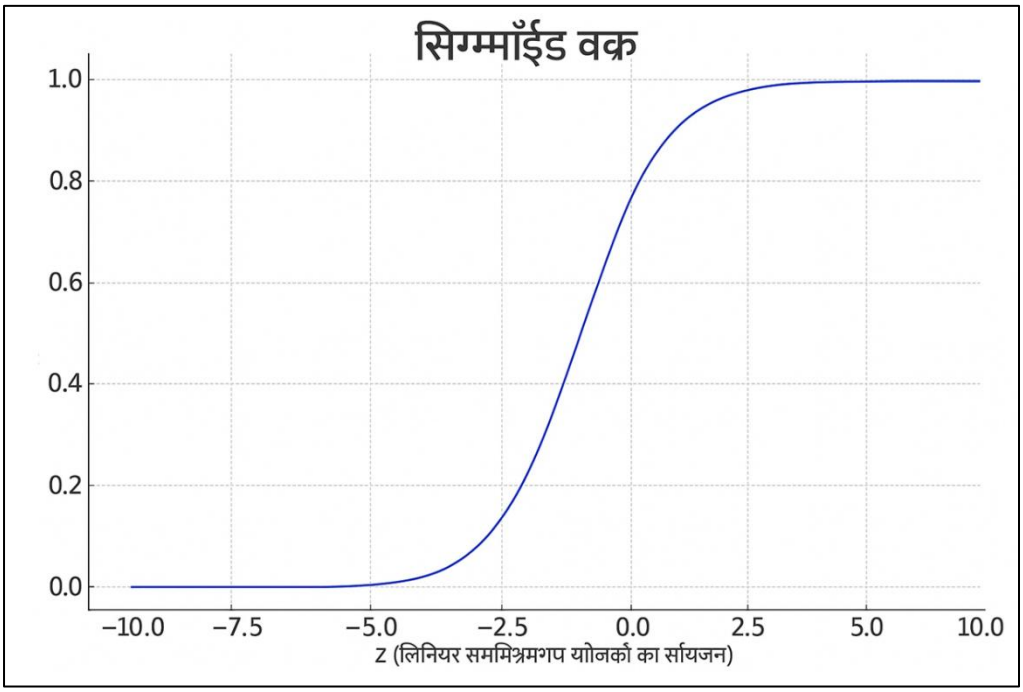
Python कोड उदाहरण:

```
python
CopyEdit
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
# डेटा बनाना
np.random.seed(0)
X = 2.5 * np.random.randn(100, 1) + 1.5
y = 2 + 0.3 * X.squeeze() + 0.5 * np.random.randn(100)
# डेटा विभाजित करना
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# मॉडल बनाना
```

```
model = LinearRegression()
# मॉडल को प्रशिक्षित करना
model.fit(X_train, y_train)
# भविष्यवाणियाँ
y_pred = model.predict(X_test)
# मूल्यांकन
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f"Mean Squared Error (MSE): {mse}")
print(f"R-squared (R2): {r2}")
```

यह कोड दिखाता है कि कैसे Linear Regression को Python में Scikit-learn का उपयोग करके लागू किया जा सकता है — डेटा निर्माण, विभाजन, मॉडल बनाना, प्रशिक्षण, भविष्यवाणी और मूल्यांकन तक की प्रक्रिया।

4.5.2 लॉजिस्टिक रिग्रेशन (Logistic Regression)



Logistic Regression का उपयोग **द्विआधारी वर्गीकरण समस्याओं** (Binary Classification) में किया जाता है। यह एक **लॉजिस्टिक फंक्शन (सिग्मॉइड)** का उपयोग करता है जो किसी वस्तु के किसी श्रेणी से संबंधित होने की **संभावना** को अनुमानित करता है।

उदाहरण:

यह निर्धारित करना कि कोई ईमेल **स्पैम है या नहीं**, उसके कंटेंट और प्रेषक के आधार पर।

यहाँ सिग्मॉइड वक्र (Sigmoid Curve) है, जो लॉजिस्टिक रिग्रेशन में संभाव्यता अनुमान (probability estimation) प्रदर्शित करने के लिए अत्यंत महत्वपूर्ण है। यह वक्र किसी भी इनपुट मान zzz (जो इनपुट्स का रैखिक संयोजन होता है) को 0 और 1 के बीच की एक संभाव्यता में परिवर्तित करता है। लॉजिस्टिक रिग्रेशन में इसका उपयोग यह भविष्यवाणी करने के लिए किया जाता है कि कोई दिया गया इनपुट किसी विशेष श्रेणी से संबंधित है या नहीं।

लॉजिस्टिक रिग्रेशन मॉडल को फिट करने का एल्गोरिद्म और कोड

एल्गोरिद्म:

Python की लोकप्रिय मशीन लर्निंग लाइब्रेरी **scikit-learn** का उपयोग करके लॉजिस्टिक रिग्रेशन मॉडल को फिट करने के लिए निम्नलिखित चरणों का पालन किया जाता है:

1. **आवश्यक लाइब्रेरी इंपोर्ट करें:** Scikit-learn और डेटा प्रोसेसिंग के लिए अन्य ज़रूरी लाइब्रेरीज़ को इंपोर्ट करें।
2. **डेटासेट लोड करें:** scikit-learn में उपलब्ध इनबिल्ट डेटासेट का उपयोग करें या Pandas से अपना कस्टम डेटा लोड करें।
3. **डेटा का पूर्वप्रसंस्करण करें:** डेटासेट को फीचर्स (X) और लक्ष्य वेरिएबल (y) में विभाजित करें। फिर उसे ट्रेनिंग और टेस्ट सेट में बाँटें।
4. **लॉजिस्टिक रिग्रेशन मॉडल बनाएँ:** Scikit-learn की LogisticRegression क्लास का उपयोग करें।
5. **मॉडल को प्रशिक्षित करें:** .fit() का उपयोग करके प्रशिक्षण डेटा पर मॉडल को फिट करें।
6. **मॉडल का मूल्यांकन करें:** टेस्ट डेटा पर मॉडल के प्रदर्शन को जाँचें।

Python कोड स्निपेट:

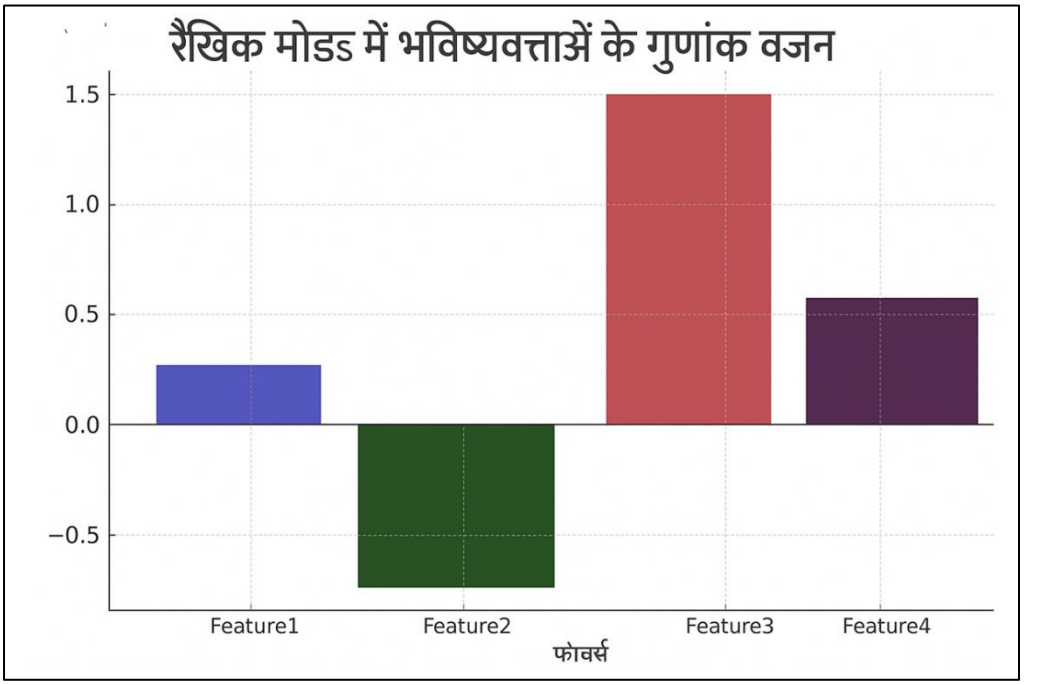
python

CopyEdit

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import datasets
from sklearn.metrics import accuracy_score
# उदाहरण: Iris डेटासेट
iris = datasets.load_iris()
X = iris.data
y = iris.target
# केवल द्विआधारी वर्गीकरण के लिए फ़िल्टर करें
is_binary_classification = y != 2
X_binary = X[is_binary_classification]
y_binary = y[is_binary_classification]
# डेटा को ट्रेनिंग और टेस्ट सेट में बाँटना
X_train, X_test, y_train, y_test = train_test_split(X_binary, y_binary, test_size=0.3,
random_state=42)
# लॉजिस्टिक रिग्रेशन मॉडल बनाना
model = LogisticRegression()
# मॉडल को प्रशिक्षित करना
model.fit(X_train, y_train)
# भविष्यवाणी करना
y_pred = model.predict(X_test)
# मूल्यांकन करना
accuracy = accuracy_score(y_test, y_pred)
print(f"Model accuracy: {accuracy:.2f}")
```

यह कोड लॉजिस्टिक रिग्रेशन के कार्यान्वयन की पूरी प्रक्रिया दर्शाता है: डेटासेट लोड करना, डेटा विभाजित करना, मॉडल बनाना, प्रशिक्षण देना, भविष्यवाणी करना, और सटीकता (accuracy) द्वारा मॉडल का मूल्यांकन करना।

4.5.3 मॉडल को समझना



दोनों प्रकार के रैखिक मॉडल — **लिनियर रिग्रेशन** और **लॉजिस्टिक रिग्रेशन** — यह मानते हैं कि इनपुट विशेषताओं (features) और लक्ष्य वेरिएबल के बीच एक रैखिक संबंध होता है।

- **लिनियर रिग्रेशन** एक सतत मान (continuous value) की भविष्यवाणी करता है।
- **लॉजिस्टिक रिग्रेशन** किसी इनपुट के एक विशिष्ट वर्ग से संबंधित होने की **संभाव्यता** का मॉडल बनाता है।

उदाहरण:

- बिक्री (Sales) का पूर्वानुमान लगाने के लिए लिनियर रिग्रेशन
- ग्राहक छूटने (Churn) की संभावना पहचानने के लिए लॉजिस्टिक रिग्रेशन

4.5.4 मॉडल मूल्यांकन

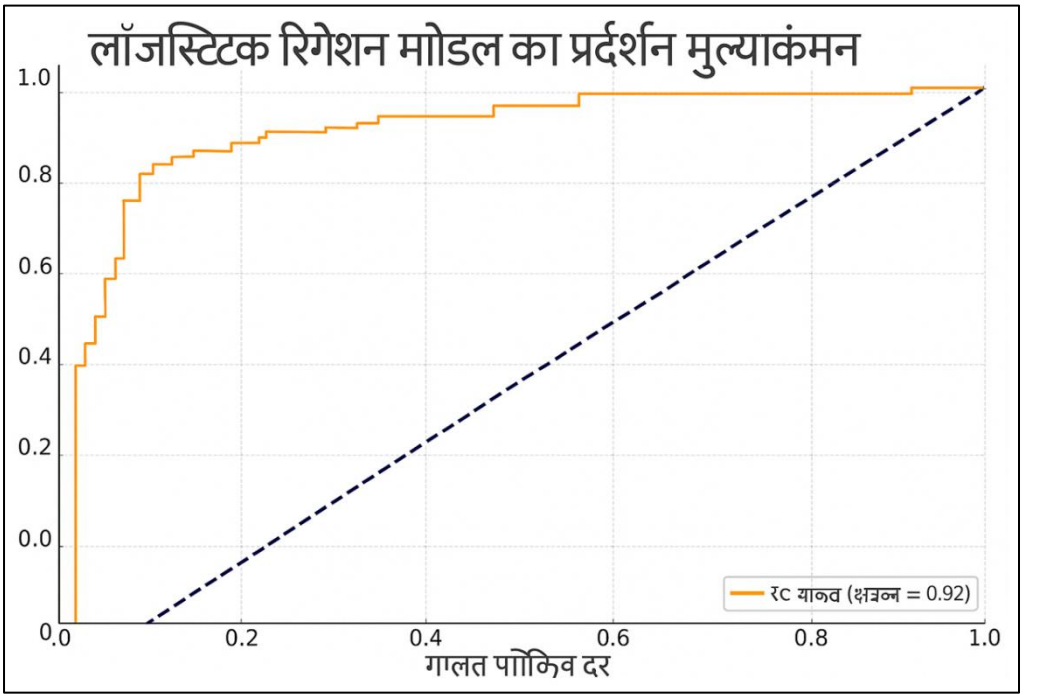
रैखिक मॉडल्स के प्रदर्शन का मूल्यांकन यह समझने के लिए महत्वपूर्ण है कि वे कितना प्रभावी हैं।

- **रिग्रेशन मॉडल के लिए:**
  - **RMSE (रूट मीन स्क्वेयर एरर)**
  - **R^2 स्कोर (निर्धारण गुणांक / Coefficient of Determination)**
- **वर्गीकरण मॉडल के लिए:**
  - **सटीकता (Accuracy)**
  - **प्रेसिजन (Precision)**
  - **रिकॉल (Recall)**
  - **AUC (ROC कर्व के नीचे का क्षेत्रफल)**

उदाहरण:

एक लॉजिस्टिक रिग्रेशन मॉडल द्वारा मरीज के पुनः प्रवेश (readmission) की भविष्यवाणी में सटीकता और प्रेसिजन का विश्लेषण करना।

4.5.5 लाभ और सीमाएँ



लाभ:

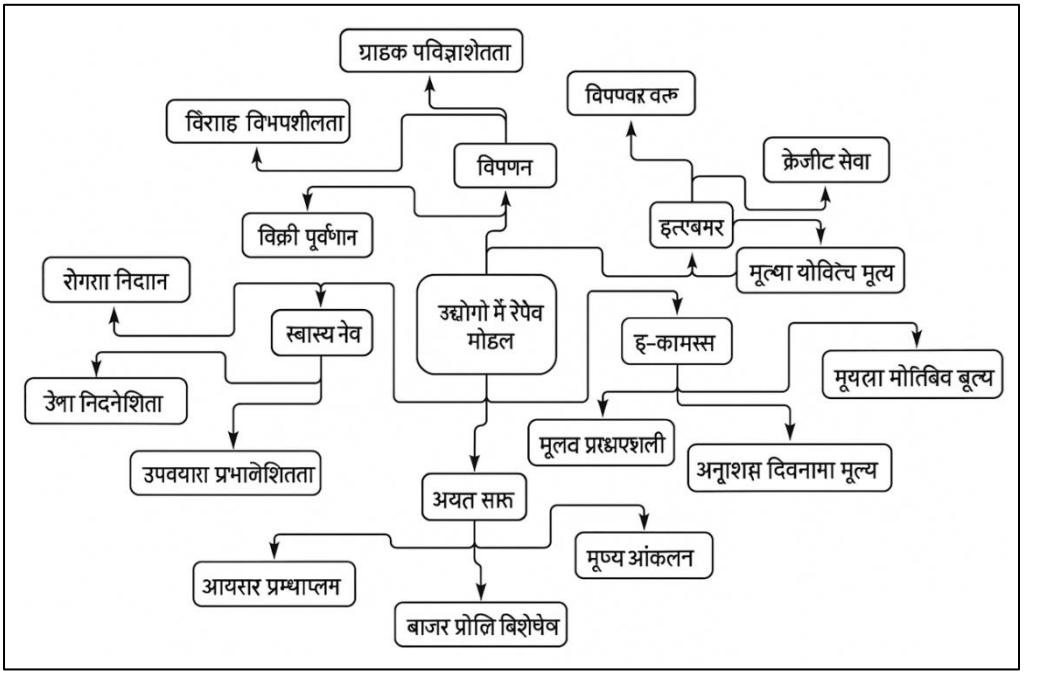
- रैखिक मॉडल्स अत्यधिक **व्याख्यायोग्य (interpretable)** होते हैं।

- यह समझना आसान होता है कि कौन-सा फ़ीचर भविष्यवाणी को कैसे प्रभावित कर रहा है।
- **कम संसाधन** में तेज़ी से प्रशिक्षण और निष्पादन।

सीमाएँ:

- इनकी **रैखिकता की धारणा** सीमित कर सकती है जटिल या गैर-रेखीय डेटा में इनका उपयोग।
- यह **आउटलायर्स** (extreme values) के प्रति संवेदनशील होते हैं।

4.5.6 उद्योग में अनुप्रयोग



चित्र 4.5.5: रैखिक मॉडलों के विभिन्न उद्योगों और अनुप्रयोगों को दर्शाने वाला आरेख

रैखिक मॉडल्स विभिन्न उद्योगों में व्यापक रूप से उपयोग किए जाते हैं:

- **वित्त (Finance):**
  - **लिनियर रिग्रेशन:** स्टॉक कीमतों का पूर्वानुमान
  - **लॉजिस्टिक रिग्रेशन:** क्रेडिट जोखिम मूल्यांकन
- **स्वास्थ्य सेवा (Healthcare):**
  - **लॉजिस्टिक रिग्रेशन:** रोग निदान

- **ऑटोमोबाइल उद्योग:**
    - **लिनियर रिग्रेशन:** गाड़ी की कीमत की भविष्यवाणी
- **मार्केटिंग:**
    - **लॉजिस्टिक रिग्रेशन:** ग्राहक खंड निर्धारण (Customer Segmentation)

उदाहरण:

ऑटोमोबाइल उद्योग में कीमत का पूर्वानुमान लगाने के लिए लिनियर रिग्रेशन का उपयोग, और विपणन (Marketing) में लॉजिस्टिक रिग्रेशन द्वारा ग्राहक वर्गीकरण।

4.5.7 विस्तार और भविष्य की दिशा

Ridge और **Lasso रिग्रेशन** जैसी रेगुलराइज़ेशन तकनीकों के आगमन के साथ, रैखिक मॉडल और भी अधिक मज़बूत और बहुपरकीय (versatile) बन गए हैं।

मशीन लर्निंग में हो रहे लगातार नवाचार इन मॉडलों को और अधिक सटीक, अनुकूल और आधुनिक चुनौतियों के लिए उपयुक्त बना रहे हैं।

एल्गोरिद्म और प्रोग्राम:

Lasso Regression के माध्यम से फ़ीचर चयन द्वारा मॉडल के प्रदर्शन को सुधारना

Lasso (Least Absolute Shrinkage and Selection Operator) एक प्रकार का रैखिक रिग्रेशन है जो **श्रीनकेज (Shrinkage)** तकनीक का उपयोग करता है, जिसमें डेटा मानों को एक केंद्रीय बिंदु (जैसे माध्य) की ओर खींचा जाता है।

यह विशेष रूप से **फ़ीचर चयन** और **ओवरफिटिंग से बचाव** के लिए उपयोगी है, क्योंकि यह कोएफिशिएंट्स के परिमाण (absolute value) को दंडित करता है और कई को शून्य तक घटा देता है।

एल्गोरिद्म:

1. **डेटा का मानकीकरण (Standardization):**
   Lasso इनपुट फ़ीचर्स के स्केल पर संवेदनशील होता है, इसलिए इन्हें मानकीकृत करना ज़रूरी है।
2. **रेगुलराइज़ेशन पैरामीटर का चयन:**
   दंड की तीव्रता एक हाइपरपैरामीटर होती है। इसके लिए आमतौर पर **क्रॉस-वैलिडेशन** का उपयोग किया जाता है।

3. **Lasso मॉडल को फिट करना:**

   प्रशिक्षण डेटा पर मॉडल को इस तरह फिट किया जाता है कि वह लॉस फंक्शन को न्यूनतम करे, जिसमें दंड शामिल हो।

4. **मॉडल का मूल्यांकन:**

   एक वैलिडेशन सेट या क्रॉस-वैलिडेशन के माध्यम से मॉडल की सटीकता और त्रुटि का विश्लेषण करें।

5. **फ़ीचर चयन:**

   जिन फ़ीचर्स के कोएफिशिएंट्स शून्य हो जाते हैं, उन्हें मॉडल से हटाया जा सकता है।

Python कोड उदाहरण:

```
python
CopyEdit
from sklearn.datasets import load_boston
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LassoCV
from sklearn.metrics import mean_squared_error
# डेटासेट लोड करना (Boston हाउसिंग डेटा)
data = load_boston()
X = data.data
y = data.target
# डेटा का मानकीकरण
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
# ट्रेनिंग और टेस्ट डेटा में विभाजन
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)
```

```
# Lasso मॉडल बनाना और फिट करना (क्रॉस-वैलिडेशन के साथ)
lasso = LassoCV(cv=5, random_state=42).fit(X_train, y_train)
# भविष्यवाणी और MSE की गणना
y_pred = lasso.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse:.2f}')
# कोएफिशिएंट्स प्रिंट करना
print("Lasso coefficients:")
features = data.feature_names
for feature, coef in zip(features, lasso.coef_):
    print(f"{feature}: {coef:.4f}")
# महत्वपूर्ण फ़ीचर्स (जिनके कोएफिशिएंट्स शून्य नहीं हैं)
important_features = features[lasso.coef_ != 0]
print(f"Important features: {important_features}")
```

यह उदाहरण:

- **Boston Housing Dataset** पर आधारित है
- सभी फ़ीचर्स को **मानकीकृत** करता है
- **LassoCV** के माध्यम से रेगुलराइज़ेशन पैरामीटर को **स्वतः चुनता** है
- **MSE** के माध्यम से प्रदर्शन को **मूल्यांकित** करता है
- और **महत्वपूर्ण फ़ीचर्स** को स्पष्ट रूप से प्रदर्शित करता है (वे जिनके कोएफिशिएंट शून्य नहीं हैं)

यह दृष्टांत दर्शाता है कि Lasso न केवल भविष्यवाणी को बेहतर करता है बल्कि फीचर चयन की प्रक्रिया को भी स्वचालित बनाता है — जो बड़े और जटिल डेटासेट्स में अत्यंत उपयोगी होता है।

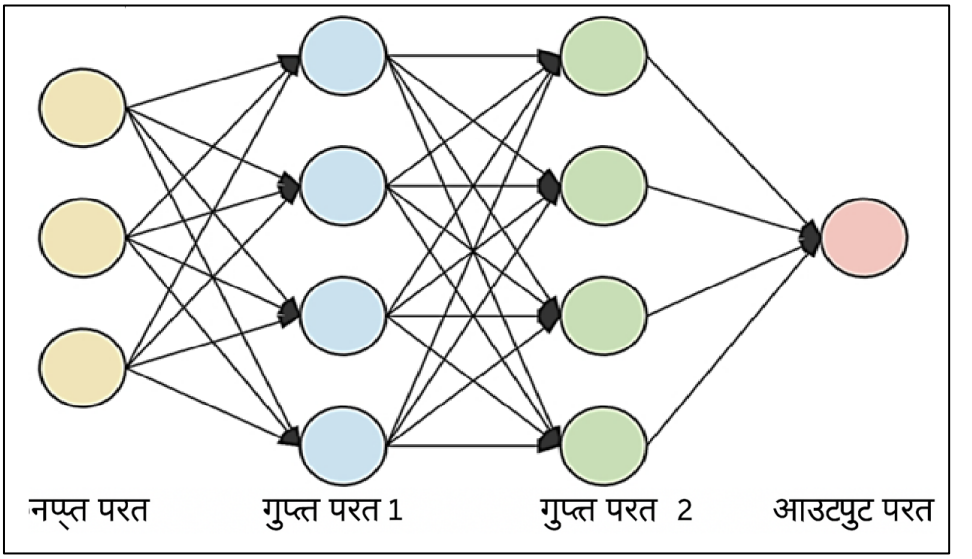
4.6 कृत्रिम न्यूरल नेटवर्क (Artificial Neural Networks)

कृत्रिम न्यूरल नेटवर्क (ANNs) ऐसे कंप्यूटिंग सिस्टम हैं जो जीवों के मस्तिष्क में पाए जाने वाले जैविक न्यूरल नेटवर्क से प्रेरित होते हैं।

ANN में इनपुट लेयर, एक या अधिक हिडन लेयर्स और एक आउटपुट लेयर सहित नोड्स की परतें होती हैं।

प्रत्येक नोड (या कृत्रिम न्यूरॉन) दूसरे से जुड़ा होता है और इसके साथ एक वेट (weight) और थ्रेशोल्ड (threshold) मान जुड़ा होता है।

यदि किसी न्यूरॉन का आउटपुट उसके थ्रेशोल्ड से अधिक होता है, तो वह सक्रिय (activate) हो जाता है और अगली लेयर को डेटा भेजता है। अन्यथा, वह डेटा को आगे नहीं बढ़ाता।



चित्र 4.6.1 इनपुट, हिडन और आउटपुट लेयर्स तथा नोड्स को दर्शाने वाला एक मूल न्यूरल नेटवर्क आरेख

4.6.1 कृत्रिम न्यूरल नेटवर्क का मूल आधार

उदाहरण:

मान लीजिए कि एक सरल न्यूरल नेटवर्क का उपयोग ईमेल को स्पैम या नॉन-स्पैम वर्ग में वर्गीकृत करने के लिए किया गया है।

इनपुट लेयर को कुछ विशेषताएँ प्राप्त होती हैं जैसे —

- कुछ शब्दों की आवृत्ति
- ईमेल की लंबाई
- अटैचमेंट का होना

हिडन लेयर्स इन इनपुट्स को प्रोसेस करती हैं और आउटपुट लेयर अंतिम वर्गीकरण (Spam / Not Spam) करती है।

एल्गोरिद्म और प्रोग्राम

नीचे एक सरल **फीडफॉरवर्ड न्यूरल नेटवर्क** को TensorFlow का उपयोग करके बनाने और प्रशिक्षित करने का उदाहरण दिया गया है।

यह एक **वर्गीकरण कार्य** (classification task) के लिए तैयार किया गया है, जिसमें हम **MNIST डेटासेट** (हाथ से लिखे अंकों की छवियाँ) का उपयोग करते हैं।

एल्गोरिद्म:

1. **आवश्यक लाइब्रेरी इंपोर्ट करें:** TensorFlow और अन्य ज़रूरी मॉड्यूल्स इंपोर्ट करें।
2. **डेटासेट लोड करें:** MNIST डेटासेट लोड करें।
3. **डेटा को प्रीप्रोसेस करें:**
    - इनपुट को नॉर्मलाइज़ करें (0–255 → 0–1)
    - लेबल्स को वन-हॉट एन्कोडिंग में बदलें
4. **मॉडल परिभाषित करें:** Sequential मॉडल बनाएं और Dense लेयर्स जोड़ें।
5. **मॉडल को कम्पाइल करें:** Optimizer, Loss Function और Accuracy मेट्रिक सेट करें।
6. **मॉडल को प्रशिक्षित करें:** .fit() का उपयोग कर ट्रेनिंग करें।
7. **मूल्यांकन करें:** टेस्ट सेट पर .evaluate() द्वारा मॉडल का प्रदर्शन जांचें।

प्रोग्राम:

python

CopyEdit

```
import tensorflow as tf
```

```
from tensorflow.keras import layers, models
from tensorflow.keras.datasets import mnist
from tensorflow.keras.utils import to_categorical
# डेटा लोड और प्रीप्रोसेसिंग
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()
train_images = train_images.reshape((60000, 28 * 28)).astype('float32') / 255
test_images = test_images.reshape((10000, 28 * 28)).astype('float32') / 255
train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)
# मॉडल परिभाषित करें
model = models.Sequential()
model.add(layers.Dense(512, activation='relu', input_shape=(28 * 28,)))
model.add(layers.Dense(10, activation='softmax'))
# मॉडल कम्पाइल करें
model.compile(optimizer='rmsprop',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
# मॉडल को प्रशिक्षित करें
model.fit(train_images, train_labels, epochs=5, batch_size=128)
# मॉडल मूल्यांकन करें
test_loss, test_acc = model.evaluate(test_images, test_labels)
print(f'Test accuracy: {test_acc:.4f}')
```

यह कोड MNIST डेटासेट पर एक न्यूरल नेटवर्क को प्रशिक्षित करता है। प्रत्येक इमेज 28x28 पिक्सेल की होती है जिसे 784 आयामों में बदला जाता है।

- हिडन लेयर में 512 न्यूरॉन होते हैं जो **ReLU** एक्टिवेशन का उपयोग करते हैं।
- आउटपुट लेयर में 10 क्लासेस के लिए **Softmax** एक्टिवेशन होता है।

- **Categorical Crossentropy** लॉस फंक्शन और **Accuracy** मेट्रिक का उपयोग किया गया है।
- मॉडल को 5 इपॉक्स तक प्रशिक्षित किया जाता है और टेस्ट सेट पर इसका मूल्यांकन किया जाता है।

4.6.2 कृत्रिम न्यूरल नेटवर्क के प्रकार

कृत्रिम न्यूरल नेटवर्क (ANNs) कई प्रकार के होते हैं, जिनकी संरचना (architecture) के आधार पर उन्हें विभिन्न कार्यों के लिए उपयुक्त बनाया गया है। यहाँ तीन मुख्य प्रकारों को विस्तार से प्रस्तुत किया गया है: **फीडफॉरवर्ड न्यूरल नेटवर्क (FNN)**, **रिकरंट न्यूरल नेटवर्क (RNN)** और **कन्वोल्यूशनल न्यूरल नेटवर्क (CNN)**

1. फीडफॉरवर्ड न्यूरल नेटवर्क (FNN)

FNN न्यूरल नेटवर्क का सबसे सरल रूप है, जिसमें नोड्स के बीच संबंध चक्र (cycle) नहीं बनाते हैं। इसका अर्थ है कि डेटा केवल एक दिशा में बहता है — इनपुट से आउटपुट की ओर, हिडन लेयर्स के माध्यम से।

FNN में किसी इनपुट से अगले इनपुट के लिए कोई स्थिति (state) नहीं रखी जाती, जिससे यह सरल और कुशल बनता है।

अनुप्रयोग:

पैटर्न पहचान:

FNN डेटा में पैटर्न और ट्रेंड पहचानने में सक्षम होते हैं, जैसे — चेहरे की पहचान, हस्तलिखित पाठ की पहचान, और बाज़ार प्रवृत्ति विश्लेषण।

वर्गीकरण कार्य:

पूर्वनिर्धारित श्रेणियों में इनपुट को वर्गीकृत करने में यह अत्यधिक प्रभावी है, जैसे — स्पैम ईमेल की पहचान, या रोग निदान।

उदाहरण:

एक ग्राहक के सेवा छोड़ने (चर्न) की भविष्यवाणी करने वाला मॉडल जिसमें ग्राहक के सेवा उपयोग पैटर्न, संतुष्टि रेटिंग और जनसांख्यिकी जानकारी जैसे इनपुट होते हैं।

2. रिकरंट न्यूरल नेटवर्क (RNN)

RNN अधिक जटिल न्यूरल नेटवर्क होते हैं जिनमें नोड्स के बीच संबंध समय क्रम (temporal sequence) में एक दिशात्मक ग्राफ बनाते हैं।
इस संरचना के कारण ये नेटवर्क अपना आंतरिक **स्मृति (memory)** बनाए रखते हैं और अनुक्रमिक इनपुट प्रोसेस कर सकते हैं।

अनुप्रयोग:

समय-श्रृंखला भविष्यवाणी:

वित्तीय बाज़ार प्रवृत्तियाँ, मौसम पैटर्न आदि की भविष्यवाणी के लिए अत्यंत उपयुक्त।

प्राकृतिक भाषा प्रोसेसिंग (NLP):

भाषा के संदर्भ को समझने में माहिर, जिससे मशीन अनुवाद, टेक्स्ट समरी, और भावना विश्लेषण जैसे कार्य संभव होते हैं।

उदाहरण:

एक टेक्स्ट जनरेशन सिस्टम जो RNN का उपयोग करके किसी अनुक्रम में अगला शब्द या अक्षर भविष्यवाणी कर सकता है।

3. कन्वोल्यूशनल न्यूरल नेटवर्क (CNN)

CNN विशेष रूप से ऐसे डेटा के लिए डिज़ाइन किए गए हैं जिनका लेआउट ग्रिड की तरह होता है — जैसे **छवियाँ (images)**।
ये नेटवर्क **कन्वोल्यूशन** नामक गणितीय प्रक्रिया का उपयोग करते हैं जो उन्हें छवि जैसे उच्च-आयामी डेटा में स्थानीय पैटर्न को प्रभावी रूप से पहचानने में सक्षम बनाती है।

अनुप्रयोग:

छवि और वीडियो पहचान:

CNN छवियों में पैटर्न पहचानकर वस्तु, चेहरा या दृश्य को पहचान सकते हैं। यह सुरक्षा प्रणालियों, सोशल मीडिया, और सर्च इंजनों में उपयोग किया जाता है।

चिकित्सीय छवि विश्लेषण:

X-ray और MRI जैसी मेडिकल इमेज में बीमारियों के संकेत पहचानकर निदान में सहायक।

उदाहरण:

एक CNN आधारित इमेज क्लासिफिकेशन सिस्टम छवियों में वस्तुओं को पहचान सकता है, जैसे — कुत्तों की नस्लों में भेद करना या त्वचा रोग की पहचान करना।

इन तीनों प्रकार के न्यूरल नेटवर्क अपनी अनूठी क्षमताओं के माध्यम से विभिन्न क्षेत्रों में जटिल समस्याओं को हल करने में सक्षम हैं।

ये मशीन लर्निंग की शक्ति और बहुपरकीयता को दर्शाते हैं।

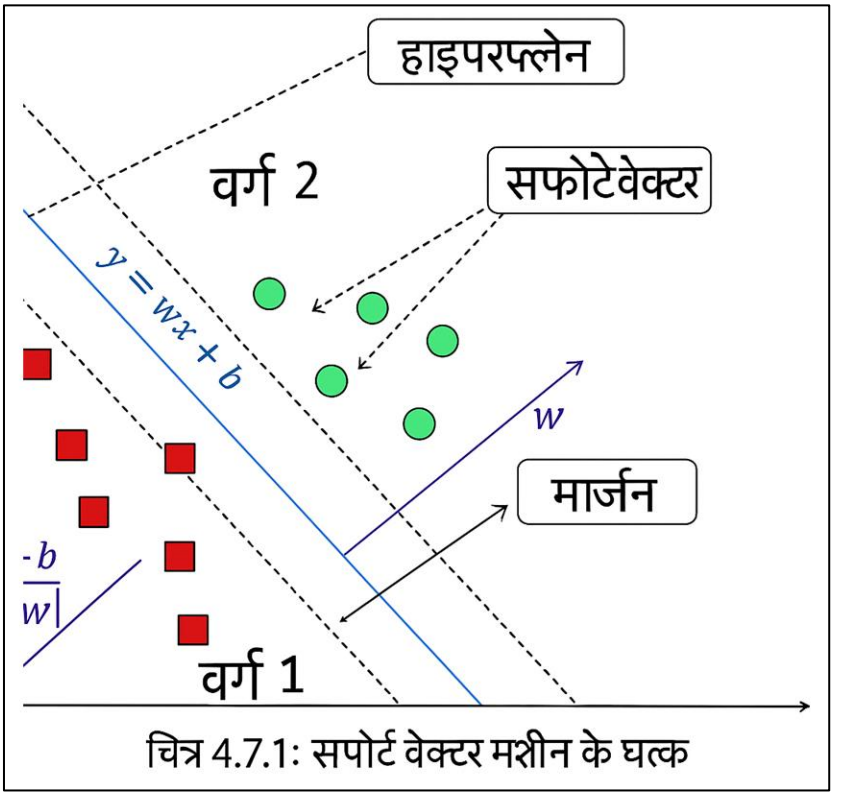
4.7 सपोर्ट वेक्टर मशीनें (Support Vector Machines)

सपोर्ट वेक्टर मशीन (SVM) एक सुपरवाइज्ड लर्निंग तकनीक है जिसका उपयोग वर्गीकरण, प्रतिगमन और आउट्लायर डिटेक्शन के लिए किया जाता है।

SVM का मुख्य विचार यह है कि वह हाइपरप्लेन (hyperplane) ढूंढे जो किसी डेटासेट को विभिन्न वर्गों में सबसे अच्छे तरीके से विभाजित करे।

यह उच्च-आयामी (high-dimensional) स्थानों में और उन मामलों में जहाँ फीचर्स की संख्या सैंपलों से अधिक होती है, विशेष रूप से उपयोगी है।

4.7.1 सपोर्ट वेक्टर मशीन को समझना



चित्र 4.7.1: सपोर्ट वेक्टर मशीन के घटक

SVM ऐसे इष्टतम हाइपरप्लेन की पहचान करता है जो फीचर स्पेस में वर्गों को अलग करता है। हाइपरप्लेन के सबसे पास के डेटा पॉइंट्स को **सपोर्ट वेक्टर** कहा जाता है, और यही पॉइंट्स वर्गों के बीच की दूरी (मार्जिन) को परिभाषित करते हैं।

उद्देश्य इस मार्जिन को अधिकतम करना होता है।

SVM का उपयोग **लाइनियर** और **नॉन-लाइनियर** दोनों प्रकार के डाटा सेट्स के लिए किया जा सकता है।

नॉन-लाइनियर विभाजन के लिए, SVM **कर्नल ट्रिक** का उपयोग करता है, जो इनपुट डेटा को उच्च-आयामी स्थान में बदल देता है जहाँ पर एक लाइनियर विभाजन संभव हो जाता है।

एल्गोरिद्म:

1. **सही कर्नल का चयन करें:**
    - लाइनियर डेटा के लिए लाइनियर कर्नल
    - नॉन-लाइनियर डेटा के लिए RBF या अन्य कर्नल
2. **डेटा रूपांतरण (यदि आवश्यक हो):**

   कर्नल फ़ंक्शन का उपयोग कर डेटा को उच्च-आयामी स्थान में मैप करें।
3. **सपोर्ट वेक्टर्स की पहचान करें:**

   निर्णय सतह के सबसे पास स्थित पॉइंट्स को खोजें।
4. **मार्जिन को अधिकतम करें:**

   ऐसा हाइपरप्लेन ढूंढें जो वर्गों को सबसे बड़े मार्जिन के साथ विभाजित करता हो।
5. **सैंपल्स का वर्गीकरण करें:**

   नए इनपुट्स को उस हाइपरप्लेन के आधार पर वर्गीकृत करें।

उदाहरण:

एक ईमेल स्पैम डिटेक्शन टास्क जिसमें फीचर्स में शब्दों की आवृत्ति, ईमेल की लंबाई, और अटैचमेंट की उपस्थिति शामिल हो सकती है।

प्रोग्राम:

```
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
# डेटासेट लोड करें
```

```
iris = datasets.load_iris()
X = iris.data
y = iris.target
# डेटा को ट्रेन और टेस्ट सेट में बाँटना
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
# फीचर्स को मानकीकृत करना
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
# SVM क्लासिफायर बनाना
clf = SVC(kernel='linear')  # लाइनियर कर्नल
clf.fit(X_train, y_train)
# मॉडल मूल्यांकन
print(f"Model accuracy: {clf.score(X_test, y_test):.2f}")
```

4.7.2 SVM में उन्नत तकनीकें

SVM की कार्यक्षमता को और बेहतर बनाने के लिए उन्नत तकनीकों का उपयोग किया जाता है, जैसे:

कर्नल्स (Kernels):

लाइनियर, पॉलीनॉमियल, और RBF जैसे विभिन्न कर्नल विकल्पों को आजमाकर डेटा के लिए उपयुक्त विभाजन प्राप्त किया जा सकता है।

मल्टी-क्लास क्लासिफिकेशन:

हालाँकि SVM मूल रूप से बाइनरी होता है, परंतु **वन-वर्सेस-वन (One-vs-One)** या **वन-वर्सेस-ऑल (One-vs-All)** तकनीकों से इसे मल्टी-क्लास समस्याओं पर लागू किया जा सकता है।

हाइपरपैरामीटर ट्यूनिंग:

C (रेगुलराइज़ेशन पैरामीटर), कर्नल कोएफिशिएंट (γ), आदि को ट्यून करके मॉडल की सटीकता और ओवरफिटिंग को नियंत्रित किया जा सकता है।

उदाहरण:

इमेज क्लासिफिकेशन टास्क जहाँ तस्वीरों को उनकी विशेषताओं के आधार पर विभिन्न श्रेणियों में वर्गीकृत करना होता है।

प्रोग्राम (उन्नत SVM एप्लिकेशन के लिए):

नीचे एक उदाहरण है जिसमें **नॉन-लाइनियर कर्नल** (जैसे RBF) और **हाइपरपैरामीटर ट्यूनिंग** को शामिल किया गया है।

```
from sklearn import datasets
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
# डेटासेट लोड करें
iris = datasets.load_iris()
X = iris.data
y = iris.target

# डेटा विभाजन
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
# फीचर्स को स्केल करना
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
# हाइपरपैरामीटर ग्रिड
param_grid = {
    'C': [0.1, 1, 10],
    'gamma': [1, 0.1, 0.01],
    'kernel': ['rbf']
```

```
}
# GridSearchCV का उपयोग करके सर्वश्रेष्ठ मॉडल ढूँढना
grid = GridSearchCV(SVC(), param_grid, refit=True, verbose=0)
grid.fit(X_train, y_train)
# परिणाम मूल्यांकन
print(f"Best parameters: {grid.best_params_}")
print(f"Model accuracy: {grid.score(X_test, y_test):.2f}")
```

यह प्रोग्राम SVM को नॉन-लाइनियर कर्नल (RBF) और ग्रिड सर्च के साथ लागू करता है, जिससे सटीकता और प्रदर्शन दोनों में सुधार होता है।

```
pip install scikit-learn
```

पायथन कोड

यह कोड यह दर्शाता है कि कैसे RBF कर्नल के साथ SVM का उपयोग किया जाता है और कैसे ग्रिड सर्च के माध्यम से सर्वश्रेष्ठ हाइपरपैरामीटर्स का चयन किया जाता है।

```
from sklearn import datasets
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import classification_report
# आइरिस डेटासेट लोड करें
iris = datasets.load_iris()
X = iris.data
y = iris.target
# डेटासेट को ट्रेनिंग और टेस्ट सेट में बाँटना
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
# फीचर स्केलिंग
scaler = StandardScaler()
```

```
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
# मॉडल परिभाषित करें
svc = SVC(kernel='rbf')
# हाइपरपैरामीटर सर्च सेट करें
param_grid = {
    'C': [0.1, 1, 10, 100],  # रेगुलराइज़ेशन पैरामीटर
    'gamma': ['scale', 'auto', 0.1, 0.01, 0.001],  # कर्नल कोएफिशिएंट
}
# ग्रिड सर्च करें
grid_search = GridSearchCV(svc, param_grid, cv=5, scoring='accuracy')
grid_search.fit(X_train_scaled, y_train)
# सर्वश्रेष्ठ पैरामीटर्स और स्कोर
print(f"Best parameters: {grid_search.best_params_}")
print(f"Best cross-validation score: {grid_search.best_score_:.2f}")
# टेस्ट सेट पर सर्वश्रेष्ठ मॉडल का मूल्यांकन
best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test_scaled)
print(classification_report(y_test, y_pred))
```

व्याख्या

· डेटासेट: इसमें आइरिस डेटासेट का उपयोग किया गया है, जो एक क्लासिक वर्गीकरण डेटासेट है।

· स्केलिंग: SVM के लिए फीचर स्केलिंग आवश्यक है, खासकर जब कर्नल का उपयोग किया जाता है, ताकि सभी फीचर्स दूरी की गणना में समान रूप से योगदान करें।

· SVC: SVC क्लास RBF कर्नल के साथ नॉन-लाइनियर डेटा को संभालने के लिए उपयोग की जाती है।

· हाइपरपैरामीटर ट्यूनिंग: GridSearchCV का उपयोग C और γ के सर्वश्रेष्ठ मान खोजने के लिए किया जाता है।

· मूल्यांकन: सर्वोत्तम पैरामीटर्स मिलने के बाद मॉडल को टेस्ट सेट पर जाँचकर उसके सामान्यीकरण प्रदर्शन का मूल्यांकन किया जाता है।

सपोर्ट वेक्टर मशीन (SVM)

सपोर्ट वेक्टर मशीनें (SVM) सुपरवाइज्ड लर्निंग एल्गोरिद्म का एक शक्तिशाली और बहुपरकीय समूह हैं, जिनका उपयोग वर्गीकरण, प्रतिगमन और आउट्लायर डिटेक्शन के लिए किया जाता है।

इनकी उच्च-आयामी डेटा को संभालने की क्षमता और विभिन्न डोमेनों में अच्छे प्रदर्शन के कारण SVM मशीन लर्निंग प्रैक्टिशनर्स के टूलकिट में एक आवश्यक उपकरण बन चुका है।

SVM की मुख्य ताकत इसके स्टैटिस्टिकल लर्निंग थ्योरी पर आधारित होने में है, जो जटिल समस्याओं में भी मज़बूत सैद्धांतिक गारंटी प्रदान करता है।

कर्नल विधियों के उपयोग से SVM फीचर स्पेस को उच्च-आयामों में बदल सकता है, जिससे वह नॉन-लाइनियर संबंधों को बिना इनपुट डेटा को स्पष्ट रूप से बदले, प्रभावी ढंग से संभाल सकता है।

SVM की कर्नल फ़ंक्शन के चयन में लचीलापन, और अनुकूलन (customization) की क्षमता इसे विशिष्ट अनुप्रयोगों के लिए अत्यंत उपयोगी बनाती है।

मल्टी-क्लास वर्गीकरण के लिए "वन-वर्सेस-वन" और "वन-वर्सेस-ऑल" जैसी रणनीतियाँ, और असंतुलित डेटा को संभालने के लिए विकसित विधियाँ, इसके अनुकूलन को और दर्शाती हैं।

हालाँकि, SVM में कुछ सीमाएँ भी हैं — जैसे कि पैरामीटर ट्यूनिंग की आवश्यकता, बड़े डेटासेट्स पर संभावित प्रदर्शन कमी, और निर्णय प्रक्रिया की व्याख्या में कठिनाई।

फिर भी, अनुकूलन तकनीकों में प्रगति इन चुनौतियों को लगातार कम कर रही है।

SVM मशीन लर्निंग के क्षेत्र में एक मजबूत, प्रभावशाली और बहुपरकीय तकनीक के रूप में स्थापित है, और भविष्य में भी डेटा विश्लेषण और पैटर्न पहचान से जुड़ी समस्याओं के समाधान में इसकी भूमिका प्रमुख बनी रहेगी।

इकाई-5: कृत्रिम बुद्धिमत्ता के अनुप्रयोग

5.1 प्राकृतिक भाषा प्रसंस्करण (Natural Language Processing - NLP)

प्राकृतिक भाषा प्रसंस्करण (NLP) कृत्रिम बुद्धिमत्ता, कंप्यूटर विज्ञान और भाषाविज्ञान के संगम पर स्थित है। इसका उद्देश्य मशीनों को मानव भाषा को समझने, व्याख्या करने और मूल्यवान ढंग से प्रतिक्रिया देने में सक्षम बनाना है। NLP में विभिन्न तकनीकों और उपकरणों का समावेश होता है जो कंप्यूटर को विशाल मात्रा में प्राकृतिक भाषा डेटा को संसाधित करने और विश्लेषण करने की क्षमता प्रदान करते हैं।

5.1.1 पाठ विश्लेषण और वर्गीकरण

चित्र 5.1.1: प्राविकण भाषा प्रसंस्करण (NLP) के अनुप्ररेयग

परिभाषा:

पाठ विश्लेषण और वर्गीकरण का तात्पर्य टेक्स्ट डेटा को इस प्रकार प्रोसेस करना है कि उसका अर्थ,

भावना या उद्देश्य समझा जा सके और उसे उपयुक्त श्रेणी में वर्गीकृत किया जा सके। यह कार्य भावनात्मक विश्लेषण, विषय पहचान और स्पैम छँटाई जैसे क्षेत्रों में अत्यंत महत्वपूर्ण है।

उदाहरण:

ग्राहकों की समीक्षाओं का विश्लेषण कर उन्हें सकारात्मक, नकारात्मक या तटस्थ भावनाओं में वर्गीकृत करना। व्यवसाय इस जानकारी का उपयोग उत्पाद की गुणवत्ता या सेवा वितरण में सुधार हेतु करते हैं।

5.1.2 मशीन अनुवाद (Machine Translation)

चित्र 5.1.2 एन.एल.पी. के अनुप्रयग

परिभाषा:

मशीन अनुवाद किसी भाषा के पाठ या भाषण को सॉफ़्टवेयर की सहायता से दूसरी भाषा में अनुवादित करने की प्रक्रिया है। इसमें NLP और संगणकीय भाषाविज्ञान का प्रयोग कर स्रोत भाषा के अर्थ को समझा जाता है और लक्ष्य भाषा में अनुकूल पाठ उत्पन्न किया जाता है।

उदाहरण:

Google Translate उपयोगकर्ताओं को वेबपेज, दस्तावेज़ और वार्तालापों का कई भाषाओं में रीयल-टाइम अनुवाद करने में सहायता करता है, जिससे भाषा की बाधा समाप्त होती है।

5.1.3 वाक् पहचान (Speech Recognition)रिभाषा:

चित्र 5.1.3: ध्वनि इनपुट को कैप्चर करने, ऑडियो संकेत को संसाधित करने और उसे पाठ प्रारूप में परिवर्तित करने की प्रक्रिया दर्शाने वाला आरेख

वाक् पहचान तकनीक बोले गए शब्दों को टेक्स्ट में परिवर्तित करती है। इसमें ध्वनि तरंगों का विश्लेषण, भाषाई संरचना की व्याख्या और वाक् को लिखित शब्दों में रूपांतरित करना शामिल होता है।

उदाहरण:

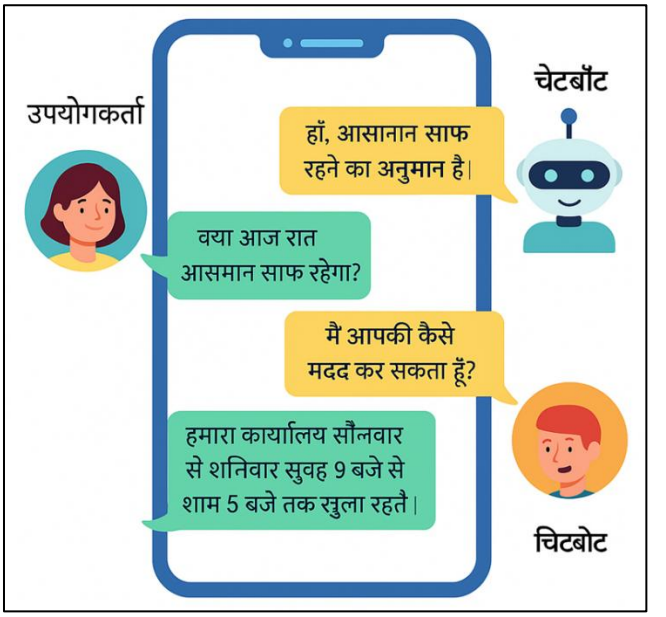
वॉयस-ऐक्टिवेटेड वर्चुअल असिस्टेंट जैसे Amazon Alexa, Apple Siri और Google Assistant उपयोगकर्ता के निर्देशों को समझने और उनके अनुसार कार्य करने हेतु वाक् पहचान का प्रयोग करते हैं।

5.1.4 चैटबॉट्स और वर्चुअल सहायक

परिभाषा:

चैटबॉट्स और वर्चुअल असिस्टेंट ऐसे AI-संचालित प्रोग्राम होते हैं जो टेक्स्ट या आवाज़ के माध्यम से मानव

संवाद का अनुकरण करते हैं। ये उपयोगकर्ता की क्वेरी को समझने और उपयुक्त, संदर्भानुकूल उत्तर देने के लिए NLP का उपयोग करते हैं।



चित्र 5.1.4: उपयोगकर्ता और चैटबॉट के बीच एक काल्पनिक संवाद, जो विभिन्न प्रश्नों को समझने और उत्तर देने की चैटबॉट की क्षमता को दर्शाता है।

उदाहरण:

वेबसाइट्स पर कस्टमर सर्विस चैटबॉट्स 24/7 सहायता प्रदान करते हैं, सामान्य प्रश्नों के उत्तर देते हैं और उपयोगकर्ताओं की समस्याओं का समाधान करते हैं, जिससे ग्राहक अनुभव और संचालन दक्षता दोनों में सुधार होता है।

5.1.5 प्राकृतिक भाषा निर्माण (Natural Language Generation - NLG)

परिभाषा:

प्राकृतिक भाषा निर्माण एक ऐसी प्रक्रिया है जिसमें संरचित डेटा को पढ़ने योग्य, अर्थपूर्ण और सहज भाषा में रूपांतरित किया जाता है। इसका उद्देश्य ऐसा टेक्स्ट उत्पन्न करना होता है जिसे मनुष्य आसानी से पढ़ और समझ सकें।

उदाहरण:

वित्तीय आँकड़ों या खेल परिणामों पर स्वचालित समाचार रिपोर्टिंग, जहाँ NLG प्रणाली कच्चे डेटा से समाचार लेख तैयार करती है जो प्रमुख तथ्यों और आँकड़ों का सार प्रस्तुत करते हैं।

5.2 पाठ वर्गीकरण और सूचना पुनःप्राप्ति

प्रस्तावना

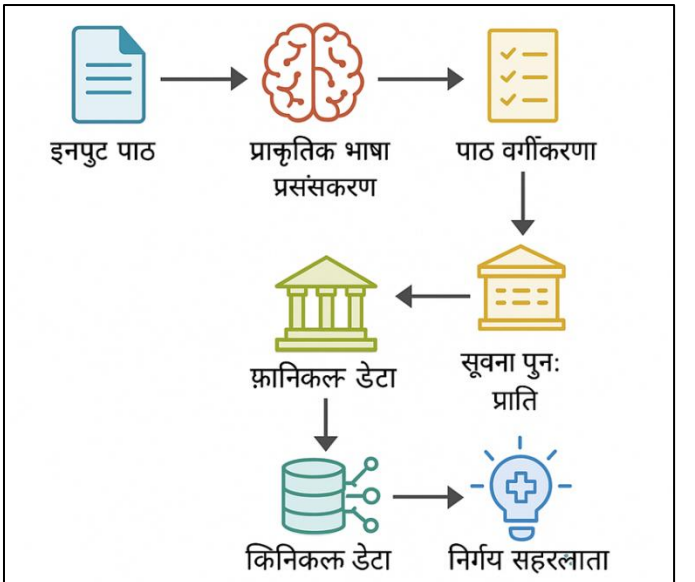
डिजिटल स्वास्थ्य सेवा के युग में, असंरचित पाठ से सार्थक जानकारी निकालने की क्षमता सटीक चिकित्सा (Precision Medicine) की एक आधारशिला बन चुकी है।

पाठ वर्गीकरण और सूचना पुनःप्राप्ति (Information Retrieval - IR) चिकित्सा साहित्य, इलेक्ट्रॉनिक स्वास्थ्य रिकार्ड्स (EHRs) और क्लिनिकल नोट्स को व्यवस्थित करने में महत्वपूर्ण भूमिका निभाते हैं, जिससे समय पर और प्रमाण-आधारित निर्णय लिए जा सकें।

कृत्रिम बुद्धिमत्ता (AI), विशेष रूप से प्राकृतिक भाषा प्रसंस्करण (NLP) और मशीन लर्निंग (ML) तकनीकों की मदद से, अब स्वास्थ्य प्रणाली स्वतः दस्तावेजों का वर्गीकरण कर सकती है, रोगी-संबंधी डेटा पुनः प्राप्त कर सकती है और पाठ्य स्वरूपों के आधार पर रोग परिणामों की भविष्यवाणी भी कर सकती है।

यह अध्याय यह दर्शाता है कि AI-आधारित पाठ वर्गीकरण और IR मॉडल कैसे स्वास्थ्य में सूचना की पहुँच, निदान और वैयक्तिक देखभाल को बेहतर बना रहे हैं।

5.2.1 स्वास्थ्य सेवा में पाठ वर्गीकरण की मूल बातें



चित्र 5.2.1: स्वास्थ्य सेवा में **AI** आभाधीत पाठ वर्गीकरण और सूचना पुनःप्राति की वैचाारिक प्रवाह

पाठ वर्गीकरण में टेक्स्ट डेटा को पूर्वनिर्धारित श्रेणियों में बाँटना शामिल है। स्वास्थ्य क्षेत्र में यह कार्य चिकित्सा साहित्य, रोगी के लक्षणों या क्लिनिकल संदेशों को श्रेणीबद्ध करने के लिए किया जाता है।

प्रयुक्त AI तकनीकें:

- सुपरवाइज्ड लर्निंग (जैसे: सपोर्ट वेक्टर मशीन, रैंडम फॉरेस्ट, डीप लर्निंग)
- ट्रांसफॉर्मर मॉडल्स (जैसे: BERT, BioBERT, ClinicalBERT)

उदाहरण:

एक अस्पताल की आपातकालीन प्रतिक्रिया प्रणाली ऐतिहासिक ट्रायाज डेटा पर प्रशिक्षित रीयल-टाइम क्लासिफायर का उपयोग करती है जो आने वाले SMS-आधारित शिकायतों को स्वतः श्रेणीबद्ध करता है। "छाती में दर्द" या "साँस लेने में कठिनाई" जैसे कीवर्ड पहचानकर, यह गंभीर संदेशों को चिन्हित कर सही चिकित्सा कर्मियों तक पहुँचाता है।

स्वास्थ्य सेवा में लाभ:

- निदान में देरी को कम करता है
- रिपोर्ट निर्माण को स्वचालित करता है
- त्वरित प्राथमिकता निर्धारण के माध्यम से रोगी की सुरक्षा बढ़ाता है

5.2.2 क्लिनिकल डेटा से AI-सक्षम सूचना पुनःप्राप्ति

सूचना पुनःप्राप्ति (Information Retrieval) का उद्देश्य किसी उपयोगकर्ता की क्वेरी के जवाब में बड़े डेटा भंडार से उपयुक्त जानकारी निकालना है।

AI-सक्षम IR सिस्टम चिकित्सकों को मैनुअल डेटा छँटाई के बिना संदर्भित चिकित्सा ज्ञान या रोगी इतिहास उपलब्ध कराते हैं।

मुख्य तकनीकें:

- सैमांटिक सर्च
- नॉलेज ग्राफ़्स
- कॉन्टेक्स्चुअल एम्बेडिंग्स (जैसे: word2vec, GloVe, ट्रांसफॉर्मर)

उदाहरण:

एक चिकित्सक "65 वर्ष से अधिक उम्र के मधुमेहग्रस्त रोगियों में मेटास्टैटिक स्तन कैंसर के लिए उपचार

दिशा-निर्देश" पूछता है। AI-सक्षम प्रणाली उसे सर्वोत्तम विश्वसनीयता और प्रासंगिकता के आधार पर साहित्य और उपचार मार्ग प्रदान करती है, जबकि पारंपरिक कीवर्ड-आधारित सिस्टम इतनी विशिष्टता में विफल रहते हैं।

सटीक स्वास्थ्य सेवा में उपयोग:

- पूर्व निदान पैटर्न की पुनःप्राप्ति
- प्रमाण-आधारित उपचार के लिए साहित्य समीक्षा
- शोध के लिए रोगी समूहों की पहचान

5.2.3 इलेक्ट्रॉनिक स्वास्थ्य रिकॉर्ड्स (EHRs) के साथ एकीकरण

एक प्रमुख प्रगति EHR सिस्टम में क्लासिफिकेशन और IR मॉडल्स को एम्बेड करना है, जिससे क्लिनिकल नोट्स, लैब रिपोर्ट्स, प्रिस्क्रिप्शन और रेडियोलॉजी सारांश से जानकारी निकाली जा सके।

उदाहरण प्रणाली:

Mount Sinai का DeepPatient मॉडल डीप लर्निंग को EHR डेटा के साथ संयोजित करता है, जो लक्षणों को वर्गीकृत कर और पिछले रिकॉर्ड पुनः प्राप्त कर रोग की भविष्यवाणी करता है।

परिणाम:

- मैन्युअल चार्ट रिव्यू की आवश्यकता घटती है
- उच्च जोखिम वाले रोगियों की पहचान होती है
- सक्रिय देखभाल हस्तक्षेप संभव होता है

5.2.4 कार्यान्वयन में चुनौतियाँ

इन तकनीकों की संभावनाओं के बावजूद, इनका क्रियान्वयन कुछ प्रमुख चुनौतियों का सामना करता है:

मुख्य चुनौतियाँ:

- **डेटा गुणवत्ता समस्याएँ:** क्लिनिकल टेक्स्ट अधूरा, असंगत और जटिल शब्दावली से युक्त होता है
- **पूर्वाग्रह और निष्पक्षता:** सीमित या पक्षपाती डेटा पर प्रशिक्षित मॉडल अनुचित परिणाम दे सकते हैं
- **स्पष्टीकरण की कमी:** ब्लैक-बॉक्स मॉडल पारदर्शिता नहीं देते, जो उच्च जोखिम वाले क्षेत्रों में चिंता का विषय बनता है

समाधान:

- डोमेन-विशिष्ट ऑंटोलॉजी जैसे SNOMED-CT और UMLS का समावेश
- चिकित्सकों के लिए स्पष्टीकरण योग्य AI (XAI) तकनीकों का उपयोग

तालिका 5.1: पारंपरिक बनाम AI-आधारित टेक्स्ट प्रोसेसिंग की तुलना

विशेषता	पारंपरिक विधियाँ	AI-आधारित विधियाँ
प्रोसेसिंग पद्धति	नियम-आधारित	डेटा-आधारित (ML/DL)
स्केलेबिलिटी	सीमित	अत्यधिक स्केलेबल
जटिल परिप्रेक्ष्य में सटीकता	मध्यम	उच्च (विशेषतः ट्रांसफॉर्मर्स के साथ)
भाषा समझ	शाब्दिक	संदर्भ-सम्मत (सैमांटिक)
नए उपयोग मामलों में अनुकूलन	कम	अधिक (ट्रांसफर लर्निंग द्वारा)

5.2.5 उभरते रुझान और भविष्य की दिशा

स्वास्थ्य सेवा में पाठ वर्गीकरण और सूचना पुनःप्राप्ति का भविष्य अधिक एकीकृत, संवादात्मक AI सिस्टम और संस्थानों के बीच फेडरेटेड लर्निंग में निहित है, जिससे मॉडल की सामान्यीकरण क्षमता में वृद्धि हो सके।

प्रमुख रुझान:

- **संवादात्मक एजेंट:** ऐसे चैटबॉट्स जो AI मॉडल द्वारा संचालित होते हैं और रोगी की पूछताछ को समझने व वर्गीकृत करने में सक्षम होते हैं।
- **रीयल-टाइम क्लिनिकल निर्णय सहायता:** त्वरित रूप से केस-आधारित डेटा पुनःप्राप्त कर निदान या दवा निर्णय में सहायता प्रदान करना।
- **बहुभाषी मेडिकल NLP:** विभिन्न क्षेत्रीय भाषाओं को मॉडल प्रशिक्षण में शामिल कर विविध जनसंख्या में समावेश सुनिश्चित करना।

उदाहरण नवाचार:

Google Health का Med-PaLM, एक बड़ा भाषा मॉडल जो मेडिकल प्रश्नोत्तर डेटा पर प्रशिक्षित है, क्लिनिकल प्रश्नों के उत्तर देने में लगभग विशेषज्ञ स्तर का प्रदर्शन करता है। यह वैश्विक परिप्रेक्ष्य में AI-सहायता प्राप्त निदान की नींव रखता है।

निष्कर्ष

पाठ वर्गीकरण और सूचना पुनःप्राप्ति AI के उस मोर्चे का प्रतिनिधित्व करते हैं जो स्वास्थ्य सेवा में क्रांति लाने की दिशा में अग्रसर है।

इनकी क्षमता असंरचित क्लिनिकल डेटा को व्यवस्थित करने और उसमें से अर्थ निकालने में सीधे तौर पर समयबद्ध, सूचित और सटीक चिकित्सा हस्तक्षेप में योगदान देती है।

इन AI क्षमताओं को EHR सिस्टम, क्लिनिकल वर्कफ़्लो और निर्णय लेने वाले उपकरणों में एकीकृत कर, स्वास्थ्य सेवा प्रदाता आधुनिक चिकित्सा की जटिलताओं का बेहतर समाधान कर सकते हैं।

जैसे-जैसे अनुसंधान आगे बढ़ रहा है—विशेष रूप से बहुभाषी NLP और स्पष्टीकरण योग्य मॉडल्स में—AI का पाठ विश्लेषण में एकीकरण, सुलभ, न्यायसंगत और रोगी-केंद्रित सटीक स्वास्थ्य देखभाल का उत्प्रेरक सिद्ध होगा।

5.3 वाक् पहचान (Speech Recognition)

प्रस्तावना

वाक् पहचान ने सटीक स्वास्थ्य सेवा (Precision Healthcare) में कृत्रिम बुद्धिमत्ता (AI) के एक परिवर्तनकारी अनुप्रयोग के रूप में उभरकर मानव और मशीन के बीच आवाज़ के माध्यम से सहज संपर्क को सक्षम किया है।

ऐसे क्षेत्र में जहाँ समय पर और सटीक संचार अत्यंत महत्वपूर्ण होता है, AI-संचालित वाक् पहचान तकनीकों ने क्लिनिकल दस्तावेज़ीकरण, रोगी सहभागिता और देखभाल की सुलभता को नए आयाम दिए हैं।

ये प्रणालियाँ चिकित्सक-रोगी वार्तालाप को ट्रांसक्राइब कर सकती हैं, दृष्टिबाधित रोगियों की सहायता कर सकती हैं, और चिकित्सा वर्कफ़्लो को सरल बना सकती हैं, जिससे प्रशासनिक बोझ कम होता है और क्लिनिकल दक्षता बढ़ती है।

जैसे-जैसे प्राकृतिक भाषा प्रसंस्करण और डीप लर्निंग विकसित हो रहे हैं, वाक् पहचान तकनीकें अधिक परिष्कृत होती जा रही हैं—जो विभिन्न उच्चारणों, चिकित्सा शब्दावली और संदर्भगत भिन्नताओं को समझने में सक्षम हैं।

यह अध्याय वाक् पहचान की संरचना, अनुप्रयोगों और प्रभावों का विश्लेषण करता है।

5.3.1 वाक् पहचान के सिद्धांत और तकनीकें

वाक् पहचान प्रणाली बोले गए शब्दों को पाठ में बदलती है, जिसके लिए ध्वनिक, भाषाई और सांख्यिकीय मॉडलों की श्रृंखला का उपयोग किया जाता है।

इस प्रक्रिया में चार मुख्य चरण होते हैं:

1. ध्वनि संकेत अधिग्रहण
2. विशेषताओं का निष्कर्षण
3. पैटर्न पहचान
4. प्राकृतिक भाषा की समझ

प्रमुख AI तकनीकें:

- ध्वनिक मॉडलिंग के लिए डीप न्यूरल नेटवर्क्स (DNNs)

- अनुक्रमिक डेटा के लिए RNNs और LSTM
- बहुभाषी और चिकित्सा शब्दावली की पहचान के लिए ट्रांसफॉर्मर मॉडल (जैसे: OpenAI का Whisper)

व्यवहार में उदाहरण:

Nuance का Dragon Medical One वाक् पहचान मॉडल का उपयोग करके चिकित्सकों के डिक्टेशन को सीधे EHR सिस्टम में ट्रांसक्राइब करता है, जिससे दस्तावेज़ीकरण का औसत समय 45% से अधिक कम हो जाता है।

स्वास्थ्य सेवा में लाभ:

- तेज़ क्लिनिकल दस्तावेज़ीकरण
- विकलांग रोगियों के साथ बेहतर संवाद
- सर्जिकल रोबोट या डायग्नोस्टिक उपकरणों के लिए वास्तविक समय में वॉइस कमांड

चित्र 5.3.1 : क्लिनिकल सेटिंग में वाक् पहचान की कार्यप्रणाली को दर्शाने वाला प्रवाह आरेख — वॉइस इनपुट से संरचित EHR प्रविष्टि तक

5.3.2 क्लिनिकल वर्कफ़्लो में अनुप्रयोग

वाक् पहचान अब कई क्लिनिकल स्पर्श बिंदुओं पर एकीकृत की जा चुकी है—प्रशासनिक कार्यों से लेकर बिस्तर के पास देखभाल तक।

उपयोग मामला 1: क्लिनिकल दस्तावेज़ स्वचालन

डॉक्टर परामर्श के दौरान या बाद में मौखिक रूप से नोट्स बोल सकते हैं। AI इस डेटा को ट्रांसक्राइब करता है, प्रासंगिक जानकारी छाँटता है और EHR को स्वतः अपडेट करता है।

उपयोग मामला 2: वॉइस-सहायता प्राप्त निदान इंटरफ़ेस

रेडियोलॉजिस्ट मौखिक रूप से पिछले स्कैन या रोगी इतिहास का अनुरोध कर सकते हैं जबकि वे वर्तमान छवियों की व्याख्या कर रहे होते हैं—जिससे निदान पर ध्यान बना रहता है।

उपयोग मामला 3: रोगी वॉइस असिस्टेंट्स

वॉइस-सक्षम चैटबॉट्स रोगियों को अपॉइंटमेंट बुक करने, दवाइयों की रिफ़िल कराने और दवा के निर्देश प्राप्त करने में सहायता करते हैं, वह भी स्क्रीन के बिना।

5.3.3 समावेशन और सुलभता हेतु वाक् पहचान

वाक् पहचान उन रोगियों के लिए स्वास्थ्य देखभाल की पहुंच बढ़ाने में महत्वपूर्ण भूमिका निभाती है, जिन्हें शारीरिक, दृष्टि या साक्षरता की बाधाएँ होती हैं।

उदाहरण:

ग्रामीण क्षेत्रों में JioHealthHub जैसे AI-संचालित मोबाइल ऐप वॉइस-टू-टेक्स्ट इंटरफ़ेस का उपयोग करके अर्ध-साक्षर उपयोगकर्ताओं को उनकी स्थानीय भाषा में स्वास्थ्य सेवाओं से जोड़ते हैं। ये प्रणालियाँ उनके बोले गए लक्षणों की व्याख्या करती हैं और उन्हें प्राथमिक स्वास्थ्य मार्गदर्शन देती हैं या किसी विशेषज्ञ से जोड़ती हैं।

समावेशी प्रभाव:

- वृद्ध रोगियों को संवाद सरल बनाकर सहायता मिलती है
- दृष्टिबाधित उपयोगकर्ता जानकारी को हैंड्स-फ़्री एक्सेस कर सकते हैं
- अविकसित क्षेत्रों में भाषाई और साक्षरता की खाई को पाटता है

तालिका 5.3.1: मैनुअल बनाम AI-आधारित वाक् पहचान की तुलना (क्लिनिकल सेटिंग में)

विशेषता	मैनुअल ट्रांसक्रिप्शन	AI-आधारित वाक् पहचान
समय दक्षता	समय-खपत	वास्तविक समय
त्रुटि दर	मानव सटीकता पर निर्भर	प्रशिक्षण और ट्यूनिंग के साथ घटती है
लागत	अधिक (मानव संसाधन)	बड़े पैमाने पर किफायती
EHR के साथ एकीकरण	मैनुअल डेटा एंट्री	सीधा एकीकरण संभव
बहुभाषीय क्षमता	सीमित	AI मॉडल्स के साथ विस्तारित हो रही है

5.3.4 चुनौतियाँ और सीमाएँ

उल्लेखनीय प्रगति के बावजूद, वास्तविक स्वास्थ्य सेवा परिवेश में वाक् पहचान प्रणालियाँ कुछ सीमाओं का सामना करती हैं:

- **पर्यावरणीय शोर हस्तक्षेप:** अस्पताल का वातावरण अक्सर शोरगुल वाला होता है, जिससे सटीकता पर प्रभाव पड़ता है
- **उच्चारण और बोली में विविधता:** सिस्टम क्षेत्रीय लहजे को पहचानने में कठिनाई महसूस कर सकते हैं यदि उन्हें विशेष रूप से प्रशिक्षित न किया जाए
- **चिकित्सा शब्दों की गलत व्याख्या:** नए या असामान्य चिकित्सा शब्दों को पहचानने में गलती हो सकती है
- **गोपनीयता और अनुपालन:** वॉइस के माध्यम से संवेदनशील स्वास्थ्य डेटा को संभालना HIPAA/GDPR जैसे मानकों की दृष्टि से चुनौतीपूर्ण हो सकता है

निरंतर अनुसंधान की दिशा:

- शोर-प्रतिरोधी मॉडल्स का विकास
- फेडरेटेड लर्निंग द्वारा मॉडल का व्यक्तिगत अनुकूलन
- चिकित्सा भाषा की गलत व्याख्या को कम करने हेतु सन्दर्भ-सम्मत AI का एकीकरण

निष्कर्ष

वाक् पहचान AI नवाचार और मानव-केंद्रित स्वास्थ्य सेवा वितरण के संगम पर खड़ी है। आवाज को क्रियात्मक डेटा में बदलने की इसकी क्षमता ने दस्तावेज़ीकरण की दक्षता को बढ़ाया है, रोगियों की पहुंच को सुधारा है और समावेशी देखभाल के अनुभव को सशक्त किया है।

जैसे-जैसे AI मॉडल अधिक संदर्भ-सम्मत, बहुभाषीय और अनुकूलनीय बनते हैं, वॉइस इंटरफ़ेस स्वास्थ्य सेवा वितरण को अधिक स्वाभाविक, न्यायसंगत और उत्तरदायी बनाते रहेंगे।

वर्तमान सीमाओं को संबोधित करते हुए यदि सुरक्षित और स्केलेबल कार्यान्वयन सुनिश्चित किया जाए, तो वाक् पहचान आने वाले वर्षों में सटीक स्वास्थ्य सेवा के विकास में एक आधारशिला सिद्ध होगी।

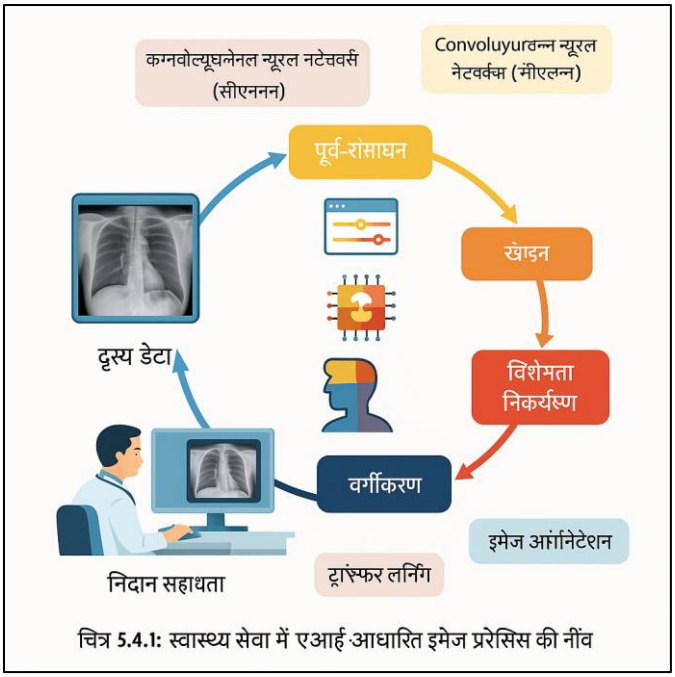
5.4 इमेज प्रोसेसिंग और कंप्यूटर विज़न

प्रस्तावना

इमेज प्रोसेसिंग और कंप्यूटर विज़न ने स्वास्थ्य सेवा में क्रांतिकारी बदलाव लाया है, जिससे मशीनें पहले से कहीं अधिक सटीकता और दक्षता के साथ दृश्य डेटा की व्याख्या कर सकती हैं। कृत्रिम बुद्धिमत्ता (AI), विशेष रूप से डीप लर्निंग और कन्वोल्यूशनल न्यूरल नेटवर्क्स (CNNs) के एकीकरण के माध्यम से, अब स्वास्थ्य प्रणालियाँ चिकित्सा छवियों का विश्लेषण कर सकती हैं, असामान्यताओं की पहचान कर सकती हैं, रोगों का वर्गीकरण कर सकती हैं, और निदान तथा उपचार योजना में चिकित्सकों की सहायता कर सकती हैं।

ये तकनीकें रेडियोलॉजी, पैथोलॉजी, डर्मेटोलॉजी और नेत्र विज्ञान जैसे विविध क्षेत्रों में अनुप्रयोग पाती हैं और सटीक स्वास्थ्य देखभाल के लिए अत्यंत आवश्यक बन चुकी हैं।

AI द्वारा संचालित स्वचालित और वास्तविक समय की छवि विश्लेषण क्षमता, डेटा अधिग्रहण और क्लिनिकल इनसाइट के बीच की दूरी को कम करती है, जिससे शीघ्र हस्तक्षेप, व्यक्तिगत उपचार और बेहतर रोगी परिणाम संभव होते हैं।



चित्र 5.4.1: स्वास्थ्य सेवा में एआई-आधारित इमेज प्रेसिस की नींव

5.4.1 स्वास्थ्य सेवा में AI-आधारित इमेज प्रोसेसिंगकी नींव

AI-आधारित इमेज प्रोसेसिंग में दृश्य डेटा का विश्लेषण और हेरफेर करके उपयोगी विशेषताएँ और पैटर्न निकाले जाते हैं। इस प्रक्रिया में सामान्यतः पूर्व-संसाधन (preprocessing), खंडन (segmentation), विशेषता निष्कर्षण और वर्गीकरण शामिल होते हैं, जो मशीन लर्निंग एल्गोरिद्म के माध्यम से अनुकूलित किए जाते हैं।

प्रमुख तकनीकें:

- **Convolutional Neural Networks (CNNs):** अधिकांश मेडिकल इमेज वर्गीकरण और खंडन मॉडल का मूल
- **Image Augmentation:** मॉडल प्रशिक्षण के लिए डेटा विविधता बढ़ाता है
- **Transfer Learning:** सीमित लेबल वाले मेडिकल डेटा पर पूर्व-प्रशिक्षित मॉडलों का पुनः उपयोग संभव बनाता है

उदाहरण:

Google Health द्वारा विकसित एक सिस्टम ने छाती के एक्स-रे पर प्रशिक्षित AI मॉडल का उपयोग करते हुए निमोनिया, टीबी और COVID-19 संबंधी असामान्यताओं का पता लगाने में रेडियोलॉजिस्ट्स के समान प्रदर्शन दिखाया और 300 से अधिक अनदेखे मामलों को चिह्नित किया।

लाभ:

- निदान में त्रुटियाँ कम होती हैं
- तेजी से छवि व्याख्या
- वस्तुनिष्ठ और दोहराने योग्य मूल्यांकन

5.4.2 रोग पहचान और निगरानी में कंप्यूटर विज़न

कंप्यूटर विज़न मशीनों को "देखने" और चिकित्सा छवियों या लाइव वीडियो स्ट्रीम का विश्लेषण करने में सक्षम बनाता है।

उपयोग मामला 1: कैंसर की पहचान और स्टेजिंग

कंप्यूटर विज़न एल्गोरिद्म हिस्टोपैथोलॉजिकल स्लाइड्स में ट्यूमर का पता लगा सकते हैं, सीमाएँ मूल्यांकित कर सकते हैं और कैंसर की आक्रामकता का अनुमान भी लगा सकते हैं।

उदाहरण: PathAI का डीप लर्निंग प्लेटफ़ॉर्म ब्रेस्ट और प्रोस्टेट कैंसर के निदान में पैथोलॉजिस्ट की सहायता करता है।

उपयोग मामला 2: डायबिटिक रेटिनोपैथी स्क्रीनिंग

IDx-DR जैसे AI मॉडल रेटिना छवियों का उपयोग करके डायबिटिक रेटिनोपैथी का स्वतः निदान करते हैं — प्राथमिक देखभाल केंद्रों में नेत्र विशेषज्ञ की आवश्यकता के बिना।

उपयोग मामला 3: सर्जिकल नेविगेशन

एंडोस्कोपिक या लैप्रोस्कोपिक प्रक्रियाओं से वास्तविक समय में दृश्य डेटा का विश्लेषण कंप्यूटर विज़न द्वारा किया जाता है ताकि सर्जिकल टूल्स को मार्गदर्शन मिल सके, शारीरिक संरचनाएँ पहचानी जा सकें और जटिलताओं से बचा जा सके।

5.4.3 वास्तविक दुनिया में कार्यान्वयन और लाभ

AI-संचालित कंप्यूटर विज़न सिस्टम अब अस्पतालों, क्लीनिकों और डायग्नोस्टिक लैब्स में बड़े पैमाने पर उपयोग हो रहे हैं।

केस स्टडी: Stanford's CheXNet

100,000+ छाती एक्स-रे पर प्रशिक्षित 121-लेयर CNN मॉडल ने निमोनिया का पता लगाने में रेडियोलॉजिस्ट्स से बेहतर प्रदर्शन किया।
अब यह टूल आपातकालीन कक्षों में त्वरित ट्रायेज निर्णय लेने के लिए उपयोग किया जा रहा है।

केस स्टडी: DermAssist द्वारा त्वचा कैंसर की पहचान

Google का मोबाइल ऐप त्वचा के घावों की छवियों का विश्लेषण करता है और संभावित कैंसर की भविष्यवाणी करता है — विशेष रूप से दूरदराज़ क्षेत्रों में दूरस्थ निदान में सहायक।

तालिका 5.4.1: पारंपरिक बनाम AI-आधारित इमेज विश्लेषण की तुलना

पैरामीटर	पारंपरिक व्याख्या	AI-आधारित विश्लेषण
गति	धीमी (मैनुअल समीक्षा)	तात्कालिक या लगभग तात्कालिक
सटीकता	मानव अनुभव पर निर्भर	प्रशिक्षण के बाद लगातार उच्च
स्केलेबिलिटी	सीमित (मानव संसाधन निर्भर)	उच्च, स्थानों में आसानी से विस्तारित
पूर्वाग्रह और थकान	उपस्थित	अनुपस्थित (डेटा पूर्वाग्रह हो सकता है)
स्वास्थ्य प्रणाली एकीकरण	मैनुअल एंट्री की आवश्यकता	EHR, PACS के साथ सहज एकीकरण

5.4.4 अपनाने की चुनौतियाँ और भविष्य की दि

तेजी से प्रगति के बावजूद, AI का चिकित्सा इमेजिंग में व्यापक उपयोग कुछ प्रमुख चुनौतियों का सामना करता है:

- **डेटा गोपनीयता और अनुपालन:** चिकित्सा छवियाँ संवेदनशील होती हैं; सुरक्षित प्रोसेसिंग आवश्यक है
- **मॉडल की सामान्यता:** विशेष आबादी पर प्रशिक्षित मॉडल अन्य जनसांख्यिकीय पर सटीक काम नहीं कर सकते
- **नियामक अनुमोदन:** क्लिनिकल-ग्रेड सिस्टम के लिए कठोर मान्यता और प्रमाणीकरण की आवश्यकता होती है
- **व्याख्यात्मकता की कमी:** कई डीप लर्निंग मॉडल "ब्लैक बॉक्स" के रूप में काम करते हैं, जिससे चिकित्सकों का विश्वास कम होता है

उभरते समाधान:

- **Explainable AI (XAI):** मॉडल के निर्णयों को विज़ुअलाइज़ करना
- **Federated Learning:** बिना रॉ डेटा साझा किए प्रदर्शन में सुधार
- **हाइब्रिड सिस्टम:** AI और चिकित्सक की देखरेख का संयोजन

निष्कर्ष

इमेज प्रोसेसिंग और कंप्यूटर विज़न चिकित्सा निदान के परिदृश्य को तेजी से बदल रहे हैं, जो बेजोड़ सटीकता, गति और स्केलेबिलिटी प्रदान करते हैं।

रोगों की प्रारंभिक पहचान से लेकर वास्तविक समय सर्जिकल प्रक्रियाओं तक, ये तकनीकें स्वास्थ्य पेशेवरों की क्षमताओं को बढ़ाती हैं और देखभाल में असमानताओं को कम करती हैं।

जैसे-जैसे मॉडल पारदर्शिता, नियामक अनुपालन और डेटा सुरक्षा में नवाचार होता है, AI-संचालित विज़न सिस्टम वैश्विक क्लिनिकल वर्कफ़्लो का एक अभिन्न हिस्सा बनेंगे।

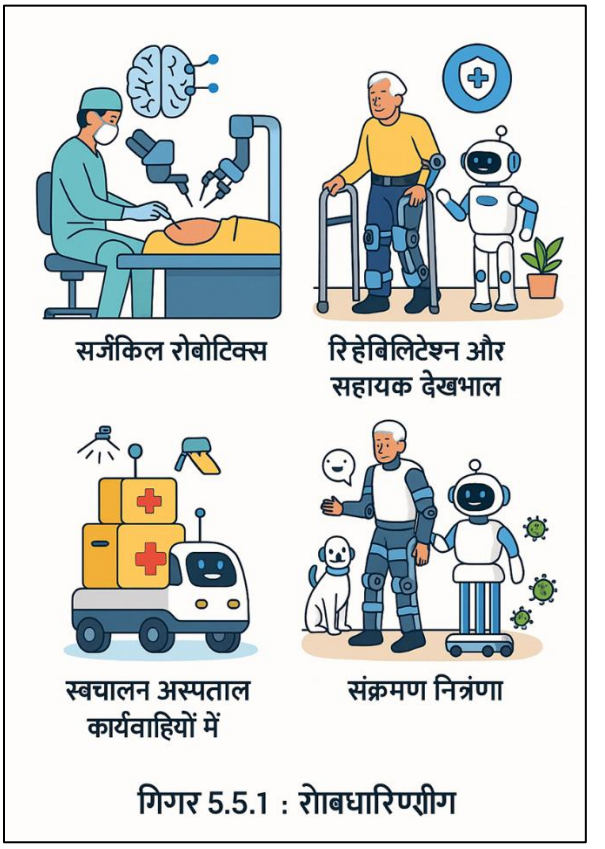
इनका निरंतर परिष्करण वास्तव में व्यक्तिगत और पूर्वानुमेय स्वास्थ्य देखभाल का वादा करता है, जो सटीक चिकित्सा के लक्ष्यों के साथ पूरी तरह से मेल खाता है।

5.5 रोबोटिक्स

परिचय

कृत्रिम बुद्धिमत्ता (AI) से सशक्त रोबोटिक्स ने आधुनिक स्वास्थ्य सेवा में क्रांतिकारी परिवर्तन ला दिया है, जिससे जटिल कार्यों को सटीकता, दक्षता और निरंतरता के साथ किया जा सकता है। सर्जरी से लेकर पुनर्वास और वृद्ध देखभाल तक, AI-चालित रोबोट स्वास्थ्य सेवा के तरीके को फिर से परिभाषित कर रहे हैं। यह तकनीक न्यूनतम इनवेसिव प्रक्रियाओं को संभव बनाती है, दोहराए जाने वाले कार्यों का स्वचालन करती है और रोगियों के साथ संवाद को बेहतर बनाती है। जैसे-जैसे हम डेटा-संचालित चिकित्सा की ओर बढ़ते हैं, रोबोटिक्स स्केलेबल, सुरक्षित और रोगी-केंद्रित देखभाल वितरण का एक प्रमुख आधार बनता जा रहा है।

5.5.1 AI-सक्षम सर्जिकल रोबोटिक्स



गिगर 5.5.1 : रोाबधारिण्णीग

स्वास्थ्य सेवा में रोबोटिक्स का सबसे चर्चित उपयोग सर्जरी में हुआ है, जहां AI के एकीकरण ने सटीकता, सुरक्षा और परिणामों को नई ऊंचाइयों पर पहुँचाया है।

मुख्य तकनीकें:

- रोगी की गति की भरपाई हेतु AI-आधारित गति भविष्यवाणी
- बेहतर नियंत्रण और जागरूकता हेतु कंप्यूटर विज़न और हैप्टिक्स
- प्रक्रियात्मक योजना और निर्णय सहायता हेतु मशीन लर्निंग एल्गोरिद्म

उदाहरण: डा विंची सर्जिकल सिस्टम

यह प्लेटफ़ॉर्म AI की मदद से सर्जनों को उच्च गति नियंत्रण और 3D दृश्यता प्रदान करता है। यह यूरोलॉजी, कार्डियोलॉजी और गायनोकोलॉजी जैसी विशेषताओं में न्यूनतम इनवेसिव सर्जरी में सहायक है।

लाभ:

- सीमित स्थानों में भी उच्च दक्षता से सर्जरी
- रियल-टाइम फ़ीडबैक
- संक्रमण की दर में कमी और कम अस्पताल प्रवास

5.5.2 पुनर्वास और सहायक देखभाल में रोबोटिक्स

AI-आधारित रोबोट अब पुनर्वास और दैनिक जीवन में सहायक भूमिका निभा रहे हैं, विशेषकर उन मरीजों के लिए जिनकी गतिशीलता सीमित है।

उपयोग मामला 1: रोबोटिक एक्सोस्केलेटन

- ReWalk और Ekso Bionics जैसे एक्सोस्केलेटन AI एल्गोरिद्म के माध्यम से रीढ़ की चोट वाले मरीजों को नियंत्रित गति सहायता और वास्तविक समय चाल विश्लेषण प्रदान करते हैं।

उपयोग मामला 2: सामाजिक और सहायक रोबोट

- PARO और Pepper जैसे सहायक रोबोट बुजुर्गों या डिमेंशिया से पीड़ित व्यक्तियों के साथ संवाद कर सकते हैं, दवाई की याद दिला सकते हैं और गिरावट का पता लगा सकते हैं।

5.5.3 अस्पताल संचालन में स्वचालन

स्वास्थ्य संस्थानों में बैकएंड संचालन और संक्रमण नियंत्रण को स्वचालित करने में रोबोट की भूमिका महत्वपूर्ण हो गई है।

उदाहरण:

- TUG रोबोट: अस्पताल के भीतर दवाएं, लिनन और आपूर्ति सामग्री पहुंचाने हेतु स्वायत्त मोबाइल रोबोट
- डिसइंफेक्शन रोबोट: UV-C या हाइड्रोजन परॉक्साइड तकनीक के ज़रिए कक्षों को रोगाणुरहित करने वाले रोबोट

संचालन लाभ:

- संक्रमण नियंत्रण में वृद्धि
- संसाधनों का अनुकूल उपयोग
- न्यूनतम डाउनटाइम में निरंतर संचालन

तालिका 5.5.1: पारंपरिक बनाम AI-सक्षम रोबोटिक्स तुलना

कार्य क्षेत्र	**पारंपरिक तरीका**	**AI-आधारित रोबोटिक सिस्टम**
सर्जरी	मानव-आधारित दृश्य सहायता	रियल-टाइम AI के साथ रोबोटिक सहायक
पुनर्वास	चिकित्सक द्वारा निर्देशित व्यायाम	फीडबैक युक्त एक्सोस्केलेटन
रोगी सहायता	मानव देखभाल और निगरानी	AI साथी रोबोट
अस्पताल लॉजिस्टिक्स	मानव संसाधनों पर निर्भर	स्वायत्त वितरण रोबोट
संक्रमण नियंत्रण	मैनुअल सफाई	UV-डिसइंफेक्शन रोबोट

5.5.4 चुनौतियाँ और नैतिक विचार

हालाँकि रोबोटिक्स स्वास्थ्य सेवा में बड़ी क्षमता रखता है, परंतु कुछ प्रमुख चुनौतियाँ बनी हुई हैं:

- उच्च प्रारंभिक लागत और रखरखाव
- डेटा गोपनीयता और नियमों का अनुपालन
- नैदानिक प्रणालियों के साथ एकीकरण की जटिलता
- देखभाल में सहानुभूति और मानवीय जुड़ाव की आवश्यकता

नवाचारशील समाधान:

- लागत-कुशल मॉड्यूलर रोबोट का विकास

- डेटा गोपनीयता के लिए फेडरेटेड लर्निंग
- मानव-रोबोट सहयोग (कोबोट्स) के माध्यम से पूरक भूमिका निभाना

निष्कर्ष

AI-सक्षम रोबोटिक्स स्वास्थ्य सेवा में नवाचार की सीमाओं को नए स्तर तक ले जा रहा है। यह न केवल सर्जिकल दक्षता को बढ़ाता है बल्कि पुनर्वास, रोगी सहायता और संचालन को भी कुशल बनाता है। नैतिक, आर्थिक और तकनीकी चुनौतियों के बावजूद, AI-रोबोटिक्स का भविष्य रोगी-केंद्रित, संवेदनशील और व्यक्तिगत स्वास्थ्य देखभाल की दिशा में एक निर्णायक कदम है।

इकाई -6: जनरेटिव एआई – बुद्धिमान प्रणालियों का भविष्य

6.1 जनरेटिव एआई का परिचय

6.1.1 परिभाषा और क्षेत्र

जनरेटिव आर्टिफिशियल इंटेलिजेंस (Generative AI) मशीन लर्निंग का एक उपसमूह है जो मौजूदा डेटा से पैटर्न सीखकर नया कंटेंट उत्पन्न कर सकता है—जैसे कि टेक्स्ट, चित्र, ऑडियो, कोड या सिमुलेशन। डिस्क्रिमिनेटिव मॉडल्स जहां इनपुट का वर्गीकरण करते हैं, वहीं जनरेटिव मॉडल्स डेटा वितरण को समझते हैं और उस जैसे नए सैंपल उत्पन्न करते हैं।

मुख्य क्षमताएं:

- यथार्थवादी टेक्स्ट जनरेट करना (जैसे GPT मॉडल्स)
- जीवंत चित्र बनाना (जैसे DALL·E, Stable Diffusion)
- संगीत बनाना या आवाजों की नकल करना
- प्रशिक्षण/परीक्षण के लिए वातावरण सिमुलेट करना

जनरेटिव एआई केवल ऑटोमेशन नहीं करता, बल्कि रचनात्मकता को मशीनों में लाकर उत्पादकता और डिज़ाइन की नई दिशा खोलता है।

6.1.2 ऐतिहासिक विकास और प्रमुख मील के पत्थर

- **2014** – जनरेटिव एडवर्सेरियल नेटवर्क्स (GANs) द्वारा इयान गुडफेलो: सिंथेटिक इमेज जनरेशन में क्रांति।
- **2017** – ट्रांसफॉर्मर आर्किटेक्चर (Vaswani et al.) की शुरुआत: NLP में उन्नति की नींव।
- **2018–2020** – BERT और GPT-2 ने भाषा को समझने और जनरेट करने में श्रेष्ठता प्राप्त की।
- **2022–2023** – GPT-3 और GPT-4 की लोकप्रियता: चैटबॉट्स, कोपायलट्स, और AI लेखकों की शुरुआत।

यह परिवर्तन कार्य-विशिष्ट बुद्धिमत्ता से रचनात्मक और अनुकूलनशील एजेंटों की ओर संकेत करता है।

6.1.3 पारंपरिक एआई मॉडल्स से तुलना

चित्र 6.1.1: जनरेटिव एआई प्रौद्योगिकियों का विकास — GANs से GPT-4 और आगे तक, प्रमुख संरचनात्मक नवाचारों का दृश्यात्मक चित्रण

विशेषता	पारंपरिक एआई	जनरेटिव एआई
कार्य	वर्गीकरण, पूर्वानुमान	कंटेंट निर्माण, सिमुलेशन
आउटपुट	लेबल, प्रेडिक्शन	टेक्स्ट, चित्र, संगीत, डिज़ाइन
आर्किटेक्चर	नियम आधारित / ML	GANs, VAEs, ट्रांसफॉर्मर्स
डेटा निर्भरता	लेबल वाले डेटा पर निर्भर	विविध और अनलेबल्ड डेटा पर निर्भर
उदाहरण	SVM, डिसीजन ट्रीज़	GPT, DALL·E, StyleGAN

महत्वपूर्ण विकास कालक्रम

1. प्रारंभिक शोध (2014)

- **GANs की शुरुआत (जून 2014):**

इयान गुडफेलो द्वारा जनरेटिव एडवर्सेरियल नेटवर्क्स का प्रस्ताव, जिसमें दो नेटवर्क—जेनरेटर और डिस्क्रिमिनेटर—एक-दूसरे से प्रतिस्पर्धा कर सीखते हैं।

- **VAEs की शुरुआत (दिसंबर 2014):**

  Kingma और Welling द्वारा वेरीशनल ऑटोएन्कोडर (VAE), जो डेटा की संरचना को सीखकर नए सैंपल उत्पन्न करते हैं।

2. आधुनिक जनरेटिव मॉडल्स की नींव (2017)

- **ट्रांसफॉर्मर आर्किटेक्चर (जून 2017):**

  "Attention is All You Need" पेपर द्वारा प्रस्तुत – NLP और स्केलेबल मॉडल्स का नया युग।

3. GPT युग (2018 - 2023)

- **GPT-1 (2018):**

  पूर्व-प्रशिक्षण + फाइन ट्यूनिंग के माध्यम से उत्कृष्ट प्रदर्शन।

- **GPT-2 (2019):**

  ज़ीरो-शॉट लर्निंग की क्षमताओं का प्रदर्शन।

- **GPT-3 (2020):**

  175 बिलियन पैरामीटर, न्यूनतम प्रशिक्षण के साथ प्रभावशाली आउटपुट।

- **DALL·E 2 (2022):**

  टेक्स्ट-टू-इमेज जनरेशन।

- **GPT-4 (2023):**

  मल्टीमॉडल इनपुट (टेक्स्ट + इमेज), अधिक सुरक्षित, अधिक समझदार।

4. GPT-4 से आगे (भविष्य)

- **2024 और उसके बाद:**

  रिसर्च का फोकस ऑटोनोमस, एजेंटिक, और मल्टीमॉडल AI पर है – जो केवल टेक्स्ट जनरेट नहीं करता, बल्कि निर्णय भी लेता है।

डिस्क्रिमिनेटिव बनाम जनरेटिव मॉडल्स

डिस्क्रिमिनेटिव मॉडल:

- इनपुट को लेबल में मैप करते हैं।

- लक्ष्य होता है सही वर्गीकरण (जैसे बिल्ली या कुत्ता)।
- उदाहरण: लॉजिस्टिक रिग्रेशन, SVM, BERT (क्लासिफिकेशन के लिए)।

जनरेटिव मॉडल:

- पूरे डेटा वितरण को समझते हैं।
- नए, असली जैसे डेटा सैंपल बना सकते हैं।
- उदाहरण: GANs, VAEs, DALL·E।

चित्र 6.1.2: विवेचक बनाम जनक मॉडल कार्यप्रवाह — इनपुट/आउटपुट प्रवाह के अंतर को दर्शाते हुए

निष्कर्ष:

डिस्क्रिमिनेटिव मॉडल्स निर्णय और वर्गीकरण में अच्छे हैं, जबकि जनरेटिव मॉडल्स रचनात्मकता, अनुकरण और कल्पना की शक्ति प्रदान करते हैं।

6.2 जनरेटिव एआई में मुख्य तकनीकें

6.2.1 जनरेटिव एडवर्सेरियल नेटवर्क्स (GANs)

GAN दो न्यूरल नेटवर्क्स से मिलकर बना होता है—जनरेटर और डिस्क्रिमिनेटर—जो एक प्रतियोगी प्रक्रिया में कार्य करते हैं। जनरेटर नकली डेटा बनाता है और डिस्क्रिमिनेटर यह पहचानने की कोशिश करता है कि डेटा असली है या नकली। समय के साथ, दोनों नेटवर्क बेहतर होते जाते हैं, जिससे अत्यंत यथार्थवादी डेटा उत्पन्न होता है।

अनुप्रयोग:

- वास्तविक मानव चेहरों का निर्माण (जैसे StyleGAN)
- रेडियोलॉजी के लिए सिंथेटिक मेडिकल इमेजेस
- कला और फैशन डिज़ाइन

6.2.2 वैरिएशनल ऑटोएन्कोडर्स (VAEs)

सारांश:

VAEs संभाव्य मॉडल होते हैं जो इनपुट डेटा को एक लेटेंट स्पेस में इनकोड करते हैं और फिर डिकोड कर पुनः डेटा जनरेट करते हैं। ये उन उपयोगों के लिए उपयुक्त हैं जहाँ लेटेंट वेरिएबल्स की व्याख्या आवश्यक होती है।

अनुप्रयोग:

- हस्तलिखित अंकों का निर्माण (MNIST)
- चेहरों का परिवर्तन और भाव-संश्लेषण
- आणविक संरचना का निर्माण

मुख्य विशेषताएँ:

- लेटेंट स्पेस पर वितरण सीखना
- डेटा सैंपल्स के बीच स्मूद ट्रांजिशन सुनिश्चित करना

6.2.3 ट्रांसफॉर्मर आधारित मॉडल (जैसे GPT, BERT, T5)

सारांश:

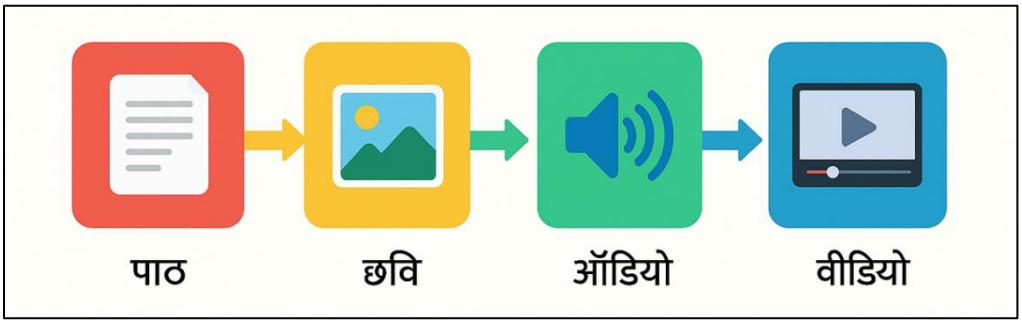
ट्रांसफॉर्मर आधुनिक NLP और मल्टीमॉडल जनरेशन का आधार हैं। ये डेटा अनुक्रमों में संबंधों को मॉडल करने के लिए *attention mechanism* का उपयोग करते हैं।

मुख्य विशेषताएँ:

- *Self-attention* द्वारा समानांतर प्रसंस्करण
- अरबों पैरामीटर तक स्केलेबल
- व्यापक डेटा पर प्री-ट्रेनिंग और टास्क-विशिष्ट फाइन-ट्यूनिंग

अनुप्रयोग:

- GPT: मानव-समान टेक्स्ट जनरेशन
- DALL·E: टेक्स्ट से इमेज जनरेशन
- Codex: एआई-सहायता प्राप्त कोड लेखन



चित्र 6.3.1: जनरेटिव एआई में कंटेंट मोडालिटी का विकास — पाठ से वीडियो तक

तकनीकों की तुलना तालिका

तकनीक	संरचना	मुख्य विशेषताएँ	लोकप्रिय मॉडल	सामान्य आउटपुट
GAN	जनरेटर + डिस्क्रिमिनेटर	तेज और स्पष्ट इमेज निर्माण	StyleGAN, Pix2Pix	इमेज, कला

VAE	इनकोडर + डिकोडर + सैंपलिंग	स्मूद लेटेंट स्पेस, नियंत्रण	Beta-VAE, CVAE	चेहरे, अंक
ट्रांसफॉर्मर	मल्टी-हेड अटेंशन + प्रीट्रेनिंग	संदर्भ की गहरी समझ, स्केलेबिलिटी	GPT-4, T5, BERT	टेक्स्ट, कोड, इमेज (एडेप्टर के साथ)

6.3 जनरेटिव एआई के अनुप्रयोग

6.3.1 कंटेंट जेनरेशन: टेक्स्ट, इमेज और ऑडियो

जनरेटिव एआई मीडिया कंटेंट निर्माण के तरीके को बदल रहा है, जिससे मनोरंजन, पत्रकारिता और डिजाइन जैसी उद्योगों में क्रांति आ रही है।

टेक्स्ट जेनरेशन

ChatGPT, Jasper और Copy.ai जैसे टूल ब्लॉग पोस्ट, सारांश, रिपोर्ट और स्क्रिप्ट तैयार करते हैं। अनुप्रयोग: शिक्षा, तकनीकी लेखन, और कोड जनरेशन (जैसे GitHub Copilot)।

इमेज जेनरेशन

DALL·E, Midjourney और Stable Diffusion जैसे टूल टेक्स्ट प्रॉम्प्ट से हाई-रेज़ोल्यूशन और रचनात्मक विज़ुअल बनाते हैं।

उपयोग: विज्ञापन, कॉन्सेप्ट आर्ट, पर्सनलाइज़्ड अवतार।

**चित्र 6.3.1: जनरेटिव एआई में कंटेंट मोडालिटी का विकास —
पाठ से वीडियो तक की प्रक्रिया**

ऑडियो जेनरेशन

Google WaveNet जैसे टेक्स्ट-टू-स्पीच मॉडल और Amper, AIVA जैसे एआई म्यूजिक कंपोज़र इंसानी आवाज़ और संगीत तैयार करते हैं।

वॉइस क्लोनिंग: डबिंग, ऑडियोबुक्स, वर्चुअल असिस्टेंट्स।

यह स्पेक्ट्रम टेक्स्ट जेनरेशन से शुरू होता है, जहाँ GPT-2 और GPT-3 जैसे मॉडल मानव-समान टेक्स्ट उत्पन्न करने में सक्षम हैं। इसके बाद इमेज जेनरेशन आया, जिसमें GANs और डिफ्यूजन मॉडल्स की मदद से AI ने टेक्स्ट से छवियाँ बनाने की क्षमता प्राप्त की।

ऑडियो जेनरेशन में VALL-E और AudioLM जैसे मॉडल्स ने इंसानी आवाज़, ध्वनि प्रभाव और संगीत की रचना संभव बनाई। यह तकनीक वॉयस असिस्टेंट, संगीत निर्माण और डबिंग जैसे क्षेत्रों में उपयोग हो रही है।

अंततः, वीडियो जेनरेशन आया, जिसमें Runway Gen-2 जैसे मॉडल्स टेक्स्ट या मिक्स्ड मीडिया से वीडियो बना सकते हैं। यह सबसे जटिल और संसाधन-गहन क्षेत्र है, जो विज़ुअल, भाषाई और समय-आधारित समझ को जोड़ता है।

इस प्रकार, चित्र 6.3.1 केवल कंटेंट टाइप्स का क्रम नहीं, बल्कि यह AI मॉडल्स की बढ़ती समझ और मल्टीमॉडल अनुभव को फिर से रचने की क्षमता को दर्शाता है।

6.3.2 स्वास्थ्य सेवा और औषधि खोज

जनरेटिव एआई सटीक चिकित्सा को समर्थन देता है, जिससे निदान, उपचार डिज़ाइन और औषधि विकास तेज़ होता है।

उपयोग के क्षेत्र:

मेडिकल इमेज सिंथेसिस: GANs का उपयोग दुर्लभ बीमारियों के सिंथेटिक MRI या CT स्कैन तैयार करने में होता है।

ड्रग मॉलिक्यूल डिज़ाइन: VAEs और GANs नए दवाओं के अणु तैयार करने में सहायक होते हैं।

पर्सनलाइज़्ड ट्रीटमेंट: ट्रांसफॉर्मर आधारित मॉडल EHR और जीनोमिक डेटा का विश्लेषण कर व्यक्तिगत उपचार सुझाव देते हैं।

उदाहरण:

Insilico Medicine ने GANs की मदद से फाइब्रोसिस उपचार के लिए मॉलिक्यूल डिज़ाइन किए, जो कुछ महीनों में प्री-क्लीनिकल ट्रायल में पहुंच गए।

6.3.3 डिज़ाइन, सिमुलेशन और क्रिएटिविटी

जनरेटिव एआई का उपयोग जटिल सिस्टम डिज़ाइन करने और रियल-वर्ल्ड सिमुलेशन में किया जा रहा है।

अनुप्रयोग:

आर्किटेक्चर: आंतरिक लेआउट और मुखौटे स्वचालित रूप से डिज़ाइन करना।
प्रोडक्ट डिज़ाइन: Autodesk Dreamcatcher जैसे टूल बाधाओं के अनुसार उत्पाद की आकृति डिज़ाइन करते हैं।
गेम डेवलपमेंट: एआई की मदद से स्तर, पात्र और कहानियाँ स्वतः उत्पन्न होती हैं।

क्रिएटिव इंडस्ट्रीज़ में:

कविता, स्क्रिप्ट लेखन, संगीत और दृश्य कहानी लेखन में ट्रांसफॉर्मर और GAN आधारित मॉडल्स का उपयोग।

केस स्टडी बॉक्स: फैशन में जनरेटिव एआई

ब्रांड: ज़ालांडो (यूरोप)

अनुप्रयोग: जनरेटिव मॉडल का उपयोग करके ग्राहकों की पसंद, ट्रेंडिंग रंगों और आकृतियों के आधार पर कपड़ों का डिज़ाइन बनाना।

परिणाम: डिज़ाइन की गति में 40% की वृद्धि और ग्राहक सहभागिता में सुधार।

चित्र 6.3.2 एक रैडियल माइंडमैप प्रस्तुत करता है जो विभिन्न उद्योगों में जनरेटिव एआई के वास्तविक जीवन के अनुप्रयोगों का दृश्य अन्वेषण करता है।

केंद्र में है "जनरेटिव एआई अनुप्रयोग", जो सीखी गई प्रवृत्तियों से नया, मूल्यवान कंटेंट बनाने की तकनीकी क्षमता को दर्शाता है।

इस केंद्रीय नोड से डायग्राम कई प्रमुख उद्योगों में शाखाओं के रूप में विस्तारित होता है, जिनमें प्रत्येक में विशिष्ट उपयोग मामले दर्शाए गए हैं:

स्वास्थ्य देखभाल

चिकित्सा छवि संश्लेषण:

जनरेटिव एआई मॉडल एमआरआई जैसी सिंथेटिक मेडिकल इमेज बनाते हैं ताकि प्रशिक्षण डेटा को बढ़ाया जा सके, डायग्नोस्टिक टूल्स बेहतर बनाए जा सकें, और रोगी गोपनीयता से समझौता किए बिना अनुसंधान को समर्थन मिल सके।

व्यक्तिगत उपचार योजनाएँ:

एआई मरीज के डेटा के आधार पर अनुकूलित उपचार सुझाव उत्पन्न करता है, जिससे चिकित्सकों को अधिक सटीक रणनीतियाँ बनाने में सहायता मिलती है।

मनोरंजन

स्क्रिप्ट और कहानी जनरेशन:

जनरेटिव एआई लेखकों को प्लॉट, संवाद और संपूर्ण पटकथा बनाने में मदद करता है।

वर्चुअल कैरेक्टर क्रिएशन:

वीडियो गेम, फिल्म और आभासी अनुभवों में उपयोग के लिए एआई जीवन-समान अवतार और डिजिटल मानव उत्पन्न करता है।

वित्त

सिंथेटिक डेटा द्वारा धोखाधड़ी का पता लगाना:

जनरेटिव एआई संवेदनशील वास्तविक डेटा के उपयोग के बिना धोखाधड़ी पहचान प्रणालियों को प्रशिक्षित करने के लिए यथार्थवादी लेकिन सिंथेटिक वित्तीय डेटा उत्पन्न करता है।

स्वचालित वित्तीय रिपोर्ट्स:

एआई रियल-टाइम में वित्तीय सारांश, बाजार रिपोर्ट और निवेश अंतर्दृष्टि उत्पन्न करता है।

शिक्षा

बुद्धिमान ट्यूटरिंग सिस्टम:

जनरेटिव एआई प्रत्येक छात्र की सीखने की गति के अनुसार पाठ योजनाएं, स्पष्टीकरण और प्रश्न उत्पन्न करता है।

स्वचालित कंटेंट जनरेशन:

एआई टेक्स्टबुक, क्विज़ और शैक्षणिक वीडियो उत्पन्न करके कंटेंट विकास को स्केल करता है।

निर्माण

डिज़ाइन प्रोटोटाइपिंग:

जनरेटिव मॉडल अभियंताओं को उत्पाद डिज़ाइन के लिए एल्गोरिदम आधारित अन्वेषण में सहायता करते हैं।

पूर्वानुमान आधारित रखरखाव:

एआई उपकरण डेटा के आधार पर रखरखाव शेड्यूल और रिपोर्ट उत्पन्न करता है ताकि डाउनटाइम कम हो सके।

विपणन

व्यक्तिगत विज्ञापन कॉपी:

एआई उपयोगकर्ता प्रोफ़ाइल और व्यवहार के आधार पर लक्षित और आकर्षक मार्केटिंग टेक्स्ट उत्पन्न करता है।

इमेज और वीडियो जनरेशन:

जनरेटिव टूल विशेष अभियानों या दर्शकों के लिए प्रचारात्मक चित्र और गतिशील वीडियो कंटेंट बनाते हैं।

6.4 जनरेटिव एआई इंडस्ट्री 5.0 में

6.4.1 मानव-एआई सहयोग

इंडस्ट्री 5.0 का केंद्र बिंदु मानव और बुद्धिमान मशीनों के बीच सामंजस्य है, जो वैयक्तिकृत, कुशल और सार्थक परिणाम उत्पन्न करता है। जनरेटिव एआई मानव रचनात्मकता और निर्णय लेने की क्षमता को प्रतिस्थापित करने के बजाय उसे बढ़ाने में महत्वपूर्ण भूमिका निभाता है।

मुख्य क्षेत्र:

• सह-डिज़ाइन प्लेटफ़ॉर्म जहाँ मनुष्य एआई द्वारा जनित विचारों को प्रेरित और समीक्षित करता है।

• लेखन, अभियांत्रिकी और वास्तुकला में पुनरावृत्त परिशोधन के लिए मानव-इन-द-लूप प्रणाली।

• सॉफ्टवेयर विकास, पत्रकारिता और कानूनी सहायता में एआई कोपायलट।

उदाहरण:

एक औद्योगिक डिज़ाइनर एआई के साथ मिलकर उत्पाद की अवधारणा बनाता है, जहाँ एआई स्थिरता और एर्गोनॉमिक्स के आधार पर प्रारंभिक डिज़ाइन तैयार करता है जिसे मानव विशेषज्ञ परिष्कृत करता है।

6.4.2 वैयक्तिकृत शिक्षा और सहायक प्रौद्योगिकी

जनरेटिव एआई व्यक्तिगत प्रोफाइल के आधार पर अनुकूलन योग्य सीखने और अनुभवों की अनुमति देता है।

अनुप्रयोग:

• एआई ट्यूटर: ScribeSense और Khanmigo जैसे उपकरण अनुकूलित पाठ योजनाएं, क्विज़ और स्पष्टीकरण उत्पन्न करते हैं।

• सहायक तकनीक: बधिर छात्रों के लिए एआई-जनित कैप्शन या दृष्टिबाधित छात्रों के लिए ऑडियो सारांश।

परिणाम:

"वन-साइज़-फिट्स-ऑल" शिक्षा से वैयक्तिक संज्ञानात्मक समर्थन प्रणाली की ओर बदलाव।

6.4.3 स्वायत्त निर्णय लेने वाली प्रणाली

जनरेटिव एआई को निर्णय समर्थन प्रणालियों में एकीकृत किया जा रहा है जो परिदृश्यों का अनुकरण करता है और सर्वोत्तम विकल्प सुझाता है।

डोमेन:

• आपदा प्रतिक्रिया अनुकरण: एआई उपग्रह डेटा का उपयोग कर संभावित बाढ़ या आग के मार्ग उत्पन्न करता है और निकासी योजनाओं की सिफारिश करता है।

• शहरी नियोजन: जनरेटिव डिज़ाइन टूल जनसंख्या वृद्धि का अनुकरण करते हैं और सड़क, परिवहन व उपयोगिता व्यवस्था का सुझाव देते हैं।

वास्तविक उपयोग मामला:

सिंगापुर अर्बन रीडेवलपमेंट अथॉरिटी भविष्यवाणी मॉडल के साथ सतत शहरी योजनाएं बनाने के लिए एआई का उपयोग करती है।

6.4.4 नैतिकता, सहानुभूति और उत्तरदायी डिज़ाइन

इंडस्ट्री 5.0 की एक विशेषता यह है कि यह मूल्यों पर बल देता है — यह सुनिश्चित करना कि एआई केवल उत्पादक ही नहीं, बल्कि सहानुभूतिपूर्ण और निष्पक्ष भी हो।

केंद्रित क्षेत्र:

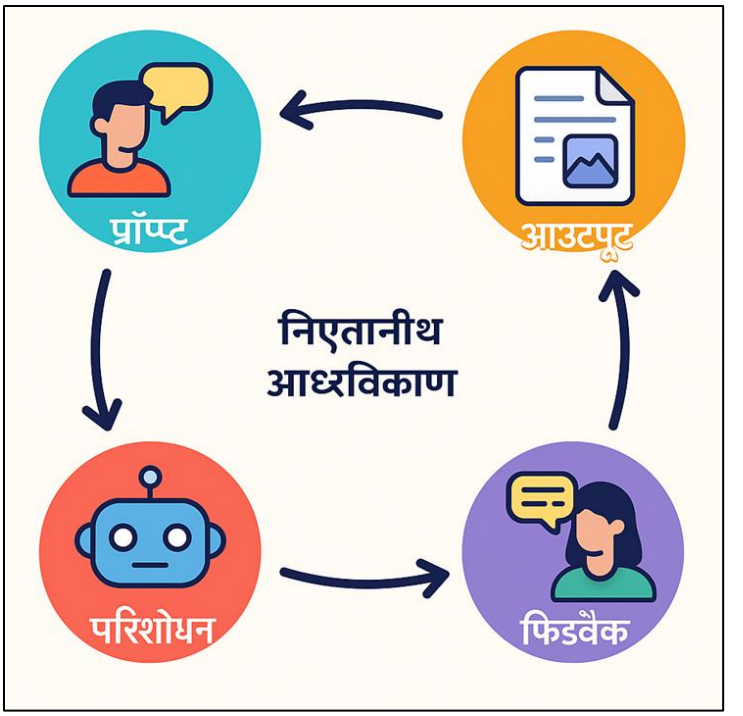
- जनित आउटपुट में पूर्वाग्रह का पता लगाना।
- भावना-संवेदनशील सामग्री निर्माण (जैसे मानसिक स्वास्थ्य ऐप्स)।
- विविध डेटा सेट के साथ समावेशी डिज़ाइन।

चित्र 6.4.1 उस मानव-केंद्रित डिज़ाइन लूप को दर्शाता है जो जनरेटिव एआई प्रणालियों में आउटपुट के विकास और परिशोधन को नियंत्रित करता है।

यह चक्र मानव द्वारा दिए गए एक प्रॉम्प्ट से शुरू होता है, जहाँ उपयोगकर्ता इनपुट प्रदान करता है — जैसे कोई प्रश्न, निर्देश या रचनात्मक सुझाव। यह इनपुट एआई मॉडल के लिए आरंभिक बिंदु होता है।

इसके बाद, एआई दिए गए प्रॉम्प्ट के आधार पर आउटपुट उत्पन्न करता है। यह आउटपुट टेक्स्ट, इमेज, ऑडियो या किसी अन्य रूप का हो सकता है।

फिर मानव उस आउटपुट की समीक्षा करता है और फीडबैक देता है — जैसे "इसे अधिक औपचारिक बनाएं" या "इस भाग को फिर से लिखें।"



चित्र 6.4.1: जनरेटिव एआई प्रणालियों में पुनरावृत्त सुधार हेतु मानव-इन-द-लूप चक्र

इसके आधार पर एआई अपने आउटपुट को परिष्कृत करता है और बेहतर संस्करण उत्पन्न करता है जो मानव की मंशा के अधिक निकट होता है।

अंततः, यह चक्र फिर से शुरू होता है: परिष्कृत आउटपुट अगली प्रतिक्रिया या स्वीकृति का आधार बनता है — जो मानव और मशीन के बीच सतत सहयोग को दर्शाता है।

यह लूप यह सुनिश्चित करता है कि एआई प्रणाली मानवीय लक्ष्यों, मूल्यों और बारीकियों के अनुरूप बनी रहे — विशेष रूप से रचनात्मक, संवेदनशील या उच्च-दांव वाले अनुप्रयोगों में।

6.5 चुनौतियाँ और भविष्य की दिशाएँ

6.5.1 नैतिक, कानूनी और सामाजिक प्रभाव

जनरेटिव एआई शक्तिशाली क्षमताएँ प्रदान करता है, लेकिन इसके साथ कई महत्वपूर्ण नैतिक और नियामकीय चिंताएँ भी जुड़ी हैं:

मुख्य चिंताएँ:

- डीपफेक: एआई द्वारा बनाए गए वीडियो या आवाज़ का दुरुपयोग गलत सूचना या किसी की नकल के लिए।
- बौद्धिक संपदा: एआई द्वारा उत्पन्न सामग्री के स्वामित्व को लेकर अस्पष्टता। पूर्वाग्रह और निष्पक्षता: जनरेटिव मॉडल प्रशिक्षण डेटा से सामाजिक पूर्वाग्रहों को दोहरा सकते हैं या बढ़ा सकते हैं।
- बड़े पैमाने पर गलत सूचना: एआई के माध्यम से नकली समाचार, घोटाले वाली ईमेल और बदले हुए प्रमाणों का स्वत: निर्माण।

उदाहरण:

2023 में एक जनरेटिव मॉडल ने झूठे कानूनी उदाहरण उत्पन्न किए, जिन्हें एक वकील ने अदालत में अनजाने में उद्धृत कर दिया — जिससे सत्यापन तंत्र की आवश्यकता स्पष्ट हुई।

6.5.2 मॉडल की व्याख्या-योग्यता और पारदर्शिता

जैसे-जैसे जनरेटिव मॉडल बड़े और अधिक जटिल होते जा रहे हैं, वे अधिक अपारदर्शी हो जाते हैं, जिससे यह समझना कठिन हो जाता है कि आउटपुट कैसे उत्पन्न हुआ।

प्रभाव:

- एआई द्वारा लिए गए निर्णयों पर भरोसा घटता है।
- परिणामों की जांच, परीक्षण या मान्यता में कठिनाई।

प्रगति में समाधान:

- एक्सप्लैनेबल एआई (XAI): अटेंशन मैप या एक्टिवेशन पैटर्न को विज़ुअलाइज़ करना।
- प्रॉम्प्ट ऑडिटिंग: इनपुट शब्दों के चयन से जनरेटिव व्यवहार में कैसे अंतर आता है, इसका विश्लेषण।

- तथ्य-प्रमाणन पाइपलाइनों का एकीकरण।

6.5.3 स्थिरता और संगणनात्मक लागत

जनरेटिव मॉडल को प्रशिक्षित करने के लिए भारी मात्रा में कंप्यूटिंग शक्ति और ऊर्जा की आवश्यकता होती है, जो पर्यावरणीय चिंताओं को जन्म देती है।

उदाहरण:

- GPT-3 को प्रशिक्षित करने में सैकड़ों GPU-वर्ष और मेगावॉट-घंटों की ऊर्जा लगी।
- बड़े पैमाने पर एआई को समर्थन देने वाले डेटा केंद्रों से कार्बन उत्सर्जन में वृद्धि।

भविष्य के रास्ते:

- ऊर्जा-कुशल ट्रांसफॉर्मर
- मॉडल डिस्टिलेशन (बड़े मॉडल से छोटे मॉडल को प्रशिक्षित करना)
- संघीय और विकेंद्रीकृत शिक्षण (जैसे एज डिवाइस पर)

6.5.4 मानव निगरानी और नियंत्रण सुनिश्चित करना

जनरेटिव सिस्टम को ऐसे सुरक्षा उपायों के साथ डिजाइन किया जाना चाहिए जो उच्च-जोखिम वाले क्षेत्रों जैसे स्वास्थ्य सेवा, कानून या युद्ध के दौरान मानव नियंत्रण को सक्षम करें।

रणनीतियाँ:

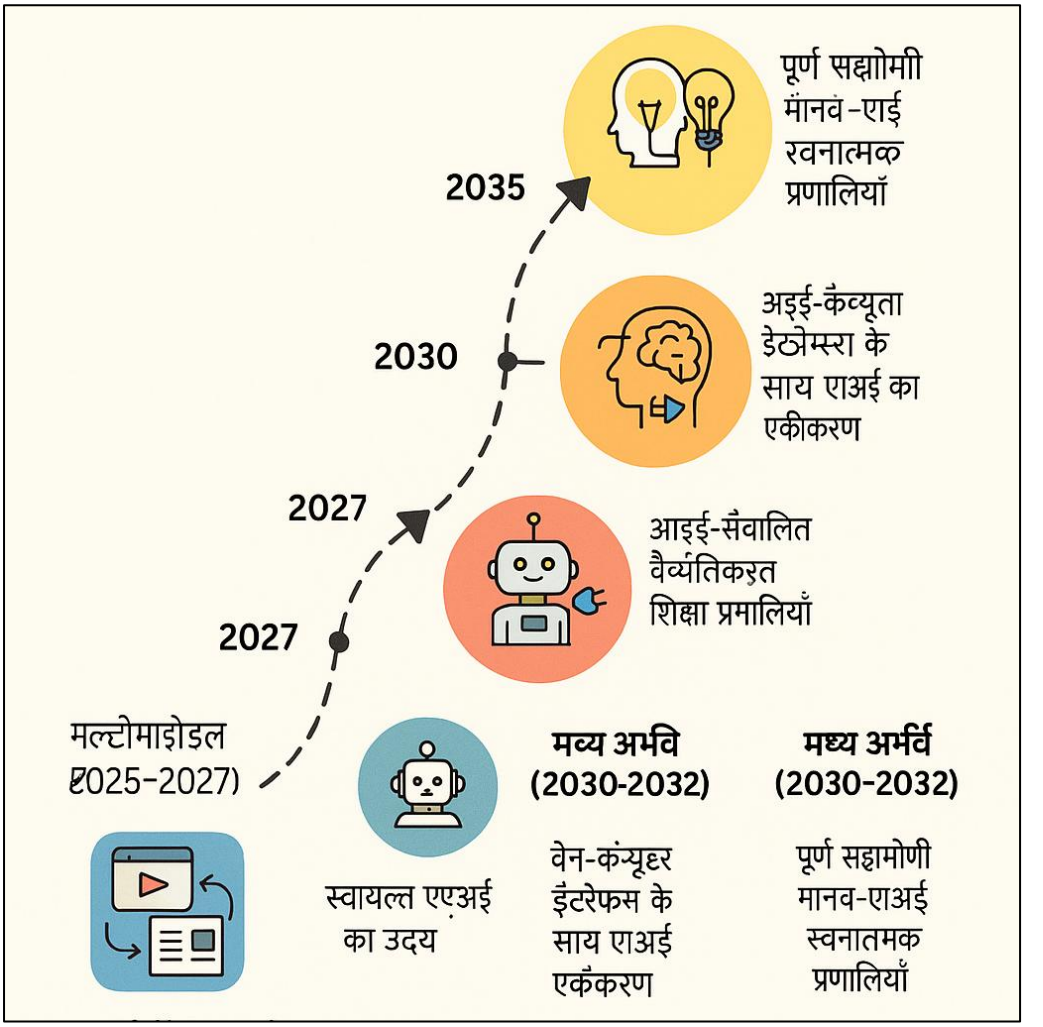
- संवेदनशील कार्यों के लिए मानव-इन-द-लूप (HITL) प्रोटोकॉल।
- विषाक्त, पूर्वाग्रही या भ्रमजनक सामग्री को हटाने के लिए परतदार आउटपुट फिल्टरिंग।
- तैनाती से पहले एआई सिस्टम का रेड टीमिंग (प्रतिकूल प्रॉम्प्ट के साथ परीक्षण)।

6.5.5 भविष्य का विकास और अभिसरण

जनरेटिव एआई संभवतः अन्य तकनीकी सीमाओं के साथ अभिसरण करेगा:

- न्यूरो-सिंबॉलिक एआई: सांख्यिकीय जनरेशन और तार्किक तर्क का संयोजन।
- क्वांटम एआई: जनरेटिव मॉडल के अनुकूलन के लिए क्वांटम कंप्यूटिंग का उपयोग।
- डिजिटल ट्विन्स + जनरेटिव एआई: एआई-निर्देशित भौतिक प्रणालियों की वास्तविक समय की वर्चुअल प्रतिकृतियाँ।

- मल्टीमॉडल एआई: एकीकृत मॉडल जो टेक्स्ट, वाक्, छवि, वीडियो और कोड को एक साथ संभाल सकते हैं।



चित्र 6.5.1: जनरेटिव एआई का भविष्य रोडमैप: 2025–2040 – प्रमुख तकनीकी एकीकरण को दर्शाते हुए

निम्नलिखित चित्र 6.5.1 वर्ष 2025 से 2040 के बीच जनरेटिव एआई प्रौद्योगिकियों के विकास के लिए एक भविष्य का रोडमैप प्रस्तुत करता है, जिसमें तीन विशिष्ट चरणों — निकट भविष्य, मध्यावधि और दीर्घकालीन — के बीच प्रमुख संभावित नवाचार और एकीकरण को दर्शाया गया है।

निकट भविष्य (2025–2027)

2025: मल्टीमॉडल मॉडलों का विस्तार

2025 तक, जनरेटिव एआई एकल-मॉडेलिटी (जैसे केवल टेक्स्ट या केवल इमेज) से आगे बढ़ते हुए पूरी तरह से मल्टीमॉडल सिस्टम को अपनाने की ओर अग्रसर होगा। ये सिस्टम टेक्स्ट, इमेज, ऑडियो, वीडियो और यहां तक कि 3D संरचनाओं के संयोजन को सहजता से प्रोसेस और उत्पन्न करेंगे, जिससे और अधिक समृद्ध और संदर्भ-संवेदनशील आउटपुट प्राप्त होंगे।

2027: स्वायत्त एआई एजेंटों का उदय

लगभग 2027 तक, हम स्वायत्त एआई एजेंटों के उद्भव की अपेक्षा करते हैं — ऐसे एआई मॉडल जो स्वयं लक्ष्य निर्धारित कर सकें, कार्यों की योजना बना सकें, विभिन्न सॉफ्टवेयर ईकोसिस्टम में इंटरैक्ट कर सकें, और लगातार मानवीय इनपुट के बिना ही गतिशील रूप से अनुकूलन कर सकें। यह बदलाव व्यापार स्वचालन, रचनात्मक मीडिया और वैज्ञानिक अनुसंधान जैसे क्षेत्रों में गहरा प्रभाव डालेगा।

मध्य अवधि (2030–2032)

2030: ब्रेन-कंप्यूटर इंटरफेस के साथ एआई का एकीकरण

2030 तक, हम जनरेटिव एआई के साथ ब्रेन-कंप्यूटर इंटरफेस (BCI) के शुरुआती एकीकरण की अपेक्षा करते हैं, जिससे उपयोगकर्ता तंत्रिका संकेतों के माध्यम से एआई मॉडल के साथ इंटरैक्ट कर सकेंगे। यह विशेष रूप से दिव्यांग व्यक्तियों या उच्च दक्षता वाली संचार आवश्यकताओं के लिए अभूतपूर्व पहुंच, रचनात्मकता और रीयल-टाइम प्रतिक्रिया को सक्षम करेगा।

2032: एआई-संचालित वैयक्तिकृत शिक्षा प्रणाली

लगभग 2032 तक, शिक्षा के क्षेत्र में बड़ा बदलाव देखने को मिलेगा। एआई व्यक्तिगत सीखने की शैली, लक्ष्य और भावनात्मक स्थिति के अनुरूप अनुकूलित लर्निंग पथ, सामग्री, सिमुलेशन और ट्यूटरिंग अनुभव उत्पन्न करेगा — जो लगातार अनुकूल होने वाले जनरेटिव मॉडलों द्वारा संचालित होगा।

दीर्घकालिक (2035–2040)

2035: पूर्ण सहयोगी मानव-एआई रचनात्मक प्रणालियाँ

2035 तक, जनरेटिव एआई केवल "मदद" करने तक सीमित नहीं रहेगा, बल्कि मनुष्यों के साथ मिलकर सह-निर्माण करेगा। हम ऐसे सहजीवी संबंधों की कल्पना करते हैं जहाँ मानव और एआई मिलकर कला,

लेखन, विज्ञान और नवाचार परियोजनाओं पर सहयोग करेंगे — जहाँ एआई सूक्ष्मता, प्राथमिकता और शैली को लगभग मानवीय स्तर पर समझेगा।

2040 तक, सामान्यीकृत कृत्रिम बुद्धिमत्ता (AGI) की दिशा में प्रमुख प्रगति की अपेक्षा है। ये मॉडल गहरे तर्क, आत्म-सुधार, रचनात्मक विचार और नैतिक निर्णय लेने की क्षमता प्रदर्शित करेंगे — जो कई डोमेन में लागू होंगे और संकीर्ण एआई व व्यापक बुद्धिमान प्रणालियों के बीच की रेखा को धुंधला कर देंगे।

यह रोडमैप जनरेटिव एआई की क्रमिक लेकिन शक्तिशाली प्रगति को दर्शाता है — जहाँ यह उन्नत मल्टीमॉडल क्षमताओं और स्वायत्त संचालन से होते हुए सीधे मानव-एआई न्यूरल एकीकरण और अंततः सामान्यीकृत बुद्धिमत्ता और सच्चे सह-निर्माण की दिशा में बढ़ता है।

6.6 निष्कर्ष: एक जनरेटिव भविष्य की ओर

जनरेटिव एआई:

पारंपरिक एआई से परे एक दृष्टिकोण बदलाव

जनरेटिव एआई केवल एआई का विस्तार नहीं है; यह उस तरीके में एक क्रांतिकारी बदलाव है जिससे मशीनें मानव परिवेश को समझती, रचती और उसमें सहभागिता करती हैं। यह उन क्षमताओं को दोहराता है जिन्हें कभी केवल मनुष्यों का विशेषाधिकार माना जाता था — जैसे फोटो-सदृश चित्र बनाना, स्वाभाविक संवाद करना, संगीत की रचना करना, और जटिल प्रणालियों का डिज़ाइन करना।

जनरेटिव एआई संज्ञानात्मक स्वचालन और रचनात्मकता के बीच की खाई को पाटता है, और इस प्रश्न को चुनौती देता है कि मशीनें क्या कर सकती हैं।

इंडस्ट्री 5.0 के युग में जनरेटिव एआई: मानव-मशीन सह-निर्माण की दिशा में इंडस्ट्री 5.0 स्वचालित दक्षता से आगे बढ़कर मानव और मशीन के सहयोग की बात करता है, जहाँ तकनीक मानवीय रचनात्मकता और मूल्यों को बढ़ाती है।

जनरेटिव एआई ऐसे उपकरण प्रदान करता है जो कल्पना को विस्तार देते हैं, अनुभवों को अनुकूलित करते हैं, और इंटरैक्टिव इंटेलिजेंस को सशक्त करते हैं।

यह स्वास्थ्य सेवा (दवा डिज़ाइन), शिक्षा (व्यक्तिगत सीखना), मनोरंजन (वर्चुअल रियलिटी), और व्यापार (गतिशील उत्पाद डिज़ाइन) जैसे क्षेत्रों को बदल रहा है।

नैतिक और सतत विकास:

भविष्य की रक्षा

जैसे-जैसे क्षमताएं बढ़ती हैं, दुरुपयोग, पक्षपात और अनपेक्षित परिणामों को रोकने हेतु नैतिक दिशानिर्देश आवश्यक हैं।

जिम्मेदार नवाचार में बड़े एआई मॉडलों के पर्यावरणीय प्रभाव को कम करने और समाज के दीर्घकालिक लाभों को सुनिश्चित करने का प्रयास भी होना चाहिए।

ऐसे एआई सिस्टम बनाना जो स्पष्ट और समझने योग्य हों, विश्वास, निष्पक्षता और लोकतांत्रिक सिद्धांतों को बनाए रखने के लिए महत्वपूर्ण है।

आगे की राह: तकनीकों और मूल्यों का एकीकरण

जनरेटिव एआई का एकीकरण क्वांटम कंप्यूटिंग (तेजी से समाधान), न्यूरोसाइंस (बेहतर संज्ञानात्मक मॉडलिंग), और डिजिटल नैतिकता (मूल्य-आधारित एआई सिस्टम) के साथ होगा। यह समेकन चिकित्सा से मनोरंजन तक के क्षेत्रों में क्रांति लाने वाले नए अंतःविषय क्षेत्रों को जन्म देगा। तकनीकविदों, नैतिकतावादियों और नेताओं को इस विकास को दिशा देनी चाहिए, ताकि भविष्य की प्रणालियों में मानवीय मूल्यों और सामूहिक आकांक्षाओं का समावेश हो।

चित्र 6.6.1: मानव-एआई सह-सृजन की दृष्टि – एक प्रतीकात्मक चित्रण जो मानव हाथों और रोबोटिक प्रणालियों के सहयोग को दर्शाता है, जो एक बेहतर विश्व की रचना कर रहे हैं।

निष्कर्ष: मानव-केंद्रित भविष्य का निर्माण

जनरेटिव एआई मानव क्षमताओं को बढ़ा सकता है, पर केवल तब जब इसे दृष्टिकोण, नैतिकता और समावेशन के साथ विकसित किया जाए।

अंततः हमें ऐसे एआई सिस्टम बनाने चाहिए जो मानव मूल्यों, रचनात्मकता और बुद्धिमत्ता को आने वाली पीढ़ियों के लिए प्रतिबिंबित करें।

चित्र 6.6.1 प्रतीकात्मक रूप से मानव और एआई के बीच सहयोग की भावना को दर्शाता है, जहाँ वे मिलकर एक बेहतर दुनिया का निर्माण कर रहे हैं।

जनरेटिव एआई के बारे में रोचक तथ्य
(Interesting Facts in Generative AI)

1. **मनुष्यों जैसे चेहरे बना सकता है**

   जनरेटिव एआई (जैसे StyleGAN) ऐसे चेहरे बना सकता है जो बिल्कुल असली लगते हैं — लेकिन वे असल में किसी व्यक्ति के नहीं होते।

2. **मूल्यांकन करना कठिन होता है**

   जनरेटिव मॉडल द्वारा बनाए गए कंटेंट की गुणवत्ता का मूल्यांकन करना अक्सर कठिन होता है क्योंकि कई बार "सही उत्तर" नहीं होता — खासकर कला, कहानी या संगीत में।

3. **"डेपफेक" वीडियो भी जनरेटिव एआई से बनते हैं**

   चेहरों और आवाज़ों की नकल कर वीडियो बनाना संभव हो गया है, जिससे फेक वीडियो को असली जैसा दिखाया जा सकता है।

4. **कोड भी लिख सकता है**

   जनरेटिव एआई अब कोड भी बना सकता है। GitHub Copilot जैसे टूल्स डेवलपर्स को कोडिंग में मदद करते हैं।

5. **भाषा और चित्र को जोड़ सकता है**

   DALL·E जैसे मॉडल टेक्स्ट इनपुट से चित्र बना सकते हैं — उदाहरण: "एक गुलाबी हाथी जो स्केटबोर्ड चला रहा है"।

6. **खुद से नए विचार उत्पन्न कर सकता है**

   कुछ मॉडल जैसे GPT-4 या MidJourney, केवल जवाब नहीं देते, बल्कि कल्पनाशील और रचनात्मक समाधान प्रस्तुत करते हैं।

7. **संगीत और ध्वनि निर्माण**

   Amper और AIVA जैसे एआई सिस्टम संगीत बना सकते हैं जो फिल्म, गेम या विज्ञापन के लिए उपयुक्त होते हैं।

8. **भविष्य में 'डिजिटल कलाकार' बन सकते हैं**

   जनरेटिव एआई भविष्य में फिल्मों, गेम्स, और किताबों के लिए सह-लेखक, डिजाइनर या निर्देशक का काम कर सकता है।

9. **वास्तविकता और कल्पना की सीमाएँ धुंधली हो रही हैं**

   जनरेटिव एआई ऐसे दृश्य और कथाएँ बना सकता है जो पूरी तरह काल्पनिक होते हुए भी अत्यधिक यथार्थवादी प्रतीत होते हैं।

10. **बहुभाषीय क्षमताएँ**

    आधुनिक जनरेटिव मॉडल एक ही समय में कई भाषाओं में टेक्स्ट उत्पन्न कर सकते हैं, जिससे वैश्विक संचार में क्रांति आ रही है।

11. **जैविक अनुसंधान में सहयोगी**

    कुछ जनरेटिव मॉडल नए प्रोटीन ढाँचे और दवाओं की संरचना डिजाइन करने में वैज्ञानिकों की मदद कर रहे हैं।

12. **प्रशिक्षण में बड़ी मात्रा में डेटा और ऊर्जा का उपयोग**

    GPT-3 जैसे मॉडल को प्रशिक्षित करने में अरबों शब्दों और बहुत अधिक बिजली की आवश्यकता होती है — जिससे पर्यावरणीय चिंताएँ उत्पन्न होती हैं।

13. **इंटरएक्टिव कहानी कहने में उपयोग**

    गेमिंग और इंटरएक्टिव फिक्शन में जनरेटिव एआई अब उपयोगकर्ता की पसंद के अनुसार कहानी को मोड़ सकता है।

14. **वॉयस क्लोनिंग**

    जनरेटिव एआई किसी व्यक्ति की आवाज़ की हूबहू नकल कर सकता है — जिससे डबिंग और वॉयस असिस्टेंट अधिक प्राकृतिक बनते हैं।

15. **सीखते हुए बेहतर बनते जाते हैं**

    जनरेटिव मॉडल फीडबैक के साथ लगातार सुधार कर सकते हैं, जिससे वे उपयोगकर्ता की प्राथमिकताओं को समझने लगते हैं।

16. **मानव रचनात्मकता का पूरक**

    जनरेटिव एआई कलाकारों, लेखकों और डिज़ाइनरों के लिए एक सहयोगी उपकरण बन चुका है — यह रचनात्मकता को प्रतिस्थापित नहीं करता, बल्कि उसे बढ़ाता है।

शब्दावली (Glossary)

1. **एआई (AI)**
   कृत्रिम बुद्धिमत्ता, एक ऐसी तकनीक जिसमें मशीनें मानव-जैसी बुद्धिमता दिखा सकती हैं — जैसे सीखना, निर्णय लेना और समस्याओं का समाधान करना।
2. **मशीन लर्निंग (Machine Learning)**
   डाटा से स्वतः सीखने और अनुभव के आधार पर प्रदर्शन सुधारने की क्षमता वाली तकनीक।
3. **डीप लर्निंग (Deep Learning)**
   न्यूरल नेटवर्क पर आधारित एक मशीन लर्निंग तकनीक जो बहुस्तरीय परतों के माध्यम से जटिल डेटा को सीखने की क्षमता देती है।
4. **न्यूरल नेटवर्क (Neural Network)**
   मस्तिष्क की संरचना से प्रेरित कंप्यूटेशनल मॉडल जो इनपुट से आउटपुट के बीच संबंधों को सीखते हैं।
5. **क्लासिफिकेशन (Classification)**
   इनपुट डेटा को पूर्वनिर्धारित श्रेणियों में वर्गीकृत करने की प्रक्रिया।
6. **रिग्रेशन (Regression)**
   इनपुट के आधार पर सतत् (continuous) मान का अनुमान लगाने की प्रक्रिया।
7. **सुपरवाइज़्ड लर्निंग (Supervised Learning)**
   ऐसी लर्निंग जिसमें मॉडल को लेबल युक्त डेटा से प्रशिक्षित किया जाता है।
8. **अनसुपरवाइज़्ड लर्निंग (Unsupervised Learning)**
   ऐसी लर्निंग जिसमें डेटा पर कोई लेबल नहीं होता और मॉडल को उसमें छिपे पैटर्न को सीखना होता है।
9. **रीइन्फोर्समेंट लर्निंग (Reinforcement Learning)**
   एक लर्निंग प्रक्रिया जिसमें एजेंट कार्य करता है और पर्यावरण से इनाम या दंड पाकर सीखता है।

10. **ट्रांसफर लर्निंग (Transfer Learning)**

    पहले से सीखे गए कार्यों को दूसरे संबंधित कार्यों पर लागू करना।

11. **नेचुरल लैंग्वेज प्रोसेसिंग (NLP)**

    कंप्यूटर को मानव भाषा को समझने और उस पर प्रक्रिया करने की क्षमता देना।

12. **वाक् पहचान (Speech Recognition)**

    बोली को टेक्स्ट में बदलने की तकनीक।

13. **मशीन अनुवाद (Machine Translation)**

    एक भाषा को दूसरी भाषा में अनुवाद करने की AI तकनीक।

14. **चैटबॉट (Chatbot)**

    एक AI-संचालित सॉफ़्टवेयर जो मनुष्यों के साथ संवाद करता है।

15. **फजी लॉजिक (Fuzzy Logic)**

    ऐसी लॉजिक प्रणाली जो अनिश्चितता और अस्पष्टता से निपटती है।

16. **जनरेटिव एआई (Generative AI)**

    ऐसी AI तकनीक जो नई सामग्री (जैसे पाठ, चित्र, ध्वनि) उत्पन्न कर सकती है।

17. **जीएएन (GAN - Generative Adversarial Network)**

    एक प्रकार का जनरेटिव मॉडल जिसमें एक जनरेटर और एक डिस्क्रिमिनेटर एक-दूसरे के खिलाफ कार्य करते हैं।

18. **वीएई (VAE - Variational Autoencoder)**

    एक जनरेटिव मॉडल जो इनपुट को लैटेंट स्पेस में इनकोड कर पुनः उसे उत्पन्न करता है।

19. **ट्रांसफॉर्मर (Transformer)**

    एक डीप लर्निंग आर्किटेक्चर जो NLP और मल्टीमॉडल कार्यों में क्रांतिकारी साबित हुआ है।

20. **अटेंशन मैकेनिज्म (Attention Mechanism)**

    डेटा अनुक्रम में महत्वपूर्ण हिस्सों पर ध्यान केंद्रित करने की तकनीक।

21. **फाइन ट्यूनिंग (Fine Tuning)**

    पूर्व प्रशिक्षित मॉडल को विशेष कार्य के लिए फिर से प्रशिक्षित करना।

22. **ह्यूमन-इन-द-लूप (Human-in-the-Loop)**
    मशीन लर्निंग प्रक्रियाओं में मानव हस्तक्षेप शामिल करना।
23. **बायस (Bias)**
    डेटा या मॉडल में मौजूद पूर्वग्रह, जो परिणामों को प्रभावित कर सकता है।
24. **मॉडल व्याख्यात्मकता (Model Interpretability)**
    मॉडल के निर्णयों को समझने और समझाने की क्षमता।
25. **एजेंट (Agent)**
    कोई भी इकाई जो पर्यावरण से इनपुट प्राप्त करती है और उस पर प्रतिक्रिया देती है।
26. **डिजिटल ट्विन (Digital Twin)**
    किसी भौतिक प्रणाली का वर्चुअल मॉडल जो वास्तविक समय में अपडेट होता है।
27. **एथिकल एआई (Ethical AI)**
    ऐसी AI प्रणाली जो निष्पक्ष, पारदर्शी और जिम्मेदार तरीके से कार्य करती है।
28. **एआई को-पायलट (AI Co-Pilot)**
    ऐसी AI प्रणाली जो मनुष्य को कार्यों में सहायक होती है — जैसे कोडिंग, लेखन या डिजाइन।
29. **यूनिफिकेशन (Unification)**
    वह प्रक्रिया जिसमें दो लॉजिकल अभिव्यक्तियों को समान बनाने के लिए उपयुक्त प्रतिस्थापन (substitution) खोजा जाता है। इसका उपयोग नियमों और तथ्यों को मेल कराने में किया जाता है।
30. **वैरिएशनल ऑटोएन्कोडर (VAE)**
    एक संभाव्य मॉडल जो डेटा को एक गुप्त (latent) स्थान में एनकोड करता है और फिर उसे डीकोड करके पुनः निर्माण करता है। यह मॉडल नियंत्रित जनरेशन के लिए उपयुक्त होता है।
31. **जनरेटिव एडवर्सेरियल नेटवर्क (GAN)**
    दो नेटवर्क—जनरेटर और डिस्क्रिमिनेटर—के बीच प्रतिस्पर्धा आधारित एक तकनीक, जिसमें एक नकली डेटा बनाता है और दूसरा उसकी सत्यता जांचता है।

32. **ट्रांसफॉर्मर (Transformer)**
    एक तंत्रिका नेटवर्क आर्किटेक्चर जो अनुक्रमिक डेटा को प्रोसेस करने के लिए सेल्फ-अटेंशन तकनीक का उपयोग करता है।
33. **फॉरवर्ड चेनिंग (Forward Chaining)**
    नियमों को लागू करने की एक विधि जिसमें ज्ञात तथ्यों से निष्कर्ष निकाले जाते हैं जब तक लक्ष्य प्राप्त न हो।
34. **ऑब्जेक्टिव फंक्शन (Objective Function)**
    वह मापक जो किसी एजेंट के प्रदर्शन को उसके लक्ष्य की पूर्ति के संदर्भ में आंकता है।
35. **सर्च एल्गोरिद्म (Search Algorithm)**
    कृत्रिम बुद्धिमत्ता में वह प्रक्रिया जो समाधान खोजने के लिए विभिन्न अवस्थाओं को अन्वेषित करती है।
36. **ह्यूरिस्टिक फलन (Heuristic Function)**
    एक अनुमान आधारित मापक जो यह निर्धारित करता है कि कौन-सा पथ लक्ष्य तक पहुँचने के लिए सबसे उपयुक्त है।
37. **बीम सर्च (Beam Search)**
    एक सीमित चौड़ाई वाला सर्च एल्गोरिद्म जिसमें सबसे अच्छे K रास्तों को ही हर चरण पर रखा जाता है।
38. **ग्रीडी बेस्ट फर्स्ट सर्च (Greedy Best-First Search)**
    एक एल्गोरिद्म जो हमेशा निकटतम लक्ष्य की दिशा में सबसे अच्छा अनुमानित पथ चुनता है।
39. **मल्टीमॉडल AI (Multimodal AI)**
    एक ऐसा AI सिस्टम जो एक साथ टेक्स्ट, चित्र, आवाज़ और अन्य इनपुट को प्रोसेस कर सकता है।
40. **आत्म-अवलोकन (Self-Attention)**
    ट्रांसफॉर्मर आर्किटेक्चर का एक घटक जो किसी अनुक्रम में प्रत्येक तत्व को बाकी तत्वों के साथ संबंध स्थापित करने की अनुमति देता है।

41. **मानव-सहयोगी एआई (Human-in-the-loop AI)**
    वह प्रणाली जिसमें एआई द्वारा उत्पन्न आउटपुट को मानव फीडबैक द्वारा संशोधित किया जाता है।
42. **अनुवांशिक एल्गोरिद्म (Genetic Algorithm)**
    एक ऑप्टिमाइज़ेशन तकनीक जो प्राकृतिक चयन और आनुवंशिक क्रियाओं के सिद्धांतों पर आधारित है।
43. **डीप लर्निंग (Deep Learning)**
    एक मशीन लर्निंग तकनीक जो गहरे (multi-layered) न्यूरल नेटवर्क का उपयोग करके डेटा से जटिल पैटर्न सीखती है।

"Generative AI is the canvas where data becomes art, and algorithms shape creativity."

"जनरेटिव एआई वह कैनवास है जहाँ डेटा कला बनता है, और एल्गोरिद्म रचनात्मकता को आकार देते हैं।"

"With Generative AI, the future is not just predicted — it is designed."

"जनरेटिव एआई के साथ, भविष्य केवल अनुमानित नहीं होता — वह डिज़ाइन किया जाता है।"

Prof. (Dr.) K P Yadav (Vishwa Guru)
(B. Tech., PGDBM, M. Tech., Ph.D. and Post Doctorate D. Sc. & D. Litt.-H.)
(Also, Doctorates from International Universities / Institutes : Engineering, Life Science and Health Science, Education & Management, Humanities & Management, Agriculture, and Advocacy/Law-Hon.)
Vice Chancellor, MATS University (NAAC A+ Grade), Raipur CG, India (Appointed by Hon'ble Governor of Chhattisgarh)
- Second Term VC

Prof. (Dr.) S. Mohan Kumar
M.Tech.[Software Engineering]
Ph.D [CSE-Medical Diagnosis CAD System]
Ph.D [Medical Imaging -Machine Learning]
Post Doctorate Degree D.Sc. [Engineering-DL]
EPLM (IIM-Calcutta)
Dean, Indra Ganesan College of Engineering (Autonomous- Higher Educational Research Institution), NAAC Accredited, Indra Ganesan Group of Institutions, Trichy, Tamil Nadu, India.)

DOI: 10.47715/978-93-86388-78-0
ISBN: 978-93-86388-78-0
Publisher: Jupiter Publications Consortium
Published URL: www.jpc.in.net
ISBN 978-93-92090-63-9