Intelligent Agents

tortrixx

2026-10-17

Outline

1. What is an Agent? 2
2. Types of Agents 9
3. Decision Making Under Uncertainty 15
4. Intelligent Agents in Practice 19
5. Summary 32

1. What is an Agent?

1.1 Definition

An **agent** is anything that can be viewed as perceiving its environment through **sensors** and acting upon that environment through **actuators**.

agent 就是一个能够通过传感器感知环境，并通过执行器对环境采取行动的实体。

1.1 Definition

An **agent** is anything that can be viewed as perceiving its environment through **sensors** and acting upon that environment through **actuators**.

agent 就是一个能够通过传感器感知环境，并通过执行器对环境采取行动的实体。

Key characteristics:

- **Autonomy**: operates without continuous human intervention
- **Reactivity**: responds to environmental changes
- **Proactivity**: takes initiative to achieve goals
- **Social ability**: interacts with other agents or humans

1.2 Simple Agent Architecture

Environment → Sensors → Agent Function → Actuators → Environment

1.2 Simple Agent Architecture

Environment → Sensors → Agent Function → Actuators → Environment

The **agent function** maps percept sequences to actions.

1.3 The PEAS Framework

PEAS is a practical way to specify an intelligent agent before implementation:

1.3 The PEAS Framework

Dimension	Question
Performance	How do we measure success?
Environment	Where does the agent operate?
Actuators	How can it change the world?
Sensors	What can it observe?

1.3 The PEAS Framework

1.3 The PEAS Framework

For an autonomous taxi, PEAS might be: safety and travel time; roads and traffic; steering and braking; cameras, GPS, and lidar.

1.4 Rationality and Bounded Rationality

An agent is **rational** when it chooses the action that maximizes expected performance, given:

- the percept sequence available so far;
- its prior knowledge of the environment;
- the actions it can execute; and
- the performance measure.

1.4 Rationality and Bounded Rationality

An agent is **rational** when it chooses the action that maximizes expected performance, given:

- the percept sequence available so far;
- its prior knowledge of the environment;
- the actions it can execute; and
- the performance measure.

Perfect rationality is rarely possible: observations are noisy, computation is limited, and the future is uncertain. Real systems therefore optimize under resource constraints.

2. Types of Agents

2.1 Four Classic Types (Russell & Norvig)

1. Simple Reflex Agents
- Condition-action rules
- No memory of past
- Fast but limited

2. Model-Based Reflex Agents
- Maintain internal state
- Track unobservable aspects
- Better in partially observable environments

3. Goal-Based Agents
- Consider future outcomes
- Search & planning
- Decide actions that achieve goals

4. Utility-Based Agents
- Maximize expected utility
- Handle trade-offs
- Prefer better outcomes among many goals

2.2 Learning Agents

A modern agent often includes a **learning element**:

> Performance Element + Critic + Learning Element + Problem Generator

2.2 Learning Agents

A modern agent often includes a **learning element**:

> Performance Element + Critic + Learning Element + Problem Generator

This allows the agent to improve its behavior over time from experience.

2.3 Environment Properties

The environment determines which agent design is appropriate:

2.3 Environment Properties

Observable vs. partially observable
Can the agent see the complete state?

Episodic vs. sequential
Do earlier actions affect later decisions?

Discrete vs. continuous
Are states, time, and actions countable?

Deterministic vs. stochastic
Does an action have a predictable result?

Static vs. dynamic
Can the world change while the agent thinks?

Single-agent vs. multi-agent
Are other decision makers present?

2.3 Environment Properties

2.3 Environment Properties

The same algorithm can behave very differently when these assumptions change. Explicitly stating them prevents hidden design failures.

3. Decision Making Under Uncertainty

3.1 Markov Decision Processes

Many sequential decision problems can be modeled as an MDP:

$$M = (S, A, P, R, \gamma)$$

- S: states, A: available actions
- $P(s' \mid s, a)$: transition dynamics
- $R(s, a)$: immediate reward
- γ: discount factor, with $0 \leq \gamma \leq 1$

3.1 Markov Decision Processes

Many sequential decision problems can be modeled as an MDP:

$$M = (S, A, P, R, \gamma)$$

- S: states, A: available actions
- $P(s' \mid s, a)$: transition dynamics
- $R(s, a)$: immediate reward
- γ: discount factor, with $0 \leq \gamma \leq 1$

The Markov property says that the current state contains all information needed to predict the future:

$$P(s_{t+1} \mid s_t, a_t, s_{t-1}, \ldots) = P(s_{t+1} \mid s_t, a_t)$$

3.2 Value, Policy, and Return

A **policy** $\pi(a \mid s)$ maps states to actions. The discounted return is:

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

The state-value function estimates the expected return under a policy:

$$V^{\pi(s)} = E_{\pi[G_t \mid s_t = s]}$$

3.2 Value, Policy, and Return

A **policy** $\pi(a \mid s)$ maps states to actions. The discounted return is:

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

The state-value function estimates the expected return under a policy:

$$V^{\pi(s)} = E_{\pi[G_t \mid s_t = s]}$$

Learning becomes a search for a policy that produces high long-term value, not merely a sequence of locally good actions.

3.3 The Reinforcement Learning Loop

State s_t → Agent chooses a_t → Environment returns r_{t+1} and s_{t+1}

3.3 The Reinforcement Learning Loop

State s_t → Agent chooses a_t → Environment returns r_{t+1} and s_{t+1}

At every step, the agent balances:

- **exploration**: try uncertain actions to learn more;
- **exploitation**: choose actions already known to work well.

This exploration-exploitation tension is central to online learning.

4. Intelligent Agents in Practice

4.1 Multi-Agent Systems (MAS)

- Multiple interacting agents
- Cooperation, competition, or negotiation
- Applications: traffic control, smart grids, distributed sensing

4.2 Large Language Model Agents

Modern LLM-based agents typically combine:

4.2 Large Language Model Agents

Modern LLM-based agents typically combine:

- **Reasoning** (Chain-of-Thought, ReAct, Tree of Thoughts)
- **Tool use** (APIs, search, code execution)
- **Memory** (short-term context + long-term vector stores)
- **Planning** (decompose tasks into sub-goals)

4.2 Large Language Model Agents

Modern LLM-based agents typically combine:

- **Reasoning** (Chain-of-Thought, ReAct, Tree of Thoughts)
- **Tool use** (APIs, search, code execution)
- **Memory** (short-term context + long-term vector stores)
- **Planning** (decompose tasks into sub-goals)

Popular frameworks: LangChain, AutoGen, CrewAI, Semantic Kernel, etc.

4.3 A Tool-Using Agent Workflow

1. Observe
Read the task and context

2. Plan
Decompose the goal

3. Act
Call tools and APIs

4. Reflect
Check results and revise

4.3 A Tool-Using Agent Workflow

4.3 A Tool-Using Agent Workflow

The language model is not the whole agent. Reliable behavior comes from the loop around it: typed tools, state management, validation, retries, and explicit stopping rules.

4.4 Grounding and Tool Contracts

Tools turn abstract reasoning into verifiable actions:

4.4 Grounding and Tool Contracts

```
{
  "name": "lookup_weather",
  "arguments": {
    "city": "Shanghai",
    "date": "2026-09-29"
  }
}
```

4.4 Grounding and Tool Contracts

4.4 Grounding and Tool Contracts

A robust tool interface should define:

- a narrow purpose and typed inputs;
- predictable outputs and error states;
- permission boundaries and audit logs;
- timeouts, retries, and idempotency where needed.

4.5 Evaluating Agentic Systems

Evaluation should measure more than whether the final answer sounds plausible:

- **Task success**: did the agent achieve the user goal?
- **Grounding**: are claims supported by observations or sources?
- **Efficiency**: how many tokens, tool calls, and seconds were used?
- **Robustness**: does it recover from malformed or unavailable tools?
- **Safety**: does it respect permissions and refuse unsafe actions?

4.5 Evaluating Agentic Systems

Evaluation should measure more than whether the final answer sounds plausible:

- **Task success**: did the agent achieve the user goal?
- **Grounding**: are claims supported by observations or sources?
- **Efficiency**: how many tokens, tool calls, and seconds were used?
- **Robustness**: does it recover from malformed or unavailable tools?
- **Safety**: does it respect permissions and refuse unsafe actions?

Trace-based evaluation makes failures diagnosable: inspect the observation, decision, tool call, result, and final response as one trajectory.

4.6 Key Challenges

- **Hallucination** and reliability
- **Long-horizon planning**
- **Grounding** in real environments
- **Safety & alignment**
- **Evaluation** of open-ended behavior

4.7 Design Principles

When building an agent, prefer:

4.7 Design Principles

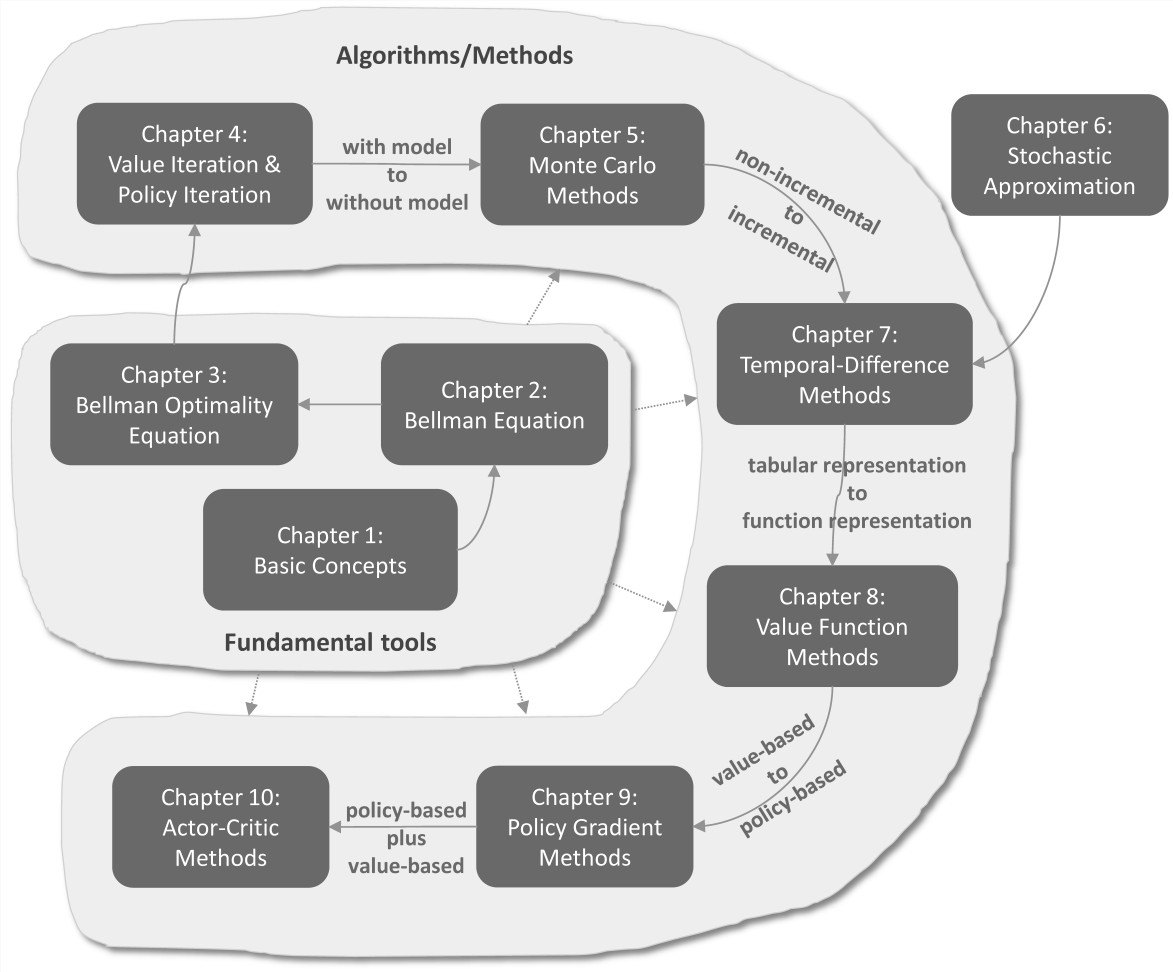
Small action spaces
Make each tool easy to validate.

Reversible actions
Preview or stage high-impact changes.

Explicit state
Store important decisions outside the prompt.

Human checkpoints
Ask for approval at meaningful risk boundaries.

4.7 Design Principles



这里写标题或重点

- 第一点说明内容
- 第二点说明内容
- 第三点说明内容

5. Summary

5.1 Takeaways

- An agent perceives and acts to achieve goals
- Classic taxonomy: reflex → model-based → goal-based → utility-based
- Learning agents improve through experience
- Modern AI agents combine LLMs + tools + memory + planning
- Multi-agent systems enable complex collaborative behaviors

Core definition of an agent

Key components of m

Thank you!

Questions?