

2. Intelligent Agents

Buzhou Tang
 Email: tangbz@hit.edu.cn
<http://icrc.hit.edu.cn>
 Harbin Institute of Technology (Shenzhen)
 Intelligent Computing Research Center
 (ICRC, HITSZ)

Outline

- ▶ Agents and environments 智能体与环境
- ▶ Rationality 理性
- ▶ PEAS (Performance measure, Environment, Actuators 执行器, Sensors 传感器)
- ▶ Environment types
- ▶ Agent types

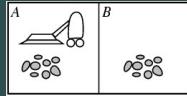
The Question to be Answered

How to define an AI problem?

What is an (Intelligent) Agent

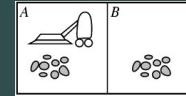
- ▶ Anything that can be viewed as **perceiving** its **environment** through **sensors** and **acting** upon that environment through its **actuators** (执行器), to maximize progress towards its **goals**.
 => PAGE (Percepts, Actions, Goals, Environment)
- ▶ Human agent: eyes, ears, and other organs for sensors; hands, legs, mouth, and other body parts for actuators.
- ▶ Robotic agent: cameras and infrared range finders for sensors; various motors for actuators.

Ho Vacuum-cleaner world



- ▶ Environment: square A and B
- ▶ Percept: location and contents, e.g., [A, Dirty]
- ▶ Actions: Left, Right, Suck
- ▶ "percept" denotes the agent's perceptual inputs at any given instance
- ▶ "percept sequence" is the complete history of everything the agent has ever perceived

Ho Vacuum-cleaner world



Percept sequence	Action
[A, Clean]	Right
[A, Dirty]	Suck
[B, Clean]	Left
[B, Dirty]	Suck
[A, Clean], [A, Clean]	Right
[A, Clean], [A, Dirty]	Suck
...	...

Rational Agents(Cont.)

- ▶ A **rational agent** is one that does the right thing
 - ▶ Every entry in the table is filled out correctly.
- ▶ What is the right thing?
 - ▶ Approximation: the most *successful* agent.
 - ▶ *Measure of success?*
- ▶ Performance measure should be objective
 - ▶ E.g. the amount of dirt cleaned within a certain time.
 - ▶ E.g. how clean the floor is.
 - ▶ ...
- ▶ **Performance measure according to what is wanted in the environment instead of how the agents should behave**

Rational Agents(Cont.)

- ▶ What is rational at a given time depends on four things: **PEAS**
 - ▶ Performance measure
 - ▶ Prior environment knowledge
 - ▶ Actions
 - ▶ Percept sequence to date (Sensors)
- ▶ **DEF:** A **rational agent** chooses whichever action maximizes the expected value of the performance measure given the percept sequence to date and prior environment knowledge.
- ▶ **Rationality ≠ omniscience** (全知者)
 - ▶ An omniscient agent knows all the actual outcome of its actions and always takes proper actions.
- ▶ **Rationality ≠ perfection**
 - ▶ Rationality maximizes *expected* performance, while perfection maximizes *actual* performance.

Rational Agents(Cont.)

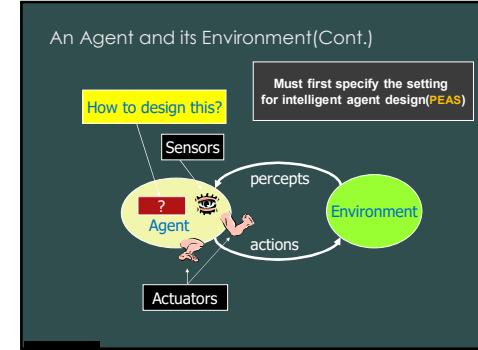
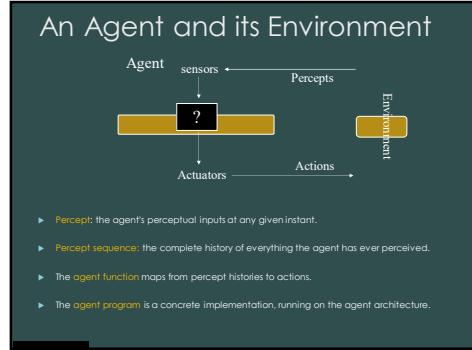
Ideal Rational Agent: For each possible percept sequence, such an agent does whatever action is expected to maximize its performance measure, on the basis of the evidence provided by the percept sequence and whatever built-in knowledge the agent has.

What do you think? Is this an acceptable definition?

Not looking left when crossing the street: If I don't see a car coming from the left, it is rational to cross the street?
 No. Should also consider taking *information gathering actions*.

Limited/costly computation time
 Bounded rationality
 Limited/costly memory
 ...

Agent and Its Environment



PEAS

- ▶ **PEAS:**
 - ▶ **Performance measure**
 - ▶ **Environment**
 - ▶ **Actuators**
 - ▶ **Sensors**

PEAS (Cont.)

- Consider, e.g., the task of designing:
An automated taxi driver, what are the PEAS?
- ▶ **Performance measure:** Safe, fast, legal, comfortable trip, maximize profits
 - ▶ **Environment:** Roads, other traffic, pedestrians(行人), customers
 - ▶ **Actuators:** Steering wheel, accelerator, brake, signal, horn(鸣笛)
 - ▶ **Sensors:** Cameras, sonar(声呐), speedometer, GPS, odometer(里程表), engine sensors, keyboard

PEAS(Cont.)

- Agent: Medical diagnosis system
- ▶ **Performance measure:** Healthy patient, minimize costs, lawsuits
 - ▶ **Environment:** Patient, hospital, staff
 - ▶ **Actuator:** Screen display (questions, tests, diagnoses, treatments)
 - ▶ **Sensor:** Keyboard (entry of symptoms, findings, patient's answers)

PEAS(Cont.)

- Agent: Part-picking robot
- ▶ **Performance measure:** Percentage of parts in correct bins (工具箱)
 - ▶ **Environment:** Conveyor belt with parts, bins
 - ▶ **Actuators:** Jointed arm and hand
 - ▶ **Sensors:** Camera, joint angle sensors

PEAS(Cont.)

- Agent: Interactive English tutor
- ▶ **Performance measure:** Maximize student's score on test
 - ▶ **Environment:** Set of students
 - ▶ **Actuators:** Screen display (exercises, suggestions, corrections)
 - ▶ **Sensors:** Keyboard

Environment Types

- ▶ 1. **Fully observable** (vs. partially observable): An agent's sensors give it access to the complete state of the environment at each point in time.
- ▶ 2. **Deterministic** (vs. stochastic) 确定的 vs. 随机的: The next state of the environment is completely determined by the current state and the action executed by the agent. (If the environment is deterministic except for the actions of other agents, then the environment is **strategic** (策略的).)
- ▶ 3. **Episodic** (vs. sequential) 片段的 vs. 延续的: The agent's experience is divided into atomic "episodes" (each episode consists of the agent perceiving and then performing a single action), and the choice of action in each episode depends only on the episode itself.

Environment Types(Cont.)

- ▶ **4. Static** (vs. dynamic) 静态的 vs. 动态的: The environment is unchanged while an agent is deliberating (计算或“思考时”).
(The environment is **semidynamic** if the environment itself does not change with the pass of time but the agent's performance score does)
- ▶ **5. Discrete** (vs. continuous) 离散的 vs. 连续的: A limited number of distinct, clearly defined percepts and actions.
- ▶ **6. Single agent** (vs. multi-agent) 单智能体 vs. 多智能体: An agent operating by itself in an environment.

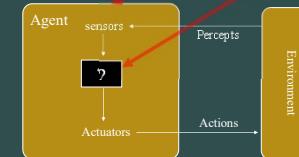
Environment Types(Cont.)

	Chess with a clock	Chess without a clock	Taxi driving
Fully observable	Yes	Yes	No
Deterministic	Strategic	Strategic	No
Episodic	No	No	No
Static	Semi	Yes	No
Discrete	Yes	Yes	No
Single agent	No	No	No

- ▶ The environment type largely determines the agent design
- ▶ The real world is (of course) partially observable, stochastic, sequential, dynamic, continuous, multi-agent

Agent Types

Agent = architecture + program



Agent Types

- ▶ How does the inside of the agent work?
 - ▶ Agent = architecture + program
 - ▶ Architecture: computing **devices** with physical sensors and actuators
- ▶ All agents have the same skeleton:
 - ▶ Input = current percepts
 - ▶ Output = action
 - ▶ Program = manipulates input to produce output
- Note difference between agent program and agent function:
 - ▶ **Agent program** implements the agent function mapping percepts to actions.
 - ▶ Agent program differs from agent function:
 - ▶ Input = Percept, **not history**

Agent Types(Cont.)

- ▶ Five basic types in order of increasing generality:
 - ▶ Table-driven agent
 - ▶ Simple reflex agents
 - ▶ Model-based reflex agents
 - ▶ Goal-based agents
 - ▶ Utility-based agents

More sophisticated

Table-Driven Agent

An agent based on a prespecified lookup table. It keeps track of percept sequence and just looks up the best action

• Problems

- Huge number of possible percepts (consider an automated taxi with a camera as the sensor) => Lookup table would be huge
- Takes long time to build the table
- Not adaptive to changes in the environment; requires entire table to be updated if changes occur

Agent Program

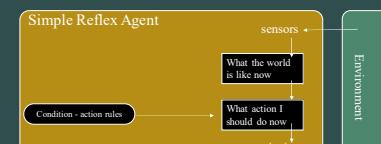
```

function TABLE-DRIVEN-AGENT(percept) returns an action
  persistent: percepts, a sequence, initially empty
  table, a table of actions, indexed by percept sequences, initially fully specified
  append percept to the end of percepts
  action ← LOOKUP(percepts, table)
  return action
  
```

Simple Reflex Agents

- ▶ Differs from the lookup table based agent is that: Based on the **current percept**, ignoring the rest of the percept history. And the condition (that determines the action) is already higher-level interpretation of the percepts.
- ▶ Percepts could be e.g. the pixels on the camera of the automated taxi

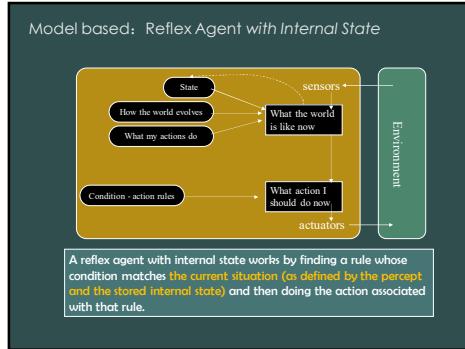
Simple Reflex Agents(Cont.)



A simple reflex agent works by finding a rule whose condition matches the **current situation** (as defined by the percept) and then doing the action associated with that rule.

Simple Reflex Agents(Cont.)

- Table lookup of condition-action pairs defining all possible condition-action rules necessary to interact in an environment
 - e.g. if car-in-front-is-breaking then initiate breaking
- Problems**
 - Table is still too big to generate and to store (e.g. taxi)
 - Takes long time to build the table
 - No knowledge of non-perceptual parts of the current state
 - Not adaptive to changes in the environment: requires entire table to be updated if changes occur
 - Looping: Can't make actions conditional



Model-based reflex agents

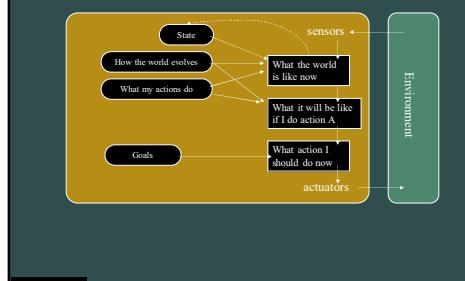
```

function MODEL-BASED-REFLEX-AGENT(percept)
  persistent state, the agent's current conception of the world state
  model, a description of how the next state depends on current state and action
  rules, a set of condition-action rules
  action, the most recent action, initially none
  state ← UPDATE-STATE(state,action,percept,model)
  rule ← RULE-MATCH(state,rules)
  action ← rule.ACTION
  return action
  
```

Reflex Agent with Internal state (Cont.)

- Encode "internal state of the world" to remember the past as contained in earlier percepts
- Needed because sensors do not usually give the entire state of the world at each input, so perception of the environment is captured over time. "State" used to encode different "world states" that generate the same immediate percept
- Requires ability to represent change in the world with/without the agent
 - one possibility is to represent just the latest state, but then cannot reason about hypothetical courses of action

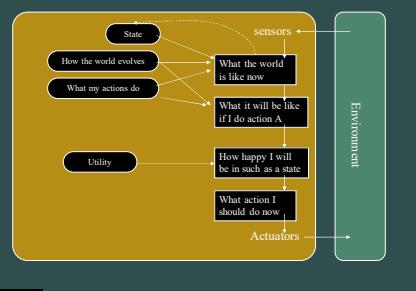
Agent with Explicit Goals



Agent with Explicit Goals (Cont.)

- Choose actions so as to achieve a (given or computed) goal = a description of desirable situations. e.g. where the taxi wants to go
- Keeping track of the current state is often not enough ~ need to add goals to decide which situations are good
- Deliberative instead of reactive** (需要思考, 而不仅仅是条件性反射)
- May have to consider long sequences of possible actions before deciding if goal is achieved ~ involves considerations of the future, "what will happen if I do...?" (search and planning)
- More flexible than reflex agent. (e.g. rain / new destination) In the reflex agent, the entire database of rules would have to be rewritten

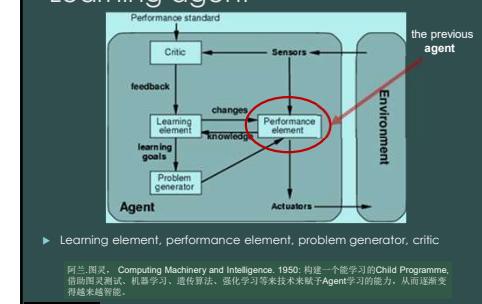
Utility-Based Agent



Utility-Based Agent (Cont.)

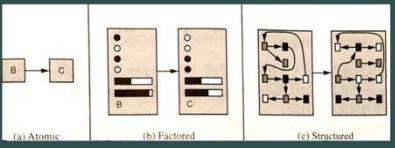
- When there are multiple possible alternatives, how to decide which one is best?
- A goal specifies a crude distinction between a happy and unhappy state, but often need a more general performance measure that describes "degree of happiness"
 - Utility function $U: \text{State} \rightarrow \text{Reals}$ indicating a measure of success or happiness when at a given state
 - Allows decisions comparing choice between conflicting goals, and choice between likelihood of success and importance of goal (if achievement is uncertain)

Learning agent



Agent Program Components

- ▶ Atomic state
- ▶ Factored state
- ▶ Structured state



Summary

▶ Intelligent Agents:

- ▶ Anything that can be *viewed as perceiving* its *environment* through *sensors* and *acting* upon that environment through its *actuators* to maximize progress towards its *goals*.
- ▶ PAGE (Percepts, Actions, Goals, Environment)
- ▶ Described as a Perception (sequence) to Action Mapping: $f: \mathcal{P}^* \rightarrow \mathcal{A}$
- ▶ Using look-up-table, closed form, etc.

- ▶ **Rational Action:** The action that maximizes the expected value of the performance measure *given the percept sequence to date*

- ▶ **Agent Types:** Reflex, state-based, goal-based, utility-based

Homework

Recently, "Agent framework" becomes a hot topic of LLM. You are asked to conducting a short survey for the LLM based or LLM oriented Agent framework, analysis each of the agent system you report under the agent framework, i.e., what's the PAGE of the agent? Which type should the agent belong to? (you can even get a conclusion like: it should not be called an agent, but must give your analysis.)

Write a report for your survey.

Here is one such of agents for your reference:
AutoGen (Microsoft) <https://github.com/microsoft/autogen>

- ▶ AutoGen is a framework that enables the development of LLM applications using multiple agents that can converse with each other to solve tasks. AutoGen agents are customizable, conversable, and seamlessly allow human participation. They can operate in various modes that employ combinations of LLMs, human inputs, and tools.

Artificial Intelligence

第2部分 推理

1. 符号推理
2. 不确定性推理
3. 因果推理

Prof. Buzhou Tang

Email: tangbz@hit.edu.cn
<http://icrc.hit.edu.cn>

Harbin Institute of Technology (Shenzhen)
Intelligent Computation Research Center
(ICRC, HITZ)

ChatGPT带来的思考	
• 大模型时代，我们还需要基于符号的知识推理吗？	<p>是的，我们依然需要基于符号的知识推理。尽管大模型在处理自然语言方面取得了显著进展，但它们在处理逻辑推理、数学证明、法律条文等需要精确符号表示和推理的任务时，表现往往不如人类。因此，在这些领域，我们仍然需要依赖基于符号的知识推理。</p>
• 新冠肺炎的治疗方案上说，患者出现呼吸困难时应及时就医，但患者出现高烧、咳嗽、乏力、嗅觉减退、味觉减退、鼻塞、流涕、咽痛、结膜炎、肌痛和腹泻等症状时，不需要就医。请问，为什么会出现这种情况呢？	<p>这是因为新冠肺炎的症状通常包括发热、咳嗽、乏力、嗅觉减退、味觉减退、鼻塞、流涕、咽痛、结膜炎、肌痛和腹泻等，而呼吸困难通常被视为一种严重的症状，可能预示着病情加重或并发症的发生。因此，当患者出现呼吸困难时，应及时就医。至于其他症状（如高烧、咳嗽、乏力等），虽然它们也是新冠肺炎常见的表现，但如果症状较轻且没有明显恶化趋势，可以先在家观察，必要时再就医。</p>
• 新冠肺炎的治疗方案上说，可以采用鼻拭子进行病毒检测，检测结果为阳性，说明患者感染了新冠病毒。但患者出现高烧、咳嗽、乏力、嗅觉减退、味觉减退、鼻塞、流涕、咽痛、结膜炎、肌痛和腹泻等症状时，不需要就医。请问，为什么会出现这种情况呢？	<p>这是因为新冠肺炎的症状通常包括发热、咳嗽、乏力、嗅觉减退、味觉减退、鼻塞、流涕、咽痛、结膜炎、肌痛和腹泻等，而呼吸困难通常被视为一种严重的症状，可能预示着病情加重或并发症的发生。因此，当患者出现呼吸困难时，应及时就医。至于其他症状（如高烧、咳嗽、乏力等），虽然它们也是新冠肺炎常见的表现，但如果症状较轻且没有明显恶化趋势，可以先在家观察，必要时再就医。</p>

ChatGPT带来的思考

- 大模型时代，我们还需要基于符号的知识推理吗？

① 什么是知识图谱：上篇讲到，进行理性推论需要知识库

② 理性推论的步骤：上篇讲到，进行理性推论需要知识库，可能以下几步操作之一：

1. 提出问题：可提出问题，如“如何治疗感冒”，“感冒的症状是什么”，“感冒的原因是什么”，等等，同时提出逻辑前提、公理、规则，即理论背景知识。
2. 找证据：根据问题，找到证据，根据理论背景知识，找出与问题相关的论据，是事实、数据、经验、理论等。
3. 评估证据：评估证据，根据证据和理论背景知识，对论据进行真伪判断，是否成立，是否可信。
4. 推理：根据评估结果，结合公理、规则，得出结论，即推论，从而完成理性推论。

因此，建议读者仔细阅读上篇讲到的理性推论的步骤和方法。

这个进行性加重，应该是出现胆囊积液，像是严重的肝炎。肝硬化，胰腺炎一般是比较轻的情况下会出现腹痛。一般还是比较少见。

③ 为什么一个普通医生就能诊断出糖尿病，胰腺炎，无非，就是对于疾病的了解程度不同而已。

④ 机器能推理吗：机器能推理，但机器的推理能力有限，目前，机器能推理的范围非常有限，主要集中在一些固定的数据和规则上，如天气预报、股票走势、交通状况等。

⑤ 机器能学习吗：机器能学习，但机器的“学习”能力有限，主要集中在一些固定的数据和规则上，如天气预报、股票走势、交通状况等。

⑥ 机器能理解吗：机器能理解，但机器的理解能力有限。

⑦ 机器能推理吗：机器能推理，但机器的推理能力有限，主要集中在一些固定的数据和规则上，如天气预报、股票走势、交通状况等。

⑧ 机器能学习吗：机器能学习，但机器的“学习”能力有限，主要集中在一些固定的数据和规则上，如天气预报、股票走势、交通状况等。

⑨ 机器能理解吗：机器能理解，但机器的理解能力有限，主要集中在一些固定的数据和规则上，如天气预报、股票走势、交通状况等。

⑩ 机器能推理吗：机器能推理，但机器的推理能力有限，主要集中在一些固定的数据和规则上，如天气预报、股票走势、交通状况等。

⑪ 机器能学习吗：机器能学习，但机器的“学习”能力有限，主要集中在一些固定的数据和规则上，如天气预报、股票走势、交通状况等。

⑫ 机器能理解吗：机器能理解，但机器的理解能力有限，主要集中在一些固定的数据和规则上，如天气预报、股票走势、交通状况等。

Chapter 8. First order logic (FOL) (or First-Order Predicate Calculus, FOPC)

Outline

- "Knowledge-Based Agent
- Why FOL?
- Syntax and semantics of FOL
- Using FOL
- Wumpus world in FOL
- Knowledge engineering in FOL

The diagram illustrates the architecture of a Knowledge-Based Agent. At the top, three lightbulbs represent "Domain independent algorithms". Below them is a central box labeled "Inference engine". To the left, two arrows point to this box: one from "ASK" (labeled "Domain specific controller") and another from "TELL" (also labeled "Domain specific controller"). A third arrow points away from the "Inference engine" box towards the right, labeled "Knowledge Base".

Knowledge-Based Agent

- Agent that uses **prior or acquired** knowledge to achieve its goals
- make more efficient decisions
- can make informed decisions
- Knowledge Base (KB)**: contains a set of **presentations** of facts about the agent's environment
 - Each representation is called a **sentence**
- use some **knowledge representation language**, to TELL (告诉) and TO KNOW (知道)
 - e.g. temperature 72F
- Then it can **ask** (询问) itself what to do - answers should follow from the KB
- Agent can use **Inference** to deduce new facts from TELLED facts

```

function KB-AGENT[percept] returns an action
  static: KB, a knowledge base
          t, a counter, initially 0, indicating time

  TELL(KB,MAKE-PERCEPT-SENTENCE[percept,t])
  action← ASK(KB,MAKE-ACTION-QUERY[t])
  TELL(KB,MAKE-ACTION-SENTENCE[action,t])
  t← t + 1
  return action

```

- TELL KB what was perceived
Uses a Knowledge Representation Language (KRL) to insert new sentences, representations of facts, into KB.
- ASK KB what to do
Uses logical reasoning to examine actions and select best.
- TELL KB the agent has in fact taken the action

Outline

- Knowledge-Based Agent
- Why FOL?
- Syntax and semantics of FOL
- Using FOL
- Wumpus world in FOL
- Knowledge engineering in FOL

First-order logic

Whereas **propositional logic** assumes the world contains facts

First-order logic (like natural language) contains:

- Objects: people, houses, numbers, colors, baseball games, wars, ...
- Relations: red, round, prime, brother of, bigger than, part of, comes between, ...
- Functions: father of, best friend, one more than, plus, ...

FOL combines the merits of both **propositional logic** and **natural language**

Examples:

"One plus two equals three"

Objects: one, two, three

Relations: equals

Properties: --

Functions: plus ("one plus two" is the name of the object obtained by applying function plus to one and two; three is another name for this object)

Examples:

"One plus two equals three"

Objects: one, two, three, one plus two

Relations: equals

Properties: --

Functions: plus ("one plus two" is the name of the object obtained by applying function plus to one and two; three is another name for this object)

Syntax of FOL: Basic elements

Constants	KingJohn, 2,
Predicates	Brother, >, ...
Functions	Sqrt, ...
Variables	(functions, x, y, a, b, ...)
Connectives	$\neg, \Rightarrow, \wedge, \vee, \Leftrightarrow$
Equality	=
Quantifiers	\forall, \exists
量词	(量词)

Syntax of FOL: Basic elements

```

Sentence → AtomicSentence | ComplexSentence
AtomicSentence → Predicate | Predicate(Term,...) | Term = Term
ComplexSentence → { Sentence } | { Sentence }
| Sentence
| Sentence ∧ Sentence
| Sentence ∨ Sentence
| Sentence → Sentence
| Sentence ↔ Sentence
| Quantifier Variable..., Sentence

Term → Function(Term,...)
| Constant
| Variable

Quantifier → ∀ | ∃
Constant → A | X, | John |
Variable → a | x | s | ...
Predicate → True | False | After | Leaves | Running | ...
Function → Mother | LeftLeg | ...
OPERATOR PRECEDENCE : ~, =, ∧, ∨, →, ↔
  
```

Semantics

There is a correspondence between

- functions (函数), which return values
- Predicates(谓词), which are true or false

e.g.

Function: father_of(Mary) = Bill

Predicate: father_of(Mary, Bill)

Atomic sentences

Atomic sentence: predicate, predicate (term₁,...,term_n) , term₁ = term₂ (等词)

term (项) is a logical expression that refers to an object, includes:

function (term₁,...,term_n) , constant , variable

e.g.,

Brother(KingJohn, RichardTheLionheart)
>(LengthLeftLegOf(Richard))
Length(LeftLegOf(KingJohn)))

Complex sentences

Complex sentences are made from atomic sentences using connectives

$\neg S, S_1 \wedge S_2, S_1 \vee S_2, S_1 \Rightarrow S_2, S_1 \Leftrightarrow S_2$,

E.g. Sibling(KingJohn, Richard) \Rightarrow Sibling(Richard, KingJohn)

$\neg(1,2) \vee \leq(1,2)$
 $\neg(1,2) \wedge \neg \neg(1,2)$

Truth in first-order logic

- Sentences** are true with respect to a **model** and an **interpretation**
- Model** contains **objects** (domain elements) and **relations** among them
- Interpretation (解释)** specifies referents for
 - constant symbols → objects
 - predicate symbols → relations
 - function symbols → functional relations

An atomic sentence $\text{predicate}(\text{term}_1, \dots, \text{term}_n)$ is true iff the objects referred to by $\text{term}_1, \dots, \text{term}_n$ are in the relation referred to by predicate

Models for FOL: Example

Models:
The formal structures that constitute the possible worlds under consideration. Each model links the vocabulary of the logical sentences to elements of the possible world, so that the truth of any sentence can be determined.

Quantifiers 量词

Existential quantification (存在量词。There exists an x such that ...)

$\exists <variables> <sentence>$

e.g. Someone at HIT is smart: $\exists x \text{At}(x, \text{HIT}) \wedge \text{Smart}(x)$

$\exists x P :$

is true in a model m iff P is true with x being some possible object in the model

• Roughly speaking, equivalent to the disjunction of instantiations of P

$\text{At}(\text{KingJohn}, \text{HIT}) \wedge \text{Smart}(\text{KingJohn})$
 $\vee \text{At}(\text{Richard}, \text{HIT}) \wedge \text{Smart}(\text{Richard})$
 $\vee \text{At}(\text{HIT}, \text{HIT}) \wedge \text{Smart}(\text{HIT})$
 $\vee \dots$

Universal quantification

(全称量词, for all $\forall <variables> <sentence>$)

e.g. Everyone at HIT is smart: $\forall x \text{At}(x, \text{HIT}) \Rightarrow \text{Smart}(x)$

$\forall x P :$

is true in a model m iff P is true with x being each possible object in the model

Roughly speaking, equivalent to the conjunction of instantiations of P

$\text{At}(\text{KingJohn}, \text{HIT}) \Rightarrow \text{Smart}(\text{KingJohn})$
 $\wedge \text{At}(\text{Richard}, \text{HIT}) \Rightarrow \text{Smart}(\text{Richard})$
 $\wedge \text{At}(\text{HIT}, \text{HIT}) \Rightarrow \text{Smart}(\text{HIT})$
 $\wedge \dots$

A common mistake to avoid

• Typically, \Rightarrow is the main connective with \forall

• Common mistake: using \wedge as the main connective with \forall , e.g.,

$\forall x \text{At}(x, \text{HIT}) \wedge \text{Smart}(x)$

means "Everyone is at HIT and everyone is smart"

Equality

$term_1 = term_2$

is true under a given interpretation iff $term_1$ and $term_2$ refer to the same object (differ from propositional terms)

e.g., definition of *Sibling* (弟兄) in terms of *Parent*
 $\forall x,y \text{Sibling}(x,y) \Leftrightarrow \neg(x = y) \wedge \exists m,f \rightarrow (m = f) \wedge \text{Parent}(m,x) \wedge \text{Parent}(f,x) \wedge \text{Parent}(m,y) \wedge \text{Parent}(f,y)$

Using FOL

Properties of quantifiers

• $\forall x \forall y$ is the same as $\forall y \forall x$

• $\exists x \exists y$ is the same as $\exists y \exists x$

• $\exists x \forall y$ is not the same as $\forall y \exists x$
e.g., $\exists x \forall y \text{Loves}(x,y)$ "There is a person who loves everyone in the world"
 $\forall y \exists x \text{Loves}(x,y)$ "Everyone in the world is loved by at least one person"

• Quantifier duality (量词互换): each \forall \exists and \exists \forall using the other

Think about De Morgan's rules and that:
 \forall denotes the conjunction of all objects in the model
 \exists denotes the disjunction of all objects in the model

e.g.,
 $\forall x \text{Likes}(x, \text{IceCream}) \quad \neg \exists x \neg \text{Likes}(x, \text{IceCream})$
 $\exists x \text{Likes}(x, \text{Broccoli}) \quad \neg \forall x \neg \text{Likes}(x, \text{Broccoli})$

Using FOL

The kinship (亲属关系) domain

Objects: persons

Unary Predicates: Male, Female

Binary Predicates: Parent, Sibling, Brother, Sister, Spouse, Grandparent, etc.

Functions: Father, Mother (it only return a unique value for each input, in general)

Brothers are siblings: $\forall x,y \text{Brother}(x,y) \Leftrightarrow \text{Sibling}(x,y)$
One's mother is one's female parent:

$\forall m,c \text{Mother}(c) = m \Leftrightarrow (\text{Female}(m) \wedge \text{Parent}(m,c))$

Theorems(定理): entailed by axioms. e.g.,
"Sibling" is symmetric: $\forall x,y \text{Sibling}(x,y) \Leftrightarrow \text{Sibling}(y,x)$

Using FOL

The set domain

Syntactic sugar (含糖语法, using infix (中缀) notation)

Objects: set, {}, (constant for empty set), element

Predicates: Set, \in , \subseteq , $=$, \cap , \cup , $\{x|s\}$ (adjoining x to set s)

- $\forall s \text{ Set}(s) \Leftrightarrow (s = \emptyset) \vee (\exists x, s_2 \text{ Set}(s_2) \wedge s = \{x|s_2\})$
- $\neg \exists x, s \in s \Leftrightarrow s = \emptyset$
- $\forall x, s \in s \Leftrightarrow \exists y, s_2 (s = \{y|s_2\} \wedge (x = y \vee x \in s_2))$
- $\forall s_1, s_2, s_1 \subseteq s_2 \Leftrightarrow (\forall x \in s_1 \rightarrow x \in s_2)$
- $\forall s_1, s_2 (s_1 = s_2) \Leftrightarrow (s_1 \subseteq s_2 \wedge s_2 \subseteq s_1)$
- $\forall x, s_1, s_2 (x \in (s_1 \cap s_2) \Leftrightarrow (x \in s_1 \wedge x \in s_2))$
- $\forall x, s_1, s_2 (x \in (s_1 \cup s_2) \Leftrightarrow (x \in s_1 \vee x \in s_2))$

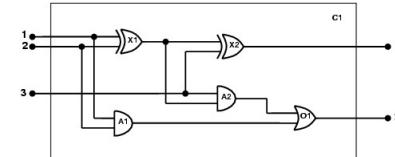
Knowledge engineering in FOL

The general process of knowledge-base construction:

1. Identify the task (PEAS?)
2. Assemble the relevant knowledge: Knowledge acquisition
3. Decide on a vocabulary of predicates, functions, and constants
4. Encode general knowledge about the domain
5. Encode a description of the specific problem instance
6. Pose queries to the inference procedure and get answers
7. Debug the knowledge base

The electronic circuits domain

One-bit full adder(1位全加器)



The electronic circuits domain

1. Identify the task
 - Does the circuit actually add properly? (circuit verification)
 2. Assemble the relevant knowledge
 - Composed of wires and gates;
 - Types of gates (AND, OR, XOR, NOT)
 - Irrelevant: size, shape, color, cost of gates
 3. Decide on a vocabulary
 - There are many of alternatives, e.g., for the type of a gate,

Type(X) = XOR
Type(X, XOR)
XOR(X)
- Which one is better? Why?

The electronic circuits domain

4. Encode general knowledge of the domain
 - $\forall t_1, t_2 \text{ Connected}(t_1, t_2) \Rightarrow \text{Signal}(t_1) = \text{Signal}(t_2)$
 - $\forall t \text{ Signal}(t) = 1 \vee \text{Signal}(t) = 0$
 - $1 \neq 0$
 - $\forall t_1, t_2 \text{ Connected}(t_1, t_2) \Rightarrow \text{Connected}(t_2, t_1)$
 - $\forall g \text{ Type}(g) = \text{OR} \Rightarrow \text{Signal}(\text{Out}(1,g)) = 1 \Leftrightarrow \exists n \text{ Signal}(\text{In}(n,g)) = 1$
 - $\forall g \text{ Type}(g) = \text{AND} \Rightarrow \text{Signal}(\text{Out}(1,g)) = 0 \Leftrightarrow \exists n \text{ Signal}(\text{In}(n,g)) = 0$
 - $\forall g \text{ Type}(g) = \text{XOR} \Rightarrow \text{Signal}(\text{Out}(1,g)) = 1 \Leftrightarrow \text{Signal}(\text{In}(1,g)) \neq \text{Signal}(\text{In}(2,g))$
 - $\forall g \text{ Type}(g) = \text{NOT} \Rightarrow \text{Signal}(\text{Out}(1,g)) \neq \text{Signal}(\text{In}(1,g))$

The electronic circuits domain

5. Encode the specific problem instance (here we define constants and connected them with common knowledge)
 - Type(X₁) = XOR
 - Type(X₂) = XOR
 - Type(A₁) = AND
 - Type(A₂) = AND
 - Type(O₁) = OR
- Connected(Out(1,X₁),In(1,X₂)) Connected(Out(1,C₁),In(1,X₁))
 Connected(Out(1,X₁),In(2,A₂)) Connected(Out(1,C₁),In(1,A₁))
 Connected(Out(1,A₂),In(1,O₁)) Connected(Out(2,C₁),In(2,X₁))
 Connected(Out(1,A₁),In(2,O₁)) Connected(Out(2,C₁),In(2,A₁))
 Connected(Out(1,X₂),Out(1,C₁)) Connected(Out(3,C₁),In(2,X₂))
 Connected(Out(1,O₁),Out(2,C₁)) Connected(Out(3,C₁),In(1,A₂))
 Connected(Out(2,C₁),In(3,C₁)) Connected(Out(1,A₂),In(3,C₁))

The electronic circuits domain

6. Pose queries to the inference procedure
 What are the possible sets of values of all the terminals for the adder circuit?

$$\exists i_1, i_2, i_3, o_1, o_2 \text{ Signal}(\text{In}(1, C_1)) = i_1 \wedge \text{Signal}(\text{In}(2, C_1)) = i_2 \wedge \text{Signal}(\text{In}(3, C_1)) = i_3 \wedge \text{Signal}(\text{Out}(1, C_1)) = o_1 \wedge \text{Signal}(\text{Out}(2, C_1)) = o_2$$
7. Debug the knowledge base
 May have omitted assertions like $1 \neq 0$

Summary

- First-order logic:
 - objects and relations are semantic primitives 语义元素
 - syntax: constants, functions, predicates, equality, quantifiers
- Increased expressive power: sufficient to define wumpus world

Homework 2: Medical Disease Diagnosis

新冠病毒确诊病例是指:

- 新冠病毒疑似病例同时具备以下病原学或血清学证据之一者:
- 1.实时荧光 RT-PCR 检测新型冠状病毒核酸阳性;
 - 2.病毒基因测序, 与已知的新型冠状病毒高度同源;
 - 3.新型冠状病毒特异性 IgM 抗体和 IgG 抗体阳性;
 - 4.新型冠状病毒特异性 IgG 抗体由阴性转为阳性或恢复期 IgG 抗体滴度较急性期呈 4 倍及以上升高
- 按照知识工程的过程, 即先描述医疗领域通用知识 (与当前疾病诊疗相关的即可), 再描述具体的疾病的诊疗知识, 将上述新冠病毒确诊病例的知识描述用一阶谓词逻辑表示出来

Chapter 9 Inference in First-order logic

Buzhou Tang
tangbuzhou@hit.edu.cn
Harbin Institute of Technology (Shenzhen)
(HIT SZ ICRC)

[Some slides were created by Dan Klein and Pieter Abbeel for CS188 Intro to AI at UC Berkeley (ai.berkeley.edu).]

Outline

- ▶ Reducing first-order inference to **propositional inference**
- ▶ Unification合一
- ▶ Generalized Modus Ponens一般化假言推理
- ▶ Forward chaining前向链接
- ▶ Backward chaining后向链接
- ▶ Resolution归结

Reducing first-order inference to propositional inference

Universal instantiation (UI)

- ▶ Every instantiation of a universally quantified sentence is entailed by it: 全称量词实例化

$$\frac{\forall v \ a}{\text{Subst}(\{v/g\}, a)}$$

for any variable v and ground term g

e.g.,

$$\frac{\forall x \text{ King}(x) \wedge \text{Greedy}(x) \Rightarrow \text{Evil}(x)}{\text{King}(\text{John}) \wedge \text{Greedy}(\text{John}) \Rightarrow \text{Evil}(\text{John})}$$

$$\frac{\forall x \text{ King}(x) \wedge \text{Greedy}(x) \Rightarrow \text{Evil}(x)}{\text{King}(\text{Richard}) \wedge \text{Greedy}(\text{Richard}) \Rightarrow \text{Evil}(\text{Richard})}$$

$$\frac{\forall x \text{ King}(x) \wedge \text{Greedy}(x) \Rightarrow \text{Evil}(x)}{\text{King}(\text{Father}(\text{John})) \wedge \text{Greedy}(\text{Father}(\text{John})) \Rightarrow \text{Evil}(\text{Father}(\text{John}))}$$

Existential instantiation (EI)

- ▶ For any sentence a , variable v , and constant symbol k that **does not** appear elsewhere in the knowledge base:

$$\frac{\exists v \ a}{\text{Subst}(\{v/k\}, a)}$$

E.g., $\exists x \text{ Crown}(x) \wedge \text{OnHead}(x, \text{John})$ yields:

$$\text{Crown } C_1 \wedge \text{OnHead } (C_1, \text{John})$$

provided C_1 is a new constant symbol, called a Skolem constant

Subst : 置换

Reduction to propositional inference

FOL 退化到命题推理

Suppose the KB contains just the following:

$$\begin{aligned} &\forall x \text{ King}(x) \wedge \text{Greedy}(x) \Rightarrow \text{Evil}(x) \\ &\text{King}(\text{John}) \\ &\text{Greedy}(\text{John}) \\ &\text{Brother}(\text{Richard}, \text{John}) \end{aligned}$$

- ▶ Instantiating the universal sentence in **all possible ways**, we have:

$$\begin{aligned} &\text{King}(\text{John}) \wedge \text{Greedy}(\text{John}) \Rightarrow \text{Evil}(\text{John}) \\ &\text{King}(\text{Richard}) \wedge \text{Greedy}(\text{Richard}) \Rightarrow \text{Evil}(\text{Richard}) \\ &\text{King}(\text{John}) \\ &\text{Greedy}(\text{John}) \\ &\text{Brother}(\text{Richard}, \text{John}) \end{aligned}$$

- ▶ The new KB is **propositionalized**: proposition symbols are $\text{King}(\text{John})$, $\text{Greedy}(\text{John})$, $\text{Evil}(\text{John})$, $\text{King}(\text{Richard})$, etc.

Reduction contd.

- ▶ Every FOL KB can be propositionalized so as to preserve entailment

(A ground sentence is entailed by new KB **iff** entailed by original KB)

Idea: propositionalize KB and query, apply resolution (归结), return result

- ▶ Problem: with function symbols (函数), there are infinitely many ground terms

e.g., $\text{Father}(\text{Father}(\text{Father}(\text{John})))$

Reduction contd.

Theorem: Herbrand (1930).

If a sentence a is entailed by an FOL KB, it is entailed by a finite subset of the propositionalized KB

Idea: For $n = 0$ to ∞ do

- create a propositional KB by instantiating with depth (n) terms
- see if a is entailed by this KB

Problem: works if a is entailed, loops if a is not entailed

Theorem: Turing (1936), Church (1936)

Entailment for FOL is **semidecidable** (半可判定的) (algorithms exist that say yes to every entailed sentence, but no algorithm exists that also says no to every non-entailed sentence.)

Problems with propositionalization

- ▶ Propositionalization (FOL的) 命题化 seems to generate lots of irrelevant sentences.

e.g., from

$$\begin{aligned} &\forall x \text{ King}(x) \wedge \text{Greedy}(x) \Rightarrow \text{Evil}(x) \\ &\text{King}(\text{John}) \\ &\forall y \text{ Greedy}(y) \\ &\text{Brother}(\text{Richard}, \text{John}) \end{aligned}$$

- ▶ it seems obvious that $\text{Evil}(\text{John})$, but propositionalization produces lots of facts such as $\text{Greedy}(\text{Richard})$ that are irrelevant
- ▶ With p k -ary predicates and n constants, there are $p \cdot n^k$ instantiations. (p 个 k 元谓词)

Outline

- ▶ Reducing first-order inference to propositional inference
- ▶ **Unification**合一
- ▶ Generalized Modus Ponens 一般化假言推理
- ▶ Forward chaining 前向链接
- ▶ Backward chaining 后向链接
- ▶ Resolution 归结

Unification

合一

Unification

- ▶ We can get the inference immediately if we can find a substitution θ such that $\text{King}(x)$ and $\text{Greedy}(x)$ match $\text{King}(\text{John})$ and $\text{Greedy}(\text{y})$:

$$\theta = \{\text{x}/\text{John}, \text{y}/\text{John}\}$$

$\text{Unify}(\text{p}, \text{q}) = \theta$ if $\text{Subst}(\theta, \text{p}) = \text{Subst}(\theta, \text{q})$

p	q	θ
Knows(John,x)	Knows(John,Jane)	$\{\text{x}/\text{Jane}\}$
Knows(John,x)	Knows(y,OJ)	
Knows(John,x)	Knows(y,Mother(y))	
Knows(John,x)	Knows(x,OJ)	

Unification

- ▶ We can get the inference immediately if we can find a substitution θ such that $\text{King}(x)$ and $\text{Greedy}(x)$ match $\text{King}(\text{John})$ and $\text{Greedy}(\text{y})$

$$\theta = \{\text{x}/\text{John}, \text{y}/\text{John}\}$$

- ▶ $\text{Unify}(\text{p}, \text{q}) = \theta$ if $\text{Subst}(\theta, \text{p}) = \text{Subst}(\theta, \text{q})$

p	q	θ
Knows(John,x)	Knows(John,Jane)	$\{\text{x}/\text{Jane}\}$
Knows(John,x)	Knows(y,OJ)	
Knows(John,x)	Knows(y,Mother(y))	
Knows(John,x)	Knows(x,OJ)	

Unification

- ▶ We can get the inference immediately if we can find a substitution θ such that $\text{King}(x)$ and $\text{Greedy}(x)$ match $\text{King}(\text{John})$ and $\text{Greedy}(\text{y})$

$$\theta = \{\text{x}/\text{John}, \text{y}/\text{John}\}$$

- ▶ $\text{Unify}(\text{p}, \text{q}) = \theta$ if $\text{Subst}(\theta, \text{p}) = \text{Subst}(\theta, \text{q})$

p	q	θ
Knows(John,x)	Knows(John,Jane)	$\{\text{x}/\text{Jane}\}$
Knows(John,x)	Knows(y,OJ)	$\{\text{x}/\text{OJ}, \text{y}/\text{John}\}$
Knows(John,x)	Knows(y,Mother(y))	
Knows(John,x)	Knows(x,OJ)	

Unification

- ▶ We can get the inference immediately if we can find a substitution θ such that $\text{King}(x)$ and $\text{Greedy}(x)$ match $\text{King}(\text{John})$ and $\text{Greedy}(\text{y})$

$$\theta = \{\text{x}/\text{John}, \text{y}/\text{John}\}$$

- ▶ $\text{Unify}(\text{p}, \text{q}) = \theta$ if $\text{Subst}(\theta, \text{p}) = \text{Subst}(\theta, \text{q})$

p	q	θ
Knows(John,x)	Knows(John,Jane)	$\{\text{x}/\text{Jane}\}$
Knows(John,x)	Knows(y,OJ)	$\{\text{x}/\text{OJ}, \text{y}/\text{John}\}$
Knows(John,x)	Knows(y,Mother(y))	$\{\text{y}/\text{John}, \text{x}/\text{Mother}(\text{John})\}$
Knows(John,x)	Knows(x,OJ)	

Unification

- ▶ We can get the inference immediately if we can find a substitution θ such that $\text{King}(x)$ and $\text{Greedy}(x)$ match $\text{King}(\text{John})$ and $\text{Greedy}(\text{y})$

$$\theta = \{\text{x}/\text{John}, \text{y}/\text{John}\}$$

- ▶ $\text{Unify}(\text{p}, \text{q}) = \theta$ if $\text{Subst}(\theta, \text{p}) = \text{Subst}(\theta, \text{q})$

p	q	θ
Knows(John,x)	Knows(John,Jane)	$\{\text{x}/\text{Jane}\}$
Knows(John,x)	Knows(y,OJ)	$\{\text{x}/\text{OJ}, \text{y}/\text{John}\}$
Knows(John,x)	Knows(y,Mother(y))	$\{\text{y}/\text{John}, \text{x}/\text{Mother}(\text{John})\}$
Knows(John,x)	Knows(x,OJ)	{fail}

Unification

- ▶ We can get the inference immediately if we can find a substitution θ such that $\text{King}(x)$ and $\text{Greedy}(x)$ match $\text{King}(\text{John})$ and $\text{Greedy}(\text{y})$

$$\theta = \{\text{x}/\text{John}, \text{y}/\text{John}\}$$

- ▶ $\text{Unify}(\text{p}, \text{q}) = \theta$ if $\text{Subst}(\theta, \text{p}) = \text{Subst}(\theta, \text{q})$

p	q	θ
Knows(John,x)	Knows(John,Jane)	$\{\text{x}/\text{Jane}\}$
Knows(John,x)	Knows(y,OJ)	$\{\text{x}/\text{OJ}, \text{y}/\text{John}\}$
Knows(John,x)	Knows(y,Mother(y))	$\{\text{y}/\text{John}, \text{x}/\text{Mother}(\text{John})\}$
Knows(John,x)	Knows(x,OJ)	{fail}

The last one failed, it is for the same variable name x for different quantifier. Standardizing [open](#) eliminates overlap of variables, e.g., let the q be $\text{knows}(z, OJ)$ for the last sentence, i.e., use z to replace x .

Unification

- ▶ To unify $\text{Knows}(\text{John}, \text{x})$ and $\text{Knows}(\text{y}, \text{z})$

$$\theta = \{\text{y}/\text{John}, \text{x}/\text{z}\} \text{ or } \theta = \{\text{y}/\text{John}, \text{x}/\text{John}, \text{z}/\text{John}\}$$

- ▶ The first unifier is **more general** than the second.

- ▶ There is a single **most general unifier (MGU)** 最一般合一置换 that is unique up to renaming of variables*.

$$\text{MGU} = \{\text{y}/\text{John}, \text{x}/\text{z}\}$$

The unification algorithm

```

function UNIFY(x, y, θ) returns a substitution to make x and y identical
  inputs: x, a variable, constant, list, or compound
          y, a variable, constant, list, or compound
          θ, the substitution built up so far
  if θ = failure then return failure
  else if x = y then return θ
  else if VARIABLE?(x) then return UNIFY-VAR(x, y, θ)
  else if VARIABLE?(y) then return UNIFY-VAR(y, x, θ)
  else if COMPOUND?(x) and COMPOUND?(y) then
    return UNIFY(ARGS[x], ARGS[y], UNIFY(Op[x], Or[y], θ))
  else if List?(x) and List?(y) then
    return UNIFY(REST[x], REST[y], UNIFY(FIRST[x], FIRST[y], θ))
  else return failure

Op[x] extract the function symbol of x, ARGS[x] extract the arguments of x.
  
```

The unification algorithm

```

function UNIFY-VAR(var, x, θ) returns a substitution
  inputs: var, a variable
          x, any expression
          θ, the substitution built up so far
  if {var=val} ∈ θ then return UNIFY(val, x, θ)
  else if {x=val} ∈ θ then return UNIFY(var, val, θ)
  else if OCCUR-CHECK?(var, x) then return failure
  else return add {var=x} to θ

OCCUR-CHECK check whether the var occurred in the compound
expression x before unify them, if so, return failure.

e.g. 调用算法UNIFY({ Knows(John,x), Knows(John,Jane)}), 给出算法调用
过程?
  
```

- ## Outline
- ▶ Reducing first-order inference to propositional inference
 - ▶ Unification
 - ▶ Generalized Modus Ponens
 - ▶ Forward chaining
 - ▶ Backward chaining
 - ▶ Resolution

Generalized Modus Ponens

一般化假言推理

Generalized Modus Ponens (GMP)

$$\frac{p_1', p_2', \dots, p_n', (p_1 \wedge p_2 \wedge \dots \wedge p_n \Rightarrow q)}{\text{Subst } (\Theta, q)}$$

where $\text{Subst } (\Theta, p_i') = \text{Subst } (\Theta, p_i)$ for all i

(一般化假言推理规则)

p_1' is $\text{King}(John)$ p_1 is $\text{King}(x)$
 p_2' is $\text{Greedy}(y)$ p_2 is $\text{Greedy}(x)$
 Θ is $[x/John, y/John]$ q is $\text{Evil}(x)$
 $\text{Subst } (\Theta, q)$ is $\text{Evil}(John)$

► GMP used with KB of **definite clauses** (exactly one positive literal, 确定子句)

► All variables assumed **universally quantified**

Soundness of GMP

- ▶ Need to show that $p_1' \wedge \dots \wedge p_n' \wedge (p_1 \wedge p_2 \wedge \dots \wedge p_n \Rightarrow q) \models \text{Subst } (\Theta, q)$
- ▶ provided that $\text{Subst } (\Theta, p_i') = \text{Subst } (\Theta, p_i)$ for all i
- ▶ **Lemma:** For any sentence p , we have $p \models \text{Subst } (\Theta, p)$ by UI (全称量词实例化)
 1. $(p_1 \wedge \dots \wedge p_n \Rightarrow q) \models \text{Subst}(\Theta, (p_1 \wedge \dots \wedge p_n \Rightarrow q))$
 $= (\text{Subst}(\Theta, p_1) \wedge \dots \wedge \text{Subst}(\Theta, p_n) \Rightarrow \text{Subst}(\Theta, q))$
 2. $p_i' \wedge \dots \wedge p_n' \models \text{Subst}(\Theta, p_i') \wedge \dots \wedge \text{Subst}(\Theta, p_n) = \text{Subst}(\Theta, p_i) \wedge \dots \wedge \text{Subst}(\Theta, p_n)$
 3. From 1 and 2, $\text{Subst}(\Theta, q)$ follows by ordinary Modus Ponens

FOL definite clauses(确定子句)

- ▶ Disjunctions of literals of which **exactly one is positive** (肯定)
- ▶ A definite clause either is **atomic** or is an **implication** whose antecedent (前项) is a conjunction of positive literals and whose consequent(后项) is a single positive literal.
- Eg. $\text{King}(x) \wedge \text{Greedy}(x) \Rightarrow \text{Evil}(x)$
 $\text{King}(John)$
 $\text{Greedy}(y)$
- ▶ Unlike propositional literals, FOL literals can include **variables (universally quantified)**
- ▶ Not every KB can be converted into a set of definite clauses. But many can.

Example knowledge base

e.g.

The law says that it is a crime for an American to sell weapons to hostile nations. The country Nono, an enemy of America, has some missiles, and all of its missiles were sold to it by Colonel West, who is American.

Prove that Colonel West is a criminal

Example knowledge base cont'd

... it is a crime for an American to sell weapons to hostile nations:
 $\text{American}(x) \wedge \text{Weapon}(y) \wedge \text{Sells}(x,y,z) \wedge \text{Hostile}(z) \rightarrow \text{Criminal}(x)$

Nono ... has some missiles, i.e., $\exists \text{Owns}(\text{Nono}, x) \wedge \text{Missile}(x)$:
 $\text{Owns}(\text{Nono}, M_1)$
 $\text{Missile}(M_1)$

... all of its missiles were sold to it by Colonel West
 $\text{Missile}(x) \wedge \text{Owns}(\text{Nono}, x) \Rightarrow \text{Sells}(\text{West}, x, \text{Nono})$

Missiles are weapons:
 $\text{Missile}(x) \rightarrow \text{Weapon}(x)$

An enemy of America counts as "hostile":
 $\text{Enemy}(x, \text{America}) \Rightarrow \text{Hostile}(x)$

West, who is American ...
 $\text{American}(\text{West})$

The country Nono, an enemy of America ...
 $\text{Enemy}(\text{Nono}, \text{America})$

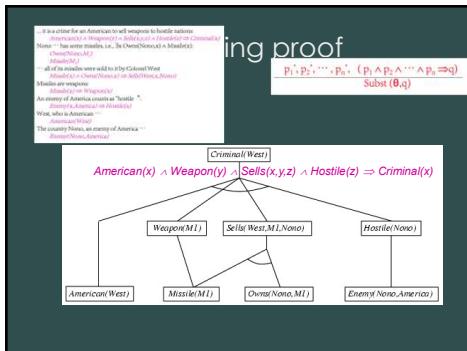
Forward chaining algorithm

```

function FOL-FC-ASK( $KB, \alpha$ ) returns a substitution or false
  inputs:  $KB$ , the knowledge base, a set of first-order definite clauses
          $\alpha$ , the query, an atomic sentence
  local variables:  $new$ , the new sentences inferred on each iteration

  repeat until  $new$  is empty
     $new \leftarrow \{\}$ 
    for each rule in  $KB$  do
       $(p_1 \wedge \dots \wedge p_n \rightarrow q) = STANDARDIZE-VARIABLES(rule)$ 
      for each  $\theta$  such that  $SUBST(\theta, p_1 \wedge \dots \wedge p_n) = SUBST(\theta, p'_1 \wedge \dots \wedge p'_n)$ 
        for some  $p'_1, \dots, p'_n$  in  $KB$ 
           $new \leftarrow SUBST(\theta, q)$ 
        if  $q'$  does not unify with some sentence already in  $KB$  or  $new$  then
          add  $q'$  to  $new$ 
           $\phi = UNIFY(q', \alpha)$ 
          if  $\phi$  is not fail then return  $\phi$ 
  add  $new$  to  $KB$ 
return false

```



Properties of forward chaining

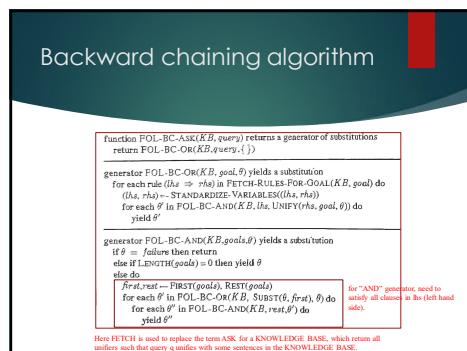
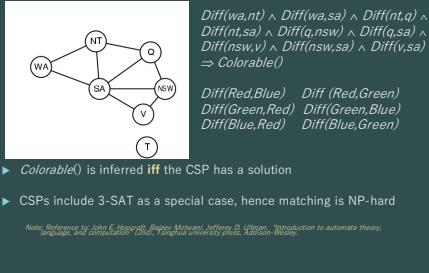
- ▶ Sound and complete for first-order definite clauses
 - ▶ Datalog = first-order definite clauses + no functions
 - ▶ FC terminates for Datalog in finite number of iterations
 - ▶ May not terminate in general if α is not entailed
(对于一般问题待证明项不成立的时候有可能不会终止)
 - ▶ This is unavoidable: entailment with definite clauses is semidecidable

Efficiency of forward chaining

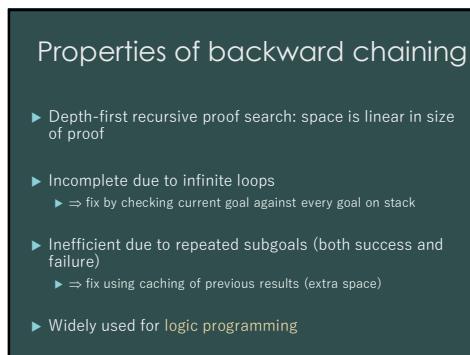
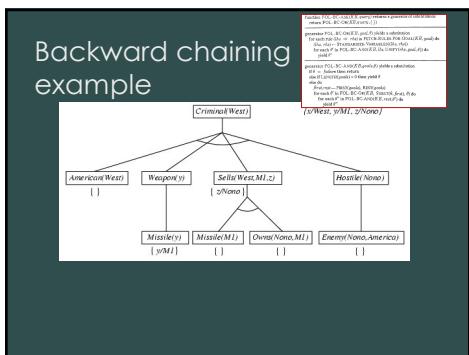
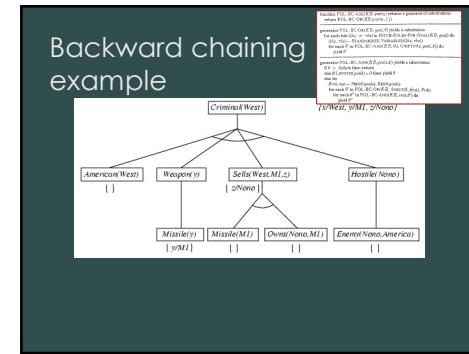
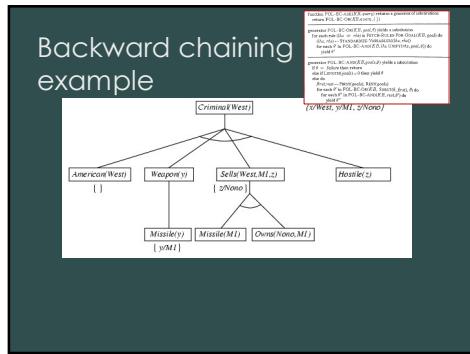
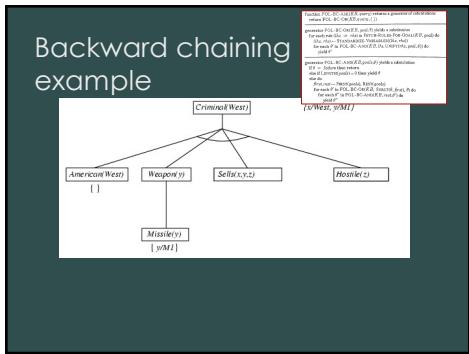
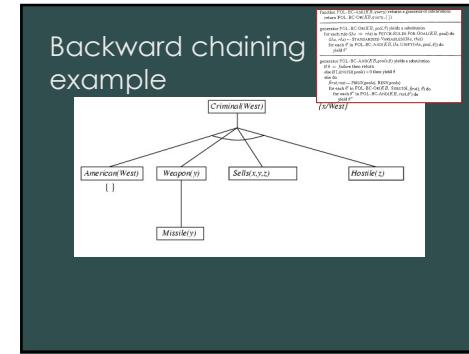
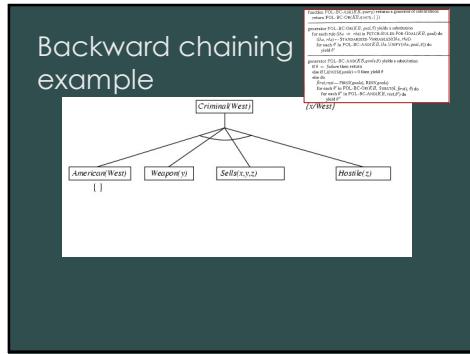
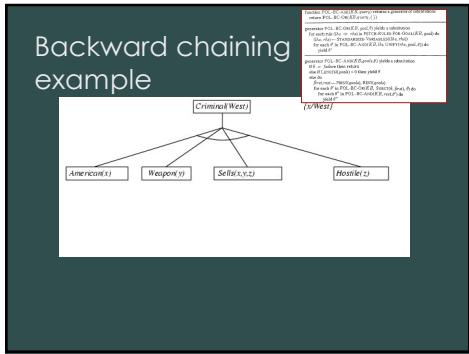
Incremental forward chaining: no need to match a rule on iteration k if a premise wasn't added on iteration $k-1$
⇒ match each rule whose premise contains a newly added positive literal

Forward chaining is widely used in deductive databases
演绎数据库

Hard matching example



Backward chaining example



Logic programming: Prolog

- ▶ Algorithm = Logic + Control
- ▶ Program = set of clauses = head :- literal₁, ..., literal_n
 criminal(X) :- american(X), weapon(Y), sells(X,Y,Z), hostile(Z)
- ▶ Depth-first, left-to-right backward chaining
- ▶ A set of built-in functions for arithmetic etc., e.g., X is Y+Z+3
- ▶ Built-in predicates that have side effects (e.g., input and output predicates, assert/retract predicates)
- ▶ Closed-world assumption ("negation as failure")
 - ▶ e.g., given alive(X) :- not dead(X)
 - ▶ alive(joe) succeeds if dead(joe) fails

Prolog

- ▶ Appending two lists to produce a third:
 $\text{append}([], Y, Y).$
 $\text{append}([A|X], Y, [A|Z]) :- \text{append}(X, Y, Z).$
- (if Z is the result of appending X to Y , then $[A|Z]$ is the result of appending $[A|X]$ on Y . A is the head of a list)
- ▶ query: $\text{append}(A, B, [1, 2]) ?$
- ▶ answers: $A = [] \quad B = [1, 2]$
 $A = [1] \quad B = [2]$
 $A = [1, 2] \quad B = []$

Outline

- ▶ Reducing first-order inference to propositional inference
- ▶ Unification
- ▶ Generalized Modus Ponens
- ▶ Forward chaining
- ▶ Backward chaining
- ▶ Resolution

Resolution 归结

Resolution: brief summary

- ▶ Full first-order logic version
 $I_1 \vee \dots \vee I_n \quad M_1 \vee \dots \vee M_m$
- where $\text{Unify}(I_i, -M_j) = \Theta$
- ▶ The two clauses are assumed to be standardized apart so that they share no variables.
- ▶ e.g.
 $\neg R(\text{Ken}) \vee \text{Unhappy}(\text{Ken}) \quad R(\text{Ken})$
 $\text{Unhappy}(\text{Ken})$
with $\Theta = \{\text{Ken}/\text{Ken}\}$
- ▶ Apply resolution steps to CNF($KB \wedge \neg Q$): complete for FOL

Conversion to CNF

- ▶ e.g. Everyone who loves all animals is loved by someone:
 $\forall y \forall x \forall z (\text{Animal}(y) \rightarrow \text{Loves}(x, y)) \Rightarrow (\exists y \forall z \text{Loves}(y, z))$

1. Eliminate biconditionals and implications
 $\forall x \forall y \forall z (\neg \text{Animal}(y) \vee \text{Loves}(x, y)) \vee (\exists y \forall z \text{Loves}(y, z))$
 $\neg \forall x \text{ 不是所有的 } x \rightarrow \exists x \text{ 不存在 } x$
2. Move \neg inwards: $\neg \forall x p \equiv \exists x \neg p, \neg \exists x p \equiv \forall x \neg p$
 $\forall x \forall y \forall z (\neg (\neg \text{Animal}(y) \vee \text{Loves}(x, y))) \vee (\exists y \forall z \text{Loves}(y, z))$
 $\forall x \forall y \forall z (\neg \text{Animal}(y) \wedge \neg \text{Loves}(x, y)) \vee (\exists y \forall z \text{Loves}(y, z))$
 $\forall x \forall y \forall z (\text{Animal}(y) \wedge \neg \text{Loves}(x, y)) \vee (\exists y \forall z \text{Loves}(y, z))$

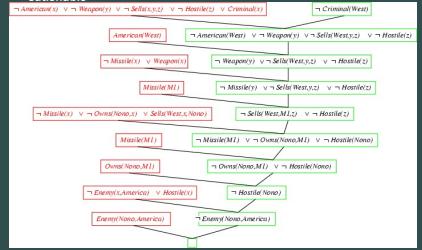
Do not need to care about the natural language meaning of the converted CNF

Conversion to CNF contd.

3. Standardize variables: each quantifier should use a different one
 $\forall x \exists y \forall z (\text{Animal}(y) \wedge \neg \text{Loves}(x, y)) \vee \exists z \forall x \text{Loves}(z, x)$
4. Skolemize: a more general form of existential instantiation
Each existential variable is replaced by a Skolem function of the enclosing universally quantified variables
 $\forall x \forall z (\text{Animal}(F(x)) \wedge \neg \text{Loves}(x, F(x))) \vee \text{Loves}(G(x), x)$
Why not $\text{Animal}(F), \neg \text{Loves}(x, F)$ and $\text{Loves}(G, x)$?
5. Drop universal quantifiers
 $[\text{Animal}(F(x)) \wedge \neg \text{Loves}(x, F(x))] \vee \text{Loves}(G(x), x)$
6. Distribute \vee over \wedge
 $[\text{Animal}(F(x)) \vee \text{Loves}(G(x), x)] \wedge [\neg \text{Loves}(x, F(x)) \vee \text{Loves}(G(x), x)]$

Resolution proof: definite clauses

Prove that West is crime. We need to prove that $KB \wedge \neg \text{Criminal}(West)$ is not satisfiable



Resolution proof example 2 (in class exercise)

e.g. the problem is given as:

Everyone who loves all animals is loved by someone.
Anyone who kills an animal is loved by no one.
Jack loves all animals.
Either Jack or Curiosity killed the cat, who is named Tuna.
Did Curiosity kill the cat?

Resolution proof example 2 contd.

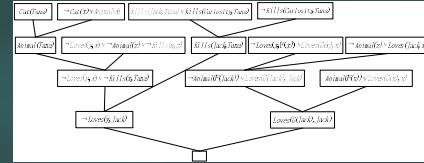
▶ First, we express the original sentences, some background knowledge, and the negated goal G in FOL:

- $\forall y \forall x (\text{Animal}(y) \rightarrow \text{Loves}(x, y)) \Rightarrow (\exists y \forall z \text{Loves}(y, z))$
- $\forall x \forall y \forall z (\exists y \forall z (\text{Animal}(y) \wedge \text{kills}(x, y)) \Rightarrow (\forall z \neg \text{Loves}(z, x)))$
- $\forall x \text{Animal}(x) \Rightarrow \text{Loves}(Jack, x)$
- $\text{kills}(Jack, Tuna) \vee \text{kills}(Curiosity, Tuna)$
- $\text{Cat}(Tuna)$
- $\forall x \text{Cat}(x) \Rightarrow \text{Animal}(x)$
- $\neg G. \quad \neg \text{kills}(Curiosity, Tuna)$

Resolution proof example 2 contd.

Resolution proof example 2 contd.

- ▶ The resolution proof that Curiosity killed the cat is given below.



Homework4:

Everyone who loves all animals is loved by someone.

Anyone who kills an animal is loved by no one.
Jack loves all animals.

Either Jack or Curiosity killed the cat, who is named Tuna.

Did Curiosity kill the cat ?

- ▶ In this homework, you are asked to write a python program to implement the Unification algorithm, the input of this program is two FOL sentences s_1 and s_2 , output is the substitution that can unify s_1 and s_2 or “fail”

V. Constraint Satisfaction Problems

Buzhou Tang
tangbuzhou@hit.edu.cn
Harbin Institute of Technology (Shenzhen)
Intelligent Computation Research Center

Reference: [Stuart Russell, Berkeley]

View Slides

Outline

- ▶ Constraint Satisfaction Problems (CSP)
- ▶ Backtracking search for CSPs (回溯搜索)
- ▶ Problem structure and problem decomposition
- ▶ Local search for CSPs

Agent Program Components

- ▶ Atomic state
- ▶ Factored state
- ▶ Structured state

(a) Atomic (b) Factored (c) Structured

Constraint satisfaction problems (CSPs)

- ▶ Standard search problem:
 - ▶ state is a "black box" – any data structure that supports successor function, heuristic function, and goal test
- ▶ CSP:
 - ▶ state is defined by variables X_i with values from domain D_i
 - ▶ goal test is a set of constraints specifying allowable combinations of values for subsets of variables
 - ▶ Simple example of a formal representation language
- ▶ Allows useful general-purpose algorithms with more power than standard search algorithms

A Logic Puzzle

- ▶ Consider the following logic puzzle: In five houses, each with a different color, live 5 persons of different nationalities, each of whom prefer a different brand of cigarette, a different drink, and a different pet.
 - ▶ The Englishman lives in the red house.
 - ▶ The Portuguese owns the dog.
 - ▶ The Norwegian lives in the first house on the left.
 - ▶ Marlboro are smoked in the yellow house.
 - ▶ The man who smokes Chesterfields lives in the house next to the man with the fox.
 - ▶ The Norwegian lives next to the blue house.
 - ▶ The Barclay smoker owns a rabbit.
 - ▶ The Lucky Strike smoker drinks orange juice.
 - ▶ The Russian drinks tea.
 - ▶ The Chinese smokes Parisienne.
 - ▶ Marlboro are smoked in the house next to the house where the horse is kept.
 - ▶ Coffee is drunk in the green house.
 - ▶ The Green house is immediately to the right (your right) of the white house.
 - ▶ Milk is drunk immediately to the right (your right) of the yellow house.
- ▶ Question: Who has a fish, where does the zebra live, and in which house do they drink water?

Example: Map-Coloring

- ▶ Variables WA, NT, Q, NSW, V, SA, T
- ▶ Domains $D_i = \{\text{red}, \text{green}, \text{blue}\}$
- ▶ Constraints adjacent regions must have different colors
 - ▶ e.g., WA ≠ NT (if the language allows this), or (WA,NT) ∈ {(\text{red},\text{green}), (\text{red},\text{blue}), (\text{green},\text{red}), (\text{green},\text{blue}), (\text{blue},\text{red}), (\text{blue},\text{green})}

Example: Map-Coloring

- ▶ Solutions are complete and consistent assignments, e.g.,
 - WA = red, NT = green, Q = red, NSW = green, V = red, SA = blue, T = green

Constraint graph

- ▶ Binary CSP (二元约束满足问题) : each constraint relates two variables
- ▶ Constraint graph: nodes are variables, arcs are constraints
- ▶ General-purpose CSP algorithms use the graph structure to speed up search. E.g., Tasmania is an independent subproblem!

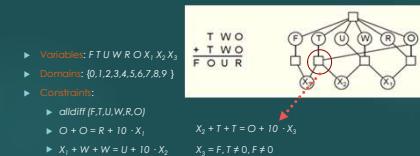
Varieties of CSPs

- ▶ Discrete variables
 - ▶ finite domains:
 - ▶ n variables, domain size $d \rightarrow O(d^n)$ complete assignments (完全赋值问题)
 - ▶ e.g., Boolean CSPs, incl. Boolean satisfiability (NP-complete)
 - ▶ infinite domains:
 - ▶ integers, strings, etc.
 - ▶ e.g., job scheduling, variables are start/end days for each job
 - ▶ need a constraint language, e.g., StartJob1 + 5 ≤ StartJob3
 - ▶ linear constraints solvable, nonlinear undecidable
- ▶ Continuous variables
 - ▶ e.g., start/end times for Hubble Space Telescope observations
 - ▶ linear constraints solvable in polynomial time by linear programming methods

Varieties of constraints

- ▶ **Unary** constraints involve a single variable. (一元约束)
 - ▶ e.g., $SA \neq \text{green}$
- ▶ **Binary** constraints involve pairs of variables.
 - ▶ e.g., $SA \neq WA$
- ▶ **Higher-order** constraints involve 3 or more variables.
 - ▶ e.g., cryptarithmic column constraints (密码算术列约束)
- ▶ **Preferences** (优先向题soft constraints), e.g., **red** is better than **green**, often representable by a cost for each variable assignment
 - constrained optimization problems

Example: Cryptarithmetic



Real-world CSPs

- ▶ Assignment problems 赋值问题
 - ▶ e.g., who teaches what class
- ▶ Timetabling problems
 - ▶ e.g., which class is offered when and where?
- ▶ Hardware configuration
- ▶ Transportation scheduling
- ▶ Factory scheduling

Notice that many real-world problems involve real-valued variables

Standard search formulation (incremental)

Let's start with the straightforward approach, then fix it.
States are defined by the values assigned so far

- ▶ Initial state: the empty assignment {}
- ▶ Successor function: assign a value to an unassigned variable that does not conflict with current assignment
 - fail if no legal assignments
- ▶ Goal test: the current assignment is complete

1. This is the same for all CSPs!
2. Every solution appears at depth n with n variables
 - use depth-first search
3. Path is irrelevant, so can also use complete-state formulation

◀ $b = (n - l) \cdot d$ at depth l , hence a^b leaves!!!!

Backtracking search (回溯搜索)

Backtracking search (回溯搜索)

- ▶ Variable assignments are **commutative**, i.e.,
 $[WA = \text{red} \text{ then } NT = \text{green}]$ same as $[NT = \text{green} \text{ then } WA = \text{red}]$
- ▶ Only need to consider assignments to a single variable at each node
 - $b = d$ and there are a^b leaves
- ▶ Depth-first search for CSPs with single-variable assignments is called **backtracking search**
- ▶ Backtracking search is the basic uninformed algorithm for CSPs
- ▶ Can solve n -queens for $n \geq 25$

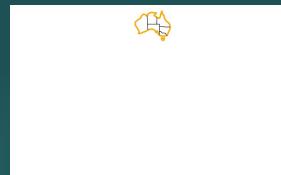
Backtracking search

```

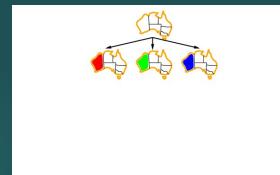
function BACKTRACKING-SEARCH(csp) returns a solution, or failure
  return BACKTRACK([], csp)

function BACKTRACK(assignment, csp) returns a solution, or failure
  if assignment is complete then
    return assignment
  var ← SELECT-UNASSIGNED-VARIABLE(csp)
  for each value in ORDER-DOMAIN-VALUES(var, assignment, csp) do
    if var is consistent with assignment then
      add {var = value} to assignment
      inferences ← INFERENCE(csp, var, value)
      if inferences ≠ failure then
        add inferences to assignment
        result ← BACKTRACK(assignment, csp)
        if result ≠ failure then
          return result
      remove {var = value} and inferences from assignment
  return failure
  
```

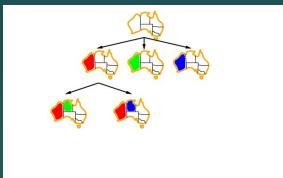
Backtracking example



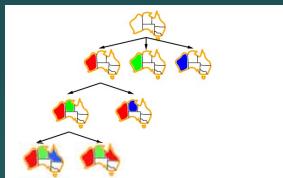
Backtracking example



Backtracking example



Backtracking example

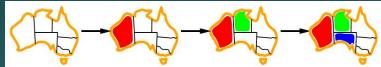


Improving backtracking efficiency

- General-purpose methods can give huge gains in speed:
 - Which variable should be assigned next?
 - In what order should its values be tried?
 - Can we detect inevitable failure early?
 - Can we take advantage of problem structure?

Most constrained variable

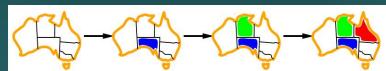
- Most constrained variable:
choose the variable with the fewest legal values



- a.k.a. minimum remaining values (MRV 最少剩余值) heuristic

Most constrained variable

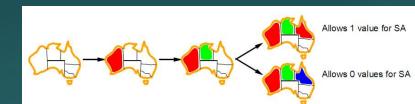
- Tie-breaker among most constrained variables
- Most constrained variable:



- choose the variable with the most constraints on remaining variables

Least constraining value

- Given a variable, choose the least constraining value



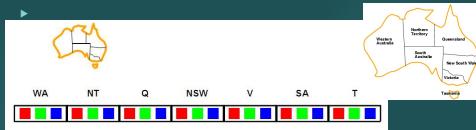
- the one that rules out (排除) the fewest values in the remaining variables

- Combining these heuristics makes 1000 queens feasible

Forward checking

Idea:

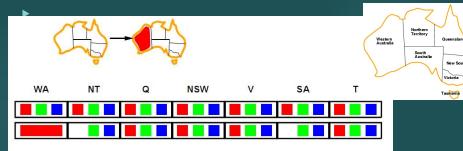
- Keep track of remaining legal values for unassigned variables
- Terminate search when any variable has no legal values



Forward checking

Idea:

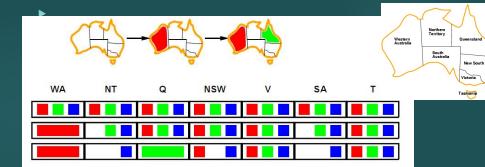
- Keep track of remaining legal values for unassigned variables
- Terminate search when any variable has no legal values



Forward checking

Idea:

- Keep track of remaining legal values for unassigned variables
- Terminate search when any variable has no legal values



Forward checking

- Idea:
 - Keep track of remaining legal values for unassigned variables
 - Terminate search when any variable has no legal values

Constraint propagation

- Forward checking propagates information from assigned to unassigned variables, but doesn't provide early detection for all failures:

NT and SA cannot both be blue!

- Constraint propagation repeatedly enforces constraints locally

Arc consistency(弧相容性)

- Simplest form of propagation makes each arc consistent
- $X \rightarrow Y$ is consistent iff for every value x of X there is some allowed y

Arc consistency

- Simplest form of propagation makes each arc consistent
- $X \rightarrow Y$ is consistent iff for every value x of X there is some allowed y

Arc consistency

- Simplest form of propagation makes each arc consistent
- $X \rightarrow Y$ is consistent iff for every value x of X there is some allowed y

If Y loses a value, neighbors of Y need to be rechecked

Arc consistency

- Simplest form of propagation makes each arc consistent
- $X \rightarrow Y$ is consistent iff for every value x of X there is some allowed y

If Y loses a value, neighbors of Y need to be rechecked

- Arc consistency detects failure earlier than forward checking
- Can be run as a preprocessor or after each assignment

Arc consistency algorithm AC-3

```

function AC-3(csp) returns the CSP, possibly with reduced domains
  inputs: csp, a binary CSP with variables { $X_1, X_2, \dots, X_n$ }
  local variables: queue, a queue of arcs, initially all the arcs in csp
  while queue is not empty do
     $(X_i, X_j) \leftarrow \text{REMOVE-FIRST}(queue)$ 
    if REMOVE-INCONSISTENT-VALUES( $X_i, X_j$ ) then
      for each  $X_k$  in NEIGHBORS( $X_j$ ) do
        add  $(X_k, X_j)$  to queue
  return csp

function REMOVE-INCONSISTENT-VALUES( $X_i, X_j$ ) returns true iff succeeds
  removed  $\leftarrow$  false
  for each  $x$  in DOMAIN( $X_i$ ) do
    if no value  $y$  in DOMAIN( $X_j$ ) allows  $(x,y)$  to satisfy the constraint  $X_i \leftrightarrow X_j$ 
      then delete  $x$  from DOMAIN( $X_i$ ); removed  $\leftarrow$  true
  return removed
  
```

Time complexity: $O(n^2d^2)$, can be reduced to $O(n^2d)$ (but detecting all is NP-hard)

Problem Structure

Extreme case:
independent subproblems
example: Tasmania and
Queensland are identifiable as
connected components

variables can be broken
into subproblems of only c
variables
Worst-case solution cost
is $O((n/c)(d^c))$, linear in n
• E.g., $n = 80$, $d = 2$, $c = 20$

Tree-Structured CSPs

Theorem: if the constraint graph has no loops, the CSP can be solved in $O(n d^c)$ time
Compare to general CSPs, where worst-case time is $O(d^n)$

This property also applies to probabilistic reasoning (later): an example of the relation between syntactic restrictions and the complexity of reasoning

Tree-Structured CSPs

- Algorithm for tree-structured CSPs:
 - Order: Choose a root variable, order variables so that parents precede children
-
- Remove backward: For $i = n : 2$, apply Remove Inconsistent($\text{Parent}(X_i), X_i$)
Assign forward: For $i = 1 : n$, assign X_i consistently with Parent(X_i)

Runtime: $O(n d^2)$ (why?)

Tree-Structured CSPs

- Claim 1: After backward pass, all root-to-leaf arcs are consistent
Proof: Each $X \rightarrow Y$ was made consistent at one point and Y 's domain could not have been reduced thereafter (because Y 's children were processed before Y)

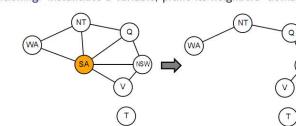


- Claim 2: If root-to-leaf arcs are consistent, forward assignment will not backtrack

Proof: Induction on position
Why doesn't this algorithm work with cycles in the constraint graph?
Note: we'll see this basic idea again with Bayes' nets

Nearly tree-structured CSPs

Conditioning: instantiate a variable, prune its neighbors' domains



Cutset conditioning: instantiate (in all ways) a set of variables such that the remaining constraint graph is a tree

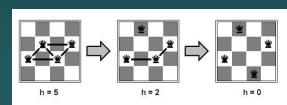
Cutset size $c \Rightarrow$ runtime $O(d^c \cdot (n - c)d^2)$, very fast for small c

Local search for CSPs

- Hill-climbing, simulated annealing typically work with "complete" states, i.e., all variables assigned
- To apply to CSPs:
 - allow states with unsatisfied constraints
 - operators **reassign** variable values
- Variable selection: randomly select any conflicted variable
- Value selection by **min-conflicts** heuristic:
 - choose value that violates the fewest constraints
 - i.e., hill-climb with $h(n)$ = total number of violated constraints

Example: 4-Queens

- States: 4 queens in 4 columns ($4^4 = 256$ states)
- Action: move queen in column
- Goal test: no attacks
- Evaluation: $h(n) = \text{number of attacks}$

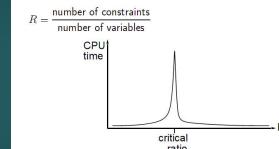


Given random initial state, can solve n -queens in almost constant time for arbitrary n with high probability (e.g., $n = 10,000,000$)

Performance of min-conflicts

Given random initial state, can solve n -queens in almost constant time for arbitrary n with high probability (e.g., $n = 10,000,000$)

The same appears to be true for any randomly-generated CSP except in a narrow range of the ratio



Summary

- CSPs are a special kind of problem:
- states defined by values of a fixed set of variables
 - goal test defined by constraints on variable values
- Backtracking** = depth-first search with one variable assigned per node
- Variable ordering** and **value selection** heuristics help significantly
- Forward checking** prevents assignments that guarantee later failure
- Constraint propagation (e.g., **arc consistency**) does additional work to constrain values and detect inconsistencies
- The CSP representation allows analysis of **problem structure**
- Tree-structured CSPs can be solved in linear time
- Iterative min-conflicts is usually effective in practice

Homework5

- Prove that the time complexity of solving n variables Tree-Structured CSPs with the domain size of d is: $O(n d^2)$
see slides page 37 for the principle of this algorithm.

Artificial Intelligence

第2部分 推理

1. 符号推理
2. 不确定性推理
3. 因果推理

Buzhou Tang
Email: tangbz@hit.edu.cn
http://icrc.hit.edu.cn

Harbin Institute of Technology (Shenzhen)
Intelligent Computation Research Center
(ICRC, HITZ)

Chapter 14 Bayesian networks (section 1-3)

Autumn 2024
Instructor: Buzhou Tang

Harbin Institute of Technology (Shenzhen) Intelligent Computation Research Center (ICRC, HITZ)

Uncertainty

Simple Example:
Let action A_1 = "leave for airport 1 minutes before flight.
Will A_1 get me there on time?"

Problems:
partial observability (road state, other drivers' plans, etc.)
ability to predict the future
uncertainty in action outcomes (flat tire, etc.)
immense complexity of modeling and predicting traffic

Hence a purely logical approach either:
risks falsehood: " A_{12} , will get me there on time", or
leads to conclusions that are too weak for decision making:
" A_{12} will get me there on time if there's no accident on the bridge and it doesn't rain and my tires remain intact etc."
(A_{12} might reasonably be said to get me there on time but I'd have to stay overnight in the airport...)

Probability

Probabilistic assertions **summarize** effects of
Laziness: failure to enumerate exceptions, qualifications, etc.
Ignorance: lack of relevant facts, initial conditions, etc.

Subjective probability (主观概率):
Probabilities relate propositions to agent's own state of knowledge

e.g., $P(A_{25} \mid \text{no reported accidents}) = 0.06$

These are not assertions (断言) about the world

Probabilities of propositions change with new evidence:

e.g., $P(A_{25} \mid \text{no reported accidents, 5 a.m.}) = 0.15$

Question

How to effectively represent uncertainty knowledge in the context of **Subjective Probability**?

Outlines

- Bayesian networks: Basic Conceptions
 - Syntax
 - Semantics
- Construction of Bayesian Networks

Outlines

- Bayesian networks: Basic Conceptions
 - Syntax
 - Semantics
- Construction of Bayesian Networks

Agent Program Components

- Atomic state
- Factored state
- Structured state

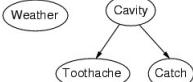
(a) Atomic (b) Factored (c) Structured

Bayesian networks

- Also known as Belief networks, probability networks, or causal networks, are a representation method for uncertain knowledge.
- A simple, graphical notation for conditional independence assertions and hence for compact specification of full joint distributions
- Syntax:
 - a set of nodes, one per variable
 - a directed, acyclic graph (link ≈ "directly influences")
 - a conditional distribution for each node given its parents:
$$P(X_i \mid \text{Parents}(X_i))$$
- In the simplest case, conditional distribution represented as a **conditional probability table** (CPT) giving the distribution over X_i for each combination of parent values

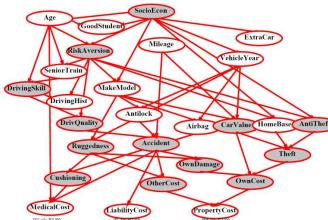
Example

- Topology of network encodes conditional independence assertions:

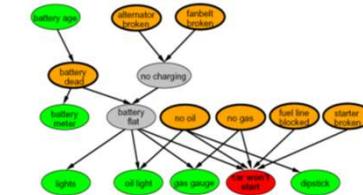


- Weather* is independent of the other variables
- Toothache* and *Catch* are conditionally independent given *Cavity*

Example Bayes' Net: Insurance



Example Bayes' Net: Car



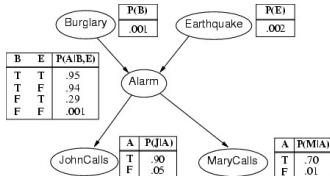
Example

- e.g.,

I'm at work, neighbor John calls to say my alarm is ringing, but neighbor Mary doesn't call. Sometimes it's set off by minor earthquakes. Is there a burglar?

- Variables: *Burglary*, *Earthquake*, *Alarm*, *JohnCalls*, *MaryCalls*
- Network topology reflects "causal" knowledge:
 - A burglar can set the alarm off
 - An earthquake can set the alarm off
 - The alarm can cause Mary to call
 - The alarm can cause John to call

Example cont'd.



Compactness (贝叶斯网络的紧致性)

- A CPT for Boolean X_i with k Boolean parents has 2^k rows for the combinations of parent values
- Each row requires one value p for $X_i = \text{true}$ (the number for $X_i = \text{false}$ is just $1-p$)
- If each variable has no more than k parents, the complete network requires $O(n \cdot 2^k)$ values
- i.e., grows linearly with n , vs. $O(2^n)$ for the full joint distribution
- For burglary net, $1 + 1 + 4 + 2 + 2 = 10$ values (vs. $2^5 - 1 = 31$?)

Semantics

- The semantics of Bayesian networks:
 - A representation of the joint probability distribution.
(Numerical semantics)
 - An encoding of a collection of conditional independence statements.
(Topological semantics)

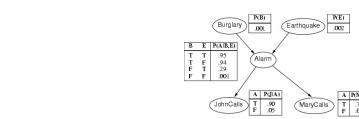
Numerical semantics

- The full joint distribution is defined as the product of the local conditional distributions:

$$P(x_1, \dots, x_n) = \prod_{i=1}^n P(x_i | \text{parents}(X_i))$$

e.g., $P(j \wedge m \wedge a \wedge \neg b \wedge \neg e)$

=



Numerical semantics

- The full joint distribution is defined as the product of the local conditional distributions:

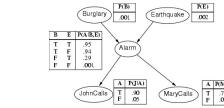
$$P(x_1, \dots, x_n) = \prod_{i=1}^n P(x_i | \text{parents}(X_i))$$

e.g., $P(j \wedge m \wedge a \wedge \neg b \wedge \neg e)$

$$= P(j|a)P(m|a)P(a|\neg b, \neg e)P(\neg b)P(\neg e)$$

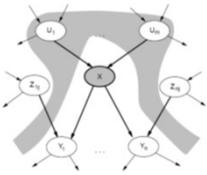
$$= 0.9 \times 0.7 \times 0.001 \times 0.999 \times 0.998$$

$$\approx 0.00063$$



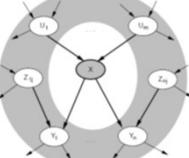
Topological semantics

- 1. Each node is conditionally independent of its non-descendants given its parents



Markov blanket

- 2. Each node is conditionally independent of all others given its **Markov blanket** 马尔科夫毯: parents + children + children's parents



- Theorem: Topological semantics \Rightarrow Numerical semantics

Outlines

- Bayesian networks: Basic Conceptions
 - Syntax
 - Semantics
- Construction of Bayesian Networks

Constructing Bayesian networks: Ordering

1. Choose an ordering of variables X_1, \dots, X_n
2. For $i = 1$ to n
 - add X_i to the network
 - select parents from X_1, \dots, X_{i-1} such that

$$P(X_i / \text{Parents}(X_i)) = P(X_i / X_1, \dots, X_{i-1})$$

This choice of parents guarantees:

$$\begin{aligned} P(X_1, \dots, X_n) &= \prod_{i=1}^n P(X_i / X_1, \dots, X_{i-1}) \quad (\text{chain rule}) \\ &= \prod_{i=1}^n P(X_i / \text{Parents}(X_i)) \quad (\text{by construction}) \end{aligned}$$

Example : The wrong order

- Suppose we choose the ordering M, J, A, B, E



$$P(J / M) = P(J)?$$

Example

- Suppose we choose the ordering M, J, A, B, E



$$P(J / M) = P(J)? \text{ No}$$

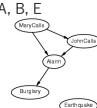
$$P(A / J, M) = P(A / J)? \quad P(A / J, M) = P(A)? \text{ No}$$

$$P(B / A, J, M) = P(B / A)?$$

$$P(B / A, J, M) = P(B)?$$

Example

- Suppose we choose the ordering M, J, A, B, E



$$P(J / M) = P(J)? \text{ No}$$

$$P(A / J, M) = P(A / J)? \quad P(A / J, M) = P(A)? \text{ No}$$

$$P(B / A, J, M) = P(B / A)? \text{ Yes}$$

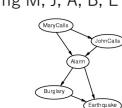
$$P(B / A, J, M) = P(B)? \text{ No}$$

$$P(E / B, A, J, M) = P(E / A)?$$

$$P(E / B, A, J, M) = P(E / A)?$$

Example

- Suppose we choose the ordering M, J, A, B, E



$$P(J / M) = P(J)? \text{ No}$$

$$P(A / J, M) = P(A / J)? \quad P(A / J, M) = P(A)? \text{ No}$$

$$P(B / A, J, M) = P(B / A)? \text{ Yes}$$

$$P(B / A, J, M) = P(B)? \text{ No}$$

$$P(E / B, A, J, M) = P(E / A)? \text{ No}$$

$$P(E / B, A, J, M) = P(E / A)? \text{ Yes}$$

Example contd.



- Deciding conditional independence is hard in noncausal directions
- (Causal models and conditional independence seem hardwired for humans!)
- Network is less compact: $1 + 2 + 4 + 2 + 4 = 13$ numbers needed

Constructing Bayesian networks: Compact conditional distributions

- CPT grows exponentially with number of parents ($O(2^n)$)
- CPT becomes infinite with continuous-valued parent or child
 - Solution: canonical distributions 规范分布 that are defined compactly
 - Deterministic nodes 确定性节点 are the simplest case:
 $X = f(\text{Parents}(X))$ for some function f
 - E.g., Boolean functions
 $\text{NorthAmerican} \Leftrightarrow \text{Canadian} \vee \text{US} \vee \text{Mexican}$
 - E.g., numerical relationships among continuous variables
 $\frac{\partial \text{Level}}{\partial t} = \text{inflow} + \text{precipitation} - \text{outflow} - \text{evaporation}$
河流量 = 入流量 + 降水量 - 出流量 - 蒸发量

Constructing Bayesian networks: Compact conditional distributions

- Noisy-OR distributions model multiple noninteracting causes
 - Parents $U_1 \dots U_n$ include all causes (can add leak node)
 - Independent failure probability q_i (抑制概率) for each cause alone
 $\Rightarrow P(X|U_1 \dots U_j, \neg U_{j+1} \dots \neg U_k) = 1 - \prod_{i=j+1}^k q_i$ denotes that X is false if all the $U_{j+1} \dots U_k$ are failure at the same time.
- | Cold | Flu | Malaria | $P(Fever)$ | $P(\neg Fever)$ |
|------|-----|---------|------------|-------------------------------------|
| F | F | F | 0.0 | 1.0 |
| F | F | T | 0.9 | 0.1 |
| F | T | F | 0.8 | 0.2 |
| F | T | T | 0.98 | $0.02 = 0.2 \times 0.1$ |
| T | F | F | 0.4 | 0.6 |
| T | F | T | 0.94 | $0.06 = 0.6 \times 0.1$ |
| T | T | F | 0.88 | $0.12 = 0.6 \times 0.2$ |
| T | T | T | 0.988 | $0.012 = 0.6 \times 0.2 \times 0.1$ |
- Number of parameters linear in number of parents ($O(k)$)

Constructing Bayesian networks: Hybrid (discrete+continuous) networks

- Discrete (Subsidy 补贴? (y or n) and Buys? (y or n); continuous (Harvest 收成 and Cost 价格)



Option 1: discretization 连续变量离散化 —— possibly large errors, large CPTs
 Option 2: finitely parameterized canonical families

- Continuous variable, discrete+continuous parents (e.g., Cost)
- Discrete variable, continuous parents (e.g., Buys?)

Constructing Bayesian networks: Continuous child variables

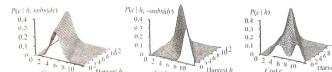
- Need one conditional density function for child variable given continuous parents, for each possible assignment to discrete parents
- Most common is the linear Gaussian (LG) model, e.g.:

$$P(Cost = c | Harvest = h, Subsidy = \text{true})$$

$$= N(a_0h + b_0, \sigma_0^2)$$

$$= \frac{1}{\sigma_0\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(c - (a_0h + b_0)\right)^2\right)$$
- Mean Cost varies linearly with Harvest, variance is fixed
- Linear variation is unreasonable over the full range, but works OK if the likely range of Harvest is narrow

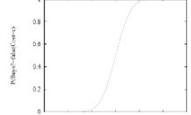
Continuous child variables (多个连续子节点)



- All-continuous network with LG distributions
 \Rightarrow full joint distribution is a multivariate Gaussian
- Discrete+continuous LG network is a conditional Gaussian network
 - i.e., a multivariate Gaussian over all continuous variables for each combination of discrete variable values

Discrete variable with continuous parents

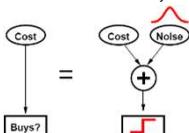
- Probability of Buys given Cost should be a "soft" threshold:



- Probit distribution (概率单位分布) uses integral of Gaussian:
 $\Phi(x) = \int_{-\infty}^x N(0, 1)(x) dx$
 $P(Buys? = \text{true} | Cost = c) = \Phi((-c + \mu)/\sigma)$

Why the probit?

- It's sort of the right shape
- Can view as hard threshold whose location is subject to noise

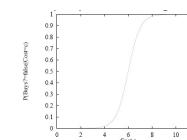


Discrete variable contd.

- Sigmoid (or logit) distribution also used in neural networks:

$$P(Buys? = \text{true} | Cost = c) = \frac{1}{1 + \exp(-\frac{c - \mu}{\sigma})}$$

- Sigmoid has similar shape to probit but much longer tails:



Summary

- Bayesian networks provide a natural representation for (causally induced) conditional independence
- Topology + CPTs = compact representation of joint distribution
- Generally easy for domain experts to construct
- Canonical distributions (e.g., noisy-OR) = compact representation of CPTs
- Continuous variables \Rightarrow parameterized distributions (e.g., linear Gaussian)

Artificial Intelligence

第2部分 推理

1. 符号推理
2. 不确定性推理
3. 因果推理

Buzhou Tang

Email: tangbz@hit.edu.cn
<http://rcrc.hit.edu.cn>

Harbin Institute of Technology (Shenzhen)
Intelligent Computation Research Center
(ICRC, HITZ)

Chapter 14 Bayesian Networks

inference (section 4-5)

Autumn 2024
Buzhou Tang

Harbin Institute of Technology (Shenzhen) Intelligent Computation Research Center (ICRC, HITZ)

Outlines

- Exact inference by enumeration
- Exact inference by variable elimination
- Approximate inference by Markov chain Monte Carlo

Inference tasks

- Simple queries: compute posterior marginal $P(X_i|E=e)$
e.g., $P(\text{NoGas}|\text{Gauge}=\text{empty}, \text{Lights}=\text{on}, \text{Starts}=\text{false})$
- Conjunctive queries: $P(X_i, X_j|E=e) = P(X_i|E=e)P(X_j|X_i, E=e)$
- Optimal decisions: decision networks include utility information;
• probabilistic inference required for $P(\text{outcome}|\text{action}, \text{evidence})$
- Value of information: which evidence to seek next?
- Sensitivity analysis: which probability values are most critical?
- Explanation: why do I need a new starter motor?

Outlines

- Exact inference by enumeration
- Exact inference by variable elimination
- Approximate inference by Markov chain Monte Carlo

Inference by enumeration

- Slightly intelligent way to sum out variables from the joint without actually constructing its explicit representation
- Simple query on the burglary network:

$$\begin{aligned} & P(B|j, m) \\ &= P(B, j, m)/P(j, m) \\ &= \alpha P(B, j, m) \\ &= \alpha \sum_e \sum_a P(B, e, a, j, m) \end{aligned}$$
- Rewrite full joint entries using product of CPT entries:

$$\begin{aligned} & P(B|j, m) \\ &= \alpha \sum_e \sum_a P(B)P(e)P(a|B, e)P(j|a)P(m|a) \\ &= \alpha P(B) \sum_e P(e) \sum_a P(a|B, e)P(j|a)P(m|a) \end{aligned}$$
- Recursive depth-first enumeration: $O(|e|)$ space, $O(d^k)$ time

Enumeration algorithm

```

function ENUMERATION-ASK(X, e, b)
  inputs: X, the query variable
  e, observed values for variables E
  b, a Bayesian network with variables {X} ∪ E ∪ Y
  Q(Y) ← {} // initially empty
  for each value x_i of X do
    extend e with x_i for X
    Q(x_i) ← ENUMERATE-ALL({VARS[i]}, e)
  return NORMALIZE(Q(X))
}

function ENUMERATE-ALL(vars, e) returns a distribution over X
  if EMPTY?(vars) then return 1.0
  y ← FIRST(vars)
  if y = e[y] then
    then return P(y | Pa(Y)) × ENUMERATE-ALL(REST(vars), e)
  else return Σ_a P(y | Pa(Y)) × ENUMERATE-ALL(REST(vars), e_y)
    where e_y is e extended with Y = y
  
```

Evaluation tree

- Enumeration is inefficient: repeated computation
e.g., computes $P(j|a)P(m|a)$ for each value of a

Outlines

- Exact inference by enumeration
- Exact inference by variable elimination
- Approximate inference by Markov chain Monte Carlo

Inference by variable elimination

- Variable elimination: carry out summations right-to-left, storing intermediate results (factors) to avoid recompilation

$$\begin{aligned}
 P(B|j, m) &= \alpha \frac{P(B)}{\overbrace{B}^E} \sum_e P(e) \sum_a \frac{P(a|B, e)}{\overbrace{A}^J} \frac{P(j|a)}{\overbrace{J}^M} \frac{P(m|a)}{\overbrace{M}^F} \\
 &= \alpha P(B) \sum_e P(e) \sum_a P(a|B, e) P(j|a) f_{JM}(a) \\
 &= \alpha P(B) \sum_e P(e) \sum_a P(a|B, e) f_J(a) f_M(a) \\
 &= \alpha P(B) \sum_e P(e) \sum_a f_A(a, b, e) f_J(a) f_M(a) \\
 &= \alpha P(B) \sum_e P(e) f_{AJM}(b, e) \quad (\text{sum out } A) \\
 &= \alpha P(B) f_{EJM}(b) \quad (\text{sum out } E) \\
 &= \alpha f_B(b) \times f_{EJM}(b) \\
 &= \alpha f_B(b) \times f_{EAJM}(b)
 \end{aligned}$$

Variable elimination: Basic operations

- Summing out a variable from a product of factors:

- move any constant factors outside the summation
- add up submatrices in pointwise product of remaining factors

$$\sum_x f_1 \times \dots \times f_k = f_1 \times \dots \times f_i \sum_x f_{i+1} \times \dots \times f_k = f_1 \times \dots \times f_i \times f_X$$

assuming f_1, \dots, f_i do not depend on X

Pointwise product of factors f_1 and f_2 :

$$\begin{aligned}
 f_1(x_1, \dots, x_j, y_1, \dots, y_k) \times f_2(y_1, \dots, y_k, z_1, \dots, z_l) \\
 = f(x_1, \dots, x_j, y_1, \dots, y_k, z_1, \dots, z_l)
 \end{aligned}$$

E.g., $f_1(n, b) \times f_2(b, c) = f(n, b, c)$

x	y	$f_1(x, y)$	$f_2(y)$	$f_1(x, y) \times f_2(y)$
T	T	2	2	4
T	F	3	1	3
F	T	1	2	2
F	F	1	1	1

Figure 14.10 Illustrating matrix multiplication $A \times B = A \otimes B$ (\otimes is the pointwise product).

Variable elimination algorithm

```

function ELIMINATION-ASK( $X, e, bn$ ) returns a distribution over  $X$ 
inputs:  $X$ , the query variable
         $e$ , evidence specified as an event
         $bn$ , a belief network specifying joint distribution  $P(X_1, \dots, X_n)$ 
factors  $\leftarrow []$ ; vars  $\leftarrow \text{REVERSE}(\text{VARS}[bn])$ 
for each var in vars do
    factors  $\leftarrow [\text{MAKE-FACTOR}(\text{var}, e)] \cup \text{factors}$ 
    if var is a hidden variable then factors  $\leftarrow \text{SUM-OUT}(\text{var}, \text{factors})$ 
return NORMALIZE(POINTWISE-PRODUCT(factors))

```

Irrelevant variables

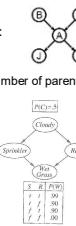
Consider the query $P(\text{JohnCalls}|\text{Burglary}=\text{true})$



Complexity of exact inference

- Singly connected networks (单连通网络 or polytrees 多形树):

- any two nodes are connected by at most one (undirected) path
- time and space cost of variable elimination are $O(d^n)$
- here d is the domain size of variables, and k is the maximum number of parents in the bn



- The complexity for multiply connected network (多连通网络) has the worst of $O(d^n)$ time and space cost

Inference by stochastic simulation

基于随机模拟的推理

Query: What's the probability of a given sequence of throwing a coin 10 times?

- Basic idea:
 - Draw N samples from a sampling distribution S
 - Compute an approximate posterior probability P
 - Show this converges to the true probability P



Inference by stochastic simulation

基于随机模拟的推理

- Outline:
 - Sampling from an empty network
 - Rejection sampling: reject samples disagreeing with evidence
 - Likelihood weighting: use evidence to weight samples
 - Markov chain Monte Carlo (MCMC): sample from a stochastic process whose stationary distribution is the true posterior

Outlines

- Exact inference by enumeration
- Exact inference by variable elimination
- Approximate inference by Markov chain Monte Carlo

Sampling from an empty network

```

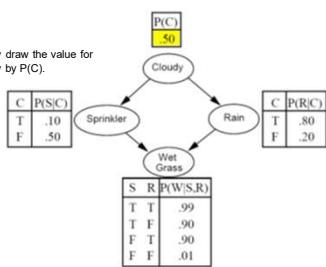
function PRIOR-SAMPLE( $bn$ ) returns an event sampled from  $bn$ 
inputs:  $bn$ , a belief network specifying joint distribution  $P(X_1, \dots, X_n)$ 
x  $\leftarrow$  an event with  $n$  elements
for  $i = 1$  to  $n$  do
     $x_i \leftarrow$  a random sample from  $P(X_i | \text{parents}(X_i))$ 
    given the values of  $\text{Parents}(X_i)$  in  $x$ 
return  $x$ 

```

Sampling from an empty network, according to the CPT given by a bn , repeat to generate samples that are consistent with the probability distribution of bn .

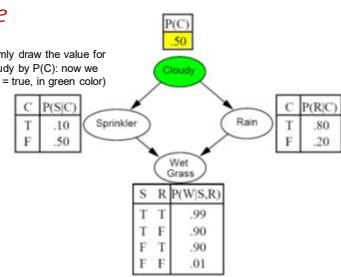
Example

First, we randomly draw the value for the variable Cloudy by $P(C)$.

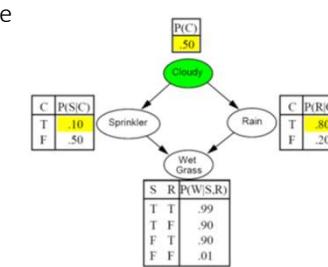


Example

First, we randomly draw the value for the variable Cloudy by $P(C)$: now we get c (means c = true, in green color)

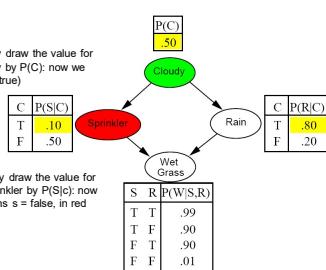


Example



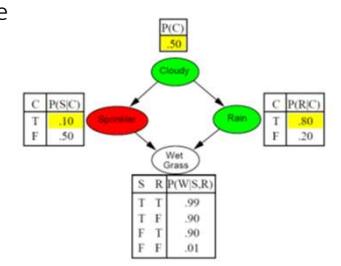
Example

First, we randomly draw the value for the variable Cloudy by $P(C)$: now we get c (means c = true)

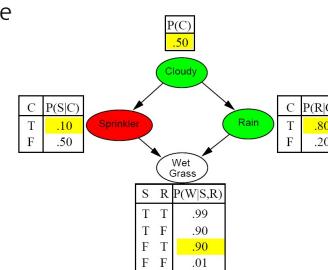


Example

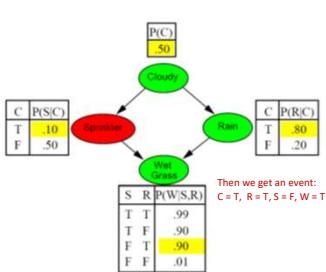
2nd, we randomly draw the value for the child node Sprinkler by $P(S|C)$: now we get s (means s = false, in red color)



Example



Example



Sampling from an empty network cont'd.

Probability that PRIORSAMPLE generates a particular event
 $S_{PS}(x_1, \dots, x_n) = \prod_{i=1}^n P(x_i | parents(X_i)) = P(x_1, \dots, x_n)$
i.e., the true prior probability

E.g., $S_{PS}(t, f, t) = 0.5 \times 0.9 \times 0.8 \times 0.9 = 0.324 = P(t, f, t, t)$

Let $N_{PS}(x_1, \dots, x_n)$ be the number of samples generated for event x_1, \dots, x_n

Then we have

$$\lim_{N \rightarrow \infty} P(x_1, \dots, x_n) = \lim_{N \rightarrow \infty} N_{PS}(x_1, \dots, x_n)/N \\ = S_{PS}(x_1, \dots, x_n) \\ = P(x_1, \dots, x_n)$$

That is, estimates derived from PRIORSAMPLE are consistent

Shorthand: $P(x_1, \dots, x_n) \approx P(x_1, \dots, x_n)$

Inference by stochastic simulation

基于随机模拟的推理

- Outline:
 - Sampling from an empty network
 - Rejection sampling: reject samples disagreeing with evidence
 - Likelihood weighting: use evidence to weight samples
 - Markov chain Monte Carlo (MCMC): sample from a stochastic process whose stationary distribution is the true posterior

Rejection sampling

$P(X|e)$ estimated from samples agreeing with e

```
function REJECTION-SAMPLING( $X, e, bn, N$ ) returns an estimate of  $P(X|e)$ 
local variables: N, a vector of counts over  $X$ , initially zero
for  $j = 1$  to  $N$  do
     $x \leftarrow \text{PRIORSAMPLE}(bn)$ 
    if  $x$  is consistent with  $e$  then
         $N[j] \leftarrow N[j]+1$  where  $j$  is the value of  $X$  in  $x$ 
    return  $\text{NORMALIZE}(N[1:N])$ 
```

E.g., estimate $P(\text{Rain}|\text{Sprinkler}=\text{true})$ using 100 samples
27 samples have $\text{Sprinkler}=\text{true}$

Of these, 8 have $\text{Rain}=\text{true}$ and 19 have $\text{Rain}=\text{false}$.

$P(\text{Rain}|\text{Sprinkler}=\text{true}) = \text{NORMALIZE}((8, 19)) = \langle 0.296, 0.704 \rangle$
Similar to a basic real-world empirical estimation procedure

Analysis of rejection sampling

$$\begin{aligned} P(X|e) &= \alpha N_{PS}(X, e) && (\text{algorithm defn.}) \\ &= N_{PS}(X, e)/N_{PS}(e) && (\text{normalized by } N_{PS}(e)) \\ &\approx P(X, e)/P(e) && (\text{property of PRIORSAMPLE}) \\ &= P(X|e) && (\text{defn. of conditional probability}) \end{aligned}$$

Hence rejection sampling returns **consistent posterior estimates**

- Problem: hopelessly expensive if $P(e)$ is small

- $P(e)$ drops off exponentially with number of evidence variables!

Inference by stochastic simulation

基于随机模拟的推理

- Outline:
 - Sampling from an empty network
 - Rejection sampling: reject samples disagreeing with evidence
 - Likelihood weighting: use evidence to weight samples
 - Markov chain Monte Carlo (MCMC): sample from a stochastic process whose stationary distribution is the true posterior

Likelihood weighting

- Idea: fix evidence variables, sample only non-evidence variables, and weight each sample by the likelihood it accords the evidence

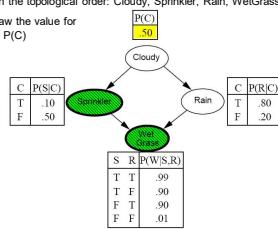
```
function LIKELIHOOD-WEIGHTING( $X, e, bn, N$ ) returns an estimate of  $P(X|e)$ 
inputs:  $X$ , the query variable
e: evidence variables, a vector of booleans
bn: a Bayesian network specifying joint distribution  $P(X_1, \dots, X_n)$ 
N: the total number of samples to be generated
local variables: W, a vector of weighted counts for each value of  $X$ , initially zero
for  $j = 1$  to  $N$  do
     $x, w \leftarrow \text{WEIGHTED-SAMPLE}(bn, e)$ 
     $W[x] \leftarrow W[x] + w$  where  $x$  is the value of  $X$  in  $x$ 
    return  $\text{NORMALIZE}(W)$ 

function WEIGHTED-SAMPLE( $bn, e$ ) returns an event and a weight
 $w \leftarrow 1$  is an event and a weight initialized from e
for each node  $x_i \in bn$  do
    if  $X_i$  is an evidence variable with value  $x_i = e$ 
        then  $w \leftarrow w \times P(X_i = e_i | \text{parents}(X_i))$ 
        else  $s_i \leftarrow$  a random sample from  $P(X_i | \text{parents}(X_i))$ 
    return  $x_i, w$ 
```

Likelihood weighting example

e.g. Query $P(\text{Rain} | \text{Sprinkler}=\text{true}, \text{WetGrass}=\text{true})$ given below bn.
Given the topological order: Cloudy, Sprinkler, Rain, WetGrass.

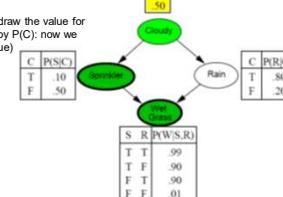
First, we randomly draw the value for the variable Cloudy by $P(C)$



$w = 1.0$

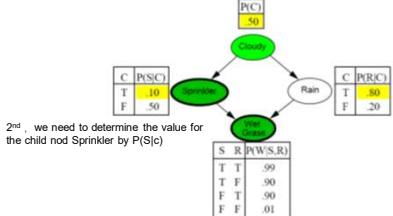
Likelihood weighting example

First, we randomly draw the value for the variable Cloudy by $P(C)$: now we get c (means c = true)



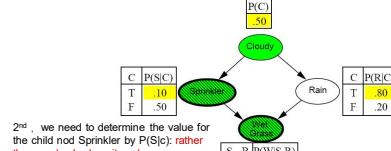
$w = 1.0$

Likelihood weighting example



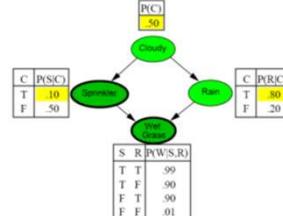
2nd, we need to determine the value for the child nod Sprinkler by $P(S|c)$

Likelihood weighting example



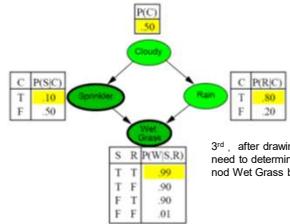
$w = 1.0 \times 0.1$

Likelihood weighting example



$w = 1.0 \times 0.1$

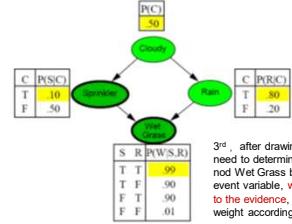
Likelihood weighting example



3rd, after drawing the value of Rain, we need to determine the value for the child node Wet Grass by $P(W|s,c)$

$$w = 1.0 \times 0.1 \times 0.99 = 0.099$$

Likelihood weighting example



3rd, after drawing the value of Rain, we need to determine the value for the child node Wet Grass by $P(W|s,c)$; also, as an event variable, we set its value according to the evidence, as "w" and give it a weight according to the CPT of the bn

$$w = 1.0 \times 0.1 \times 0.99 = 0.099$$

Likelihood weighting analysis

Sampling probability for WEIGHTEDSAMPLE is

$$S_{WS}(\mathbf{z}, \mathbf{e}) = \prod_{i=1}^n P(z_i | parents(Z_i))$$

Note: pays attention to evidence in **ancestors** only
=> somewhere "in between" prior and posterior distribution

Weight for a given sample \mathbf{z}, \mathbf{e} is
 $w(\mathbf{z}, \mathbf{e}) = \prod_{i=1}^n P(e_i | parents(E_i))$

Weighted sampling probability is
 $S_{WS}(\mathbf{z}, \mathbf{e})w(\mathbf{z}, \mathbf{e})$
= $\prod_{i=1}^n P(z_i | parents(Z_i)) \prod_{i=1}^n P(e_i | parents(E_i))$
= $P(\mathbf{z}, \mathbf{e})$ (by standard global semantics of network)

Hence likelihood weighting returns consistent estimates but performance still degrades with many evidence variables because a few samples have nearly all the total weight*



Inference by stochastic simulation

基于随机模拟的推理

- Outline:
 - Sampling from an empty network
 - Rejection sampling: reject samples disagreeing with evidence
 - Likelihood weighting: use evidence to weight samples
 - Markov chain Monte Carlo (MCMC):** sample from a stochastic process whose stationary distribution is the true posterior

Approximate inference using MCMC

Markov Chain Monte Carlo 马尔可夫链蒙特卡洛采样

- "State" of network = current assignment to all variables.
- Generate next state by sampling one variable given Markov blanket 马尔可夫毯(mb)
- Sample each variable in turn, keeping evidence fixed

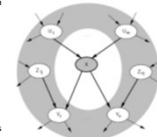
```

function MCMC(X,V,e,N) returns estimate of P(V|e)
local variables Z, x, c, w, N
Z, the evidence variables in e
x, the current state of the network, initially copied from e
initialise Z with random values for the variables in e
for i = 1 to N do
  for each Zi in Z do
    sample the value of Zi in x from P(Zi|mld(Zi))
    given the value of Zi, mld(Zi) is x
    Ni ← Ni + 1, where i is the value of Zi in x
    Ni ← Ni/N, where i is the value of Zi in x
return NORMALIZE(Ni)
  
```

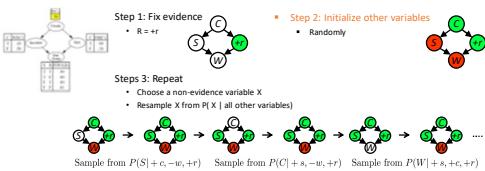
- Can also choose a variable to sample at random each time

Gibbs Sampling

- Procedure: keep track of a full instantiation x_1, x_2, \dots, x_n . Start with an arbitrary instantiation consistent with the evidence. Sample one variable at a time, conditioned on all the rest, but keep evidence fixed. Keep repeating this for a long time.
- Property: in the limit of repeating this infinitely many times the resulting samples come from the correct distribution (i.e. conditioned on evidence).
- Rationale: both upstream and downstream variables condition on evidence.*
- In contrast: likelihood weighting only conditions on upstream evidence, and hence weights obtained in likelihood weighting can sometimes be very small. Sum of weights over all samples is indicative of how many "effective" samples were obtained, so we want high weight.



Gibbs Sampling Example: $P(S | +r)$



Gibbs Sampling

- How is this better than sampling from the full joint?
 - In a Bayes' Net, sampling a variable given all the other variables (e.g. $P(R|S,C,W)$) is usually much easier than sampling from the full joint distribution
 - Only requires a join on the variable to be sampled (in this case, a join on R)
 - The resulting factor only depends on the variable's parents, its children, and its children's parents (this is often referred to as its Markov blanket)

Efficient Resampling of One Variable

- Sample from $P(S | +c, +r, -w)$

$$P(S | +c, +r, -w) = \frac{P(S, +c, +r, -w)}{P(+c, +r, -w)}$$

$$= \sum_i P(s_i, +c, +r, -w)$$

$$= \sum_i P(+c)P(+r)P(-w | S_i, +r)$$

$$= \sum_i P(+c)P(+r)P(+r | +c)P(-w | S_i, +r)$$

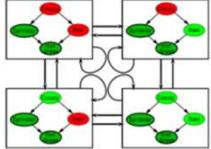
$$= P(+c)P(+r) + c \sum_i P(s_i | +c)P(-w | s_i, +r)$$

$$= P(S) + cP(-w | S, +r)$$

$$= \sum_i P(s_i | +c)P(-w | s_i, +r)$$
- Many things cancel out – only CPTs with S remain! (Markov blanket)
- More generally: only CPTs that have resampled variable need to be considered, and joined together

The Markov chain

With *Sprinkler = true, WetGrass = true*, there are four states:



- Wander about for a while, average what you see

MCMC example contd.

Estimate $P(Rain|Sprinkler = \text{true}, WetGrass = \text{true})$

Sample *Cloudy* or *Rain* given its Markov blanket, repeat.
Count number of times *Rain* is true and false in the samples.

E.g., visit 100 states
31 have *Rain = true*, 69 have *Rain = false*
 $P(Rain|Sprinkler = \text{true}, WetGrass = \text{true})$
= NORMALIZE((31, 69)) = (0.31, 0.69)

- Theorem: chain approaches **stationary distribution**:
long-run fraction of time spent in each state is exactly proportional to its posterior probability

Markov blanket sampling

Markov blanket of *Cloudy* is
Sprinkler and *Rain*
Markov blanket of *Rain* is
Cloudy, *Sprinkler*, and *WetGrass*

Probability given the Markov blanket is calculated as follows:

$$P(x_i^j | \text{mb}(X_i)) = P(x_i^j | \text{parents}(X_i) \cup \prod_{Z_j \in \text{Children}(X_i)} P(z_j | \text{parents}(Z_j)))$$

Easily implemented in message-passing parallel systems, brains

- Main computational problems:
- 1) Difficult to tell if convergence has been achieved
 - 2) Can be wasteful if Markov blanket is large:
 $P(X | \text{mb}(X_i))$ won't change much (law of large numbers)

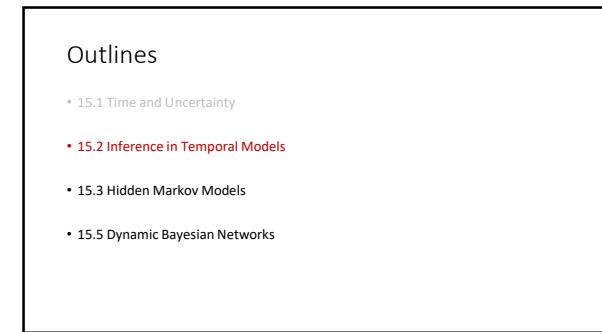
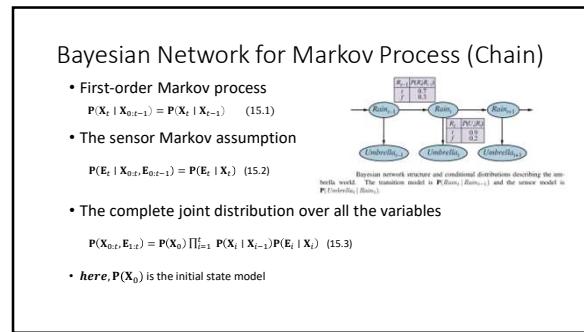
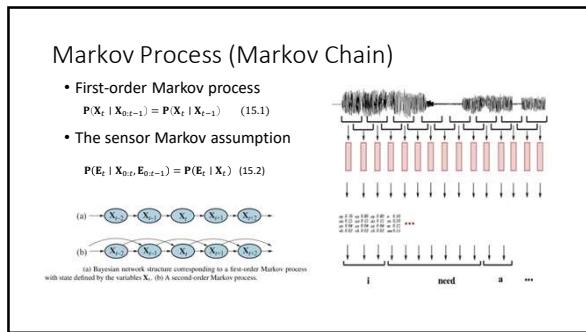
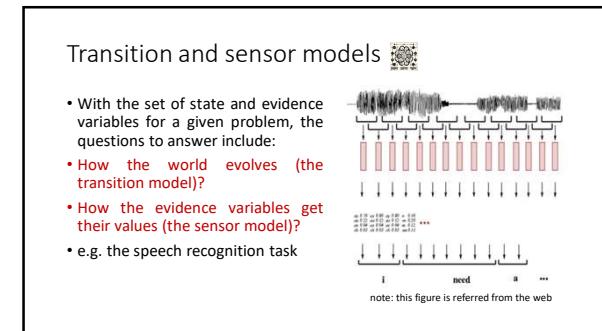
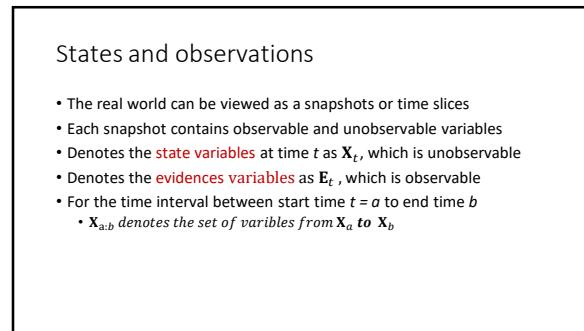
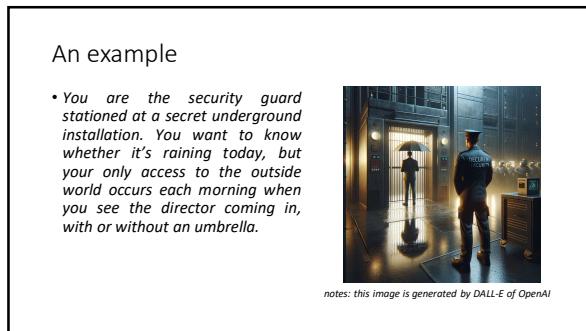
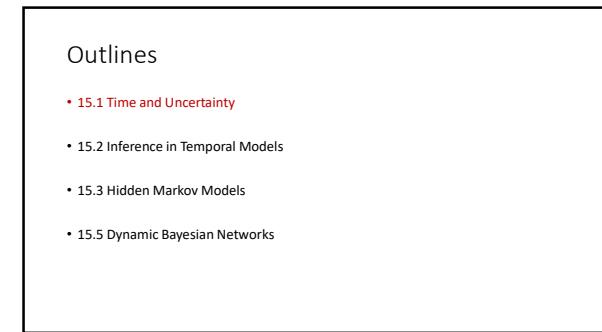
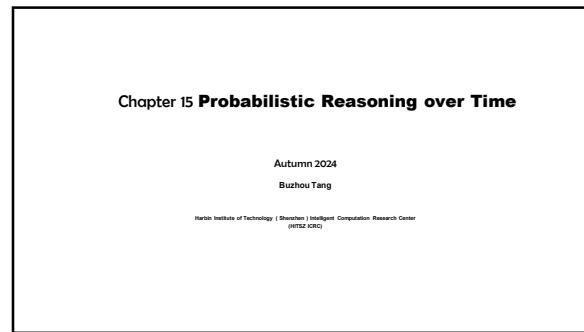


Summary

- Exact inference by variable elimination:
 - polytime on polytrees, NP-hard on general graphs
 - space = time, very sensitive to topology
- Approximate inference by LW, MCMC:
 - LW does poorly when there is lots of (downstream) evidence
 - LW, MCMC generally **insensitive** to topology
 - Convergence can be very slow with probabilities close to 1 or 0
 - Can handle arbitrary combinations of discrete and continuous variables

Assignments

- Ex 14.1 14.14(a, b)



Inference in Temporal Models

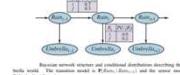
- Filtering (state estimation)**: computing the **belief state**, $P(X_t | e_{1:t})$, i.e., the posterior distribution over the most recent state, given all evidence to date
 - Prediction**: computing the posterior distribution $P(X_{t+k} | e_{1:t})$ over the future state, given all evidence to date, $k > 0$
 - Smoothing**: computing the posterior distribution over a past state, $P(X_k | e_{1:t})$, $0 \leq k < t$
 - Most likely explanation**: given a sequence of observation, find the sequence of states that is most likely to have generated those observations (e.g., speech recognition)
- $$\text{argmax}_{x_{1:t}} P(X_{1:t} | e_{1:t})$$
- Learning**: learn the transition and sensor models from observations.

Filtering (state estimation)

- recursive estimation**: given the result of filtering up to time t , compute the result for $t+1$ from the new evidence e_{t+1}

$$P(X_{t+1} | e_{1:t+1}) = f(e_{t+1}, P(X_t | e_{1:t}))$$

$$\begin{aligned} P(X_{t+1} | e_{1:t+1}) &= P(X_{t+1} | e_{1:t}, e_{t+1}) \quad (\text{证据分解}) \\ &= \alpha P(e_{t+1} | X_{t+1}, e_{1:t}) P(X_{t+1} | e_{1:t}) \quad (\text{使用贝叶斯规则}) \\ &= \alpha P(e_{t+1} | X_{t+1}) P(X_{t+1} | e_{1:t}) \quad (\text{根据传感器马尔可夫假设}) \end{aligned}$$



Filtering (state estimation)

- recursive estimation**: given the result of filtering up to time t , compute the result for $t+1$ from the new evidence e_{t+1}

$$P(X_{t+1} | e_{1:t+1}) = f(e_{t+1}, P(X_t | e_{1:t}))$$

$$\begin{aligned} P(X_{t+1} | e_{1:t+1}) &= P(X_{t+1} | e_{1:t}, e_{t+1}) \quad (\text{证据分解}) \\ &= \alpha P(e_{t+1} | X_{t+1}, e_{1:t}) P(X_{t+1} | e_{1:t}) \quad (\text{使用贝叶斯规则}) \\ &= \alpha P(e_{t+1} | X_{t+1}) P(X_{t+1} | e_{1:t}) \quad (\text{根据传感器马尔可夫假设}) \end{aligned}$$

one-step prediction for X_{t+1} by condition on X_t

$$\begin{aligned} P(X_{t+1} | e_{1:t+1}) &= \alpha P(e_{t+1} | X_{t+1}) \sum_{x_t} P(X_{t+1} | x_t, e_{1:t}) P(x_t | e_{1:t}) \\ &= \alpha P(e_{t+1} | X_{t+1}) \sum_{x_t} P(X_{t+1} | x_t) P(x_t | e_{1:t}) \quad (\text{马尔可夫假设}) \end{aligned}$$

denotes as the forward message propagation as: $f_{1:t+1} = \alpha \text{ FORWARD}(f_{1:t}, e_{t+1})$

Filtering (state estimation): example

- the filtering process for two steps in the basic umbrella example: for both Day 1 and Day 2, the security guard see the umbrella appear, i.e., evidences $U_1 = \text{true}$, and $U_2 = \text{true}$

$$P(R_0) = (0.5, 0.5) \quad \text{since there is no observation input}$$

on Day 1:



$$\begin{aligned} P(R_1) &= \sum_{r_0} P(R_1 | r_0) P(r_0) = (0.7, 0.3) \times 0.5 + (0.3, 0.7) \times 0.5 = (0.5, 0.5) \\ P(R_1 | u_1) &= \alpha P(u_1 | R_1) P(R_1) = \alpha(0.9, 0.2)(0.5, 0.5) \\ &= \alpha(0.45, 0.1) \approx (0.818, 0.182) \end{aligned}$$

on Day 2:

$$P(R_2 | u_1, u_2) = \sum_{r_1} P(R_2 | r_1) P(r_1 | u_1) = (0.7, 0.3) \times 0.818 + (0.3, 0.7) \times 0.182 \approx (0.627, 0.373)$$

$$P(R_2 | u_1, u_2) = \alpha P(u_2 | R_2) P(R_2 | u_1) = \alpha(0.9, 0.2)(0.627, 0.373) = \alpha(0.565, 0.075) \approx (0.883, 0.117)$$

Prediction

- The further steps of **prediction** only involve the transition model since there is no new evidence input after the step t .

$$P(X_{t+k+1} | e_{1:t}) = \sum_{x_{t+k}} P(X_{t+k+1} | x_{t+k}) P(x_{t+k} | e_{1:t}) \quad (15.6)$$

- Stationary distribution of the Markov process: without new evidence, the predicted state distribution finally converges to a fix point, and after which, it remains constant for all time.

Smoothing

$$\begin{aligned} P(X_k | e_{1:t}) &= P(X_k | e_{1:k}, e_{k+1:t}) \\ &= \alpha P(X_k | e_{1:k}) P(e_{k+1:t} | X_k, e_{1:k}) \quad (\text{使用贝叶斯规则}) \\ &= \alpha P(X_k | e_{1:k}) P(e_{k+1:t} | X_k) \quad (\text{使用条件独立性}) \\ &= \alpha f_{1:k} \times b_{k+1:t} \quad (15.8) \end{aligned}$$

here $b_{k+1:t} = P(e_{k+1:t} | X_k)$ is define as "backward" message

Smoothing

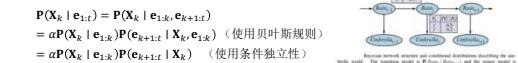
$$\begin{aligned} P(e_{k+1:t} | X_k) &= \sum_{x_{k+1}} P(e_{k+1:t} | X_k, x_{k+1}) P(x_{k+1} | X_k) \quad (\text{将 } X_{k+1} \text{ 条件化}) \\ &= \alpha \sum_{x_{k+1}} P(e_{k+1:t} | X_k, x_{k+1}) P(x_{k+1} | X_k) \quad (\text{根据条件独立性}) \\ &= \alpha \sum_{x_{k+1}} P(e_{k+1:t} | e_{k+2:t}, x_{k+1}) P(x_{k+1} | X_k) \quad (\text{分离证据变量}) \\ &= \alpha \sum_{x_{k+1}} P(e_{k+1:t} | x_{k+1}) P(e_{k+2:t} | x_{k+1}) P(x_{k+1} | X_k) \quad (15.9) \end{aligned}$$

BACKWARD: compare the 2nd term in the summation and the left hand of the equation, we can let:
 $b_{k+1:t} = \text{BACKWARD}(b_{k+2:t}, e_{k+1:t})$, and $b_{1:t+1} = P(e_{1:t+1} | X_t) = P(|X_t|=1, 1 \text{ is a vector of 1s.})$

Smoothing : Forward-Backward Algorithm

```
function FORWARD-BACKWARD(ev, prior) returns a vector of probability distributions
inputs: ev, a vector of evidence values for steps 1, ..., t
prior, the prior distribution on the initial state, P(X_0)
local variables: fv, a vector of forward messages for steps 0, ..., t
b, a representation of the backward message, initially all 1s
sv, a vector of smoothed estimates for steps 1, ..., t

fv[0] = prior
for i = 1 to t do
  fv[i] = FORWARD(fv[i - 1], ev[i])
for i = t down to 1 do
  sv[i] ← NORMALIZE(fv[i] × b)
  b = BACKWARD(b, ev[i])
return sv
```



Smoothing : example

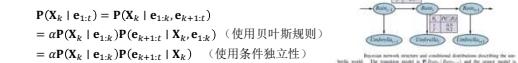
- Computing the smoothed estimate for the probability of rain at time $k=1$, given the umbrella observations on days 1 and 2:

$$P(R_1 | u_1, u_2) = \alpha P(R_1 | u_1) P(u_2 | R_1) \quad (15.10)$$

$$\begin{aligned} P(u_2 | R_1) &= \sum_{r_1} P(u_2 | r_1) P(r_1 | R_1) \\ &= (0.9 \times 1 \times 0.7, 0.3) + (0.2 \times 1 \times 0.3, 0.7) = (0.69, 0.41) \end{aligned}$$

$$P(R_1 | u_1, u_2) = \alpha(0.818, 0.182) \times (0.69, 0.41) \approx (0.883, 0.117)$$

note: $b_{k+1:t} = P(e_{k+1:t} | X_k) = P(|X_k|=1)$
 $P(R_1 | u_1)$ had been computed in the example of filtering



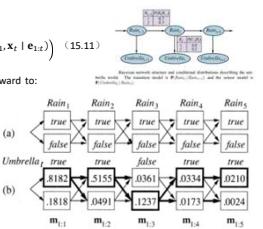
Finding the most likely sequence: Viterbi algorithm

Suppose that [true, true, false, true, true] is the umbrella sequence, what is the most likely weather sequence to explain it?

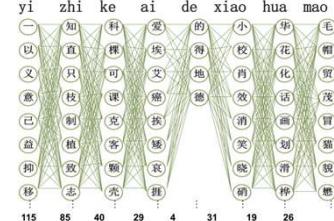
$$\max_{\mathbf{x}_1 \dots \mathbf{x}_t} P(\mathbf{x}_1, \dots, \mathbf{x}_t, \mathbf{x}_{t+1} | \mathbf{e}_{1:t+1}) \\ = \alpha P(\mathbf{e}_{t+1} | \mathbf{x}_{t+1}) \max_{\mathbf{x}_t} \left(P(\mathbf{x}_{t+1} | \mathbf{x}_t) \max_{\mathbf{x}_1 \dots \mathbf{x}_{t-1}} P(\mathbf{x}_1, \dots, \mathbf{x}_{t-1}, \mathbf{x}_t | \mathbf{e}_{1:t}) \right) \quad (15.11)$$

Here we replace the forward message $f_{1:t}$ in forward-backward to:

$$\mathbf{m}_{1:t} = \max_{\mathbf{x}_1 \dots \mathbf{x}_{t-1}} P(\mathbf{x}_1, \dots, \mathbf{x}_{t-1}, \mathbf{x}_t | \mathbf{e}_{1:t})$$



Finding the most likely sequence: Viterbi algorithm



Outlines

- 15.1 Time and Uncertainty
- 15.2 Inference in Temporal Models
- 15.3 Hidden Markov Models
- 15.5 Dynamic Bayesian Networks

Hidden Markov Models (HMM)

- HMM: an HMM is a temporal probabilistic model in which the **state** of the process is described by a **single discrete** random variable.
- For multiple state variables, combining them into "megavariable", each value of this megavariable is the full assignment of the variables.
- The inference of this model can be simplified to matrix algorithms as follows:

Transition model : $\mathbf{T}_{ij} = P(X_t = j | X_{t-1} = i)$ Sensor model:

$$\mathbf{T} = P(X_t | X_{t-1}) = \begin{pmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{pmatrix} \quad \mathbf{O}_i = \begin{pmatrix} 0.9 & 0 \\ 0 & 0.1 \end{pmatrix}, \mathbf{O}_3 = \begin{pmatrix} 0.1 & 0 \\ 0 & 0.8 \end{pmatrix}$$

Forward message: $\mathbf{f}_{1:t+1} = \alpha \mathbf{O}_{t+1} \mathbf{T}^T \mathbf{f}_{1:t}$ Backward message: $\mathbf{b}_{k+1:t} = \mathbf{T} \mathbf{O}_k \mathbf{b}_{k+2:t}$.

More efficient smoothing: (vs. equation 15.8)

Propagating the forward message in a backward way: $\mathbf{f}_{1:t} = \alpha' (\mathbf{T}^T)^{-1} \mathbf{O}_{t+1}^{-1} \mathbf{f}_{1:t+1}$

Application of HMM: Speech recognition

- Finding the words given the acoustics is an HMM inference problem
- Which state sequence $\mathbf{x}_{1:T}$ is most likely given the evidence $\mathbf{e}_{1:T}$?

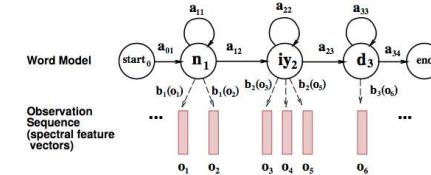
$$\mathbf{x}_{1:T}^* = \arg \max_{\mathbf{x}_{1:T}} P(x_{1:T} | e_{1:T}) = \arg \max_{\mathbf{x}_{1:T}} P(x_{1:T}, e_{1:T})$$

- From the sequence \mathbf{x} , we can simply read off the words



[This page is referred from Dan Klein and Pieter Abbeel for CS188 intro to AI at UC Berkeley. All CS188 materials are available at <http://ai.berkeley.edu>.]

HMM for Speech Recognition:



[This page is referred from Dan Klein and Pieter Abbeel for CS188 intro to AI at UC Berkeley. All CS188 materials are available at <http://ai.berkeley.edu>.]

Outlines

- 15.1 Time and Uncertainty
- 15.2 Inference in Temporal Models
- 15.3 Hidden Markov Models
- 15.5 Dynamic Bayesian Networks

Dynamic Bayesian Networks

- a dynamic Bayesian network (DBN), is a Bayesian network that represents a temporal probability model.
- each slice of a DBN can have any number of state variables X_t and evidence variables E_t .
- assume variables and their links are exactly replicated from slice to slice, each variable can have parents only in its own slice or the immediately preceding slice (a first-order Markov process).
- every discrete-variable DBN can be represented as an HMM, and vice versa.

• Then, what's the difference between DBN and HMM?



Video of Demo Pacman Sonar Ghost DBN Model

Pacman spends his life running from ghosts, but things were not always so. Legend has it that many years ago, Pacman's great grandfather Grandpac learned to hunt ghosts for sport. However, he was blinded by his power and could only track ghosts by their banging and clanging.



[This page is referred from Dan Klein and Pieter Abbeel for CS188 intro to AI at UC Berkeley. All CS188 materials are available at <http://ai.berkeley.edu>.]

Constructing DBNs

By specifying three kinds of information:

- the prior distribution over the state variables, $P(X_0)$
- the transition model $P(X_{t+1} | X_t)$
- the sensor model, $P(E_t | X_t)$

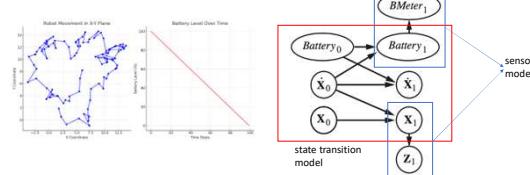
Example: monitoring a battery-powered robot moving in the X-Y plane.



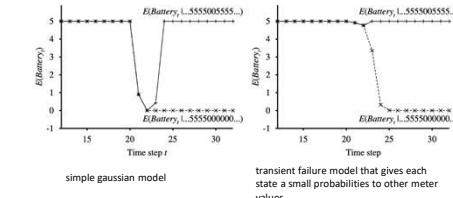
DBN Example

monitoring a battery-powered robot moving in the X-Y plane.

- Transient failure model: e.g., In idea case, the CPT $P(BMeter_1 | Battery_1)$ should have 1.0 along the diagonal and 0.0 elsewhere, which may cause strange behavior for the transient failure of the meter.



Transient failure model: example



Exact Inference in DBNs: unrolling (展开)

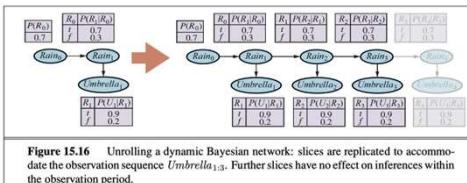
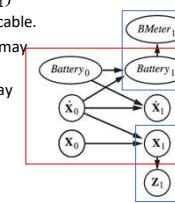


Figure 15.16 Unrolling a dynamic Bayesian network: slices are replicated to accommodate the observation sequence $Umbrella_{1:3}$. Further slices have no effect on inferences within the observation period.

Approximate Inference in DBNs

- computing the probabilities: $P(X_{t+1} | e_{1:t+1})$
- the likelihood weighting and MCMC is applicable.
- The challenge is: unrolling a complex DBNs may greatly increase space and time complexity.
- Particle filtering: a recursive and efficient way



Particle filtering

- First, a population of N initial-state samples is created by sampling from the prior distribution $P(X_0)$;
- Repeat for each time step:
 - each sample is propagated forward by sampling the next state value x_{t+1} given the current value x_t for the sample, based on the transition model $P(X_{t+1} | x_t)$
 - each sample is weighted by the likelihood it assigns to the new evidence: $P(e_{t+1} | x_{t+1})$
 - the population is resampled to generate a new population of N samples. Each new sample is selected from the current population; the probability that a particular sample is selected is proportional to its weight. The new samples are unweighted.
- The major unique of the algorithm:
 - use the samples themselves as an approximate representation of the current state distribution
 - focus the set of samples on the high-probability regions of the state space.

Particle filtering: Algorithm

```
function PARTICLE-FILTERING( $e$ ,  $N$ ,  $dbn$ ) returns a set of samples for the next time step
  inputs:  $e$ , the new incoming evidence
   $N$ , the number of samples to be maintained
   $dbn$ , a DBN defined by  $P(X_0)$ ,  $P(X_t | X_0)$ , and  $P(E_t | X_t)$ 
  persistent:  $S$ , a vector of samples of size  $N$ , initially generated from  $P(X_0)$ 
  local variables:  $W$ , a vector of weights of size  $N$ 

  for  $i = 1$  to  $N$  do
     $S[i] \leftarrow$  sample from  $P(X_1 | X_0 = S[i])$  // step 1
     $W[i] \leftarrow P(e | X_1 = S[i])$  // step 2
     $S \leftarrow$  WEIGHTED-SAMPLE-WITH-REPLACEMENT( $N$ ,  $S$ ,  $W$ ) // step 3
  return  $S$ 
```

Particle filtering: example

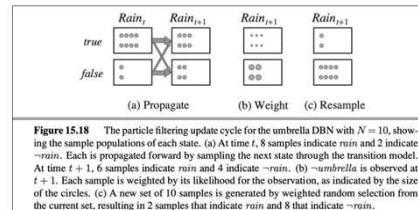


Figure 15.18 The particle filtering update cycle for the umbrella DBN with $N = 10$, showing the sample populations of each state. (a) At time t , 8 samples indicate rain and 2 indicate $\neg rain$. Each is propagated forward by sampling the next state through the transition model. A new set of 10 samples is generated. (b) At time $t + 1$, 8 samples indicate rain and 2 indicate $\neg rain$ observed at $t + 1$. Each sample is weighted by its likelihood for the observation, as indicated by the size of the circles. (c) A new set of 10 samples is generated by weighted random selection from the current set, resulting in 2 samples that indicate rain and 8 that indicate $\neg rain$.

Summary

- The changing state of the world is handled by using a set of random variables to represent the state at each point in time.
- Representations can be designed to satisfy the **Markov property**, so that future is **independent of the past given the present**.
- A temporal probability model can be thought of as containing a **transition model** describing the state evolution and a **sensor model** describing the observation process.
- Inference tasks in temporal models include **filtering**, **prediction**, **smoothing** and **most likely explanation**.
- HMM, dynamic Bayesian networks and particle filtering algorithm were studied. In practice, **particle filtering** algorithm seems to be an effective approximation algorithm.

Artificial Intelligence

第2部分 推理

1. 符号推理
2. 不确定性推理
3. 因果推理

Buzhou Tang

Email: tangbz@hit.edu.cn
<http://irc.hit.edu.cn>

Harbin Institute of Technology (Shenzhen)
 Intelligent Computation Research Center
 (ICRC, HITZ)

『CAUSAL INFERENCE IN STATISTICS: A Primer』

2024
 Instructor: Buzhou Tang

Harbin Institute of Technology (Shenzhen) Intelligent Computation Research Center
 (HITZ/CRC)

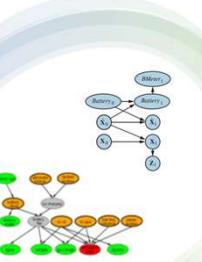
Chapter 1 Introduction of Causal Inference

Autumn 2024
 Instructor: Buzhou Tang

Harbin Institute of Technology (Shenzhen) Intelligent Computation Research Center
 (HITZ/CRC)

Question

Since we have the powerful tool of Probability, the Bayesian Network, and Dynamic Bayesian Network etc., why we still need to take the causal inference?



Part 1 Simpson's Paradox

Simpson's Paradox: Example 1

Edward Simpson (born 1922)
 Table 1.1 Results of a study into a new drug, with gender being taken into account.

	Drug	No drug
Men	81 out of 87 recovered (93%) ①	234 out of 270 recovered (87%) ②
Women	192 out of 263 recovered (73%) ③	55 out of 80 recovered (69%) ④
Combined data	273 out of 350 recovered (78%) A	289 out of 350 recovered (83%) B

Subpopulation data: Men ① (drug) > ② (no drug)
 Women ③ (drug) > ④ (no drug)

Paradox: Population data: A (drug) < B (no drug)

□ The data says if the gender is known, we should prescribe treat; Otherwise we should not prescribe treat. **This is obviously ridiculous!**

□ A statistical association that holds for an entire population is **reversed** in every subpopulation

Simpson's Paradox: Example 2

The paradox refers to the existence of data in which a statistical association that holds for an **entire population** is reversed in **every subpopulation**.

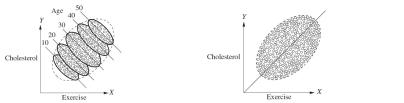


Figure 1.1 Results of the exercise–cholesterol study, segregated by age
Figure 1.2 Results of the exercise–cholesterol study, unsegregated.

Subpopulation data: Exercise ↑ Cholesterol ↓
Paradox: Population data: Exercise ↑ Cholesterol ↑

Simpson's Paradox: Why these happened?

For example 1:

```

graph TD
    sex[sex] --> treat[treat]
    sex --> recovered[Recovered]
    treat --> recovered
  
```

- Sex is a common cause of both treat and recovered;
- Estrogen(雌激素) has a negative effect on recovery, so women are less likely to recover than men, regardless of the drug;
- women are significantly more likely to take the drug than men are.

In this case, you should use **subpopulation** data.

Simpson's Paradox: Why these happened?

For example 2:

```

graph TD
    age[age] --> exercise[Exercise]
    age --> cholesterol[Cholesterol]
    exercise --> cholesterol
  
```

- Age is a common cause of both exercise and cholesterol
- Older people, who are more likely to exercise are also more likely to have high cholesterol regardless of exercise.

In this case, you should use **subpopulation** data.
 Then, in which case should we use the whole data ?

Simpson's Paradox: Example 3

Table 1.2 Results of a study into a new drug, with posttreatment blood pressure taken into account.

	No drug	Drug
Low BP	81 out of 87 recovered (93%)	234 out of 270 recovered (87%)
High BP	192 out of 263 recovered (73%)	55 out of 80 recovered (69%)
Combined data	273 out of 350 recovered (78%)	289 out of 350 recovered (83%)

Subpopulation data: no drug > drug. Blood pressure is fixed and we would not see drug effect, we would only see drug's toxic effect.

Paradox:

Population data: no drug < drug. Drug improve recovery by lowering blood pressure.

Simpson's Paradox: Example 3



In this case, you should use **population** data

Simpson's Paradox: conclusion & definition

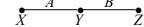
- ◻ There is **no statistical method** that can determine the causal story from the data alone. Consequently, there is no statistical method that can aid in our decision.
- ◻ We primarily focus on **causal inference**, used to express and explain causal hypotheses.
- ◻ The definition of **causation** is simply given as: **a variable X is a cause of a variable Y if Y in any way relies on X for its value**

Part 2 Structural Casual Models

Graphs: a quick glimpse

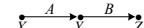
A mathematical **graph** is a collection of vertices (or nodes) and edges. The nodes in a graph are connected (or not) by the edges.

Undirected Graph (X,Y and Z are nodes, A and B are edges)



- Two nodes are adjacent if there is an edge between them. (e.g. X and Y)
- A **path** between two nodes X and Y is a sequence of nodes beginning with X and ending with Y.

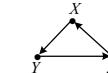
Directed Graph



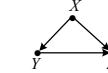
- X is the **parent** of Y, conversely, Y is the **child** of X.
- X is the **ancestor** of both Y and Z, and both Y and Z are **descendants** of X

Graphs: a quick glimpse

Cyclic: A directed path exists from a node to itself.



Acyclic: A directed graph with no cycles



Structure Causal Models

Structural Causal Model (SCM): a way of describing the relevant features of the world and how they interact with each other

SCM consists two sets of variables U and V , and a set of functions f

$$f = \{f_X: W_X \rightarrow X | X \in V\} \quad W_X \subseteq (U \cup V) - \{X\}$$

- variables in U are called **exogenous variables** (外生变量), they have **no ancestors**.
- variables in V are called **endogenous variables** (内生变量), every endogenous variable in a model is a descendant of **at least one** exogenous variable

SCM and Graph Models

- Every SCM is associated with a graphical model.
- X is a **cause** of Y if it is a direct cause of Y, or of any cause of Y
- U are called **exogenous variables**, V are **endogenous variables**
- We deal primarily with SCMs for which the graphical models are **directed acyclic graphs (DAGs)**.

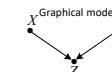
Structure Causal Models

SCM 1.5.1 (Salary Based on Education and Experience)

$$U = \{X, Y\}, V = \{Z\}, F = \{f_Z\}$$

$$f_Z: Z = 2X + 3Y$$

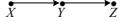
Z: salary that an employer pays to an individual, X: years of schooling, Y: years in the profession.



- The Graphical model usually owns less information, is a qualitative model with more intuitive for causality.

Structure Causal Models

Product Decomposition



We can write directly from this SCM.

$$P(X = x, Y = y, Z = z) = P(X = x)P(Y = y|X = x)P(Z = z|Y = y)$$

It's not easy to get:

$$P(\text{clouds}, \text{no-rain}, \text{dry pavement}, \text{slippery pavement}) = 0.23^*$$

But we can decomposition this into

$$P(\text{clouds})P(\text{no rain}|\text{clouds})P(\text{dry pavement}|\text{no rain})P(\text{slippery pavement}|\text{dry pavement})$$

And we can know:

$$P(\text{clouds}) = 0.5, P(\text{no rain}|\text{clouds}) = 0.75,$$

$$P(\text{dry pavement}|\text{no rain}) = 0.9,$$

$$P(\text{slippery pavement}|\text{dry pavement}) = 0.05$$

So we get result.

Chapter 2 Graphical Models and Their Application

Autumn 2024

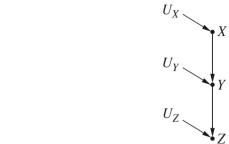
Instructor: Buzhou Tang

Harbin Institute of Technology (Shenzhen) Intelligent Computation Research Center (HITZIC ICRC)

Graphical Models : Chains 链结构

e.g. SCM 2.2.1 (School Funding: X, SAT Scores: Y, and College Acceptance: Z)

Graphical Model : Chains



1. **Z and Y are likely dependent.**

For some $x, z, P(Z = z|X = x, Y = y) \neq P(Z = z)$

2. **Y and X are likely dependent.**

For some $y, x, P(Y = y|X = x) \neq P(Y = y)$

3. **Z and X are likely dependent.**

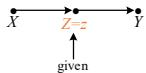
For some $z, x, P(Z = z|X = x) \neq P(Z = z)$

4. **Z and X are independent, conditional on Y.**

For all $x, y, z, P(Z = z|X = x, Y = y) = P(Z = z|Y = y)$

Graphical Models : Chains

Rule 1 (Conditional Independence in Chains) Two variables, X and Y, are conditionally independent given Z, if there is **only one unidirectional path** between X and Y, and Z is any set of variables that intercepts that path.

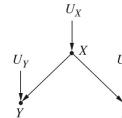


Graphical Models: Forks 分叉结构

e.g. SCM 2.2.5 (Temperature: X, Ice Cream Sales: Y, and Crime: Z)

$$V = \{X, Y, Z\}, U = \{U_X, U_Y, U_Z\}, F = f_X, f_Y, f_Z$$

Graphical Model: Forks



1. **X and Y are likely dependent.**

For some $x, y, P(X = x|Y = y) \neq P(X = x)$

2. **X and Z are likely dependent.**

For some $x, z, P(X = x|Z = z) \neq P(X = x)$

3. **Z and Y are likely dependent.**

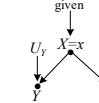
For some $z, y, P(Z = z|Y = y) \neq P(Z = z)$

4. **Y and Z are independent, conditional on X.**

For all $x, y, z, P(Y = y|Z = z, X = x) = P(Y = y|X = x)$

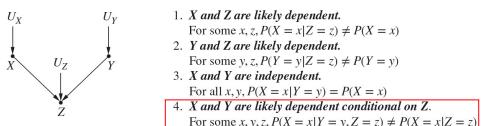
Graphical Models: Forks

Rule 2 (Conditional Independence in Forks) If a variable X is a common cause of variables Y and Z, and there is **only one path** between Y and Z, then Y and Z are independent conditional on X.



Graphical Models: Colliders 对撞结构

The variable Z is dependent on both the variable X and variable Y, on the condition of the independence of variables in set U



see the example SCM 1.5.1. Salary(Z) Based on Education(X) and Experience(Y)

Graphical Models: Colliders

Example: Let the outcomes of the two coins be denoted X and Y, respectively, and let Z stand for the state of the bell, with $Z = 1$ representing ringing, and $Z = 0$ representing silence.

Table 2.1 Probability distribution for two flips of a fair coin, with X representing flip one, Y representing flip two, and Z representing a bell that rings if either flip results in heads

X	Y	Z	$P(X, Y, Z)$
Heads	Heads	1	0.25
Heads	Tails	1	0.25
Tails	Heads	1	0.25
Tails	Tails	0	0.25

Graphical Models: Colliders

Example: Let the outcomes of the two coins be denoted X and Y, respectively, and let Z stand for the state of the bell, with $Z = 1$ representing ringing, and $Z = 0$ representing silence.

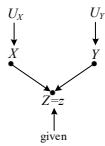
Table 2.2 Conditional probability distributions for the distribution in Table 2.1. Top: Distribution conditional on $Z = 1$. Bottom: Distribution conditional on $Z = 0$

X	Y	$P(X, Y Z = 1)$
Heads	Heads	0.333
Heads	Tails	0.333
Tails	Heads	0.333
Tails	Tails	0

X	Y	$P(X, Y Z = 0)$
Heads	Heads	0
Heads	Tails	0
Tails	Heads	0
Tails	Tails	1

Graphical Models: Colliders

Rule 3 (Conditional Independence in Colliders) If a variable Z is the collision node between two variables X and Y, and there is **only one path between X and Y**, then X and Y are unconditionally independent, but are dependent conditional on Z and any descendants of Z.



Graphical Models: d-separation

Question: for any type of real world problems, is there existing some rules to predict the properties of data generated by the corresponding complex model?

d-separation: a tool of describing whether two variables are independent in the model.

Graphical Models: d-separation

Concepts:

d-separation (the d stands for "directional") allows us to determine, for any pair of nodes, whether the nodes are **d-connected**, meaning there exists a connecting path between them, or **d-separated**, meaning there exists no such path.

two types of d-separation:

Unconditional d-separation (not conditioning on any variable)
• only colliders can block a path.

Conditional d-separation

- A collider that is not conditioned on Z (i.e., not in Z), and that has **no descendants in Z**.
- A chain or fork whose middle node is **in Z**.

Graphical Models: d-separation

Definition 2.4.1 (d-separation) A path p is blocked (阻断) by a set of nodes Z if and only if:

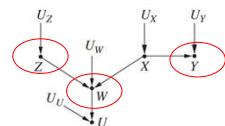
- p contains a chain of nodes $A \rightarrow B \rightarrow C$ or a fork $A \leftarrow B \rightarrow C$ such that the middle node **B is in Z** (i.e., B is conditioned on), or
- p contains a collider $A \rightarrow B \leftarrow C$ such that the collision **node B is not in Z**, and **no descendant of B is in Z**.

Graphical Models: d-separation

Example 1: Let's look at the relationship between Z and Y.

Using an empty conditioning set, they are **d-separated**.

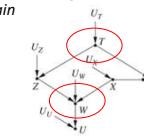
But suppose we condition on W. d-separation tells us that Z and Y are **d-connected**, conditional on W.



Graphical Models: d-separation

Example 2: We add another path between Z and Y. Z and Y are now **unconditionally dependent**.

If we condition on T, however, that path is blocked, and Z and Y become **independent again**. Conditioning on {T, W}, on the other hand, makes them **d-connected again**



Homework

- 思考题2.4.1 (中文版)

Artificial Intelligence

第2部分 推理

1. 符号推理
2. 不确定性推理
- 3. 因果推理**

Buzhou Tang

Email: tangbz@szit.edu.cn
<http://icrc.hitsz.edu.cn>

Harbin Institute of Technology (Shenzhen)
Intelligent Computation Research Center
(ICRC, HITSZ)

《CAUSAL INFERENCE IN STATISTICS: A Primer》

Autumn 2024

Instructor: Buzhou Tang

Harbin Institute of Technology (Shenzhen) Intelligent Computation Research Center (HITSZ ICRC)

Chapter 3 The Effects of Interventions
干预的效果

Autumn 2024

Instructor: Buzhou Tang

Harbin Institute of Technology (Shenzhen) Intelligent Computation Research Center (HITSZ ICRC)

Outlines

- Interventions 干预
- The Adjustment Formula 纠正公式
- The Backdoor Criterion & The Front-Door Criterion 后门准则 & 前门准则

Interventions (干预)

Golden standard of statistics:
the randomized controlled experiment

Is it practical on all conditions?
weather, participants...

observational studies(观察性研究)

Problem: hard to untangle the causal from the merely correlative

Eg1: heavy text messaging (过度发短信) in teens and shallowness (肤浅)

Interventions (干预)

Intervening a variable: fixing value of the variable, and thus changing the model, which is an important way of causal analysis on observation data.

Eg: ice cream sales example for intervening

Figure 3.1 A graphical model representing the relationship between temperature (Z), ice cream sales (X), and crime rates (Y)

Figure 3.2 A graphical model representing an intervention on the model in Figure 3.1 that lowers ice cream sales

Interventions (干预)

Difference between intervening and conditioning on a variable:
Change the model?

Eg: ice cream sales example

Figure 3.1 A graphical model representing the relationship between temperature (Z), ice cream sales (X), and crime rates (Y)

Figure 3.2 A graphical model representing an intervention on the model in Figure 3.1 that lowers ice cream sales

Interventions (干预)

Distinguish between cases where a variable X takes a value x naturally and cases where we fix X = x (intervening), by denoting the latter

$\text{do}(X = x)$.

- ◻ $P(Y = y | X = x)$ is the probability that $Y = y$ conditional on finding $X = x$, w
- ◻ $P(Y = y | \text{do}(X = x))$ is the probability that $Y = y$ when we **intervene to make $X = x$**
- ◆ $P(Y = y | \text{do}(X = x), Z = z)$ denotes the conditional probability of $Y = y$, given $Z = z$, in the distribution created by the intervention $\text{do}(X = x)$.

Interventions (干预)

Intervening assumptions:

- The graph constitutes a valid representation of reality.
- The intervention has no “side effects”, which means assigning the value x for the variable X for an individual does not alter subsequent variables in a direct way.

Eg: Being “assigned” a drug might have a different effect on recovery than being forced to take the drug against one’s religious objections.

➢ When side effects are present, they need to be specified explicitly in the model

Outlines

- Interventions 干预
- The Adjustment Formula 纠正公式
- The Backdoor Criterion&The Front-Door Criterion 后门准则&前门准则

The Adjustment Formula (矫正公式)

Simpson's Paradox

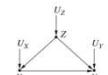
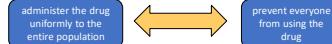


Figure 3.3 A graphical model representing the effects of a new drug.
Z representing gender, X standing for drug usage, and Y standing for recovery.

a hypothetical intervention for the input variable of possible causal relation



$$\text{Difference } P(Y = 1 | \text{do}(X = 1)) - P(Y = 1 | \text{do}(X = 0)) \quad (3.1)$$

eq (3.1) is called "causal effect difference" or "average causal effect" (ACE, 平均因果效应).

The Adjustment Formula (矫正公式)

Compute the magnitude of the causal effect with the aid of the graph.

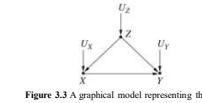


Figure 3.3 A graphical model representing the effects of a new drug.

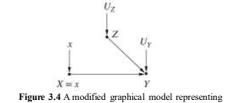


Figure 3.4 A modified graphical model representing an intervention on the model in Figure 3.3.

Simulate the intervention in the form of a graph surgery: set drug usage in the population

- Manipulated probability P_{dr} (操组概率)

The Adjustment Formula (矫正公式)

- invariance for manipulated probability:

$$P_m(Y = y | Z = z, X = x) = P(Y = y | Z = z, X = x) \quad \text{and} \quad P_m(Z = z) = P(Z = z)$$

- Z and X are d-separated in the modified model and are, therefore, independent under the intervention distribution.

$$P_m(Z = z | X = x) = P_m(Z = z) = P(Z = z)$$

- Then we have

$$P(Y = y | \text{do}(X = x)) = P_m(Y = y | X = x) \quad (\text{by definition}) \quad (3.2)$$

$$= \sum_z P_m(Y = y | X = x, Z = z) P_m(Z = z | X = x) \quad (3.3)$$

$$= \sum_z P_m(Y = y | X = x, Z = z) P_m(Z = z) \quad (3.4)$$

- Using the invariance relations, we get the adjustment formula:

$$P(Y = y | \text{do}(X = x)) = \sum_z P(Y = y | X = x, Z = z) P(Z = z) \quad (3.5)$$

The Adjustment Formula (矫正公式)

$$P(Y = y | \text{do}(X = x)) = \sum_z P(Y = y | X = x, Z = z) P(Z = z) \quad (3.5)$$

This procedure is referred to as "adjusting for Z" or "controlling for Z"

Eg. numerical analysis to Simpson's story:

	Drg	No drg
Men	81 out of 87 recovered (93%)	234 out of 270 recovered (87%)
Women	192 out of 263 recovered (73%)	55 out of 80 recovered (69%)
Combined data	275 out of 530 recovered (52%)	289 out of 350 recovered (81%)

$$P(Y = 1 | \text{do}(X = 1)) = P(Y = 1 | X = 1, Z = 1) P(Z = 1) + P(Y = 1 | X = 1, Z = 0) P(Z = 0)$$

$$\text{Substituting the figures given in Table 1.1 we obtain: } P(Y = 1 | \text{do}(X = 1)) = \frac{0.93(87) + 270}{700} + \frac{0.73(263 + 80)}{700} = 0.832$$

while, similarly,

$$P(Y = 1 | \text{do}(X = 0)) = \frac{0.87(87) + 270}{700} + \frac{0.69(263 + 80)}{700} = 0.7818$$

Thus, comparing the effect of drug-taking ($X = 1$) to the effect of not-taking ($X = 0$), we obtain

$$ACE = P(Y = 1 | \text{do}(X = 1)) - P(Y = 1 | \text{do}(X = 0)) = 0.832 - 0.7818 = 0.0502$$

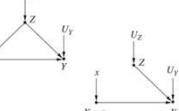


Figure 3.3: A graphical model representing the effects of a new drug.



Figure 3.4: A modified graphical model representing an intervention on the model in Figure 3.3.

The Adjustment Formula (矫正公式)

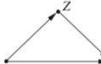


Figure 3.5 A graphical model representing the effects of a new drug, with X representing drug usage, Y representing recovery, and Z representing blood pressure.

Eg. the blood pressure example of Simpson's paradox

- Under such conditions, the intervention graph is equal to the original graph—no arrow need be removed—and the adjustment formula reduces to

$$P(Y = y | \text{do}(X = x)) = P(Y = y | X = x)$$

The Adjustment Formula (矫正公式)

To Adjust or not to Adjust?

- Rule 1 (The Causal Effect Rule)** Given a graph G in which a set of variables PA are designated as the parents of X, the causal effect of X on Y is given by

$$P(Y = y | \text{do}(X = x)) = \sum_z P(Y = y | X = x, PA = z) \quad (3.6)$$

where z ranges over all the combinations of values that the variables in PA can take.

- multiply and divide the summand in (3.6) by the probability $P(X = x | PA = z)$, we get a more convenient form:

$$P(Y = y | \text{do}(x)) = \sum_z \frac{P(X = x, Y = y, PA = z)}{P(X = x | PA = z)} \quad (3.7)$$

- propensity score (倾向分数) $P(X = x | PA = z)$

The Adjustment Formula (矫正公式)

the Truncated Product Rule: (截断乘积公式)

$$P(z, y | \text{do}(x)) = \frac{P(x, y, z)}{P(x | z)} \quad (3.10)$$

- According to the Rule of Product Decomposition, the preintervention (干预前分布) distribution in the model of Figure 3.3

$$P(x, y, z) = P(z)P(x | z)P(y | x, z) \quad (3.8)$$

the postintervention distribution (干预后分布)

$$P(z, y | \text{do}(x)) = P(z)P(y | x, z) = P(z)P(y | x, z) \quad (3.9)$$

Or by (3.8) and (3.9):

$$P(z, y | \text{do}(x)) = \frac{P(x, y, z)}{P(x | z)} \quad (3.10)$$

the conditional probability $P(x | z)$ is all we need to know in order to predict the effect of an intervention $\text{do}(x)$ from nonexperimental data governed by the distribution $P(x, y, z)$.

Eg. the adjustment formula for Figure 3.3 is by remove $P(x | z)$:

$$P(y | \text{do}(x)) = \sum_z P(z)P(y | x, z)$$

The Adjustment Formula (矫正公式)

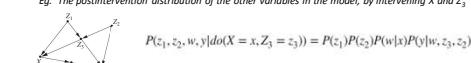
Multiple Interventions (多重干预) and the Truncated Product Rule

- Generalize the adjustment formula to multiple interventions

the truncated product formula or g-formula

$$P(x_1, x_2, \dots, x_n | \text{do}(x)) = \prod_i P(x_i | p_{a_i}) \quad \text{for all } i \text{ with } X_i \text{ not in } X.$$

Eg. The postintervention distribution of the other variables in the model, by intervening X and Z_3



$$P(z_1, z_2, w, y | \text{do}(X = x, Z_3 = z_3)) = P(z_1)P(z_2)P(w | x)P(y | w, z_3, z_2)$$

Outlines

The Effects of Interventions

- Interventions
- The Adjustment Formula
- The Backdoor Criterion & The Front-Door Criterion

The Backdoor Criterion

To determine whether we can compute a causal effect.

Definition 3.3.1 (The Backdoor Criterion) :
Given an ordered pair of variables (X, Y) in a directed acyclic graph G , a set of variables Z satisfies the backdoor criterion relative to (X, Y) if no node in Z is a descendant of X , and Z blocks every path between X and Y that contains an arrow into X .

- If a set of variables Z satisfies the backdoor criterion for X and Y , then the causal effect of X on Y is given by the formula

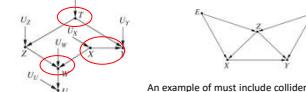
$$P(Y = y|do(X = x)) = \sum_z P(Y = y|X = x, Z = z)P(Z = z)$$

- Condition on the set of nodes Z

- block all spurious paths (伪路径), the paths contain an arrow to X) between X and Y .
- leave all directed paths from X to Y unperturbed.
- create no new spurious paths

The Backdoor Criterion

Eg: Figure 2.8



An example of must include collider node Z for $X \rightarrow Y$.

for the causal effect of X on Y , there is no back-door path, so its $P(Y = y|do(X = x))$. if we add w into the adjustment formula, then we may generate a spurious paths $X \rightarrow W \leftarrow Z \leftarrow T \rightarrow Y$. But if we do need to analysis the sensitive of $X \rightarrow Y$ with w , then we need to add other variables to compose the backdoor path: e.g. we add T into the backdoor path:

$$P(Y = y|do(X = x), W = w) = \sum_t P(Y = y|X = x, W = w, T = t)P(T = t|X = x, W = w) \quad (3.11)$$

The Backdoor Criterion

Other applications of the Backdoor Criterion

Any suitable variable or set of variables that we adjust on—whether it be $PA(X)$ or any other set that conforms to the backdoor criterion—must return the same result for $P(Y = y|do(X = x))$

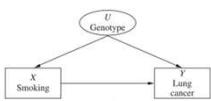
$$\sum_w P(Y = y|X = x, W = w)P(W = w) = \sum_z P(Y = y|X = x, Z = z)P(Z = z)$$

This equality is useful in two ways:

- it provides us with a choice of which variables to adjust for.
- the equality constitutes a testable constraint on the data when all the adjustment variables are observed

The Front-door Criterion: eg. the century-old debate

Eg: the century-old debate on the relation between smoking and lung cancer, the tobacco industry provided a theory to explain the relation between smoking and lung cancer as below:



Here the hypothesized carcinogenic genotype (致癌基因) that inborn craving for nicotine still not been observed

By this model, how can we analyze the causal effect of smoking on Lung cancer?

The Front-door Criterion: eg. the century-old debate

Eg: Assume the following factors were measured simultaneously on a randomly selected sample of 800,000 subjects considered to be at very high risk of cancer.

Consider the factors: 1. Whether the subject smoked 2. Amount of tar in the subject's lungs 3. Whether lung cancer has been detected in the patient.

		Smokers 400		Non-smokers 400		All subjects 800		
		Yes	No	Yes	No	Yes	No	
No cancer	321	3	18	38	341	39	56	
	(80%)	(0%)	(48%)	(100%)	(89%)	(97%)	(98%)	
Cancer	57	19	2	30	19	342	76	344
	(13%)	(95%)	(100%)	(100%)	(100%)	(100%)	(100%)	(100%)

Table 3.1 A hypothetical data set of randomly selected samples showing the percentage of cancer cases for smokers and nonsmokers in each tar category (numbers in thousands)

Smokers get lower risk than nonsmokers!

		Smokers 400		Non-smokers 400		All subjects 800		
		Yes	No	Yes	No	Yes	No	
Tar	323	18	3	38	56	56	3	
	(80%)	(48%)	(0%)	(100%)	(97%)	(98%)	(100%)	
No tar	67	19	2	30	19	342	76	344
	(13%)	(95%)	(100%)	(100%)	(100%)	(100%)	(100%)	(100%)

Table 3.2 Recomputation of the data set of Table 3.1 showing the percentage of cancer cases in each smoking-tar category (numbers in thousands)

Smokers get higher risk than nonsmokers in Tar!

The Front-door Criterion: eg. the century-old debate

The do-operator can be applied to graphical patterns that do not satisfy the backdoor criterion to identify effects.

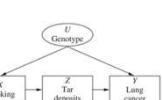


Figure 3.10 A graphical model representing the relationships between smoking (X) and lung cancer (Y), with unobserved confounder (U) and a mediating variable Z (中介变量)

- This model does not satisfy the backdoor criterion, because there is still no variable capable of blocking the spurious path $X \leftarrow U \rightarrow Y$.
- The causal effect $P(Y = y|do(X = x))$ is identifiable in this model, through two consecutive applications of the backdoor criterion.

The Front-door Criterion: eg. the century-old debate

two consecutive applications of the backdoor criterion, X is the backdoor variable in 3.13 (to block the Z - X - Y path):

$$P(Z = z|do(X = x)) = P(Z = z|X = x) \quad (3.12)$$

$$P(Y = y|do(Z = z)) = \sum_x P(Y = y|Z = z, X = x)P(X = x) \quad (3.13)$$

Then the causal effect for $X \rightarrow Y$ is (since we need to count all the effect of Z on Y , so the summation is taken on Z):

$$P(Y = y|do(X = x)) = \sum_z P(Y = y|do(Z = z))P(Z = z|do(X = x)) \quad (3.14)$$

$$P(Y = y|do(X = x)) = \sum_{z'} \sum_{x'} P(Y = y|Z = z, X = x')P(X = x')P(Z = z|X = x) \quad (3.15)$$

notice the distinguish between the x that appears in (3.12) and the one that appears in (3.13), the latter of which is merely an index of summation denoted as x .

Equation (3.15) is known as the front-door formula

The Front-door Criterion

Definition 3.4.1 (Front-Door)

A set of variables Z is said to satisfy the front-door criterion relative to an ordered pair of variables (X, Y) if

- Z intercepts (拦截) all directed paths from X to Y . (Z only intercepts the directed path $X \rightarrow Y$)
- There is no backdoor path from X to Y .
- All backdoor paths from Z to Y are blocked by Z .

Theorem 3.4.1 (Front-Door Adjustment) If Z satisfies the front-door criterion relative to (X, Y) and if $P(x, z) > 0$, then the causal effect of X on Y is identifiable and is given by the formula

$$P(Y|do(x)) = \sum_z P(z|x) \sum_{x'} P(Y|x', z)P(x') \quad (3.16)$$

Chapter4 counterfactuals (反事实) (section 4)

Autumn 2024
Buzhou Tang

Harbin Institute of Technology (Shenzhen) Intelligent Computation Research Center (HITZIC ICRC)

Outlines

- Defining Counterfactuals
- Computing Counterfactuals
- Nondeterministic Counterfactuals

Defining Counterfactuals

Eg. While driving home last night, I came to a fork in the road, where I had to make a choice: to take the freeway ($X = 1$) or go on a surface street ($X = 0$). I took Sepulveda, only to find out that the traffic was tough and go. As I arrived home, an hour later, I asked myself: "Gee, I should have taken the freeway."

What's that mean in Causality?

- It means that my mental estimate of the expected driving time on the freeway, on the same route, was lower than what actually happened, and governed by the same idiosyncratic driving habits that I have, would have been lower than my actual driving time?

Defining Counterfactuals

Definition: This kind of statement—an "if" statement in which the "if" portion is untrue or unrealized—is known as a **counterfactual**. The "if" portion of a counterfactual is called the **hypothetical condition**, or more often, the antecedent (前件条件).

Try to express in do-expressions:
 $E(\text{driving time} | \text{do(freeway)}, \text{driving time} = 1 \text{ hour})$

We must distinguish symbolically between:
Actual driving time
Hypothetical driving time under freeway conditions when actual surface driving time is known to be 1 hour.

Clash occurs (这里有冲突)

Defining Counterfactuals: Symbolic denotes

In the example case:
 $Y_{x=1}$ (or Y_x , where context permits), denoting the freeway driving time.
 $Y_{x=0}$ (or Y_0), and Sepulveda driving time. Y_0 is the value of Y that is actually observed.

The quantity we wish to estimate is denoted as:
 $E(Y_{x=1}|X = 0, Y = Y_0 = 1)$

Here the hypothetical variable $Y_{x=1}$ is predicated upon one event ($X = 1$) and conditioned upon the conflicting event, $X = 0$, which was **actually observed**.

In General, a counterfactuals could be given as the sentence:
"Y would be y had X been x, in situation U = u"

denoted as:
 $Y_x(U) = y$

where Y and X are any two variables in V

Defining Counterfactuals: SCM Example

Eg. SCM contains three variables X, Y, U, and defined by two equations:
 $X = aU$
 $Y = bX + U$

The counterfactual $Y_x(u)$ is computed as: (i.e. what Y would be had X been x, in situation U = u .)

- Replacing the first equation with $X = x$ gives the "modified" model M_x :
 $X = x$
 $Y = bx + U$
- Substituting $U = u$ and solving for Y gives: $Y_x(u) = bx + u$
- Given the values of x and u, the values of $Y_x(u)$ could be given in the table (let a = b = 1)

U	$X(u)$	$Y(u)$	$Y_x(u)$	$Y_x(u)$	$X_x(u)$	$X_x(u)$	$X_x(u)$	
1	1	2	2	3	-4	1	1	1
2	2	4	3	4	5	2	2	2
3	3	6	4	5	6	3	3	3

Which means that we can do compute the value of the hypothesis: Y would be y had X been x !

Structural Causal Model—SCM

Outlines

- Defining Counterfactuals
- Computing Counterfactuals
- Nondeterministic Counterfactuals

Computing Counterfactors

The fundamental law of Counterfactuals
Let M_x stand for the modified version of M, with the equation of X replaced by $X = x$. The formal definition of the counterfactual $Y_x(u)$ reads

$$Y_x(u) = Y_{M_x}(u)$$

It means that the values that these counterfactuals receive are not totally arbitrary, but must cohere with each other to be consistent with an underlying model.

Consistency rule
if $X = x$ then $Y_x = Y$
If X is binary, the consistency takes the convenient form ($X=0$ or 1):

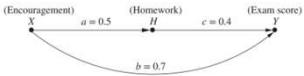
$$Y = XY_1 + (1 - X)Y_0$$

Computing Counterfactors

The three steps in computing counterfactuals

- Abduction (溯因):** Use evidence $E = e$ to determine the value of U.
- Action (作用):** Modify the model, M, by removing the structural equations for the variables in X and replacing them with the appropriate functions $X = x$, to obtain the modified model, M_x .
- Prediction (预测):** Use the modified model, M_x , and the value of U to compute the value of Y, the consequence of the counterfactual.

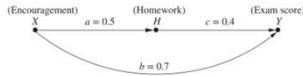
Computing Counterfactuals: an example



An "encouragement design":

X: the amount of time a student spends in an after-school remedial program (课后补习班)
H: the amount of homework a student does
Y: a student's score on the exam

Computing Counterfactuals: an example



Corresponding SCM equation:

$$\begin{aligned} X &= U_X \\ H &= a \cdot X + U_H \\ Y &= b \cdot X + c \cdot H + U_Y \\ \sigma_{U_i U_j} &= 0 \quad \text{for all } i, j \in \{X, H, Y\} \end{aligned}$$

Let us consider a student named Joe, for whom we measure $X = 0.5$, $H = 1$, and $Y = 1.5$. Suppose we wish to answer the following query: What would Joe's score have been had he doubled his study time H?

Computing Counterfactuals

Let us consider a student named Joe, for whom we measure $X = 0.5$, $H = 1$, and $Y = 1.5$. Suppose we wish to answer the following query: What would Joe's score have been had he doubled his study time H?

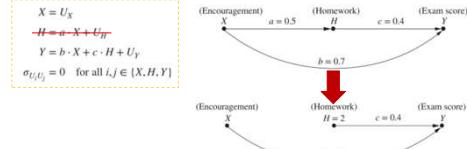
step(i) Abduction: In this case, we are able to obtain the specific characteristics of Joe from the evidence

$X = U_X$	$U_X = 0.5$,
$H = a \cdot X + U_H$	$U_H = 1 - 0.5 \cdot 0.5 = 0.75$, and
$Y = b \cdot X + c \cdot H + U_Y$	$U_Y = 1.5 - 0.7 \cdot 0.5 - 0.4 \cdot 1 = 0.75$.
$\sigma_{U_i U_j} = 0 \quad \text{for all } i, j \in \{X, H, Y\}$	

Computing Counterfactuals

Let us consider a student named Joe, for whom we measure $X = 0.5$, $H = 1$, and $Y = 1.5$. Suppose we wish to answer the following query: What would Joe's score have been had he doubled his study time H?

step(ii) Action: we simulate the action of doubling Joe's study time by replacing the structural equation for H with the constant $H = 2$. The modified model is depicted in Figure



Computing Counterfactuals

Let us consider a student named Joe, for whom we measure $X = 0.5$, $H = 1$, and $Y = 1.5$. Suppose we wish to answer the following query: What would Joe's score have been had he doubled his study time H?

step(iii) Prediction:

$X = U_X$	$Y_{H=2}(U_X = 0.5, U_H = 0.75, U_Y = 0.75)$
$H = a \cdot X + U_H$	$= 0.5 \cdot 0.7 + 2.0 \cdot 0.4 + 0.75$
$Y = b \cdot X + c \cdot H + U_Y$	$= 1.90$
$\sigma_{U_i U_j} = 0 \quad \text{for all } i, j \in \{X, H, Y\}$	

Outlines

- Defining Counterfactuals
- Computing Counterfactuals
- Nondeterministic Counterfactuals

Nondeterministic Counterfactuals

To examine how nondeterminism is reflected in the calculation of counterfactuals, let us assign probabilities to the values of U in the model of the Eqs below, where $a=1$, $b=1$.

$$\begin{aligned} X &= aU \\ Y &= bX + U \\ &\begin{array}{ccccc} a & X(0) & X(1) & Y(0) & Y(1) \\ \hline 1 & 2 & 2 & 2 & 4 \\ 2 & 2 & 2 & 3 & 5 \\ 3 & 3 & 6 & 4 & 5 \end{array} \\ &\sigma_{U_i U_j} = 0 \quad \text{for all } i, j \in \{X, Y\} \end{aligned}$$

Imagine that $U = \{1, 2, 3\}$ represents three types of individuals in a population, occurring with probabilities

$$P(U = 1) = \frac{1}{2}, \quad P(U = 2) = \frac{1}{3}, \quad P(U = 3) = \frac{1}{6}$$

We can similarly compute the probability of any counterfactual statement, e.g.,

$$P(Y_1 = 4) = \frac{1}{6}, \quad P(Y_2 = 3) = \frac{1}{3}, \quad P(Y_2 > 3) = \frac{1}{2}$$

And joint probabilities of every combination of counterfactual and observable events:

$$P(Y_2 > 3, Y_1 < 4) = \frac{1}{3}, \quad P(Y_1 < 4, Y - X > 1) = \frac{1}{3}, \quad P(Y_1 < Y_2) = 1$$

Here cross-world (like Y_2 and Y_1) probabilities are as simple to derive as intra-world ones.

Nondeterministic Counterfactuals

Let us examine a model with the below equations and structure, where $X = 1$ stand for having a college education, $U_1 = 1$ for having professional experience, Z for the level of skill needed for a given job, and Y for salary.

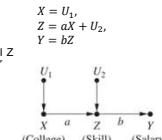
What we want to know:

$$E[Y|do(X = 1)]$$

which stands for the expected salary of individuals with skill level $Z = 1$, had they received a college education (假如当初他们接受了大学教育的话).

In general, the do-expression will not capture our counterfactual question (Z blocks the causal connection between X and Z):

$$E[Y|do(X = 1), Z = 1] \neq E[Y_{Z=1}|Z = 1]$$



Nondeterministic Counterfactuals

counterfactuals can capture both single-world and cross-world probabilities flexible

- counterfactual notation can capture the postintervention, single-world expression $E[Y|do(X = 1), Z = 1]$
- The translation of $E[Y|do(X = 1), Z = 1]$ into counterfactual notation is simply (用反事实的符号来表示干预)

$$E[Y_{Z=1}|Z_{X=1} = 1]$$

which explicitly designates the event $Z = 1$ as postintervention (the single-world). And the calculation for nondeterministic counterfactuals can be simplified by the method we used before

$$P(Y = y|do(X = 1), Z = z) = \frac{P(Y = y, Z = z|do(X = 1))}{P(Z = z|do(X = 1))}$$

Nondeterministic Counterfactors

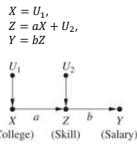
We can calculate counterfactuals associated with the model and gain the table below, with all the subscripts denoting the state of X

u_1	u_2	$X(u)$	$Z(u)$	$Z = u_1$	$Z = aX + u_2$	$Y = bZ$
0	0	0	0	0	0	0
0	1	0	0	0	($a+1)b$)	a
1	0	1	ab	1	ab	$a+1$
1	1	1	$a+1$	$(a+1)b$	b	$(a+1)b$

We get the table by replacing the equation $X = u$ with the appropriate constant (0 or 1) and solving for Y and Z.

What we want to know: $E[Y_{X=1}|Z=1]$

Notice that condition $Z = 1$ occurs only for $u_1 = 0$ and $u_2 = 1$ (assuming that $a \neq 0$ and $a \neq 1$), and under these conditions, each of Y, Y_1 , and Y_2 has a definite value.



Nondeterministic Counterfactors

Assuming that $a \neq 0$ and $a \neq 1$, then we can directly get the expectation values:

$$\begin{aligned} E[Y_{X=1}|Z=1] &= (a+1)b \quad \text{and} \quad E[Y_{X=0}|Z=1] = b \\ \text{and} \quad E[Y_{X=1} - Y_{X=0}|Z=1] &= ab \end{aligned}$$

if $a = 1$, then we need to consider the probability of Y since we get two rows satisfied the condition of $Z = 1$ (or a):

$$E[Y_{X=1}|Z=1] = b \left(1 + \frac{P(u_1 = 0)P(u_2 = 1)}{P(u_1 = 0)P(u_2 = 1) + P(u_1 = 1)P(u_2 = 0)} \right)$$

$$E[Y_{X=0}|Z=1] = b \left(\frac{P(u_1 = 0)P(u_2 = 1)}{P(u_1 = 0)P(u_2 = 1) + P(u_1 = 1)P(u_2 = 0)} \right)$$

The fact that, in both cases, the first expression is larger than the second, which demonstrates again that **the skill-specific causal effect of education on salary is nonzero**, despite the fact that salaries are determined by skill only, not by education.

Causal Inference Summary

- Basic Conceptions

- Simpson's Paradox
- Structural Causal Models
- Graph Models: Chains, Forks, Colliders

- Intervention

- Interventions
- The Adjustment Formula 矫正公式
- The Backdoor Criterion&The Front-Door Criterion 后门准则&前门准则

- Counterfactuals

- Definition
- Computing of Counterfactors
- Nondeterministic Counterfactors

Homework

- 思考题3.4.1(p98), 4.3.1(p148)

Artificial Intelligence

第3部分 生成式人工智能方法

- 语言的传统表示方法
- 语言的神经网络表示
- 生成式大语言模型

Buzhou Tang
Email: tangbz@hit.edu.cn
<http://icrc.hit.edu.cn>

Harbin Institute of Technology (Shenzhen)
Intelligent Computation Research Center
(ICRC, HITZ)

生成式人工智能的兴起

ChatGPT:2022.11
Midjourney生成:《太空歌剧院》(Théâtre D'opéra Spatial), 2022.09
Gemini: Google, 2023.12

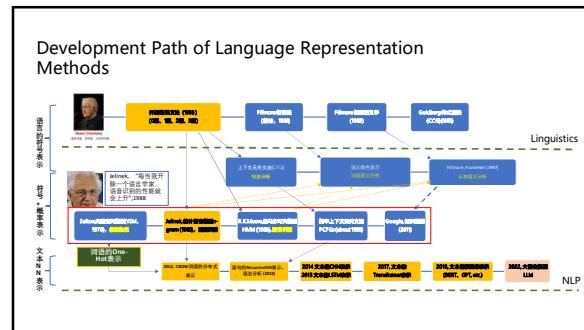
Part 1 The Classical Representation of Language

Oct 2024
Instructor: Buzhou Tang

Harbin Institute of Technology (Shenzhen) Intelligent Computation Research Center (HITZ) ICRC

Outlines

- Development Path of Language Representation Methods
- Symbolist School: Grammars of Language
- Statistical School: Statistical Language Model: n-gram, Vector Space Model
- Connectionist School: Distributed Representation of Word Sense



Chomsky Grammars

Chomsky Hierarchy: This includes a hierarchy of formal grammars that classifies languages based on their expressive power. It consists of four levels:

- Type 0: Unrestricted grammars
- Type 1: Context-sensitive grammars
- Type 2: Context-free grammars
- Type 3: Regular grammars

Context-sensitive grammars are more powerful than context-free grammars and can describe more complex language structures. They are called context-sensitive because the rules for producing sentences depend on the context of non-terminal symbols in a production rule.

Context-free Grammar

A context-free grammar G is defined by the 4-tuple $G = (V, \Sigma, R, S)$, where (1)

- V is a finite set; each element $v \in V$ is called a **nonterminal character** or a **variable**. Each variable represents a different type of phrase or clause in the sentence. Variables are also sometimes called **syntactic categories**. Each variable defines a sub-language of the language defined by G .
- Σ is a finite set of **terminals**, disjoint from V , which make up the **actual content** of the sentence. The set of terminals is the alphabet of the language defined by the grammar G .
- R is a finite relation in $V \times (V \cup \Sigma)^*$, where the asterisk represents the Kleene star operation (zero or more repetitions). The members of R are called the (rewrite) rules or productions of the grammar. (also commonly symbolized by a P)
- S is the start variable (or start symbol), used to represent the **whole sentence** (or program). It must be an element of V .

[1] Sipser, Michael (1997). Introduction to the Theory of Computation, PWS Publishing, ISBN 978-0-534-94726.

Context-free Grammar: An example

Example of generating:

```

terminals
  [green, hate, great, green, ideas, linguist]
  nonterminals
    [SENTENCE, NOUNPHRASE, VERBPHRASE, NOUN, VERB, ADJ]
  production rules
    SENTENCE -> NOUNPHRASE VERBPHRASE
    NOUNPHRASE -> ADJ NOUN
    NOUNPHRASE -> VERB NOUNPHRASE
    VERBPHRASE -> VERB
    VERBPHRASE -> VERB NOUNPHRASE
    VERB -> ideas
    VERB -> linguist
    VERB -> generate
    VERB -> hate
    ADJ -> great
    ADJ -> green
  SENTENCE -> NOUNPHRASE VERBPHRASE -> ADJ NOUNPHRASE VERBPHRASE ->
  ADJ NOUN VERB NOUNPHRASE -> ADJ NOUN VERB ADJ NOUNPHRASE -> ADJ NOUN VERB ADJ ADJ NOUNPHRASE ->
  great linguist generate ADJ ADJ NOUN -> great linguist generate great ADJ NOUN ->
  great linguist generate great green NOUN -> great linguist generate great green ideas.
  
```

Context-free Grammar: An example

Example of syntax parsing: for the sentence "小猫坐在垫子上。"

- First, we need to recognize the syntax component of the sentence, which include the subject (小猫) , predicate (坐) and object (垫子) .
- Then, choose the syntax rules:
 - $S \rightarrow NP VP$ (句子由名词短语和动词短语组成)
 - $NP \rightarrow Det N$ (名词短语由限定词和名词组成)
 - $NP \rightarrow AdjP$ (名词短语由形容词和名词组成)
 - $Det \rightarrow "a"$
 - $N \rightarrow "猫"$
 - $V \rightarrow "坐"$
 - $VP \rightarrow NP VP$ (名词短语也可以由介词和名词短语组成)
 - $P \rightarrow "在"$
 - $N \rightarrow "垫子"$
 - $NP \rightarrow "垫子上"$ (这里我们将"垫子上"视为一个整体)
- Applying the rules to generate the parsing results

Through the above steps, we can use Chomsky grammar to analyze and understand the syntactic structure of natural language sentences. This analysis helps us better understand the complexities and subtleties of language.

Case Grammar

Case Grammar is a linguistic theory developed by **Charles Fillmore** in the 1960s. It focuses on the **syntactic and semantic relationship of words** in a sentence. Some cases:

- 1. Agentive Case (施事格): This case indicates the entity that performs the action of the verb.
- Example: In "She gave the book to John," "John" is the recipient.
- 2. Dative Case (Recipient受事格): This case indicates the entity that receives the action or is indirectly affected by it.
- Example: In "She cut the paper with scissors," "scissors" is the instrument.
- 3. Instrumental Case (Instrument, 工具格): This involves an entity used by the agent to perform an action.
- Example: In "He ate the sandwich with a knife." "Knife" is the instrument.
- 4. Locative Case (Location, 处所格): This case denotes the location where the action occurs.
- Example: In "He sat at the table." "The table" is the source.
- 5. Objective Case (Object, 宾格): This case refers to the entity that is directly affected by the action.
- Example: In "He hit the mouse," "the mouse" is the object.
- 6. Source Case (Source, 源格): This indicates the origin from which something moves or changes.
- Example: In "She left the room," "the room" is the source.
- 7. Benefactive Case (Beneficiary, 利益格): This represents the entity for whom the action is performed.
- Example: In "He gave a cake to his mother." "His mother" is the beneficiary.

Case Grammar is a useful tool for understanding the underlying structure of sentences and the roles that different nouns play in relation to the verbs. This theory has been influential in the fields of syntax, semantics, and language teachings.

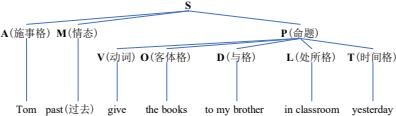
Case Grammar

There are three basic rules in case grammar parsing:

- Rule 1: $S \rightarrow M + P$ i.e., 句子 (sentence, S) -> 情态 (modality, M) + 命题 (proposition, P)
 Rule 2: $P \rightarrow V + Cl \leftarrow \dots \leftarrow C2$ V (动词), Cl (格)
 Rule 3: $C \rightarrow K + NP$ K(格标, Label of a case, e.g., L 代表处所格), could be empty.

Example 1:
 $Tom \text{ give } the \text{ books } to \text{ my } brother \text{ in } classroom \text{ yesterday}$

The parsing tree by the case grammar is:

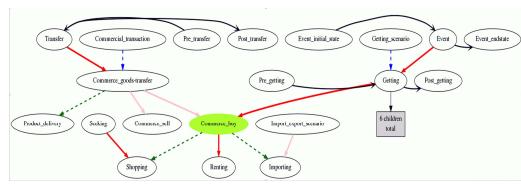


Frame semantics

Frame Semantics is a theory of linguistic meaning that focuses on understanding how the meanings of words are linked to the knowledge structures or 'frames' they evoke in our minds.

1. **Frames as Mental Structures:** words evoke certain mental structures or 'frames', which are essentially conceptual backgrounds or contexts within which words make sense. These frames are built from our experiences and cultural knowledge.
 2. **Lexical Units and Frames:** every word or lexical unit is associated with a specific frame. For instance, the word "self" evokes a commercial transaction frame, which includes roles like buyer, seller, goods, and price.
 3. **Role and Frame Elements:** Each frame consists of various elements or roles. For example, the word "self" is seen as the buyer in a commercial transaction frame.
 4. **Contextual Meaning:** the meaning of a word can change depending on the frame it is used in. Words are flexible and can evoke different frames based on the context, altering their meaning.
 5. **Cultural and Experiential Influence:** frames are influenced by cultural and experiential factors, meaning that different linguistic communities might have different frames for the same lexical units.
- Fillmore's theory emphasizes the **interconnectedness of language, knowledge, and experience**, suggesting that understanding language involves more than just knowing vocabulary and grammar.

Frame Semantics: example from FrameNet



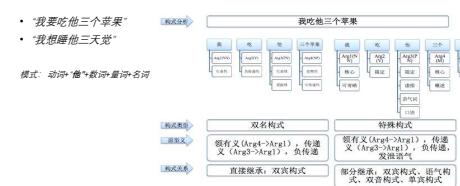
<http://framenet.icsi.berkeley.edu>

Cognitive Construction Grammar (CCG)

Cognitive Construction Grammar (CCG), a branch of cognitive linguistics, focuses on understanding language as a set of constructions, or learned pairings of form and meaning. These constructions can range from simple words to complex sentence structures.

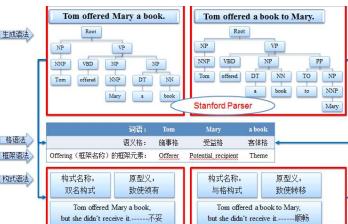
1. **Constructions as Fundamental Units:** In CCG, language is seen as a collection of constructions, each with its own syntax and semantics.
 2. **Form-Meaning Pairing:** Each construction embodies a specific meaning paired with a particular form. Example: "Can you [NP]" is as in "Can you help?" Here, "[NP]" is a construction that requests an action.
 3. **Usage-Based Approach:** CCG emphasizes that our knowledge of constructions arises from our experience with language.
 4. **Embodyed Cognition:** constructions are embodied in our physical and social experiences.
 5. **Construction Across Linguistic Levels:** CCG looks at constructions across all levels of language, from simple morphemes to complex sentence structures. Example: The passive construction in English, as in "The ball was kicked by the boy," involves rearranging the typical subject-verb-object order to highlight or background the construction.
 6. **Constructions and Context:** In CCG, the meaning of a construction can change depending on the context in which it is used.
- Example: The word "check" can mean different things in different constructions. "Check the box" (mark) vs. "Check the store" (visit).
- Cognitive Construction Grammar provides a comprehensive framework for understanding language as a dynamic system of meaningful constructions shaped by our cognitive processes and linguistic experiences.

Cognitive Construction Grammar: example



注：以上例子来自王宾：认知构式语法[J].外语学刊,2011(2)

Comparison of Grammars: by Example



Outlines

- | | |
|---|---|
| Development Path of Language Representation Methods | Symbolist School: Grammars of Language |
| Statistical School: Statistical Language Model: n-gram | Statistical Language Model: n-gram |
| Connectionist School: Representation of Word Sense | Distributed Model |

Statistical Language Model: n-gram

Markov Assumption: The probability of a word only depends on the previous $n-1$ words. This simplifies language modeling but also limits the context that the model can consider.

Probability Estimation:

$$p(W_1 W_2 \dots W_n) = p(W_1) p(W_2 | W_1) \dots p(W_n | W_1 W_2 \dots W_{n-1})$$

$$p(W_n | W_1 W_2 \dots W_{n-1}) = p(W_n | W_{n-1}) = \text{count}(W_{n-1}, W_n) / \text{count}(W_{n-1}) (\text{Bigram})$$

Perplexity on a given text or sentence:

$$\log(\text{Perplexity}) = -\frac{1}{N} \sum_{i=1}^N \log(P(\text{token}_i | \text{context for token}_i))$$

*Unigrams (1-gram): e.g. in the sentence "The cat sat on the mat," "cat" would be a unigram.

*Bigrams (2-grams): e.g. In the same sentence, the bigrams would be "The cat," "cat sat," "sat on," "on the," "the mat."

*Trigrams (3-grams): e.g. In the sentence, the trigrams would be "The cat sat," "cat sat on," "sat on the," "on the mat."

The Zero-frequency Problem

- Suppose some event not in our observation S
 - Model will assign zero probability to that event
 - And to any set of events involving the unseen event
 - Happens very frequently with language
 - It is incorrect to infer zero probabilities
 - especially when creating a model from short samples



- Reference: James Allan, University of Massachusetts Amherst, Pandu Nayak and Prabhakar Raghavan, Stanford University

Vector Space Model for Text

Salton G, Wong A, Yang C S. A vector space model for automatic indexing[J]. Communications of the ACM, 1975, 18(11):613-620.

Technical Memo Example

- Titles:**

 - c1: Human machine interface for Lab ABC computer applications
 - c2: A survey of user opinion of computer system response time
 - c3: The EPS user interface management system
 - c4: System and human system engineering testing of EPS

1. Base vector dimensions are high and change rapidly.
2. The selection of base vectors lacks theoretical guidance
3. Document vectors are sparsified and discretized.

- m1: The generation of random, binary, unordered trees
- m2: The intersection graph of paths in trees
- m3: Graph minors IV: Widths of trees and well-quasi-ordering
- m4: Graph minors: A review

III. *Orbital motors: A survey*

Smoothing methods: example

- Laplace correction:
 - add 1 to every count, normalize
 - problematic for large vocabularies
 - Lindstone correction:
 - add a small constant ϵ to every count, re-normalize

$P(\text{blue}) = (1 + \varepsilon) / (3 + 5\varepsilon)$
 $P(\text{red}) = (1 + \varepsilon) / (3 + 5\varepsilon)$
 $P(\text{yellow}) = (1 + \varepsilon) / (3 + 5\varepsilon)$
 $P(\text{green}) = (0 + \varepsilon) / (3 + 5\varepsilon)$
 $P(\text{white}) = (0 + \varepsilon) / (3 + 5\varepsilon)$

- Absolute Discounting
 - subtract a constant ϵ , re-distribute the probability mass

References: James Allan, University of Massachusetts Amherst, Pandu Nayak and Prabhakar Raghavan, Stanford University S. F. Chen and J. T. Goodman. An empirical study of smoothing techniques for language modeling. In Proceedings of the 34th Annual Meeting of the ACL-1996

Application of n-gram

- Search Engine
 - Chinese Input Method: Pinyin, Handwriting, Speech etc.

Vector Space Model for Text

Soltan G, Wang A, Yang C S. A vector space model for automatically indexing of D. *Communications of the ACM*, 1975, 18(11):612-620.

- | Technical Memo Example | |
|---|---|
| Titles: | |
| c1: Human machine interface for Lab ABC computer applications | 1. Base vector dimensions are high and change rapidly. |
| c2: A survey of user selection of computer system response time | 2. The selection of base vectors lacks theoretical guidance |
| c3: The EPS user interface management system | 3. Document vectors are overspecified and discretized. |
| c4: Design of a graphical user interface for the EPS | |

e5: Relation of user-perceived response time to error measurement

- m1: The generation of random, binary, unordered trees
- m2: The intersection graph of paths in trees
- m3: Graph minors IV: Widths of trees and well-quasi-ordering

Vector Space Model for Text-SVD decomposition and LSI

SVD decomposition
and dimension



- 1. Base vector dimensions are high and change rapidly.
 - 2. The selection of base vectors lacks theoretical guidance.
 - 3. Document vectors are sparsified and discretized.

Summary

- Context-free Grammar
 - Case Grammar
 - Frame semantics
 - Cognitive Construction Grammar (CCG)
 - Statistical Language Model: n-gram
 - Vector Space Model for Text

Artificial Intelligence

第3部分 生成式人工智能方法

- 语言的传统表示方法
- 语言的神经网络表示
- 生成式大语言模型

Buzhou Tang

Email: tangbz@hit.edu.cn
http://erc.hit.edu.cn

Harbin Institute of Technology (Shenzhen)
Intelligent Computation Research Center
(ICRC, HITZ)



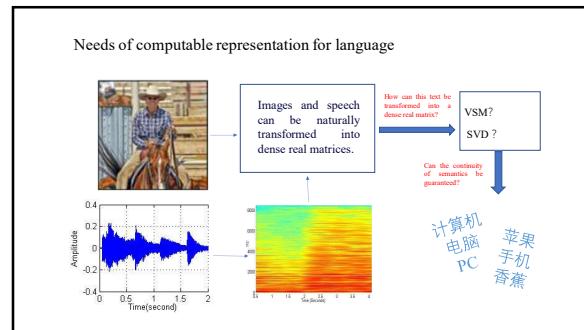
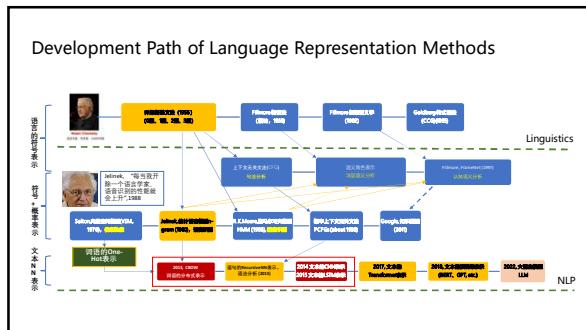
Part 2 Neural Network Representation of Language

Oct 2024
Instructor: Buzhou Tang

Harbin Institute of Technology (Shenzhen) Intelligent Computation Research Center (ICRC, HITZ)

Outlines

 Development Path of Language Representation Methods
 Symbolist School: Grammars of Language
 Statistical School: Statistical Language Model: n-gram Vector Space Model
 Connectionist School: Distributed Representation of Word Sense CNN Representation of Text



Word Embedding Models

Test of Time

This year, following the usual practice, we chose a NeurIPS paper from 10 years ago to receive the Test of Time Award, and "Distributed Representations of Words and Phrases and their Compositionality" by Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean, won.

Published at NeurIPS 2013 and cited over 40,000 times, the work introduced the seminal word embeddings technique word2vec, demonstrating the power of learning from large amounts of unstructured text and work catalyzed progress that marked the beginning of a new era in natural language processing.

Tomas Mikolov and Jeffrey Dean will be giving a talk about this work and related research on Tuesday, Dec. 4, 5:05 - 5:25 pm CST in Hail 2.



Distributed Representation Learning of Words : Word Embeddings

- Distributed Representation of words**
 - In comparison to **the one-hot representation** of a vector space model
 - a fixed low-dimensional dense vector
- The foundation of deep learning-based NLP

Visual comparison between One-Hot and Distributed representations:

One-Hot: A 5x5 matrix where the first column has a value of 1 and others are 0. Labels: 'V' is 词向量大小 (vector size), '0' is 0, '1' is 1, '2' is 2, '3' is 3, '4' is 4.

Distributed: A 5x5 matrix with values ranging from 0.00 to 0.12. Labels: 'V' is 词向量大小 (vector size), '0' is 0.00, '1' is 0.01, '2' is 0.02, '3' is 0.03, '4' is 0.04.

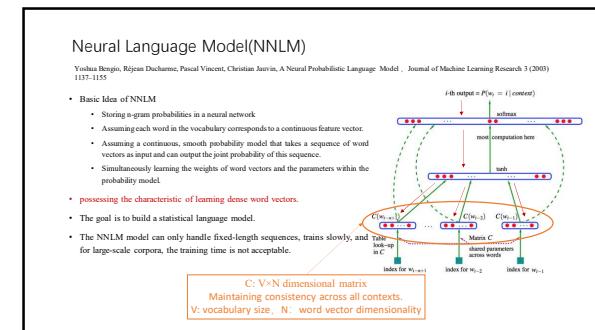
Distributed Representation Learning of Words : Word Embeddings

Word Embeddings 词嵌入、Word2Vec 词向量

- Neural Probabilistic Language Model (NNLM) [1] (2000-2003)
 - To solve the problem of n-gram dimensionality
- Mikolov and others' CBOW model, Skip-Gram model. (2013)
 - model architecture
 - training optimization
- Dissociation between Nouns and Verbs[4] (2016)

References:

- Yoshua Bengio, R\'ejean Duchaineau, Pascal Vincent, Christian Jauvin, A Neural Probabilistic Language Model, Journal of Machine Learning Research 3 (2003) 1137-1155
2. Mikolov T, Chen K, Corrado G, et al. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013).
3. Mikolov T, Sutskever I, Chen K, et al. Distributed representations of words and phrases and their compositionality(C) International Conference on Machine Learning, 2013:111-119.
4. Baohe He, Buzhou Tang, Qingsong Chen, and Longtuo Kang, "A Novel Word Embedding Learning Model Using the Dissociation between Nouns and Verbs," Neurocomputing, 2016.



The diagram illustrates two word embedding models: CBOW and Skip-Gram.

CBOW Model:

- Input:** A context window of words (e.g., w₀, w₁, w₂) is fed into a **PROJECTION** layer.
- Projection Layer:** The input words are mapped to vectors in a latent space.
- Middle Layer (Mapping Layer):** The vectors from the projection layer are aggregated using a **SUM** operation.
- Output:** The aggregated vector is passed through a **WEIGHT** matrix (w_t) to predict the target word.

Skip-Gram Model:

- Input:** A target word (e.g., w_t) is fed into a **PROJECTION** layer.
- Projection Layer:** The target word is mapped to a vector in the latent space.
- Output:** The vector is passed through three **WEIGHT** matrices (w₀, w₁, w₂) to predict the context words (w₀, w₁, w₂).

References:

Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean. "Efficient estimation of word representations in vector space." *arXiv preprint arXiv:1301.3781* (2013).

Word Embedding Learning: CBOW

Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean. "Efficient estimation of word representations in vector space." *arXiv preprint arXiv:1301.3781* (2013).

INPUT PROJECTION OUTPUT

$x = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} v_{w1}$

$h1 = \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1N} \\ w_{21} & w_{22} & \dots & w_{2N} \\ w_{31} & w_{32} & \dots & w_{3N} \end{bmatrix} v_{w1} = \begin{bmatrix} w_{11} \\ w_{21} \\ w_{31} \end{bmatrix}$

$h1 = \begin{bmatrix} w_{11} \\ w_{21} \\ w_{31} \end{bmatrix}$

$y1 = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$

$y2 = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$

$y3 = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$

A simplest CBOW model, considering only a single-word context.

The word vector for the first word $w1$ is:

$J(w1|w2) = \frac{\exp(v_{w1})}{\sum_{j=1}^N \exp(v_{w_j})}$

If we define J as the loss function, then we have

$u_j = -\log \sum_{j=1}^N \exp(u_j) := E - u_j = v_{w_j}^T h$

Xin Rong, word2vec Parameter Learning Explained, arXiv:1411.2738, 2016

Word Embedding Learning: CBOW

Mikolov, Tomáš, Kai Chen, Greg Corrado, and Jeffrey Dean. "Efficient estimation of word representations in vector space." arXiv preprint arXiv:1301.3781 (2013).

INPUT → INFLECTION → OUTPUT

$x = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \\ 0 \end{bmatrix}_{V \times 1}$

$W_{1 \times N} = \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1N} \\ w_{21} & w_{22} & \dots & w_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ w_{V1} & w_{V2} & \dots & w_{VN} \end{bmatrix}_{V \times N}$

$h = \frac{1}{C} (x_1 + x_2 + \dots + x_C)$

$= \frac{1}{C} (w_{11}x_1 + w_{12}x_2 + \dots + w_{1N}x_N)^T$

When multiple context words are involved*

Input layer
Hidden layer
Output layer

$y_{11} \quad y_{12} \quad \dots \quad y_{1N}$
 $y_{21} \quad y_{22} \quad \dots \quad y_{2N}$
 \vdots
 $y_{V1} \quad y_{V2} \quad \dots \quad y_{VN}$

$V \text{-dim}$

* Xin Rong, word2vec Parameter Learning Explained, arXiv

If we define E as the loss function, then we have

$$\begin{aligned}
 E &= -\log p(w_{y1}, w_{y2}, \dots, w_{yC}|w_i) \\
 &= -\log \prod_{j=1}^C \frac{\exp(w_{yj})}{\sum_{k=1}^C \exp(w_{yk})} \\
 &= -\sum_{j=1}^C x_{yj} + C - \log \sum_{j=1}^C \exp(w_{yj})
 \end{aligned}$$

Clearly, this part is different from CBOW

- Given a current word w_i , predict a total of C words in the context.
- The output layer expands from a single word's output vector in CBOW to C word's output vectors (refer to the right diagram*).
- All output vectors share a weight matrix from the hidden layer to the output layer.

* [Xin Rong](#), word2vec Parameter Learning Explained, arXiv1411.2738, 2016



The optimization of model training

Word Embedding Learning: The training optimization of CBOW and Skip-Gram

Both CBOW and Skip-Gram can express weight matrix updates using the following two formulas:

$$w_{ij}^{(\text{new})} = w_{ij}^{(\text{old})} - \eta \cdot e_j \cdot h_i,$$

$$w_{ij}^{(\text{new})} = v_j^{(\text{old})} - \eta \cdot e_j \cdot h \quad \text{for } j = 1, 2, \dots, V.$$

For each column corresponding to W^j , it is of the same length as the word vectors.

- The model update during training
- The computational complexity of updating the original CBOW or Skip-Gram's weight matrices is $O(N \cdot V)$.
- High complexity, lengthy training time.
- Optimization of model updates
- Hierarchical Softmax
- Negative Sampling

Word Embedding Learning: Hierarchical Softmax training optimization

- All words are stored using Huffman tree encoding, where the encoding is based on the frequency of words in the corpus: the higher the frequency, the shorter the path, and words only appear at leaf nodes.
 - It can be proven that: the total number of internal nodes in the binary tree is $V-1$ (for V words).
- For each internal node, define an N -dimensional weight vector (corresponding to the previous v).
- The leaf nodes do not define vectors. The probability of outputting a leaf node is defined as:

$$p(w = w_i) = \prod_{j=1}^{\ell(w)-1} \sigma\left(\langle h(n(w,j+1)) = \text{ch}(n(w,j)) \rangle \cdot v_{n(w,j)}^\top \tilde{h}\right) \quad [x] = \begin{cases} 1 & \text{if } x \text{ is true;} \\ -1 & \text{otherwise.} \end{cases}$$

($\ell(w)$ is the path length of output word ' w ' in the tree (the number of nodes passed from the root to the leaf).
 $[x]$ is 1 if corresponding to the probability of choosing left, $[x] = -1$ is the probability of choosing right child).
 $h(n)$ The left internal node of internal node n .
 $\text{ch}(n)$ The right internal node of internal node n .
 $v_n^\top \tilde{h}$ The j th internal node or the path from the root node to word ' w '.
 $\sigma(x) = 1 / (1 + \exp(-x))$

Loss function: $E = -\log p(w = w_i)/v_i) = -\sum_{j=1}^{\ell(w)-1} \log \sigma(v_j^\top \tilde{h})$

Update the weight matrix: $v_j^{(\text{new})} = v_j^{(\text{old})} - \eta \left(\sigma(v_j^\top \tilde{h}) - t_j \right) \mathbf{h}$ if $t_j = 1$ if $[x] = 1$ and $t_j = 0$ otherwise.

Clearly, the number of weight vectors updated for each training sample depends on the length of the output word's path, not the size of V .

* [Xin Kong](#), word2vec Parameter Learning Explained, arXiv:1411.2738, 2016

Word Embedding Learning: Negative Sampling training optimization

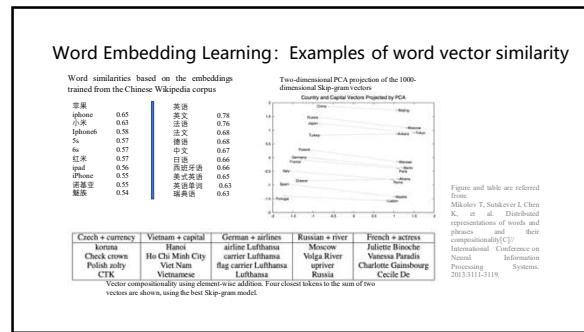
- To reduce the time complexity of weight matrix updates, only update the sampled weight vectors each time.
- The weight vectors corresponding to the target output word (the predicted word for CBOW or the context words for Skip-Gram) must be updated.
- Sampling determines which vectors are updated each time from the weight vectors corresponding to other words.

Loss function $E = -\log \sigma(\mathbf{v}_{w_j}^T \mathbf{h}) - \sum_{w_j \in W_{\text{neg}}} \log \sigma(-\mathbf{v}_{w_j}^T \mathbf{h})$

Weight update: $\mathbf{v}_{w_j}^{(\text{new})} = \mathbf{v}_{w_j}^{(\text{old})} - \eta \left(\sigma(\mathbf{v}_{w_j}^T \mathbf{h}) - t_j \right) \mathbf{h} \quad w_j \in \{w_O\} \cup W_{\text{neg}}$

Clearly, the number of weight vectors updated for each training sample depends on the number of samples taken, not on the size of V .

* [Xin Rong](#), word2vec Parameter Learning Explained, arXiv1411.2738, 2016

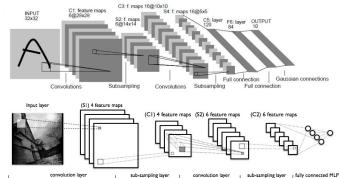


3

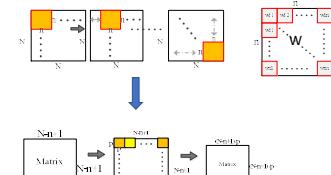
CNN Representation of Sentences and It's Applications

CNN-Convolution Neural Network

The recognition model, proposed by Japanese scholar Fukushima in 1984, can be regarded as the first implementation of a convolutional neural network.
Deep Convolutional Neural Networks (DCNN) have achieved tremendous success in fields such as images, speech, and text in recent years, thanks to the enhanced capability for large-scale training.



CNN-Convolution & Pooling



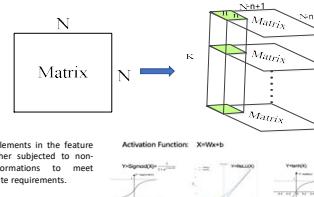
CNN-Convolution



*Image Source: http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution

CNN-Convolution

The number of kernels in each layer varies, leading to a different number of W matrices. If there are K kernels, there are K W matrices, resulting in K feature maps.



Usually, the elements in the feature map are further subjected to non-linear transformations to meet specific attribute requirements.

To the right are the image features obtained by applying different kernel transformations to the original image.*

Operation	Kernel	Image result
Identity	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (meanfilter)	$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}/9$	
Gaussian blur 3x3 (approximation)	$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}/16$	
Gaussian blur 5x5 (approximation)	$\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}/256$	
Unsharp masking 3x3 (approximation)	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}/16$	

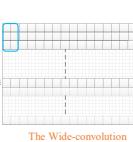
How to obtain the values of each element in the kernel matrix?

*Image Source: [https://en.wikipedia.org/wiki/Kernel_\(image_processing\)](https://en.wikipedia.org/wiki/Kernel_(image_processing))

CNN Representation of Sentences

Baodan Hu, Zhengdong Lu, Hang Li and Qingcai Chen, "Convolutional Neural Network Architectures for Matching Natural Language Sentences", Neural Information Processing Systems (NIPS), 2014

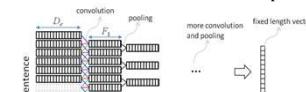
- Convolution of Word Vectors: Unlike image convolution, in NLP, there are mainly two ways in which convolution operates on word vector matrices.



CNN Representation of Sentences

Baodan Hu, Zhengdong Lu, Hang Li and Qingcai Chen, "Convolutional Neural Network Architectures for Matching Natural Language Sentences", Neural Information Processing Systems (NIPS), 2014

• Convolutional Neural Network-based Sentence Representation Model



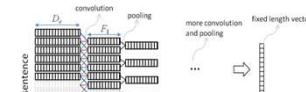
- Convolution Operation: Applied over adjacent rows of vectors, it's equivalent to obtaining a combined representation vector for adjacent words through convolution.

$$\mathbf{z}_i^{(\ell)} \stackrel{\text{def}}{=} \mathbf{z}_i^{(0)}(\mathbf{x}) = \sigma(\mathbf{W}^{(\ell)} \mathbf{z}_i^{(\ell-1)} + b^{(\ell)}).$$

$$\mathbf{z}_i^{(0)} = \mathbf{x}_{i:i+h_1-1} \stackrel{\text{def}}{=} [\mathbf{x}_i^T, \mathbf{x}_{i+1}^T, \dots, \mathbf{x}_{i+h_1-1}^T]^T$$

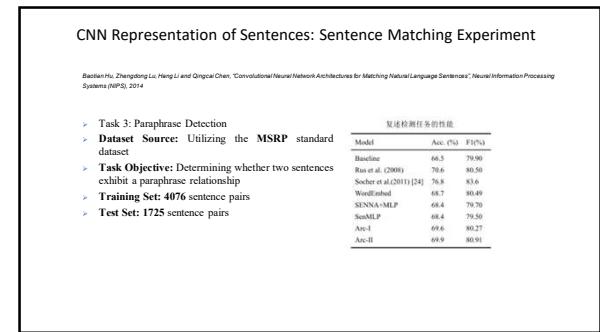
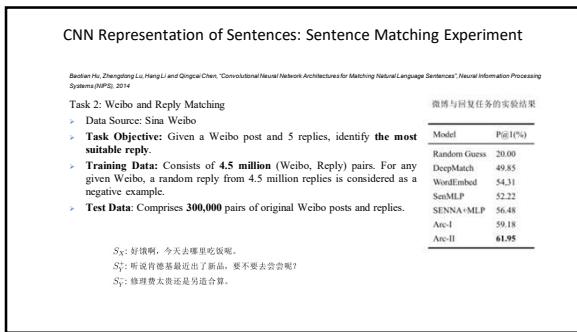
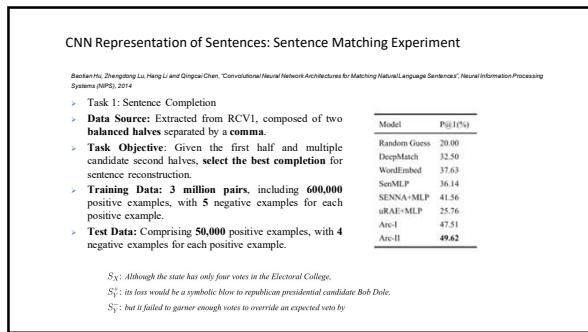
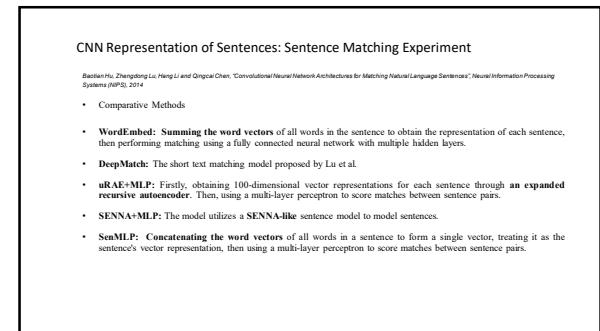
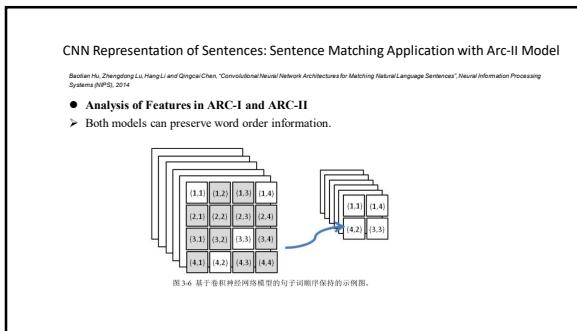
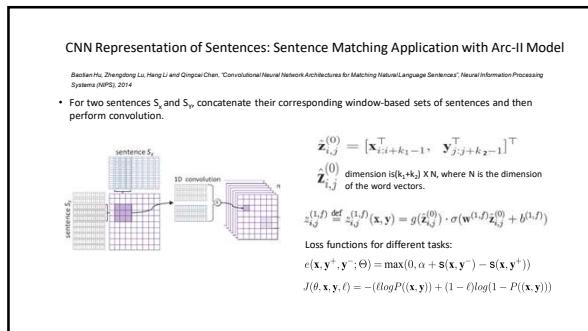
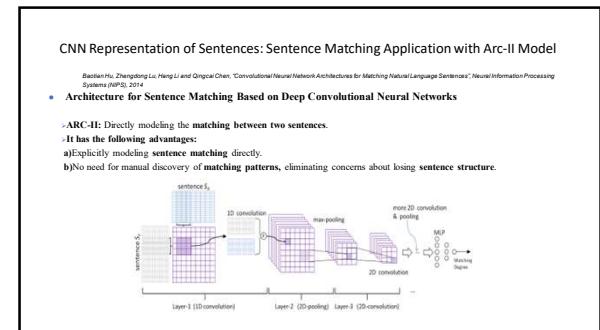
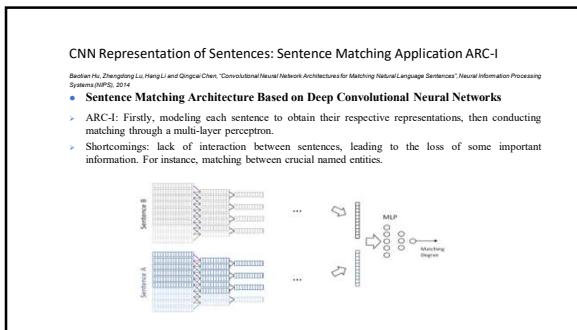
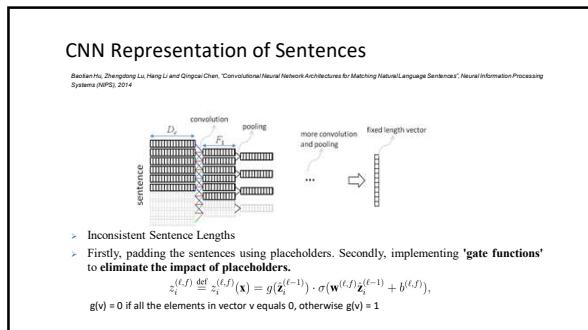
CNN Representation of Sentences

Baodan Hu, Zhengdong Lu, Hang Li and Qingcai Chen, "Convolutional Neural Network Architectures for Matching Natural Language Sentences", Neural Information Processing Systems (NIPS), 2014



- Max Pooling Operation: Applied over the representations obtained from different combinations derived by convolution of adjacent sliding windows.

$$z_i^{(\ell,f)} = \max(z_{2i-1}^{(\ell-1,f)}, z_{2i}^{(\ell-1,f)}), \quad \ell = 2, 4, \dots$$



Homework

- Using any public corpus to train a Skip-gram or CBOW word embeddings, and random select 10 words or phrase, output the top 10 most similar words for each of your selected words and their cosine similarity values, write a 1-page report for your training of the word embeddings.

Artificial Intelligence

第3部分 生成式人工智能方法

1. 语言的传统表示方法
2. 语言的神经网络表示
3. 生成式大语言模型

Buzhou Tang

Email: tangbz@hit.edu.cn
<http://icrc.hit.edu.cn>

Harbin Institute of Technology (Shenzhen)
 Intelligent Computation Research Center
 (ICRC, HITZ)

Part 1 Recurrent Neural Network

Oct 2024

Instructor: Buzhou Tang

Harbin Institute of Technology (Shenzhen) Intelligent Computation Research Center (HITZ ICRC)

RNN-Recurrent Neural Network

The earliest prototype of recurrent neural networks was the Hopfield neural network proposed by Hopfield in 1982.

John Hopfield

RNN-Recurrent Neural Network

Forward Neural Network

Recurrent Neural Network

time sequence representation

Unfolded over two time points

computation of an artificial neuron

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

$a_j = \sum_i w_{ij}x_i$

$h_t = \tanh(Wh_{t-1} + Ux_t + b)$

Ref. A Critical Review of Recurrent Neural Networks for Sequence Learning, Zachary C. Lipton et al. 2012

RNN-Recurrent Neural Network

It has a strong modeling capability for sequential data:

- The distributed representation of hidden states ensures the adequate retention of historical information.
- Multiple dynamic nonlinear operations can represent complex information processing.

“With enough neurons and time, RNNs can compute anything that can be computed by your computer”

--- Geoffrey E. Hinton

RNN-Recurrent Neural Network

Training

Back Propagation Through Time(BPTT): Assuming the use of the last state h_n as the representation of the sequence, with the loss function L , then the learning of model parameters is,

$$h_t = \tanh(Wh_{t-1} + Ux_t + b)$$

$$W = W - \alpha \frac{\partial L}{\partial W}$$

Challenges:

- Gradient Vanishing 梯度消失
- Gradient Explosion 梯度爆炸

Solution: LSTM(长短期记忆), GRU (门控循环神经元) etc

LSTM: Long short-term memory

(Long short-term memory)

Jürgen Schmidhuber

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735-1780, 1997

LSTM: Long short-term memory

Key idea:

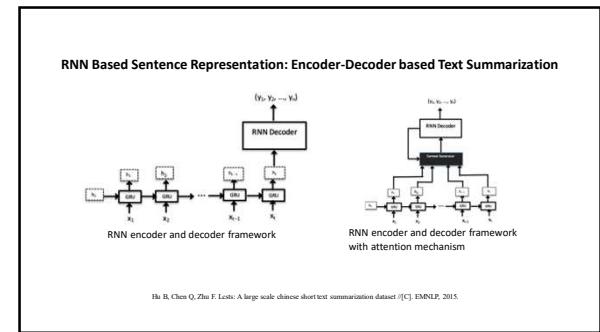
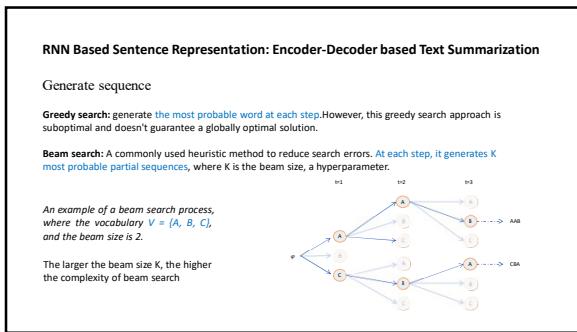
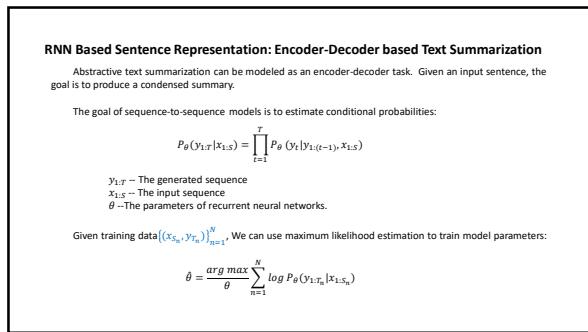
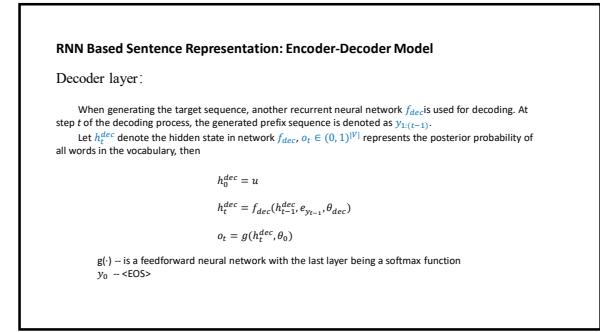
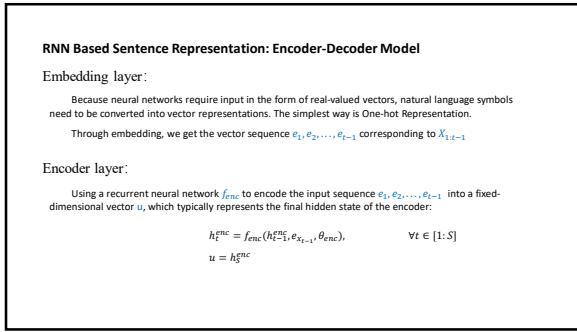
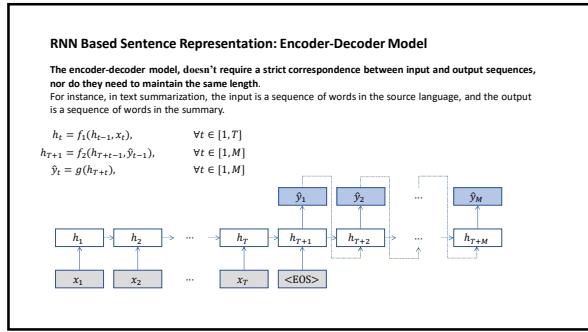
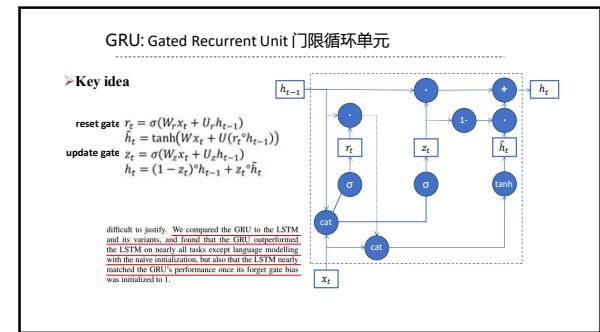
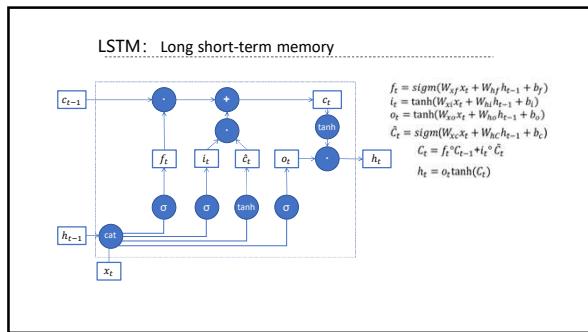
- Memory Unit: C_t
- Forget gate f_t , Input gate: i_t , Output gate o_t

i_t : Input Gate : The input gate is responsible for updating the cell state with new information.
 f_t : Forget Gate : The forget gate is responsible for deciding what information from the previous cell state should be thrown away or kept.
 c_t : Cell State : The cell state decides what the next external state h_t should be.
 h_t : Hidden State : represents the output of the LSTM cell at a given time step "t". It can also be considered as the cell state.
 C_t : Internal State : represents the output of the LSTM cell at a given time step "t". It can also be considered as the hidden state.
 C_t : Internal State : represents a temporary value that carries new information to update the LSTM cell's internal state.

LSTM: Long short-term memory

Legend:

- \oplus : vector concatenation
- σ : logistic function
- x_t : input at time step t
- \odot : element-wise vector multiplication
- $+$: vector addition



Example of generated summaries

Figure 7: An example of the generated summaries with UNKs.

Hu B, Chen Q, Zhu F. Lests: A large scale chinese shorttext summarization dataset // [C]. EMNLP, 2015.

Evaluation: ROUGE

The ROUGE (Recall-Oriented Understudy for Gisting Evaluation) algorithm was initially applied in the field of text summarization.

$$ROUGE - N(x) = \frac{\sum_{k=1}^K \sum_{w \in W} \min(c_w(x), c_w(s^{(k)}))}{\sum_{k=1}^K \sum_{w \in W} c_w(s^{(k)})}$$

$c_w(x)$ -- The frequency of the N-gram combination w appearing in the generated sequence x .
 $c_w(s^{(k)})$ -- The frequency of the N-gram combination w appearing in the reference sequence $s^{(k)}$.

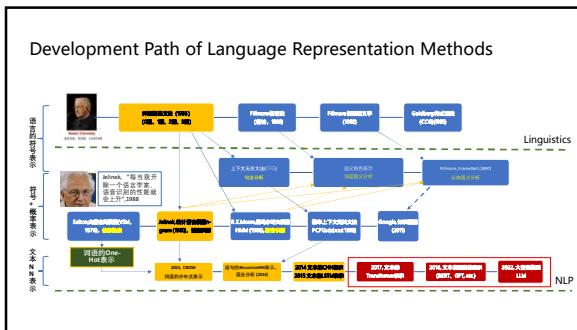
Artificial Intelligence

第3部分 生成式人工智能方法

- 语言的传统表示方法
- 语言的神经网络表示
- 生成式大语言模型**

Buzhou Tang
Email: tangbuzhou@hit.edu.cn
http://ccrc.hit.edu.cn

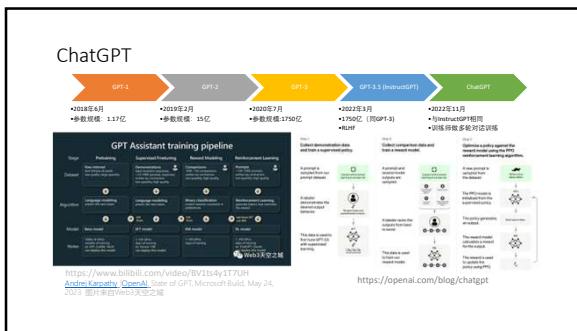
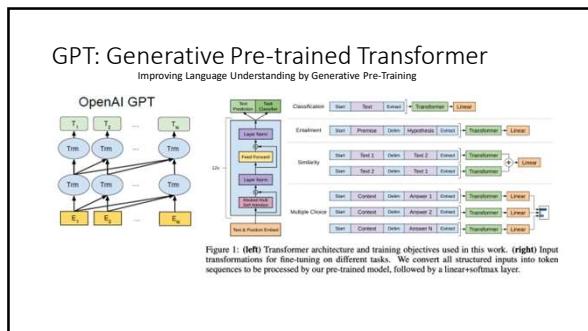
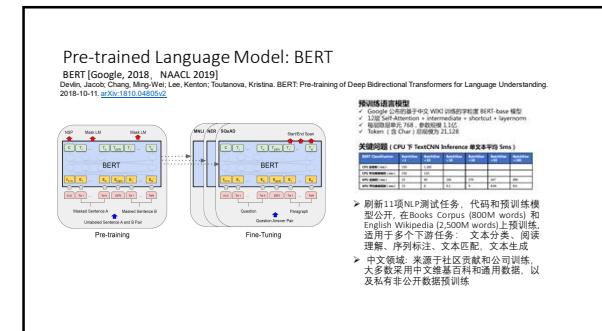
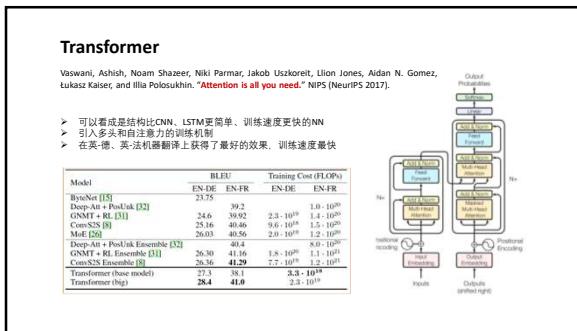
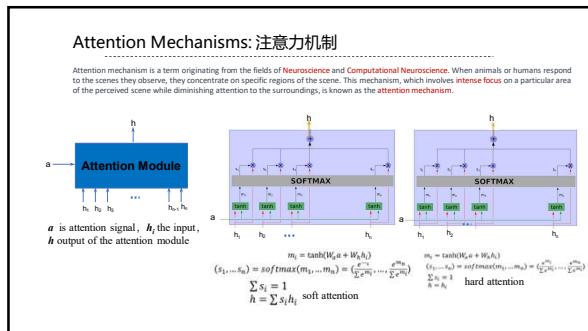
Harbin Institute of Technology (Shenzhen)
Intelligent Computation Research Center
(ICRC, HITZ)



Part 2 Large Language Models

Oct 2024
Instructor: Buzhou Tang

Harbin Institute of Technology (Shenzhen) Intelligent Computation Research Center (HITZ, ICRC)



Computing of Transformer

Following slides are created by Hung-Yi Lee
https://speech.ee.ntu.edu.tw/~tkagk/courses/ML_2019/Lecture/Transformer%20(v5).pptx

