UNIVERSITY OF TORONTO
SCHOOL OF CONTINUING STUDIES

3547 Intelligent Agents & Reinforcement Learning

Module 3: Logical Inference

Course Plan

Module Titles
Module 1 – Introduction to Intelligent Agents
Module 2 – Current Focus: Search
Module 3 – **Logical Inference**
Module 4 – Planning and Knowledge Representation
Module 5 – Probabilistic Reasoning
Module 6 – Intro to Reinforcement Learning and Finite Markov Decision Processes
Module 7 – Dynamic Programming and Monte Carlo Methods
Module 8 – Temporal Difference Learning
Module 9 – Function Approximation for RL
Module 10 – Deep Reinforcement Learning and Policy Gradient Methods
Module 11 – Introduction to Advanced DRL
Module 12 – Presentations (no content)

Learning Outcomes for this Module

- Make you familiar with the terminology of propositional logic
- Enable you to model problems as facts and rules
- Prepare you to formulate propositional constraint problems for solvers

Topics for this Module

- **3.1** Logical Agents
- **3.2** Language and Logic
- **3.3** Propositional Logic
- **3.4** Definite Clauses
- **3.5** Queries and Proofs
- **3.6** Reasoning with Contradiction
- **3.7** Resources and Wrap-up

Module 3 – Section 1

Logical Agents

What Kinds of Questions Does AI Struggle With?

- Adversarial images
- Researchers at the University of Maryland have compiled a set of 1,200 questions that usually stump conversational AI's
  - Paraphrasing
  - Distracting references
  - **Requiring logic, calculation or a chain of reasoning**

Logical Agents

- The story so far:
  - We have represented the world as a set of states
  - In a partially-observable environment, our agents could only model what it knows about the current state by listing all possible concrete states
  - But we have started to represent states as assignments to variables
- The next step:
  - Give our agents the ability to begin to reason i.e. use a **knowledge base** of facts and rules, and derive new facts from these and percepts
  - Challenges such as the Wumpus World can be solved more efficiently if the agent can reason about where the Wumpus, pits and gold are
  - This approach is much more scalable to larger environments than trying to work with concrete states

Knowledge Bases

- The basic idea:
  - Draw from the idea of thinking as tacitly using language
  - Maintain a **knowledge base** of facts and rules that perhaps begins with background knowledge but can evolve as the agent interacts with its environment
  - Use the formalisms of logic to create new sentences that we can be confident are true from sentences that we already know to be true
  - Use this knowledge to take better actions given the state we find the environment in

- **Axiom**: A fact that is taken as given
- **Inference**: Start with a fact and apply rules to create new facts

Computation vs. Inference/Deduction

- **Computation**: Start with an expression and apply fixed rules to produce a guaranteed result
- **Deduction**: Start with a conjecture, and with a fixed set of axioms and inference rules attempt to construct a proof of the conjecture

- **Logic Programming**: Using a fixed strategy for proof search so we can implement it as an algorithm

Module 3 – Section 2

Language and Logic

Declarative vs. Procedural Approaches

- Procedural: How to do it
  - Can be highly efficient if well-written
  - Hand-tuned optimizable
  - Often simple things require substantial work
  - Hard to get right
  - e.g. Python, Java
- Functional: Between Procedural and Declarative
  - Higher-level abstractions that provide safety and programming convenience
  - Less efficient
  - e.g. LISP, Haskell
- Declarative: What to do
  - Very high-level abstractions make common tasks easy
  - Built-in efficient general-purpose algorithms
  - Unusual tasks may be very difficult, computational infeasible, or not even possible
  - e.g. SQL, Datalog

Logics in General

- **Syntax**: Form of sentences of a representation language
- **Semantics**: Meanings of those sentences i.e. their truth with respect to each possible world
- **Possible World / Model**: A fixing of the truth value of all relevant sentences
- **Satisfaction**: If a sentence is true in a specific model, we say that the model satisfies (or is a model of) the sentence
- **Entailment**: The relation between two sentences such that one follows logically from another (symbolized by $\models$)

Desirable Properties of a Logic

- **Soundness**: The property of an inference algorithm such that it only derives entailed sentences
- **Completeness**: The property that it can derive any sentence that is entailed
- **Grounding**: Knowledge in the knowledge base corresponds to what is true in the real world
- **Efficiency**: We would like inferences to be fast and use reasonable amounts of memory

UNIVERSITY OF TORONTO
SCHOOL OF CONTINUING STUDIES

Module 3 – Section 3

Propositional Logic

Propositions

- Statements about the world can be thought of as constraints on what could be true
- These constraints can be described succinctly in either of two ways:
  - Extensionally: enumeration
  - Intensionally: rules
- Propositions are an intensional form
  - Often more compact and readable
  - Enable efficient reasoning and query responses
  - Often more efficient to alter as new information becomes available

Terminology for Propositional Logic

- **Proposition** (or **logical formula**): A sentence, written in a language, that has a truth value in a world. It's constructed from atomic propositions and logical connectives. It is either an atom or a compound proposition.
- **Atom** (or **atomic proposition** or **atomic formula**): A formula with no deeper propositional structure. We will use a word starting with a lower-case letter.

Compound Propositions

A **proposition** (or **logical formula**) is either an atomic proposition (or atom) or a **compound proposition** of the form:

Form	Read As	Formal Name
$\neg p$	not p	**negation** of p
$p \wedge q$	p and q	**conjunction** of p and q
$p \vee q$	p or q	**disjunction** of p and q
$p \rightarrow q$	p implies q	**implication** of q from p
$p \leftarrow q$	p if q	**implication** of p from q
$p \leftrightarrow q$	p if and only if q	**equivalence** of p and q

where p and q are propositions and the operators are called **logical connectives**.

Truth Table Defining the Logical Connectives

p	q	$\neg p$	$p \wedge q$	$p \vee q$	$p \leftarrow q$	$p \rightarrow q$	$p \leftrightarrow q$
true	*true*	*false*	*true*	*true*	*true*	*true*	*true*
true	*false*	*false*	*false*	*true*	*true*	*false*	*false*
false	*true*	*true*	*false*	*true*	*false*	*true*	*false*
false	*false*	*true*	*false*	*false*	*true*	*true*	*true*

Rules (if-then)

p: billy_is_wonderful	billy_is_a_dog	dogs_are_wonderful	q: billy_is_a_dog ∧ dogs_are_wonderful	$p \leftarrow q$
true	true	true	true	true
true	true	false	false	true
true	false	true	false	true
true	false	false	false	true
false	true	true	true	false
false	true	false	false	true
false	false	true	false	true
false	false	false	false	true

p: billy_is_wonderful
q: billy_is_a_dog ∧ dogs_are_wonderful

billy_is_wonderful ← billy_is_a_dog ∧ dogs_are_wonderful

Implication

p: billy_is_wonderful	billy_is_a_dog	dogs_are_wonderful	q: billy_is_a_dog ∧ dogs_are_wonderful	$q \rightarrow p$
true	true	true	true	true
true	true	false	false	true
true	false	true	false	true
true	false	false	false	true
false	true	true	true	false
false	true	false	false	true
false	false	true	false	true
false	false	false	false	true

$\boldsymbol{p}$: billy_is_wonderful
$\boldsymbol{q}$: billy_is_a_dog ∧ dogs_are_wonderful

billy_is_a_dog ∧ dogs_are_wonderful → billy_is_wonderful

Equivalence (iff)

p: billy_is_wonderful	billy_is_a_fog	dogs_are_wonderful	q: billy_is_a_dog ∧ dogs_are_wonderful	$p \leftrightarrow q$ $q \leftrightarrow p$
true	true	true	true	true
true	true	false	false	false
true	false	true	false	false
true	false	false	false	false
false	true	true	true	false
false	true	false	false	true
false	false	true	false	true
false	false	false	false	true

p: billy_is_wonderful
q: billy_is_a_dog ∧ dogs_are_wonderful

billy_is_wonderful ↔ billy_is_a_dog ∧ dogs_are_wonderful

Semantics of Propositional Calculus

- **Semantics** define the meaning of the sentences
- An **interpretation, possible world** or **model** is a function that maps atoms to true or false
- An interpretation that makes a sentence true is said to **satisfy** it
- An **intended interpretation** is an interpretation that connects the meaning of the symbols to the “real world”
- A sentence is **logically consistent** if it is true in all intended interpretations of interest

Example Interpretations

Atoms: building_burning, child_inside, elevator_operational

Interpretation 1

building_burning = true
child_inside = true
elevator_operational = false

In Interpretation 1:

building_burning ∧ child_inside is true
elevator_operational ∧ ¬child_inside is false
child_inside → elevator_operational is false

Interpretation 2

building_burning = false
child_inside = false
elevator_operational = false

In Interpretation 2:

building_burning ∧ child_inside is false
elevator_operational ∧ ¬childInside is false
child_inside → elevator_operational is true

Knowledge Bases

- A **knowledge base** is a set of propositions that are stated to be true
- An **axiom** is an element of a knowledge base
- The KB designer selects for the intended interpretation:
  - atoms to represent relevant propositions
  - meanings for the symbols (an **ontology**)
  - axioms (propositions that state what is true in that world)

Entailment

- **Entailment** is a semantic relation between a set of propositions (i.e. a knowledge base) and a proposition that follows logically
- Denoted KB $\models p$
- To be entailed, p must be true in all interpretations of KB
- Once the knowledge base is created, users can ask whether a new logical proposition is entailed by the knowledge base

Example Knowledge Base

house_on_fire = true

house_rear_entrance = false

robot_battery_low = true

robot_extinguisher_empty = false

fire_sensed_in_room = true

robot_extinguisher_enabled ← fire_sensed_in_room ∧ ¬robot_extinguisher_empty

robot_extinguisher_usable ← robot_extinguisher_enabled ∧ ¬ robot_battery_low

- Ask: robot_extinguisher_usable?

Module 3 – Section 4

Definite Clauses

Definite Clauses

- By restricting the forms of propositional clauses we can increase the efficiency of some reasoning tasks
- **Propositional Definite Clauses** is an unambiguous sublanguage of propositional calculus that enables more efficient solvers than full propositional logic
  - Atoms have the same form
  - Clauses have the form:
    - An atom
    - $h \leftarrow a_1 \wedge \cdots \wedge a_m$ where h and all of the a_i's are atoms
  - Atoms start with a lower case letter e.g. $robotBatteryLow$ which means $robotBatteryLow = true$
- Clauses in the "just an atom" form are called **facts**
- Clauses in the "if" form are called **rules** and the h is called the **head** of the rule

Definite Clauses Example

$robotBatteryLow$
$robotExtinguisherEnabled \leftarrow fireSensedInRoom \wedge robotExtinguisherNotEmpty$
$robotExtinguisherUsable \leftarrow robotExtinguisherEnabled \wedge robotBatteryNotLow$

Not definite clauses:

$\neg robotBatteryLow$
$robotExtinguisherEnabled \wedge robotBatteryNotLow$
$RobotExtinguisherEnabled$
$robotExtinguisherUsable \leftarrow robotExtinguisherEnabled \vee robotHasHose$

Module 3 – Section 5

Queries and Proofs

Queries

- Inference engines can answer questions about the knowledge base
- Queries have the form $ask\ a.$ where a is an atom or a conjunction of atoms
- The inference engine will respond with either:
  - yes if the body is a logical consequence of the knowledge base
  - no if not
- Important! no does not mean $false$; it means $cannot\ be\ determined$

Proofs

- How does an inference engine work?
- Two approaches:
  - Enumeration
  - Theorem Proving
    - Bottom-Up aka **forward chaining**: derive everything possible from what we already know using modus ponens
    - Top-Down aka **backward chaining** using **SLD resolution**

Judgments and Proofs

- **Judgment**: An assertion (usually that something is true)
- **Inference rules**:

$$\frac{J1 \ldots Jn}{J}\,R$$

e.g. $$\frac{HaveCar\ HaveGas\ HaveMap}{GoodToGo}$$

Proof Procedures

- A **proof** is a mechanically-derivable demonstration that a proposition logically follows from a knowledge base
- A **theorem** is a provable proposition
- A **proof procedure** is an algorithm for deriving consequences of a knowledge base
- Given a proof procedure, KB $\vdash g$ means g can be **proved** or **derived** from the knowledge base KB

Forward Chaining and Modus Ponens

- **Modus Ponens**: If $h \leftarrow a_1 \wedge \cdots \wedge a_m$ is a definite clause in the knowledge base, and each a_i has been derived, then h can be derived
- Forward chaining is:
  - Sound
  - Complete
  - Linear in the size of the knowledge base

Forward Chaining Example

$a \leftarrow b \wedge c.$
$b \leftarrow d \wedge e.$

$c \leftarrow e.$

$d.$

$e.$

$f \leftarrow a \wedge g.$

One possible solving trace:

$\{\}$
$\{d\}$
$\{e, d\}$
$\{c, e, d\}$
$\{b, c, e, d\}$
$\{a, b, c, e, d\}$

Backward Chaining

- If the query is $ask\ q_1 \wedge \cdots \wedge q_m$ we want to see if we can prove $yes \leftarrow q_1 \wedge \cdots \wedge q_m$
- We create an **answer clause** $ask\ a_1 \wedge \cdots \wedge a_m$ with the a's initially set to the q's
- We select one of the a's, say a_i then look for a rule in the KB with a_i as its head and substitute it for a_i
- Because $true \wedge anything$ is $true$ we can drop any subclause once we find an atom confirming it is $true$
- We proceed until the body of the query becomes empty, in which case we return yes or there are no rules in the KB with a head matching any of the atoms in the answer clause

Backward Chaining Example

$a \leftarrow b \wedge c.$

$b \leftarrow d \wedge e.$

$c \leftarrow e.$

$d.$

$e.$

$f \leftarrow a \wedge g.$

One possible solving trace for the query $ask\ a$:

$yes \leftarrow a$

$yes \leftarrow b \wedge c$

$yes \leftarrow d \wedge e \wedge c$

$yes \leftarrow e \wedge c$

$yes \leftarrow c$

$yes \leftarrow e$

$yes \leftarrow$

Explaining Queries

- **How** was that answer proved?
  - The answer is the rule that led immediately to the proof
  - If asked again, you can typically move to how its terms were proved, etc.
- **Why** did the system ask me that question?
  - Because this atom was needed to determine if the body of this rule is true
  - Provides assurance that the system is working properly
- **Why Not** i.e. wasn't this provable?
  - Which rule broke the chain of reasoning?

Module 3 – Section 6

Reasoning with Contradiction

Horn Clauses

- Definite clauses do not allow a contradiction to be stated
- We can expand definite clauses to allow a clause of the form $false \leftarrow a_1 \wedge \cdots \wedge a_k$ where the a_i are atoms and $false$ is a special atom that is false in all interpretations
- These additional clauses are sometimes called **integrity constraints**
- **Horn Clause**: Either a definite clause or an integrity constraint

Horn Clause Examples

$false \leftarrow a \wedge b.$
$a \leftarrow c.$
$b \leftarrow c.$

$false \leftarrow a \wedge b.$
$a \leftarrow c.$
$b \leftarrow d.$
$b \leftarrow e.$

Open and Closed Worlds

- **Closed-World (or Complete Knowledge) Assumption**: Anything unknown is considered to be false
  - Allows us to use negation to derive $not\ p$
- **Open-World Assumption**: Anything unknown is considered as such

Monotonic and Non-Monotonic Reasoning

- **Monotonic Reasoning**: Adding new knowledge to a knowledge base does not invalidate previously known facts or derivable propositions
  - Avoids needing to recompute what is known when new facts arrive
- **Non-Monotonic Reasoning**: New information can overrule previously believed facts or rules
  - More realistic
  - Allows us to create default facts in the knowledge base that can be overridden by new ones
  - The logic of definite clauses with negation is non-monotonic

Many Logics!

- First and higher Orders
- Intuitionistic
- Modal
- Linear
- Temporal
- Epistemic
- …

UNIVERSITY OF TORONTO
SCHOOL OF CONTINUING STUDIES

Module 3 – Section 7

Resources and Wrap-up

Resources

- Eric Wallace, Pedro Rodriguez, Shi Feng, Ikuya Yamada, Jordan Boyd-Graber. **Trick Me If You Can: Human-in-the-Loop Generation of Adversarial Examples for Question Answering**. *Transactions of the Association for Computational Linguistics*, 2019; 7: 387
  DOI: 10.1162/tacl_a_00279
- Magnus, Button, et. al. *forall* x: An Introduction to Formal Logic. https://forallx.openlogicprojct.org

Summary

- Philosophers have long considered language to be intrinsic to thought
- Propositional logic reliably formalizes deduction
- Problems can be expressed as facts, rules and variables to be solved for rather than as procedures

Next Week

- We'll will extend propositional logic to first order logic and give you the tools to build knowledge-based agents
- We'll see how logic can be extended to model the passage of time and how to enable an intelligent agent to plan a sequence of strategic moves
- We'll learn more Prolog

Any questions?

UNIVERSITY OF TORONTO
SCHOOL OF CONTINUING STUDIES

Thank You

Thank you for choosing the University of Toronto School of Continuing Studies