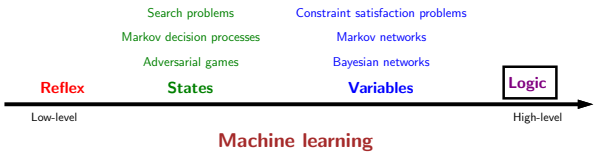
Logic: overview

Question

If $X_1 + X_2 = 10$ and $X_1 - X_2 = 4$, what is X_1?

- Think about how you solved this problem. You could treat it as a CSP with variables X_1 and X_2, and search through the set of candidate solutions, checking the constraints.
- However, more likely, you just added the two equations, divided both sides by 2 to easily find out that $X_1 = 7$. This is the power of **logical inference**, where we apply a set of truth-preserving rules to arrive at the answer. This is in contrast to what is called **model checking** (for reasons that will become clear), which tries to directly find assignments.
- We'll see that logical inference allows you to perform very powerful manipulations in a very compact way. This allows us to vastly increase the representational power of our models.

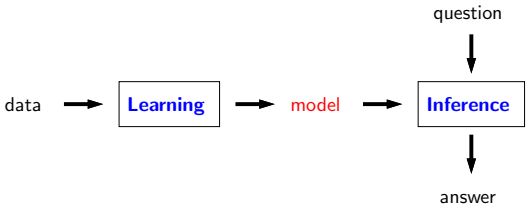
Course plan



Machine learning

- We are at the last stage of our journey through the AI topics of this course: logic. Before launching in, let's take a moment to reflect.

Taking a step back



Examples: search problems, MDPs, games, CSPs, Bayesian networks

- For each topic (e.g., MDPs) that we've studied, we followed the modeling-inference-learning paradigm: We take some data, feed it into a learning algorithm to produce a model with tuned parameters. Then we take this model and use it to perform inference (turning questions into answers).
- For search problems, the question is "what is the minimum cost path?" Inference algorithms such as DFS, UCS or A* produced the minimum cost path. Learning algorithms such as the structured Perceptron filled in the action costs based on data (minimum cost paths).
- For MDPs and games, the question is "what is the maximum value policy?" Inference algorithms such as value iteration or minimax produced this. Learning algorithms such as Q-learning or TD learning allow you to work when we don't know the transitions and rewards.
- For CSPs, the question is "what is the maximum weight assignment?" Inference algorithms such as backtracking search, beam search, or variable elimination find such an assignment. We did not discuss learning algorithms here, but something similar to the structured Perceptron works.
- For Bayesian networks, the question is "what is the probability of a query given evidence?" Inference algorithms such as Gibbs sampling and particle filtering compute these probabilistic inference queries. Learning: if we don't know the local conditional distributions, we can learn them using maximum likelihood.
- We can think of learning as induction, where we need to generalize, and inference as deduction, where it's about computing the best predicted answer under the model.

Modeling paradigms

State-based models: search problems, MDPs, games

Applications: route finding, game playing, etc.
Think in terms of **states, actions, and costs**

Variable-based models: CSPs, Bayesian networks

Applications: scheduling, tracking, medical diagnosis, etc.
Think in terms of **variables and factors**

Logic-based models: propositional logic, first-order logic

Applications: theorem proving, verification, reasoning
Think in terms of **logical formulas and inference rules**

- Each topic corresponded to a modeling paradigm. The way the modeling paradigm is set up influences the way we approach a problem.
- In state-based models, we thought about inference as finding minimum cost paths in a graph. This leads us to think in terms of states, actions, and costs.
- In variable-based models, we thought about inference as finding maximum weight assignments or computing conditional probabilities. There we thought about variables and factors.
- Now, we will talk about logic-based models, where inference is applying a set of rules. For these models, we will think in terms of logical formulas and inference rules.

A historical note

- Logic was dominant paradigm in AI before 1990s

- Problem 1: deterministic, didn't handle **uncertainty** (probability addresses this)
- Problem 2: rule-based, didn't allow fine tuning from **data** (machine learning addresses this)
- Strength: provides **expressiveness** in a compact way

- Historically, in AI, logic was the dominant paradigm before the 1990s, but this tradition fell out of favor with the rise of probability and machine learning.
- There were two reasons for this: First, logic as an inference mechanism was brittle and did not handle uncertainty, whereas probability offered a coherent framework for dealing with uncertainty.
- Second, people built rule-based systems which were tedious and did not scale up, whereas machine learning automated much of the fine-tuning of a system by using data.
- However, there is one strength of logic which has not quite yet been recouped by existing probability and machine learning methods, and that is the expressivity of the model.

Motivation: smart personal assistant

- How can we motivate logic-based models? We will take a little bit of a detour and think about an AI grand challenge: building smart personal assistants.
- Today, we have systems like Apple's Siri, Microsoft's Cortana, Amazon's Alexa, and Google Assistant.

Motivation: smart personal assistant

Tell information

Ask questions

Use natural language!

[demo: `python nli.py`]

Need to:

- Digest **heterogenous** information
- Reason **deeply** with that information

- We would like to have more intelligent assistants such as Data from Star Trek. What is the functionality that's missing in between?
- At an abstract level, one fundamental thing a good personal assistant should be able to do is to take in information from people and be able to answer questions that require drawing inferences from the facts.
- In some sense, telling the system information is like machine learning, but it feels like a very different form of learning than seeing 10M images and their labels or 10M sentences and their translations. The type of information we get here is both more heterogenous, more abstract, and the expectation is that we process it more deeply (we don't want to have to tell our personal assistant 100 times that we prefer morning meetings).
- And how do we interact with our personal assistants? Let's use natural language, the very tool that was built for communication!

Natural language

Example:

- A **dime** is better than a **nickel**.
- A **nickel** is better than a **penny**.
- Therefore, a **dime** is better than a **penny**.

Example:

- A **penny** is better than **nothing**.
- **Nothing** is better than **world peace**.
- Therefore, a **penny** is better than **world peace**???

Natural language is slippery...

- But natural language is tricky, because it is replete with ambiguities and vagueness. And drawing inferences using natural languages can be quite slippery. Of course, some concepts are genuinely vague and slippery, and natural language is as good as it gets, but that still leaves open the question of how a computer would handle those cases.

Language

Language *is a mechanism for expression.*

Natural languages (informal):

English: *Two divides even numbers.*

German: *Zwei dividieren geraden zahlen.*

Programming languages (formal):

Python: `def even(x): return x % 2 == 0`

C++: `bool even(int x) { return x % 2 == 0; }`

Logical languages (formal):

First-order-logic: $\forall x.\text{Even}(x) \rightarrow \text{Divides}(x, 2)$

- Let's think about language a bit deeply. What does it really buy you? Primarily, language is this wonderful human creation that allows us to express and communicate complex ideas and thoughts.
- We have mostly been talking about natural languages such as English and German. But as you all know, there are programming languages as well, which allow one to express computation formally so that a computer can understand it.
- This lecture is mostly about logical languages such as propositional logic and first-order logic. These are formal languages, but are a more suitable way of capturing declarative knowledge rather than concrete procedures, and are better connected with natural language.

Two goals of a logic language

- **Represent** knowledge about the world

- **Reason** with that knowledge

- Some of you already know about logic, but it's important to keep the AI goal in mind: We want to use it to represent knowledge, and we want to be able to reason (or do inference) with that knowledge.
- Finally, we need to keep in mind that our goal is to get computers to use logic automatically, not for you to do it. This means that we need to think very mechanistically.

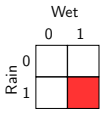
Ingredients of a logic

Syntax: defines a set of valid **formulas** (Formulas)

Example: Rain $\wedge$ Wet

Semantics: for each formula, specify a set of **models** (assignments / configurations of the world)

Example:



Inference rules: given f, what new formulas g can be added that are guaranteed to follow ($\frac{f}{g}$)?

Example: from Rain $\wedge$ Wet, derive Rain

- The **syntax** defines a set of valid formulas, which are things which are grammatical to say in the language.
- **Semantics** usually doesn't receive much attention if you have a casual exposure to logic, but this is really the important piece that makes logic rigorous. Formally, semantics specifies the meaning of a formula, which in our setting is a set of configurations of the world in which the formula holds. This is what we care about in the end.
- But in order to get there, it's helpful to operate directly on the syntax using a set of **inference rules**. For example, if I tell you that it's raining and wet, then you should be able to conclude that it is also raining (obviously) without even explicitly mentioning semantics. Most of the time when people do logic casually, they are really just applying inference rules.

Syntax versus semantics

Syntax: what are valid expressions in the language?

Semantics: what do these expressions mean?

Different syntax, same semantics (5):

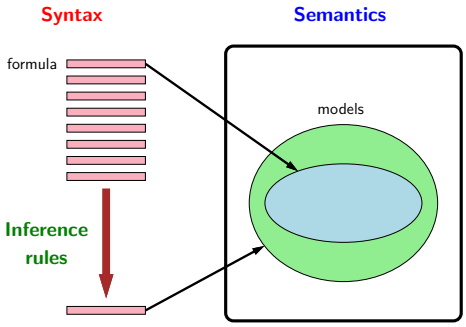
$$2 + 3 \quad \Leftrightarrow \quad 3 + 2$$

Same syntax, different semantics (1 versus 1.5):

$$3 \ / \ 2 \text{ (Python 2.7)} \quad \not\Leftrightarrow \quad 3 \ / \ 2 \text{ (Python 3)}$$

- Just to hammer in the point that syntax and semantics are different, consider two examples from programming languages.
- First, the formula 2 + 3 and 3 + 2 are superficially different (a syntactic notion), but they have the same semantics (5).
- Second, the formula 3 / 2 means something different depending on which language. In Python 2.7, the semantics is 1 (integer division), and in Python 3 the semantics is 1.5 (floating point division).

Propositional logic



Logics

- **Propositional logic with only Horn clauses**
- **Propositional logic**
- Modal logic
- **First-order logic with only Horn clauses**
- **First-order logic**
- Second-order logic
- ...

Key idea: tradeoff

Balance **expressivity** and **computational efficiency**.

- There are many different logical languages, just like there are programming languages. Whereas most programming languages have the expressive power (all Turing complete), logical languages exhibit a larger spectrum of expressivity.
- The bolded items are the ones we will discuss in this class.

Roadmap

Modeling	Inference
Propositional Logic Syntax	Inference Rules
Propositional Logic Semantics	Propositional modus ponens
First-order Logic	Propositional resolution
	First-order modus ponens
	First-order resolution

- Here are the rest of the modules under the logic unit.
- We will start by talking about the core elemenst of logics: syntax, semantics, and inference rules. We will start by defining syntax and semantics for propositional logic.
- We will then discuss a set of inference rules for propostional logic including modus ponens and resolution. We will discuss soundness and correctness of these inference algorithms.
- We then describe a more expressive logic, i.e., first order logic. We will go over its syntax and semantics, and then extend the notions of modus ponens and resolution to first order logic using unification and substitution.