CSE 513T Theory of Artificial Intelligence & Machine Learning Oct 13, 2016

Lecture 14

Lecturer: Brendan Juba *Scribe: Shiying Ni, Editor: Alex Durgin*

Today's Topics

- Proof System - formally
- Chaining
- Resolution
- Complexity Measures for Resolution

1 Proof System

1.1 Overview

In analogy to learning, there are inherent trade-offs between expressive power and the tractability of reasoning in proof systems. We are interested in which proof systems have efficient algorithms for reasoning. Conversely, which proof systems capture the abilities of efficient reasoning algorithms? We compare the abilities of algorithms by comparing their proof systems.

1.2 Definition

Definition: A proof system is given by a class of representations $\mathcal{F}$ (lines of the proof) and a relation

$$R \subseteq \mathcal{F} \times \mathcal{F}^* (\text{where } \mathcal{F}^* \text{denotes the sequences of elements of } \mathcal{F} \text{ of finite length})$$

capturing the "legal steps":
A proof of $\varphi \in \mathcal{F}$ from premises $\psi_1, ..., \psi_k \in \mathcal{F}$ is given by a sequence $\pi = \pi_1, \pi_2, ..., \pi_\ell$ where each $\pi_i \in \mathcal{F}$ is s.t. either (a) $\pi_i \in \{\psi_1, ..., \psi_k\}$ or (b) there is a subsequence $\pi'_1, ..., \pi'_k$, of $\pi_1, \pi_2, ..., \pi_{i-1}$ s.t. $(\pi_i, \pi'_1, ..., \pi'_k) \in R$. And finally $\pi_\ell = \varphi$.
Let ε denote the empty sequence in $\mathcal{F}^*$. When $(\pi_i, \varepsilon) \in R$, we say π_i is an axiom of the proof system. Otherwise, we say that it follows from a rule of inference.
Our proof system will generally be sound, that is if $(\pi_i, \pi'_1, ..., \pi'_k) \in R$, then $\{\pi'_1, ..., \pi'_k\} \models \pi_i$
In particular, the axioms must satisfy $\varnothing \models \pi_i$ (written $\models \pi_i$) i.e. $\mathcal{X} = \{x : \pi_i(x)\}$ i.e axioms are tautologies.
It follows that if we have a proof of φ from premises $\psi_1, ..., \psi_k$ in a sound proof system, then

$$\{\psi_1, ..., \psi_k\} \models \varphi$$

2 Chaining ("forward chaining")

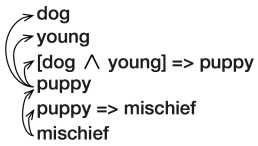
In this example of a proof system $\mathcal{F}$ consists of representations of the form $[l_1 \wedge ... \wedge l_r] \to l^* \quad (r \geq 0)$ where each of $l_1, ..., l_r$ and l^* are literals. If $r = 0$, then $l^* \in \mathcal{F}$ is a "fact", otherwise $[l_1 \wedge ... \wedge l_r] \to l^*$ is called a "rule", with body $l_1, ...l_r$ and head l^*. The only rule of ineference, given the rule $[l_1 \wedge ... \wedge l_r] \to l^*$ and the facts $l_1, ...l_r$, we can conclude the fact l^*.

2.1 Examples

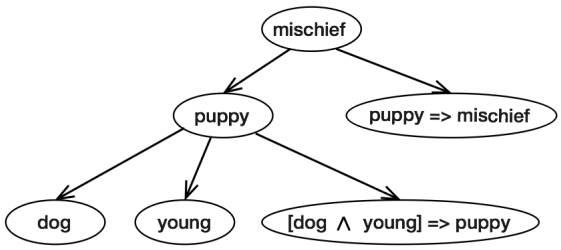
(1) Let $dog \to furry\ animal$ be a rule, and dog a fact in our chaining proof system. We can infer $furry\ animal$ from these two representations.

(2) Similarly, if we have the rule $[dog \wedge young] \to puppy$ and the facts $dog, young$. We can infer the fact "$puppy$".

(3) Let's add $puppy \to mischief$ to example (2). Then a chaining proof of $mischief$ from the above rules, facts is: $\pi =$



Given a proof π, there is a natural graph associated with it. Each π_i has an edge to each earlier π'_j for which the tuple $(\pi_i, \pi'_1, ...\pi'_k) \in R$ was used to infer π_i (as seen in our proof above, each arrow corresponds to such an edge).
In this example, the graph is:



Notice: any embedding of this graph into a sequence that respects the direction of the edges is also a proof of mischief from $\{dog, young, [dog \wedge young] \to puppy, puppy \to mischief\}$
An important syntactic restriction of a proof system is when this graph is a tree (equivalently, each line is used only once to reach the conclusion). We then say that the proof is treelike.

3 Resolution

$\mathcal{F}$ consists of disjunctions (ORs of literals, aka "clauses"). Resolution has a single rule of inference rule called "cut", given clauses C_i and C_j of the form

$$\begin{aligned} C_i &= x \vee C'_i \\ C_j &= \neg x \vee C'_j \end{aligned}$$

we can infer $C'_i \vee C'_j$.
Note:
if $x = 1$, $\neg x$ is false. So if $C_j(x) = 1$, it has to be that $C'_j(x) = 1$
Similarly, if $x = 0$, x is false. $C_i(x) = 1 \to C'_i(x) = 1$
So either way $C'_i \vee C'_j$ is true. Hence, cut is sound.

3.1 How to use

We typically use resolution to obtain proofs of more complex (DNF) formulas by contradiction. Suppose we have KB consisting of clauses $C_1, ..., C_k$ that we assume to be consistent, i.e. there is some $x \in \mathcal{X}$

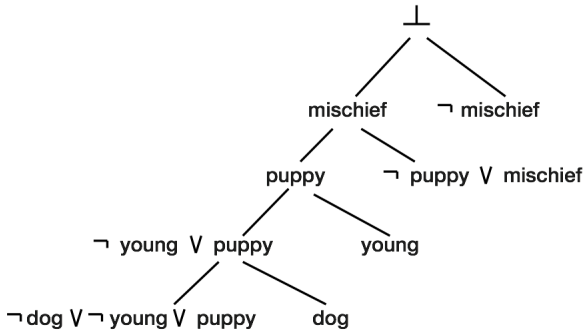
from which $C_i(x) \wedge ... \wedge C_j(x) = 1$, and we wish to prove that this KB entails a DNF ϕ. We do this by showing that $\neg\phi \wedge C_i \wedge ... \wedge C_k$ can't be satisfied by giving a resolution proof of the empty clauses "$\perp$", that we take by convention to be unsatisfiable.

Observe that the only way to derive $\perp$ is from the pair of formula x and $\neg x$, so the formula should imply both x and $\neg x$, this is a contradiction. In particular, since the cut rule only allows us to derive clauses that are satisfied by the same $x \in \mathcal{X}$ as the input clauses ($\forall x \in \mathcal{X}$ $((x_i \vee C_i') \wedge (\neg x_i \vee C_j'))(x) = 1 \implies (C_i' \vee C_j')(x) = 1$) , and the cut rule is our only rule, we find that there can't be any such common, satisfying x if we derive $\perp$. So, whenever x satisfies $C_i \wedge ... \wedge C_k$, it cannot satisfy $\neg\phi$. i.e. must satisfy ϕ so this proves $\{C_1, .., C_k\} \models \phi$.

We use proof by contradiction because it gives us a standard form for resolution proofs, their graphs always have the empty clauses at the root.

3.2 Example

Recall $A \to B$ is logically equivalent to $\neg A \vee B$, so chaining rules can be written as clauses. We prove *mischief* from our KB, given in example (3) above, as follows:



Note: It's a treelike proof.

Some algorithmic issues:
We know that reasoning is NP-complete if KB can contain all possible ORs of size ≥ 3. It turns out that resolution is complete (for clause consequences or for DNF consequences if used for proof by contradiction - we'll see next time).

So deciding whether or not there exists any resolution proof is also NP-complete. The natural first weakening of our ambitions: what about algorithms that run in time polynomial in the number of attributes/KB size/size of the unknown resolution proof? Open question - We don't know of such an algorithm.

Then, what about algorithms for reasoning that run in time polynomial in the size of an unknown treelike resolution proof? Also open (But better algorithms exist for this than the first).

4 Complexity Measures for Resolution

(1) The width of a resolution proof is the max number of literals appearing in any clauses of the proof. Our previous example (3.2) has width of 3.

(2) **Clause Space Complexity:**
We extend our notion of proof to include at each step a set of representations of $\mathcal{F}$ consisting of earlier lines of the proof.

Consider the step where we infer some π_i (and its associated set). The representations $\pi'_1, ...\pi'_k$ used to infer π_i using R must be in the set ("on the black board already") and π_i must be added to the "black board"(to the set associated with the next line). At any point, we can also drop representations from the set("erasing them from the black board"). The clause space of such a proof is the max number of representations that appear simultaneously on the black board during the proof (the number of representations in any of these associated sets). Our example can be written using clause space 2.

Notice: The condition that the proof is treelike is equivalent to being able to erase a represenation immediately after we have used it for an inference.