CSE 513T Theory of Artificial Intelligence & Machine Learning December 1, 2016

Lecture 26

Lecturer: Brendan Juba *Scribe: Kiron Sukesan*

Agenda

1. Learning and Planning analysis part 2- "completeness"
2. Analysis Part 3 - Running Time
3. Using Negation as failure in KBs for integrated learning and planning.

Find DT Algorithm

FindDT $(G, KB, k, t, \Pi, H, \mu, \epsilon, \delta, \gamma)$

If Π takes more actions than the horizon of H
  Return FAIL
Put $H^t \leftarrow$ histories $(\rho_t^{(j)}, a_{t-1}^{(j)}, \rho_{t-1}^{j}, .., a_1^{(j)}, \rho_1^{(j)}) \in H$
  with weight $w^{(j)} = \Pi_{i=1}^{t-1} w_i^{(j)}$ for histories j with $w^{(j)} \leq \frac{4}{\mu}$
If reasoning with weighted ex's on H^t for the query $\neg\Pi \cap G$ on KB returns $\tilde{p} \geq 1 - \epsilon$ **Then**
  Return Π
If $k > 0$ for each literal l such that neither $l^{(t)}$ nor $\neg l^{(t)}$ appear on Π **then**
  Put $\Pi_1 \leftarrow \Pi_0 \rightarrow l \rightarrow 1$
  Put $\Pi_1 \leftarrow$ Find-DT$(G, K, B, k-1, \Pi_l, H, ...)$
  If $\Pi_1 \neq FAIL$ **then**
    $\Pi_0 \leftarrow$ Find-DT$(G, K, B, k, \Pi_{\neg l}, H, ...)$
    If $\Pi_0 \neq FAIL$ **Then**
      Return policy $[l \rightarrow \Pi_1 if l = 1, l \rightarrow \Pi_0 if l = 0]$
    else
      Return **FAIL**
    for all action $a \in A$ such that $h^{(j)} \in H, \alpha_{a,t}^{(j)} = 1 \geq \frac{\mu}{2} m$ **do**
      Put $\Pi_\alpha \leftarrow \Pi \rightarrow \alpha \rightarrow$
      Put $H_\alpha \leftarrow h^{(j)} \leftarrow H, \alpha_{a,t}^{(j)} = 1$
      Put $\Pi'_\alpha \leftarrow Find - DT(G, K, B, k-1, \Pi_{\neg l}, H, ...)$
      If $\Pi' \neq FAIL$ **then**
        Return policy $\alpha \rightarrow \Pi'_\alpha$
      Return FAIL

Theorem 1 *Given* $m > \frac{8}{\mu^2 \gamma}(2nTln(3) + ln(\frac{4T|A|}{\mu}) + ln(\frac{3}{\delta}))$ *example histories of a reference policy* Π_0*, the algorithm runs in time* $O(m|KB|(4nT)^{2k+2})$*. It either returns 'FAIL' or a rank k T-Horizon decision tree policy. If there is a rank k Horizon decision tree policy that only takes* μ*-typical actions on any branch (w.r.t.* Π_0*) or (2). For each branch there exists a* $(1 - \epsilon + \delta)$*-testable ("implicit KB") CNF* φ *s.t. there is a space-2 tree like resolution proof of G from KB and* φ *and the literals asserted on the branch, then w.p.*$\geq 1 - \delta$*, the algorithm returns a policy that achieves G with probability* $1 - O((nT)^t(\epsilon + \gamma))$

last time we saw..
w.p. $1 - \frac{2}{3}\delta$

-Each sequence of actions appears a number of times in the histories that is within an addition $\frac{\mu}{t}$ of its true probability of being choosen by Π_0

-The algorithm either returns 'FAIL' or returns a rank-k Horizon decision tree that achieves G w.p. $1 - O((nT)^k(\epsilon + \delta))$ ("soundness").

Analysis Part 2: "Completeness"

Suppose that Π^* is a policy as required by the theorem (satisfies (1) and (2)). We first Note that whenever Find-DT finds a literal l s.t. the recursive rank k-1 call on Π_l succeeds, then Π^* is still a rank-k subtree that meets conditions (1) and (2). (By induction on the number of attributes NOT asserted on the branch).

Observe that even if we add $\neg l$ to a branch of Π^* (that's consistent with the actions taken so far). This only decreases the number of examples that are consistent with this new longer branch

So, in particular, if there was a proof of $\neg(branch) \cup G$ in a history $h^{(i)}$, there will still be a proof of $\neg$(extended branch)$\cup$G, since $\neg$(extended branch) simply asserts more literals than $\neg$(branch). Hence, estimate $\tilde{p}$ for this branch only increases, and Find-DT is guaranteed to find a policy on such a branch and return.

We're given that every branch of Π^* is only taking μ-typical actions and since our fractions of occurrences of these sequences is at least $\mu - \frac{\mu}{4} > \frac{\mu}{\epsilon}$, we know that whenever Π^* takes an action on a branch the next action is still tried by the loop in the 2nd part. So when Find-DT makes a recursive call on a branch that is extended by this next action α_t^*, or when Find-DT makes the recursive call on the literal l leading to the rank < k branch of Π^* (if its root is a branching node) inductively Find-DT return a policy.

(Note in particular that Find-DT eventually tries the seq. of actions of Π^*).

Analysis Part 3: "Running Time"

The key observation is that Find-DT partitions the set of histories H into sets H_α depending on which $a_{\alpha,t}^{(j)} = 1$, and our time bound will (only) depend linearly on $|H|$.

The point is otherwise similar to the analysis of our algorithm for space-bold treelike resolution.

Claim for the function $f(n', t, k, |H|)$ s.t. Find-DT takes time $\leq f$ when n' attributes remain unasserted we search for a rank-k tree with t steps remaining to the horizon T and $|H|$ histories in H.

then

$f(n', t, k, |H|) \leq 2n' f(n' - 1, t, k - 1, |H|)$

Max($\sum_f (n', t - 1, k, |H_\alpha|)$ *and* $f(n' - 1, t, k, |H|)$

$+W * |H|$

and so $f(n', t, k, |H|) \leq O(W|H|(2(n' + t))^{2k})$

This will hold by induction on k and n' and t. When n'+t=0, the algorithm can't branch on any literals or actions, and so simply takes $\leq W * |H|$ time to answer the reasoning with example query.

Since the algorithm only branches on actions that are $\leq 3\frac{\mu}{t}$-typical (and in particular, have histories with $a_{\alpha,t} = 1$ in H). Inductively $f(n', t - 1, k, |H_\alpha|) \leq C * W(2(n + t - 1))^{2k} \sum_| H_\alpha|$ and likewise by $|H, f(n - 1, t, k, |H|) \leq CW(\Sigma(n' - 1 + t)^{2k})$

So we find

$f(n', t, k, |H|) \leq CW|H|(2(n' + t - 1))^{2k} + CW|H|(2(n' + t - 1))^{2k-2} * 2n' + W|H|$

$\leq CW|H|[(2(n' + t - 1))^{2k-2}[(2(n' + t - 1))^{2k-2}[(2(n' + t) - 2)^2 + 2n'] + 1]$

$\leq CW|H|(2(n' + t))^{2k}$

So since $n \leq nT, \epsilon \leq T$

$n' + T \leq 2nT$

$2(n' + t) \leq 4nT$

and $W|H| \leq O(m|KB|)$

Thus we get a $O(m|KB|(4nT)^{2k})$ time bound. $\square$

Learning and Planning

So, in particular if the POMDP is a noisy STRIPs environment with noise rate η. Then if G can be achieved using the (noiseless) STRIPs rate, then we get that as long as the STRIPS rules are (say) $(1-\eta)$-testable, the algorithm finds a tree policy for G, since we can prove that a fixed (typical) plan achieves G using unit propagation implies a space-2 treelike resolution.

Recall the strips action rules had the form for action α

for each precondition P_i

$a_{\alpha,t} \rightarrow P_{r,t}$

For each effect e

$a_{\alpha,t} \rightarrow e_{t+1}$

If we've assumed that the actions are never masked, the only question for testability concerns the rules for the action α taken at step t.

Frame axiom..

$$(l_t \cap \neg l_{t+1}) \rightarrow V a_{\alpha,t}$$

We get that this is witnessed whenever we do take some α with l as an effect (we dont need a frame axiom here). Otherwise the frame axiom is only witness if one of $\neg l_t \cup l_{t+1}$ is observed.

But, if $\neg l_t$ is known the frame axiom "didn't apply", and if l_{t+1} is observed I didn't need the axiom to tell me.

We want to give frame axioms in KB... but they require knowing the effects of α. This is where generic naf frame axioms $[l_t \cap not(\neg l_{t+1})] \rightarrow l_{t+1}$ are crucial. This requires naf! (To be continued...)