CS 717 Statistical Relational Learning

January 27, 2010

A depth-first search inspired algorithm for listing all hypotheses satisfying a quality criterion

Assumptions

The hypotheses form a finite lattice

The quality criterion is polynomially computable [Sanity check : No point developing a polynomial algorithm if the check takes exponential time]

In the example discussed in class, the quality criterion was anti-monotonic. We develop the algorithm for the anti-monotonic case [the only place this is used is in the optimizations, not the essence of the search algorithm].

We shall assume the Downward Refinement operator as defined in class :

$$\rho : S \longrightarrow 2^S$$

such that for all **h'** in ρ(**h**), **h'** $\leq$ **h**

Note : S is the set of all hypotheses

We define ρ(**h**) to be the "unvisited" elements of the complete set of downward closures of h

[If this does not make sense, we can explicitly enforce this by maintaining "visited" information with each node in the lattice, and verifying that a particular node is not "visited" before visiting it.]

The Algorithm

```
Initialize
    stack S = Empty Stack
    set Answer = ϕ
    S.push(ϕ)        // Not to be confused with the Answer set. This is the T of the Hypothesis Lattice
While ( S is not empty) :
    h' = S.pop()
    if ( h' qualifies) :
        Answer = Answer U {h'}
        set Children = ρ(h')
        for each element {h''} of Children :
            S.push( h'')
    else :
        // Trying to optimize here. Very very hand-wavy
        set All_Descendents = ρ(h')
        for each element {h''} of All_Descendents :
            [Set node h'' as visited]
            All_Descendents = All_Descendents U ρ(h'')
return Answer
```

A hill-climbing search inspired algorithm for listing at most one hypothesis satisfying a quality criterion

Assumptions as in DFS algorithm

We shall assume the Upward Refinement operator as defined in class :

$$\rho : S \longrightarrow 2^S$$

such that for all **h'** in ρ(**h**), **h** ≤ **h'**

We define ρ(**h**) to be the complete set of upward closures of **h**

We also define a heuristic function "utility", that takes a hypothesis and outputs a real number (score).

$$\mu : S \longrightarrow R$$

We shall "climb a hill" assuming the utility μ gives the slope. We shall be greedy about it.

The Algorithm

Initialize

h *=∑*　　　　*// h is the current hypothesis being examined*

// ∑ is the **_|_** *of the hypothesis lattice*

While (not ***h*** *qualifies AND not* ***h*** *== ϕ) :*

*set Parents = ρ(****h****)*

h *= argmax* $_{h'' \text{ in Parents}}$ *μ(****h''****)*

return h　　　　*// What if* ***T*** *of hypothesis lattice does not qualify? We shall interpret* ***ϕ*** *as unable*

// to find any qualifying hypothesis. ***ϕ*** *is a degenerate case anyway*