Chapter 6

Covering strategies

6.1 Basic idea

The covering strategy is used for searching the hypothesis space for *disjunctive hypotehses* – hypotheses consisting of more than one component (e.g. a set of attribute-value pairs, a propositional or relational rule). Each of this components covers a subset of the set of examples and all the components jointly (the whole hypothesis) cover the whole set of examples. The basic covering algorithm consists of three steps:

1. Applying some induction technique (e.g. a generalization opertor) to infer a correct component of the hypothsis (e.g. a rule or a clause), that covers a subset (possibly maximal) of the positive examples.

2. Excluding the covered examples from the set of positive examples.

3. Repeating the above two steps until the set of positive examples becomes empty.

The final hypothesis is a dijunction of all components found by the above procedure. Step 1 of this algorithm is usualy based on searching the hypothesis space where hypotheses that are correct (not covering negative examples) and covering more positive examples are prefered.

6.2 Lgg-based propositional induction

In the attribute-value language as well as in the language of first order atomic formulas the examples and the hypotheses have the same representation. This allows the generalization operators (e.g. the lgg) to be applied on examples and hypotheses at the same time, which in turn simplifies the learning algorithms.

Consider a learning problem where a set of examples E, belonging to k different classes is given, i.e. $E = \cup_{i=1}^{k} E_i$. The following algortihm finds a set of hypotheses jointly covering E.

1. Select (randomly or using some criterion) two examples/hypotheses from the same class, say k, i.e. $e_i, e_j \in E_k$.

2. Find $h_{ij}^k = lgg(e_i, e_j)$.

3. If h_{ij}^k does not cover examples/hypotheses from classes other than k, then add h_{ij}^k to E_k and remove from E_k the elements covered by h_{ij}^k (these are at least e_i and e_j). Otherwise go to step 1.

4. The algorithm terminates when step 1 or 3 is impossible to acomplish. Then the set E contains the target hypotesis.

To illustrate the above algorithm let us consider the following set of examples (instances of animals):

```
1, mammal,    [has_covering=hair,milk=t,homeothermic=t,habitat=land,eggs=f,gills=f]
2, mammal,    [has_covering=none,milk=t,homeothermic=t,habitat=sea,eggs=f,gills=f]
3, mammal,    [has_covering=hair,milk=t,homeothermic=t,habitat=sea,eggs=t,gills=f]
4, mammal,    [has_covering=hair,milk=t,homeothermic=t,habitat=air,eggs=f,gills=f]
5, fish,      [has_covering=scales,milk=f,homeothermic=f,habitat=sea,eggs=t,gills=t]
6, reptile,   [has_covering=scales,milk=f,homeothermic=f,habitat=land,eggs=t,gills=f]
7, reptile,   [has_covering=scales,milk=f,homeothermic=f,habitat=sea,eggs=t,gills=f]
8, bird,      [has_covering=feathers,milk=f,homeothermic=t,habitat=air,eggs=t,gills=f]
9, bird,      [has_covering=feathers,milk=f,homeothermic=t,habitat=land,eggs=t,gills=f]
10,amphibian,[has_covering=none,milk=f,homeothermic=f,habitat=land,eggs=t,gills=f]
```

After termination of the algorithm the above set is transformed into the following one (the hypothesis ID's show the way they are generated, where "+" means lgg):

```
8+9, bird, [has_covering=feathers,milk=f,homeothermic=t,eggs=t,gills=f]
6+7, reptile, [has_covering=scales,milk=f,homeothermic=f,eggs=t,gills=f]
4+(3+(1+2)), mammal, [milk=t,homeothermic=t,gills=f]
5, fish, [has_covering=scales,milk=f,homeothermic=f,habitat=sea,eggs=t,gills=t]
10, amphibian, [has_covering=none,milk=f,homeothermic=f,habitat=land,eggs=t,gills=f]
```

A drawback of the lgg-based covering algorithm is that the hypotheses depend on the order of the examples. The generality of the hypotheses depend on the similarity of the examples (how many attribute-value pairs they have in common) that are selected to produce an lgg. To avoid this some criteria for selection of the lgg candidate pairs can be applied. For example, such a criterion may be the maximum similarity between the examples in the pair.

Another problem may occur when the examples from a single class are all very similar (or very few, as in the animals example above). Then the generated hypothesis may be too specific, although more general hypotheses that correctly separate the classes may exists. In fact, this is a general problem with the covering strategies, which is avoided in the separate and conquer aproaches (as desicion trees).

Lgg-based relational induction

θ-subsumption. Given two clauses C and D, we say that C *subsumes* D (or C is a *generalization* of D), if there is a substitution θ, such that $C\theta \subseteq D$. For example,

```
parent(X,Y):-son(Y,X)
```

θ-subsumes ($\theta = \{X/john, Y/bob\}$)

```
parent(john,bob):- son(bob,john),male(john)
```

because

$\{parent(X,Y), \neg son(Y,X)\}\theta \subseteq \{parent(john,bob), \neg son(bob,john), \neg male(john)\}$.

The θ-subsumption relation can be used to define an *lgg* of two clauses.

lgg **under θ-subsumption (*lgg*θ)**. The clause C is an *lgg*θ of the clauses C_1 and C_2 if C θ-subsumes C_1 and C_2, and for any other clause D, which θ-subsumes C_1 and C_2, D also θ-subsumes C. Here is an example:

$C_1 = parent(john, peter) : -son(peter, john), male(john)$

$C_2 = parent(mary, john) : -son(john, mary)$

$lgg(C_1, C_2) = parent(A, B) : -son(B, A)$

The *lgg* under θ-subsumption can be calculated by using the *lgg* on terms. $lgg(C_1, C_2)$ can be found by collecting all *lgg*'s of one literal from C_1 and one literal from C_2. Thus we have

$$lgg(C_1, C_2) = \{L | L = lgg(L_1, L_2), L_1 \in C_1, L_2 \in C_2\}$$

Note that we have to include in the result *all* literals L, because any clause even with one literal L will θ-subsume C_1 and C_2, however it will not be the least general one, i.e. an *lgg*.

When background knowledge BK is used a special form of *relative lgg* (or rlgg) can be defined on atoms. Assume BK is a set of facts, and A and B are facts too (i.e. clauses without negative literals). Then

$$rlgg(A, B, BK) = lgg(A : -BK, B : -BK)$$

The relative lgg (rlgg) can be used to implement an inductive learning algorithm that induces Horn clauses given examples and background knowledge as first order atoms (facts). Below we illustrate this algorithm with an example.

Consider the following set of facts (desribing a directed acyclic graph): $BK = \{link(1, 2), link(2, 3), link(3, 4), link(3, 5)\}$, positive examples $E^+ = \{path(1, 2), path(3, 4), path(2, 4), path(1, 3)\}$ and negative examples E^- – the set of all instances of $path(X, Y)$, such that there is not path between X and Y in BK. Let us now apply an rlgg-based version of the covering algorithm desribed in the previous section:

1. Select the first two positive examples $path(1, 2)$, $path(3, 4)$ and find their *rlgg*, i.e. the *lgg* of the following two clauses (note that the bodies of these clauses include also all positive examples, because they are part of BK):
   $path(1, 2) : -link(1, 2), link(2, 3), link(3, 4), link(3, 5),$
   $path(1, 2), path(3, 4), path(2, 4), path(1, 3)$
   $path(3, 4) : -link(1, 2), link(2, 3), link(3, 4), link(3, 5),$
   $path(1, 2), path(3, 4), path(2, 4), path(1, 3)$
   According to the above-mentioned algorithm this is the clause:
   $path(A, B) : -path(1, 3), path(C, D), path(A, D), path(C, 3),$
   $path(E, F), path(2, 4), path(G, 4), path(2, F), path(H, F), path(I, 4),$
   $path(3, 4), path(I, F), path(E, 3), path(2, D), path(G, D), path(2, 3),$
   $link(3, 5), link(3,\_), link(I,\_), link(H,\_), link(3,\_), link(3, 4),$
   $link(I, F), link(H,\_), link(G,\_), link(G, D), link(2, 3), link(E, I),$
   $link(A,\_), link(A, B), link(C, G), link(1, 2).$

2. Here we perform an additional step, called *reduction*, to simplify the above clause. For this purpose we remove from the clause body:

   - all ground literals;
   - all literals that are not connected with the clause head (none of the head variables A and B appears in them);
   - all literals that make the clause *tautology* (a clause that is always true), i.e. body literals same as the clause head;
   - all literals that when removed do not reduce the clause coverage of positive examples and do not make the clause incorrect (covering negative examples).

   After the reduction step the clause is $path(A, B) : -link(A, B)$.

3. Now we remove from E^+ the examples that the above clause covers and then $E^+ = \{path(2,4), path(1,3)\}$.

4. Since E^+ is not empty, we further select two examples ($path(2,4)$, $path(1,3)$) and find their $rlgg$, i.e. the lgg of the following two clauses:
   $path(2,4) : -link(1,2), link(2,3), link(3,4), link(3,5),$
   $\quad path(1,2), path(3,4), path(2,4), path(1,3)$
   $path(1,3) : -link(1,2), link(2,3), link(3,4), link(3,5),$
   $\quad path(1,2), path(3,4), path(2,4), path(1,3)$
   which is:
   $path(A,B) : -path(1,3), path(C,D), path(E,D), path(C,3),$
   $\quad path(A,B), path(2,4), path(F,4), path(2,B), path(G,B), path(H,4),$
   $\quad path(3,4), path(H,B), path(A,3), path(2,D), path(F,D), path(2,3),$
   $\quad link(3,5), link(3,\_), link(H,\_), link(G,\_), link(3,\_), link(3,4),$
   $\quad link(H,B), link(G,\_), link(F,\_), link(F,D), link(2,3), link(A,H),$
   $\quad link(E,\_), link(C,F), link(1,2).$
   After reduction we get $path(A,B) : -link(A,H), link(H,B)$.

The last two clauses form the sandard definition of a procedure to find a path in a graph.