Chapter 2

Concept learning

2.1 Learning semantic nets

The basic ideas of concept learning are introduced by P. Winston in his early system ARCH [4]. This system solves a concept learning task defined as follows: the input to the system are a priori knowledge (the knowledge that the system has before the learning stage takes place) and examples of some concept. The examples are of two types – positive (objects belonging to the concept) and negative (objects that do not belong to the concept). The task is to create a concept description (definition) that accepts (includes) the positive examples and rejects the negative ones.

The a priori knowledge consists of a language for describing the examples and other facts from the domain. The language used by ARCH is based on semantic networks. It includes objects (e.g. arch, block, pyramid) and relations (e.g. isa, partof, supports, touches, doesnottouch). The domain knowledge is represented by an object taxonomy, specified by a set of instances of the "isa" (is a) relation, such as `isa(pyramid, polygon)`, `isa(block, polygon)`.

Let us consider an example of building the concept of arch, given positive and negative examples. These exampes are the following (also shown in Figures 2.1 and 2.2):

```
Example1 = {partof(block1,arch), partof(block2,arch), partof(block3,arch),
            supports(block1,block3), supports(block2,block3)}
Example2 = {partof(block1,arch), partof(block2,arch), partof(pyramid1,arch),
            supports(block1,pyramid1), supports(block2,pyramid1)}
Apriori_knowledge = {isa(block1,block), isa(block2,block), isa(block3,block),
       isa(block,polygon), isa(pyramid1,pyramid), isa(pyramid,polygon)}
```

`Example1` and `example2` have the same structure and differ only in the object that is supported by `block1` and `block2`. This object is a block in example1 and pyramid in example2. According to the a priori knowledge both the block and the pyramid are polygons. This allows us to construct an object where these two parts are replaced by a polygon. Such an object will include both examples.

A transformation that carries relations from objects to other objects including the former ones as instances (successors in the object taxonomy) is called *generalization*. Thus the generalization of `example1` and `example2` is the object `hypothesis1`, shown below. Hereafter we call such an object *hypothesis*, since it is a kind of approximation of the *target object* (the object we are learning), and later on it will be a subject of verification (acception, rejection or update).

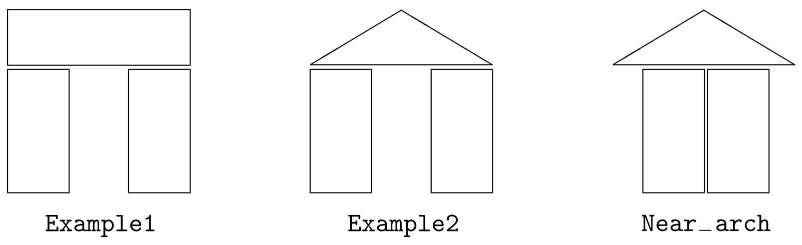

Figure 2.1: Examples of arch and "near arch"

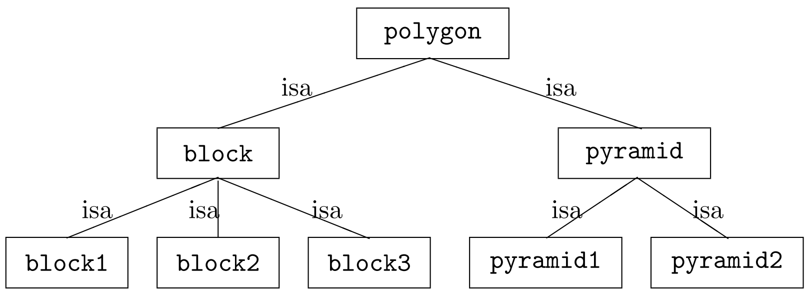

Figure 2.2: Object taxonomy (isa-hierarchy)

```
Hypothesis1 = {partof(block1,arch), partof(block2,arch), partof(polygon,arch),
               supports(block1,polygon), supports(block2,polygon)}
```

While the positive examples are instances of the target concept (arch), the negative ones are not just non-arch objects. They must be "near misses", i.e. objects with just one property (or relation) that if removed would make them belong to the target concept. This specific choice of negative examples is important for reducing the number of potential negative examples (imagine how many non-arhes exist) and possible hypotheses as well. Let us consider the following negative example of an arch (see Figure 2.1):

```
Near_miss = {partof(block1,arch), partof(block2,arch), partof(polygon,arch),
             supports(block1,polygon), supports(block2,polygon),
             touches(block1,block2)}
```

As the above object is a near miss, it is obvious that `touches(block1, block)` must be excluded from the hypothesis. This can be done by either imposing an explicit restriction to discard any objects with this relation, or to include a new relation that semantically excludes the "thouches" relation. For example, this might be the relation `doesnottouch(block1, block2)`. Thus the final hypothesis now is:

```
Hypothesis2 = {partof(block1,arch), partof(block2,arch), partof(polygon,arch),
               supports(block1,polygon), supports(block2,polygon),
               doesnottouches(block1,block2)}
```

This hypothesis satisfies the initial task by describing the concept of arch, including `example1` and `example2`, and excluding the negative example `near_miss`. The process of generating `hypothesis2` from `hypothesis1` is called *specialization*. That is, `hypothesis2` is *more specific* than `hypothesis1` (or conversely, `hypothesis1` is *more general* than `hypothesis2`), because `hypothesis1` includes (or covers) more examples (instances) than `hypothesis2`.

The ARCH algorithm developed by Patrick Winston in the early ages of AI implements the ideas described above. It is based on searching the space of possible hypotheses generated by generalization and specialization operators. The examples are processed one at a time and the for each one the best hypothesis is created. Then the system proceeds with the next example and modifies the currents hypothesis to accommodate the example (still accepting all previous examples). This kind of learning algorithm is called *incremental learning*.

The ARCH algorithm is applicable in other domains where the objects are described by relations (graphs or semantic nets). Despite of its simplicity, this algorithm illustrates the basic concepts and techniques of *inductive learning*: applying generalization/specialization operators, searching the hypothesis space, using background (domain) knowledge. It also exhibits some typical problems with inductive learning systems: the importance of the inductive bias and sensitivity to the type and the order of the training examples.

2.2 Induction task

Consider a formal system with a language L, three subsets of L – L_B (*language of background knowledge*), L_E (*laguage of examples*) and L_H (*language of hypotheses*), and *a derivabilty relation* "$\rightarrow$" – a mapping between elements from L. An example of such a system is First-Order Logic (Predicate calculus), where the derivability relation is logical implication.

The induction task is defined as follows: Given *background knowledge* $B \in L_B$[1], *positive examples* $E^+ \in L_E$ and *negative examples* $E^- \in L_E$, find a hypothesis $H \in L_H$, under the following conditions:

[1]Hereafter we denote that sentence X is from language L as $X \in L$.

1. $B \not\rightarrow E^+$ (*nessecity*);

2. $B \not\rightarrow E^-$ (*consistency of* B);

3. $B \cup H \rightarrow E^+$ (*sufficiency*);

4. $B \cup H \not\rightarrow E^-$ (*consistency of* H).

There is an obvious solution to the above stated problem. This is the hypothesis $H = E^+$. However this solution is inappropriate due to the following reasons:

- This hypothesis derives only the positive examples E^+. However the solution to the induction task supposes a kind of inductive reasoning, i.e. the hypothesis must be able to accept new examples, that the learning system has not seen before. In other words, the hypothesis must be a piece of knowledge or a general rule applicable to a whole population of objects, where the examples are just a small sample from this population.

- The hypothesis does not explain the examples. Inductive reasoning assumes that the derived hypothesis not only accepts or rejects new examples (i.e. play the role of a classifier), but describes the examples in terms of the background knowledge as well. This, in turn, would allow the system to extend the background knowledge by adding hypotheses to it.

Despite of the above deficiencies the hypothesis $H = E^+$ is useful because it plays an essential role in searching the hypothesis space. It is called the *most specific* hypothesis and is denoted by the symbol $\perp$.

Obviously we need hypotheses more general than $\perp$. Here, the relation "more general" can be easily defined by using the intuitive notion of generality – the number of objects that a concept includes, i.e. a concept is more general than another concept if the former includes (derives, explains) more objects than the latter does.

Generality (subsumption, coverage) of hypotheses. Let H and H' be hypotheses, where $H \rightarrow E$ and $H' \rightarrow E'$. *H is more general than (subsumes, covers) H'*, denoted $H \geq H'$, if $E \supseteq E'$.

This ordering between hypotheses is often called *semantic ordering*, because it is based on the meaning of the hypothesis defined by the examples it covers and can be defined independently from the particular representation languages used.

Given the language of examples in most cases we can easily find the set of all possible examples and thus, the *most general* hypothesis $\top$, that covers all examples from L_E. Again, this hypothesis is unsuitable as a solution to the induction task, because it cannot be used to distinguish between positive and negative examples (it derives both E^+ and E^-). Even when $E^- = \emptyset$, $\top$ is not suitable either, because it is not constructive (does not describe the concept).

It is known that all subsets of a given set X form an algebraic structure called *lattice*. In this particular case the lattice is induced by the subset relation $\subseteq$ and usually denoted 2^X (because this is the number of elements in it). According to our definition of generality each hypothesis is associated with a set of examples. This fact suggests that the set of hypotheses might be a lattice. This in turn might be helpful when studying the hypothesis space because lattices are well studied algebraic structures with a lot of nice properties.

Unfortunately this approach gives just a theoretical framework for solving the induction task and cannot be used directly in practice. This is because the orderings between hypotheses generally do not conform to some specific requirements needed to define correct lattices. Even when all these requirements are met, further problems exists such as:

- Every hypothesis can be associated with a set of examples. The inverse however is not true. In many cases an explicit hypothesis for a given set of examples does not exits (within the given language) or if it does exist, there is a large number of such hypotheses (often infinite).
- In more complex languages (e.g. First-Order Logic) constructive operators for generalization/specialization cannot be found. In most cases such operators either do not exist or they are non-computable.

Due to the listed above reasons the orderings between hypotheses used in practice are mostly *syntactical*, i.e. they are determined by the representation language and have no relation to the examples they derive. These syntactical orderings are usually stronger (i.e. they hold for fewer objects) than the semantic ones. Consequently the syntactical orderings are *incomplete* – they do not guarantee exhaustive search in the hypothesis space. In other words, when searching the hypothesis space it is possible to skip over the desired hypothesis and generate a hypothesis that is either too specific or too general. These problems are known as *overspecialization* and *overgerenalization.*

It is clear that the hypotheses that meet the requirements of the induction task (extended with the requirements of inductive reasoning) can be found in a stepwise manner by generalizations of the most specific hypothesis $\perp$ or by specializations of the most general hypothesis $\top$. In other words the solution to the induction task comes to searching the hypothesis space, which is a kind of a hierarchical structure with an uppermost ($\top$) and a lowermost ($\perp$) elements. Each generalization/specialization is accomplished by the so called *generalization/specialization operators.* The choice of such operators and the way they are applied are determined by the following:

- The languages of examples and hypotheses (the so called *syntactic* or *language bias*);
- The strategy for searching the hypothesis space (*search bias*);
- The criteria for hypothesis evaluation and selection.

The choices of languages and a search strategy are the most important characteristics of the inductive learning system. These choices determine the type of examples and knowledge the system can deal with, and its performance as well. These two important choices are called *inductive bias.* Since in most cases there exist more than one hypotheses that satisfy the induction task, we need criteria for evaluating and selecting hypotheses as well.