Hypothesis language

Hendrik Blockeel

Katholieke Universiteit Leuven, Belgium

Leiden Institute of Advanced Computer Science, The Netherlands

Synonyms

None.

Definition

The **hypothesis language** used by a machine learning system is the language in which the hypotheses (also referred to as patterns or models) it outputs are described.

Motivation, background

Most machine learning algorithms can be seen as a procedure for deriving one or more hypotheses from a set of observations. Both the input (the observations) and the output (the hypotheses) need to be described in some particular language. This language is respectively called the observation language or the hypothesis language. These terms are mostly used in the context of symbolic learning, where these languages are often more complex than in subsymbolic or statistical learning. For instance, hypothesis languages have received a lot of attention in the field of inductive logic programming, where systems typically take as one of their input parameters a declarative specification of the hypothesis language they are supposed to use (which is typically a strict subset of full clausal logic). Such a specification is also called a language bias.

Examples of hypothesis languages

The hypothesis language used obviously depends on the learning task that is performed. For instance, in predictive learning, the output is typically a function, and thus the hypothesis language must be able to represent functions; whereas in clustering the language must have constructs for representing clusters (sets of points). Even for one and the same goal, different languages may be used; for instance, decision trees and rule sets can typically represent the same type of functions, so the difference between these two is mostly syntactic.

Figure 1: A decision tree and an equivalent rule set.

Below we discuss briefly a few different formalisms for representing hypotheses. For most of these, there are separate entries in this volume that offer more detail on the specifics of that formalism.

Decision trees and rule sets

A decision tree represents a decision process where consecutive tests are performed on an instance to determine the value of its target variable, and at each step in this process, the test that is performed depends on the outcome of previous tests. Each leaf of the tree contains the set of all instances that fulfill the conjunctions of all conditions on the path from the root to this leaf, and as such a tree can easily be written as a set of if-then rules where each rule contains one such conjunction. If the target variable is boolean, this format corresponds to disjunctive normal form.

Figure 1 shows a decision tree and the corresponding rule set.

Graphical models

The term “graphical models” usually refers to probabilistic models where the joint distribution over a set of variables is defined as the product of a number of joint distributions over subsets of these variables (i.e., a factorization), and this factorization is defined by a graph structure. The graph may be directed, in which case we speak of a Bayesian network, undirected, in which case we speak of a Markov network, or even a mix of the two (so-called chain graphs). In a Bayesian network, the constituent distributions of the factorization are conditional probability functions associated with each node. In a Markov network, the constituent distributions are potential functions associated with each clique in the graph.

Two learning settings can be distinguished: learning the parameters of a graphical model given the model structure (the graph), and learning both structure and parameters of the model. In the first case, the graph is in fact a language bias specification: the user forces the learner to return a hypothesis that lies within the set of hypotheses representable by this particular structure. In

Figure 2: A Bayesian network, a Markov network, and a neural network.

the second case, the structure of the graph makes explicit certain independencies that are hypothesized to exist between the variables (thus it is part of the hypothesis itself).

Figure 2 shows examples of possible graphical models that might be learned from data. For details about the interpretation of such graphical models, we refer to the respective entries in this encyclopaedia.

Neural networks

Neural networks are typically used to represent complex non-linear functions. A neural network can be seen as a directed graph where the nodes are variables and edges indicate which variables depend on which other variables. Some nodes represent the observed input variables x_i and output variables y , and some represent new variables introduced by the network. Typically, a variable depends in a non-linear way on a linear combination of those variables that directly precede it in the directed graph. The parameters of the network are numerical edge labels representing the weight of a parent variable in that linear combination.

As with graphical models, one can learn the parameters of a neural network with a given structure, in which case the structure serves as a language bias; or one can learn both the structure and the parameters of the network.

Figure 2 shows an example of a neural network. We refer to the respective entry for more information on neural networks.

Instance based learning

In the most basic version of instance based learning, the training data set itself represents the hypothesis. As such, the hypothesis language is simply the powerset of the observation language. Because many instance based learners rescale the dimensions of the input space, the vector containing the rescaling factors can be seen as part of the hypothesis. Similarly, some methods derived from instance based learning build a model in which the training set instances are replaced by prototypes (one prototype being representative for a set of instances) or continuous functions approximating the instances.

Figure 3: A hierarchical clustering: left, the data set; middle: an extensional clustering shown on the data set; right, above: the corresponding extensional clustering tree; right, below: a corresponding intensional clustering tree, where the clusters are described based on color and shape of their elements.

Clustering

In clustering tasks, there is an underlying assumption that there is a certain structure in the data set, namely, that the data set is really a mixture of elements from different groups or clusters, with each cluster corresponding to a different population. The goal is to describe these clusters or populations and to indicate which data elements belong to which cluster.

Some clustering methods define the clusters extensionally, that is, they describe the different clusters in the dataset by just enumerating the elements in the dataset that belong to them. Other methods add an intensional description to the clusters, defining the properties that an instance should have in order to belong to the cluster; as such, these intensional methods attempt to describe the population that the cluster is a sample from. Some methods recursively group the clusters into larger clusters, building a cluster hierarchy. Figure 3 shows an example of such a cluster hierarchy.

The term “mixture models” typically refers to methods that return a probabilistic model (e.g., a Gaussian distribution with specified parameters) for each separate population identified. Being probabilistic in nature, these methods typically also assign data elements to the populations in a probabilistic, as opposed to deterministic, manner.

First-order logic versus propositional languages

In symbolic machine learning, a difference is often made between so-called attribute-value (or propositional) and relational (or first-order) languages. The

terminology “propositional” versus “first-order” originates in logic. In propositional logic, only the existence of propositions, which can be true or false, is assumed, and these propositions can be combined with the usual logical connectives into logical formulae. In first-order predicate logic, the existence of a universe of objects is assumed, as well as the existence of predicates that can express certain properties of and relationships between these objects. By adding variables and quantifiers, one can describe deductive reasoning processes in first order logic that cannot be described in propositional logic. For instance, in propositional logic, one could state propositions *Socrates_is_human* and *all_humans_are_mortal* (both are statements that may be true or false), but there is no inherent relationship between them. In first order logic, the formulae *human(Socrates)* and $\forall x : \text{human}(x) \rightarrow \text{mortal}(x)$ allow one to deduce *mortal(Socrates)*. A more extensive explanation of the differences between propositional and first-order logic can be found in the entry on first-order logic.

Many machine learning approaches use an essentially propositional language for describing observations and hypotheses. In the fields of inductive logic programming and relational learning, more powerful languages are used that are closer to first order logic. Many of the representation languages mentioned above originate in a propositional context but have been extended towards the first-order logic context.

The simplest example is that of rule sets. If-then-rules have a straightforward counterpart in first order logic in the form of clauses, which are usually written as logical implications where all variables are interpreted as universally quantified. For instance, the rule “IF Human=true THEN Mortal=true” can be written in clausal form as

$$\text{mortal}(x) \leftarrow \text{human}(x)$$

Propositional rules essentially correspond to clauses that refer to only one object (and the object reference is implicit). A rule such as

$$\text{grandparent}(x, y) \leftarrow \text{parent}(x, z), \text{parent}(z, y)$$

(expressing that, for any x, y, z , whenever x is a parent of z and z is a parent of y , x is a grandparent of y) has no translation into propositional logic that retains the inference capacity of the first-order logic clause.

Clauses are a natural first order logic equivalent for the if-then-rules typically returned by rule learners, but also many of the other representation languages have been upgraded to the relational or first-order-logic context. For instance, several researchers (e.g., [1]) have upgraded the formalism of decision trees towards “structural” or “first order logic” decision trees. Probabilistic relational models [5] and Bayesian logic programs [6] are examples of how Bayesian networks have been upgraded, while Markov networks have been lifted to “Markov logic” [8].

Recommended Reading

Most of the literature on hypothesis and observation languages is found in the area of inductive logic programming. Excellent starting points, containing extensive examples of bias specifications, are *Relational Data Mining* by Lavrač and Džeroski [4], *Logic for Learning* by Lloyd [7], and *Logical And Relational Learning* by De Raedt [3].

De Raedt [2] compares a number of different observation and hypothesis languages with respect to their expressiveness, and indicates relationships between them.

References

- [1] H. Blockeel and L. De Raedt. Top-down induction of first order logical decision trees. *Artificial Intelligence*, 101(1-2):285–297, June 1998.
- [2] L. De Raedt. Attribute-value learning versus inductive logic programming: the missing links (extended abstract). In D. Page, editor, *Proceedings of the Eighth International Conference on Inductive Logic Programming*, volume 1446 of *Lecture Notes in Artificial Intelligence*, pages 1–8. Springer-Verlag, 1998.
- [3] L. De Raedt. *Logical and Relational Learning*. Springer, 2008.
- [4] S. Džeroski and N. Lavrač, editors. *Relational Data Mining*. Springer-Verlag, 2001.
- [5] L. Getoor, N. Friedman, D. Koller, and A. Pfeffer. Learning Probabilistic Relational Models. In S. Džeroski and N. Lavrac, editors, *Relational Data Mining*, pages 307–334. Springer-Verlag, 2001.
- [6] K. Kersting and L. De Raedt. Towards combining inductive logic programming and Bayesian networks. In C. Rouveirol and M. Sebag, editors, *Proceedings of the 11th International Conference on Inductive Logic Programming*, volume 2157 of *Lecture Notes in Computer Science*, pages 118–131. Springer-Verlag, 2001.
- [7] J.W. Lloyd. *Logic for Learning*. Springer-Verlag, 2003.
- [8] M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62(1–2):107–136, 2006.

See also:

first-order logic, hypothesis space, inductive logic programming, observation language