Machine Learning,ML

Prof. Carl Gustaf Jansson
Prof. Henrik Boström
Prof. Fredrik Kilander
Department of Computer Science and Engineering
KTH Royal Institute of Technology, Sweden

Lecture 16
Logic Programming

Welcome to this lecture the lecture number six of the third week of the machine learning course and the topic of this lecture is logic programming. So logic programming is an abstract model of computation, in this lecture logic program is illustrated by the Prolog language other dialects of logic programming exists with slightly different properties. The logic program is an encoding of a problem in logic from which a problem solution is logically derivable. The execution of the program can be considered as a side effect of a theorem proving process. So an important concept here in logic program is of separation of programs into their logic components and their control components as captured by this local slogan, ‘Algorithm is equal to logic plus control’. So essentially the statements of logic programming represent the logic component while the machinery of the Prolog system in this case as we discussed represents that the control is use. Also a logic program can be regarded as a generalized relational database actually including both rules and facts.

So logic programming is a computational model emanating from theorem proving in predicate logic as investigated in Theoretical Philosophy. The logic program is a set of axioms and Logic programming can be viewed as control deduction. The logic programming engine tries to find a resolution refutation of the negated query. The desired computation is a side-effect the refutation proof. The resolution method used in logic program is called SLD resolution.

So let's a look a little closer to the core components of logic programming. Logic programming is based on based terms and statements, statements can be facts, rules and queries which are all syntactically so-called Horn clauses. Actually facts and queries have a very similar form, they consist of a single goal we come back to what the goal is a fact is ended with a period and the goal is ended by a question mark, so the discrimination is very discrete I'm gonna say, you can see to the right. So the fact is like father, John is the father of

top and an query can be X is the father of Tom please give me X. That's how it should be interpreted. So you tactically similar but with a very different function so a rule has the form A is implied by BB2,BN where A is the head and B's are the body and all these the A's and the B's are Goals.

So what is a goal then, so goal facts queries and rules are statements however a key part of all these are goal. So Goal is a compound term and actually compound term is a functor which is an atom which is a constant followed by a set of arguments that in turn can be a term and a term can become compound term of course or variable or an atom which is equivalent to a constant. So for example in the little example to the right you can see father is a functor actually but it's also an atom. John is an atom, Thomas an atom, X is a variable. Grandfather father, and Parents there are all factors. Variables is very important here because variable used for a matching process in the computation and Variables are Universally quantified within the scope of the statement, but so for every statement doesn't matter when there's a fact or rule if variable is mentioned there it's only variable in that limited context. So variable is not a variable in the whole set of all the statements as it is a manmade often the case in a many other programming settings. Also statements of all kinds that have only atoms are called round statements there are essentially facts or generalization of facts and a statement we have a variable is called non ground. So these are the basic ingredients the basic way of expressing yourself for logic programming.

So what I show you here is a slightly larger example of logic programming and as you can see in the top there are facts which is essentially predicates, so the predicate male states that John is a male, Tom is a male, George is a male, Tim is a male, and Ann is a female and dianna is a female and so on, so all these are facts and they are supposed to be all facts that supposed to be terminated by period. Then someone can say that this kind of all facts corresponds to two features or attributes of objects then we have relations between objects oh so I think you see in the next part so we have facts that gives relation between those objects and you know in this case is their family relationships. And then follows different kinds of rules more or less complex that expresses some of them are simple some of them express more complex things into several other sub-samples and so on. And then in the end at the bottom line you find the kind of queries you can put to a system which has the same form of facts but they are all followed by question mark and they may many times involve variables.

So the execution of a logic program and the idea here is to give you a flavour of this computation happens. So the execution is initiated by the user's posting of a single goal

called a Query. You'll see an example of query. The logic program uses that tries to find a resolution refutation of the negated query. The negated query can be refuted, it follows that a query with the appropriate variable bindings in place it's a logical consequence of the program. In that case all generated variable bindings are reported to the user and the query said to have succeeded. So essentially the computation is the indirect effect of the proof and the result or the variable bindings, so operationally the logic programming execution strategy can be thought of as a generalization of function calls in other languages because you can see that you have a query and the query is matched to some to some of the goals in the set of rules and there is a match and if there is a match that results in the invocation of the right side of the rules, so you can see this is a recursive set of function calls. The difference is that in contrast to normal function calls where we're totally deterministic which function is called in this kind of language multiple Clause heads that multiple rules can match a given call. So in that case this kind of system makes it choice so it takes the first rule that match the goal and notes that point and continues. So with any goal fails in the course of executing the program in the continuing execution all the variable bindings that were made since the most recent choice point of made are undone, so essentially if everything works well then you always take the first choice in every step, but if something goes wrong along the line you're going to go back and then you have to redo all the bindings you have made up to the point where at the first point way where you can take a next option. So this kind of execution strategy is called chronological backtracking. So essentially what the Prolog system does is tries different options. So one can see it as a search strategy through the various options that exist for this particular computation.

So what does learning mean for this kind of representation actually there is a specific subfield a special subfield called inductive logic programming the subfield of machine learning where Prolog statements are induced from examples. The standard logic programming representation is used to uniformly represent both examples hypotheses and background knowledge.This is a big feature, it's actually a big advantage because in many other learning settings you're forced to put a lot of effort on the syntax of the of the language in which you express your examples, the language in which you are supposed to express your hypotheses, and thirdly a language for its expressing background knowledge, and in many cases there are different and there are a lot of mappings and other discussions needed to make a good engineering here.

So the uniform language it is a big advantage. So what do you even input them to a inductive logic programming system, yeah what you input is actually the dataset that you expressed in

Prolog if this now is the language we look at and also the relevant background knowledge and this is pretty straightforward. The output is actually not the program statements that created usually created by the system that can entails all positive and no negative examples, of course learning program is such does not really support learning but I think what I'm argued for here is that given that some learning algorithm can apply we applied logic program is a very uniform and nice framework for integrating these components in the whole system.

So this was the end of this lecture thanks for your attention we are now finished with the main lectures of this week so the next lecture will be understand a topic of tutorial for week 3