[Music]

Welcome to the second lecture of the fifth week of the course in machine learning. So as you know this week we will focus on machine learning techniques enabled by prior theories and the topic for this particular lecture is Explanation Based Learning. Explanation based learning abbreviated EBL is a machine learning technique where we try to learn general problem-solving schemata by observing and analyzing solutions to specific problems. So we are interested in how this can be done in machine, but it should be obvious to you that this kind of technique on a conceptual level also is very much relevant for the way we learn as humans in everyday life, and this is illustrated in the very rather famous cartoon to the right where the these three guys looks at their friend and see what he does and probably at the later stage will abstract from that. Explanation based learning creates learning general problem-solving schemata by observing and analyzing solutions to specific problems and EBL spans the spectrum of inferences from deduction to abduction, however I would say that if we look at what's particularly interested by explanation learning is his ability to formalize abduction. And in this process one uses prior knowledge to maintain or explain why training example is an instance of a concept, and where that generated explanation as one thing identifies what features are relevant to the target concept among potentially many features described for the problem situation. Prior knowledge is used to reduce the hypothesis space and focus the learner on hypotheses that are consistent with that background knowledge and typically accurate learning is possible even if we have only very few instances even a single instance could be possible. There is a trade-off here obviously between the need to collect many examples to be able to induce which we are typically done now for some weeks without prior knowledge and/or the ability to explain single examples in the presence of a strong domain theory. Let us try to describe the exact scenario for explanation based learning algorithms. So given is a target concept defined by some predicate statement, also as an input to this kind of algorithm you have training examples typically a few, you have a domain theory which are built up from facts and inference rules that express background knowledge potentially relevant for the target concept. Probably most cases you have more background knowledge than you actually will use but of course it's not optimal to have too much background knowledge because then you have the problem of choosing the right subset, and finally there should be an operationality criteria which specify which predicates can occur in the right concept definition. Typically the predicates are those used to the training instance case. So actually we do not want to define hypotheses that are too general in the sense that these the

definition of those concepts are expressed in such terms, that they are not operational in a domain specific context. So the goal could be refreshed to find alternative operational more efficient definition on the target concept that is still consistent with logic entailed by both the domain theory and training examples. So typically we have some definition but rather abstract of the target concept and the idea here is to specialize it. Also typically and (this kind of algorithm has the following steps ,so first you have a step to where we explain, by which we mean that we can also more or less say we build up some kind of inference chain or proof on why this example actually should be a member of the target concept we're interested in. And then the idea is that when we look at that explanation or that chain of inferences, then that can then be in the beginning be pretty concrete in terms of the example term or the idea is to generalize that explanation in transgene (06:28) To define the most general rule but still being operational in the sense that we express ourselves in that kinds of terms or predicates that we have defined as being on the operational level. And finally if when we created this new definition of this concept we can add that to the domain theory. So the idea is to incrementally augment the domain theory for each round in this EBL iterative process.

I want to say a few words about different perspectives can have on what EBL actually means. So one perspective which is that the closest one to the mainstream or in in inductive learning techniques is as Theory guided generalization of examples. So essentially we have an induction process but we guide that process and we focus the search for the appropriate hypothesis by using the theory. And as part of that which is always important is of course that the explanations we built should be as simple as possible so we should use as few features for example as possible, but still being able to explain why the example is an instance of the concept. So that's one perspective that is more close to the inductive machine learning string. Another perspective is us example guided reformulation of theory, so then we can say we start more from a theory and can be the case that the have a theory this is over general that is not consider operational in the sense that it's effective to use for problem solving in a domain. So what we want to do is to specialize the theory so it becomes more operational and then we can do that by looking at these examples and the examples can guide us in which specializations we should make .

The third way of looking at EBL which now we move more and more actually by a step by step in this little list of perspectives into the detective realm so we can view EBL as knowledge compilations, so actually the domain knowledge is complete enough to perform some reasoning for specific cases but what we can do with through EBL is to simplify this

inference change, so they've become more efficient in in particular case. Then one can of course question whether that is learning or not, because the competence in there to solve the problem, the only we do is that we learn problem-solving skills to become more efficient. So on this line you can see some graphical depiction on what I hopefully have expressed earlier that what we really do in EBL is we try to flatten the structure, the chain of inferences so in same instance of expressing the problem-solving through a rather complex chain of inferences we can express ourselves in a more flat structure, which is actually always more efficient often more efficient. So here on this slide you can see a little picture, the purpose of that is to give you a feeling for the components of an EPL system, so I think central is you have some kind of problem solver and you specify a case and then you have this we have this knowledge base which is actually background knowledge plus some rudimentary hypothesis relevant for the case we there is a explanatory process in where we try to actually solve we can say we can try to solve the actual problem using the knowledge base, and that generates an explanation or some kind of inference chain, and then that inference is generalized and when in generalized and we have a new more general concept definition we had that concept definition to the knowledge base we get a new problem and so on and so on.

At this point we're going to become a little more formal, and we are going to look at specifically the generalization part of the explanation based learning algorithm, and actually and as you remember from one of the earlier slide where I showed you an hierarchical structure and a flat structure. So roughly you can say that a typical explanation that we start from in the concrete form is a hierarchical structure where with a lot of inference depth that are based on specific rule application. So really what we want to do here is that from that hierarchical structure we will create as flat structure as possible. Yeah however of course we want the new structure to still correspond to a valid proof, the valid proof we started from. So this means then we in transform their article structure to the flat structure we must also see to that there are appropriate connections among all variables and constants occurring in the original concrete explanation or proof and actually when on this slide the word literal is used is actually literal just a common term for four could be a constant it could be a variable so to say. So actually what we need is a machinery that do this advanced pattern matching which is called Unification which sees to that we do with the appropriate binding of variables and constants or variables or variables which is then called unification. So that's an important thing in ingredients in this. So essentially when we if we want to look a little informally on the algorithm that is described on this slide and you should always have in mind that we start

from hierarchical rule application proof and what we went in the end is a flat structure, so you can see on the last row that is a variable called P which is actually a list of predicates, so essentially these predicates or literals are collected during the process and essentially the end result as we want to set our target concepts equal to this flat list this conjunction or predicates so that's the end. So really what happens in this algorithm is that we iteratively build up this list and we actually do that through a process where we take the target, we target the predicate and as it is expressed here regress is true each rule on each level through the proof structure, always then involving this unification algorithm that it sees to that the variables and constants are unified together in the appropriate way. So this is intuitively what is described here and I hope you can look at a little in more detail, we'll also I will also show you an example and in the coming slide. Here we will look at a simple example, actually it's a problem where we look at boxes and that can be put on a table and there you can put on each other and actually the target predicate we want to look at in this example is whether it's safe to put one box on the other and then there are a number of these predicates who say whether they are light or not if we try to define light in terms of their weight, we which are try to define their weight even the in terms of their volume, we can also say that one of these elements we have is actually it constitutes this table and so on. So we have a little domain theory that describes these kinds of boxes on their property including the table. And then we also have a list as I already mentioned it typically for this kind of methods of operational predicate, this means that those predicates we are allowed to use in our final target definition of the goals concept. Then we have a training example which is actually the situation where we have two objects and the properties of those objects and then finally we have the target or goal concept that is with this case this predicate expressing whether it's safe to put one object on the other. So this is our domain theory in this case here and now comes the concrete explanation which is then the chain of inferences through which we motivate why this particular configuration of two objects with this particular kind of features are such that they can safely be put on each other, and as you can see his tree hierarchical inference structure use a certain number of the rules from our little domain theory. What we do now is actually that we go through the structure we saw a radical one and actually we start from the top with the goal predicate and then we start what was termed in the description of the algorithm as regressing that that predicate through this proof structure, and we start from the top and we take it level by level. So here we can say we look at the first step, the connection between the SageToStack and the lighter predicate. And we take a step in that way we find appropriate

unifier and we build up start to build up our P variable which is actually the list of predicate that we in the end should form our final result but at this point it has only one element.

Here you can see step two, so we take the second level where we reduce lighter into the simpler predicate and we do the same we take that we take that predicate and regress that through this this segment of the proof structure with appropriate unifier and then you can see how the P variable is built up with an asset the repertoire of consistent variables ethics as a secured by the unification algorithm. You can see the third step in a generalization process and which goes in exactly the same fashion and you can see then in the end that P variable continues to accumulate the relevant predicate with the appropriate variable bindings and then there is a last step and then at the bottom you can see the final P variable and so now this regression process is over and actually we only now have to set the target predicate equal to this conjunction of predicates collected during the process.

EBL is of course very much related to the kind of theory background series we work with and the quality of those theories I mean for this kind of toy examples we looked at is it's like we have a theory that is complete even if it's small and they have one little corner we have one target concepts which we want to operationalize, so it's very well-defined but in general of course there can be all kinds of properties of domain theories all kinds of imperfections so they can be incomplete they can be inconsistent, they can be incorrect, incorrect here means that the theory is consistent internally but the theory does not match reality. So anyway when I say it's wrong because it's not a good model of domain we want to make a model for but it can also be intractable because it's so computationally complex to handle and of course there are two way reasoning here so EBL techniques can of course be used to make theories better to complete what is incomplete and so on, but also of course other kinds of imperfections of theories can also disable the EBL algorithms, the function of the EPL algorithms. So the message is here that that EBL is not a general technique that can be used in a very simple-minded way for all possible domains theories. Actually in order to be useful the use of EBL techniques must be must be prepared for by a careful analysis of the available domain theories and the properties of those and some prognosis of the kind of imperfections they have and then making some clever choices of what kind of imperfections could be fixed using these techniques and also being sure that are not other improve imperfections that can that can hinder or complicate the use of the techniques.

Another issue that I want to discuss a little is the utility of the created new operation concept definitions or rules that we handle. EBL in many cases do not create entirely new knowledge but as rather the goal to improve the domain theory so that the problem solving becomes more efficient. So one way of looking at EBL is as knowledge compilation, of course one can always discuss what is learning is this learning them because it's not new knowledge is just becoming more efficient on the other hand, if we look at ourselves a lot of the time when we say that we learn something is that actually that we learn to become more efficient in doing things. So it could be of course a battery of debate where we are in on the borderline between planning and learning. But in this perspective EBL represents a dynamic form of knowledge compilation or a system is tuned to incrementally improve efficiency on a particular distribution of problems. So it's not so necessarily that the system becomes more efficient for all problems because we started with a more general theory and then by handling a series of cases we in a way optimize the system for that particular sample of instances. It's also the case if we learn many new rules many of these new operational rules we can call our macro operators, macro rules, search control rules, the terminology is pretty complex here, and it can also be so unfortunately that to add more rules to the system in spite of the good purpose we had will deteriorate the problem solving efficiency. So that the amount of generated additional rules outweigh the benefit that we were after. But of course as always we can come up with countermeasures so we can be more careful in our selection of rules to store, we can also see to that sometimes we throw away rules that are rarely used. So by making our algorithm more complicated we can anyway fight this phenomena that the growing rule set deteriorate performance.

EBL systems is it's not a new invention, these kind of systems have been built for a very long time and you'll find on this line a list of examples and for example if you look at the earlier systems, one famous system this combination called STRIPS+MACROPS is very famous planning system from the early 1970s which actually also in on top of its planning performance had had had a capability to building but what was called at that point as macro operators. Also the hacker system were by Sussman at MIT was from that time and both of this system if we look at what really happened was very very very similar to what we now call explanation based learning. So one can say that the evolution of the term explanation based learning was really starting more or less in the mid 1980s where there were a number of papers one of them is mentioned here by Mitchell Keller and colleagues where they try to give kind of create an abstraction for this kind of systems. So you will also get some

reference to some of these systems in the recommended readings. This was the end of the lecture on explanation much learning. thanks for your attention we will now turn in to the news next subtopic inductive logic programming which will come up in in the next lecture thank you.