[Music]

Welcome to this lecture on Generalization Search which is the second lecture of the fourth week of this course in machine learning. For practical reasons this lecture will divide it up in two videos, so this is part one. The outline of this lecture is as follows we will first start with some general characterization of the aim of this approach, we will then look on the kind of language and formalism we need in order to work with this framework and then we will look at the two main categories of strategies for doing this, one we can say Generate and Test is essentially it's a top-down approach where we in rate hypothesis and then search through the instance space in order to iteratively test which hypothesis is optimal. The other category called data-driven strategies for generalization where we actually start bottom up from the instances and we will systematically through control analysis but instead search what we call the hypothesis space that we have built up, and we will look at three ways of searching the hypothesis Base first, Breadth first and Version Space approach.

So the purpose of this lecture is to compare various approaches to generalization in terms of a single framework which we term here generalization as search. Towards this end, we want to cast the generalization problem as a search problem, and alternative methods for generalization are characterized in terms of the such strategies that they employ. This lecture is tightly based on a seminal paper by Tom Mitchell from 1977 called “Virtual Spaces: A Candidate Elimination Approach to Rule Learning”.

So the preconditions for the problem we're going to look at are the following, so we have a language in which to describe the instances which we call an instance language. We have a set of positive and negative training instances of some target generalization that we want to learn. We also have another language we call hypothesis languages, the purpose of this is to describe our generalizations and this hypothesis language which spans what we call our hypothesis space. And the hypothesis space as we are going to construct it is a Poset (a partially ordered set) organized by a more specific than relation that relates more general concept to more specific. And then finally we need a kind of matching function or predicate that can test at every point in time whether given instance and sum over our generalizations match. So what we want to determine Generalizations within the provided hypothesis language consistent with the training instances we have which mean that they are consistent if and only if hypothesis matches every positive instance and no negative instance in the data set. This approach is dependent on two constraints, so first of all it's assumed the training instance is contained no errors and it's also is constrained by the fact that it's important in

necessary that the target generalization can be described in hypothesis language we have designed, So we can say that this is a theoretical framework because it cannot handle some of these practical problems which are formulated here as constraints.

Let's start with saying something about the hypothesis language, so the choice of the hypothesis language has a major influence on the capability visually of the learning system. So by choosing a generalization language you fixes the domain of generalization which a program may can describe and therefore learn. So most systems in some way use a generalization language that are biased in the sense that the language is capable of representing only some of the possible sets of describe people instances not all. These biases causes both the strengths in the weakness if it's the biases inappropriately chosen it can prevent the system from inferring the correct generalizations, if it is well chosen it can guide the induction so it can actually enable inductive leaps steps beyond the information directly given by the instances. So there is a bad side a good side of this kind of language, also the choice of language has a strong influence on the resource requirements, so the complexity of the hypothesis space, if you represent hypothesis space by graphs because the problem can be exponential, while if you use feature vectors for example you can keep the complexity down to linear, and also a language where the ordering you introduce is shallow and branchy that will typically create a larger hypothesis set in contrast to wherever the ordering is narrow not so branch in each step but rather deep.

With the approach we have taken the most important choice to make for a language is which relation to use to relate our hypothesis. So our choice here is a more specific than relation between hypothesis and the semantics of this relation is given as follows, so if we have to generalization G1 and G2. G1 is more specific than G2 if and only if the set of instances that G1 matches it's a proper subset of the instances that G 2 matches considering all the instances and the matching predicate. So you should note that this definition of the relation is extensional is based upon the instance sets that generalization covers, of course the definition of the relation is also dependent on the language we have in the matching predicate. So the more specific than relation imposes of what is called a partial ordering over the generalizations in the hypothesis space. It provides a powerful basis for organizing the search through the hypothesis space. And finally one important fact is that in order for the more specific than relation to be practically computable by some program, it must be possible to determine whether G1 is more specific than G2, only by looking at the descriptions of G1 and G2 without computing all the sense of instances that very much. So even if the relation is defined by the instances it's not tractable - to make judgments of the relation between

hypothesis by locating instances. So this requirement that we have to be able to decide on the validity of a relation in space of the description places restrictions on the way our language is formulated. Before we continue I want to say a few words about partial orderings, in mathematics a partially ordered set formalize the ordering of the elements of a set. So a partially ordered set or Poset consists of a set together with a binary relation indicating that ordered pairs of elements in the set, such that one of the elements precedes the other in the ordering. The word partial means that only a subset of pairs of elements are directly ordered, as you can see for example in the example to the right where we actually look at subsets of a three element set XYZ, we can you can see obviously that the subset consisting only the element X is not directly order with respect to the subset Y said as an example, while it's order with respect to the subset XY and XZ if every element in the set is order would we talk about the total order instead.

Before we go on I want to introduce a simple example that I will use for this lecture, so the in this example our instances will be unordered pairs of simple objects where each object is described by three features, the features are shape, color and size and the feature values of shape is square, circle and triangle the feature values of color are red, orange and yellow and the features of size are large and small. And so actually the instance language if we look at an example there you can see instance one is consist of two objects a large red square and a small yellow circle. So that's a instance the hypothesis language in this case follows more or less the same syntax, the only thing we do is a add a wildcard symbolized by a question mark and actually the purpose is that is to function as a generalization for the features values of a specific feature.

So it's a possibility here to generalize feature values but just in one step using this wildcard.

Let's now look at an example so a small example with just four, hypothesis structured in a small network. So we have G 1 G 2 G 3 and G2 you to more general then the others and as you see in the example, ordered from general to specific. So you can see here G1 is more specific than G 2 and you can also see the G3 is more specific than G2 but you can also see that because the way we define this network through the more specific than relation G1 and G3 are not comparable generalizations even though the instances of G3 and G1 intersect, but the sets belonging to these two generalizations do not contain each other.

So much about the hypothesis language and its properties so now let's look at different kinds of generalization strategies. So on one hand we have the data-driven strategies and those will be the focus of this lecture and, in that case the instance base is traversed systematically and as a consequence the hypothesis space is then searched, and when we say we use one or the

other search strategy we talk about applying that such on to the hypothesis space. In Generate and test strategies we essentially we start from the hypothesis space by traversing the hypothesis space, we look at the instance space and such that so it's a general to specific approach.

First a few words about the generate and test strategies. So in generating and test we generate new hypothesis according to some procedure and this procedure they're typically independent of the data set of the input data. So each generated hypothesis in the hypothesis space is tested against the entire data set available and for every step the candidate hypothesis is either identified as an acceptable hypothesis generalization or it could be viewed as a node to be expanded further to create new hypothesis or one can say one can say it's a dead end and prune it away. So generate and test strategies typically consider it is all data instances in the data set at each step, for each new generated hypothesis to be tested. It is also an property of this kind of algorithm is that because for each hypothesis they look at all the instances not only single instances. They are not so prone to deteriorated in the presence of noise because the noise will just be part of the stream of all the instances. On the other hand it's a problem because if you if we have a batch learning approach it's fine but if you have an incremental situation it's more of a problem bit because if new data comes up later in the process you may have to re-execute the whole generate and test procedure. Also because of the fact that the generate hypothesis is not influenced at all by the data the search can be quite expensive.

We will now start to discuss a number of data-driven generalization strategies so actually many generalization programs employ search studies that are data-driven, which means that we build up and hypothesis space and then we revise what we believe to be the current hypothesis or the current hypothesis based on all the incoming data instances. So we will look at three kinds of approaches Depth first search, Breadth first search and something called Version Space approach, and for each of these approaches we will do four things we will give some general characterization of the approach, we will sketch the prototypical area, we will trace a simple example and give some comments on that little trace.

Let us start with the Depth-first strategy. So one of the strategies for utilization from a company is depth first in this strategy. We keep a single generalization like single hypothesis as the current best hypothesis and we call this presentation we call it CBH, so the start search start by choosing taking the first instances and choose a CBH consistent with that first instance. And then we test systematically the CBH against the new training instances and we altered the CBH and we can either alter it so it becomes more restrictive and this is because if we encounter a negative observation that we will do not want to be covered, we can relax it

more because we also have to make it want to make it cover a new positive observation. However when we alter it we'll also have to look back at all the previous observation to ensure that we do not create an inconsistency with those instances already managed. And also of course at every step there are not just a single way of altering the CBH consistent or good you see there are many options we have to choose one and for all the options there are we have to try them one by one and we do that in some order and if one of the orders later we became obvious that what we have choice we have made is not the optimal one we made you have to do some backtracking going back to that decision point and chain and take a second choice. So drawbacks with this is it's very costly at every step to check for consistent with all past training instances and also if we happen to make the wrong choice at one of these decision points to backtrack, to reconsider the alternative choices at that decision point.

On this slide you see some pseudo code for the depth-first search strategy procedure. I already outlined for you informally how it works and essentially the role of the similar code is just to formalize this one step further. So what you can see here is pseudo codes are actually two parts in the beginning, one part that handles the negative case and as I already said when you when we encountered a new negative instance is in this process of revising the currently by best hypothesis, we need to constrain the CBH, so that it do not cover that new negative instance. On the other hand there may be many ways of changing the CBH in that direction, so therefore we have to consider all the earlier instances so we don't get an inconsistency. Similarly in the positive case, we may normally need to extend the CBH so that it also covers the new instance to generalize the CBH, but when we do that we also have many choices so therefore we also need to reconsider they're all the instances so we are not extending it so much that it covers some earlier negative instances. And finally in the psuedo code you can see the section where we come into a situation that we discover that we cannot find a suitable revision which means that the probable situation is that we made the wrong choice at an earlier point among all the earlier points we had, so therefore we need them to backtrack to it to the to that to an earlier decision point and I mean backtracking can have many forms what we talk about here is the same version form of backtracking which is called a Logical backtracking essentially we go back backward to the nearest decision, and find a new alternative choice there, and so and of course that can be recursive because we have many levels of decisions, so as already said earlier this can be a pretty costly procedure.

So let's look now at an example on how to run the depth-first search and on a trace for that example. So in this example we have three instances, first we have two instances which are positive and then we have one negative instance and what we what you see on this slide is the

current best hypothesis in three versions representing three steps of the depth-first search a generalisation algorithm.

So some comments on this example, so the first positive training example leads to the initialization of the current best hypothesis to CBH1 which matches no instances other than the first positive instances. When the second positive instance is observed CBH1 must be revised in order to match the new positive instance and you should notice here of course there are many plausible revisions to CBH1 in addition to see CBH2. What is not visible here is of course the order of these alternatives but the system picks what is considered the first alternative, yeah so now we go to CBH2. So after these two positive examples comes a third training n cells which is negative and being negative it conflicts with CBH2 in this case all of the ways to could be specialized to exclude the new negative instance. No such relation is considered with the earlier observed positive instances already looked at.

Also one observation we should have her in order to understand the example is that the every instance is an unordered pair of objects, so it doesn't matter in which order the two objects are placed within the instance. But actually now and considering the third negative instance the system must and backtrack to an earlier version of this current based hypothesis reconsidering its previous revisions to determine a CBH3 that is consistent with the new negative instance as observed positive instance.

So this is the result in the end. For practical reasons not to make this video too long we make a break here and continue the treatment of the other algorithms in the part 2 video. thank you bye.