[Music]

Welcome back to the lecture on generalization as search which is part of the fourth week of the machine learning course. We did make a technical break in in this lecture in order not to have too long video sessions and we did make the break when we were starting to discuss generalization methods. We did look at depth-first search algorithm and what we are not going to do is to look at the second kind of algorithm, the breadth-first. So let's now turn to the breadth-first search strategy once on this slide you can see it's also termed specific to general and we will come back to that prefix so in contrast to depth first search breadth first search maintains the set of several alternative hypothesis not only one. The current set of hypothesis and as you will see the most specific such hypothesis is termed S and S is a generalization consistent with the observed instances such that there is no generalization which is more both more specific than S and consistent with the observed. So again we in this method we work from the most specific hypothesis that can cover the instances seen so far and generalize those generalizations when needed. So starting with the most specific realization the search is organized to follow the branches of the partial ordering so that progressively more general generalization or considered each time the current set must be
Modified. So we initialize as to the S of maximally specific generalizations consistent with the first observed positive training instance. In general positive training instances force the set S to contain progressively more general hypothesis if we encounter negative instances we need to eliminate some generalizations from S thereby pruning branches of the search which have become overly general. So this search proceeds monotonically from specific to general hypothesis. The candidate revision of S must be tested for consistency still with the past positive and negative instances. So in light this respect that this algorithm shares the properties of the depth first. One advantage of this strategy over the first search stems from the fact that one can see say that the set S represents a threshold in hypothesis space generalizations more specific than this threshold are not consistent with all observed positive instances where is those more general than this threshold are. Let’s now look at the pseudo code for the breadth-first search strategy. So we start by initializing S to a set of generalizations consistent with the first training instance. For each subsequent instance **i** we have two cases either we encounter a negative instance and if we went out the negative the instances we are we need to modify the set S so that we only retain those generalizations which do not match this new negative instance. In the other case if we encounter a positive instance, then there are two things to do, so first of all what we do is we generalize the

members of S, the who which do not match **i** along each branch of the partial ordering. But only to the extent required to allow them to match **i** which means that we're conservative we generalize of course in order to cover new but not too much so we keep to the most specific generalizations needed to be done. And after that we'll also have to release it S and look at it and see to that if there is a more general element of S then the newly created, then we should remove that element or it could also be so that our elements in S now that matches a previously observed negative instance and that we cannot allow, so also those elements have to be removed. So this is pseudo code for this algorithm.
Let us now look at the example again. So we have three instances, two positive or negative and this allows us to look at three iterations of running the algorithm. So we can then also create three versions of the hypothesis space, S1, S2, S3. So the first thing we do is that we instantiate S to the most specific generalization consistent with the first example. Then we in the next step we need to revise S1 in response to the second post instances. Here S1 it generalized along each branch of the partial ordering to the extent needed to match the new positive instance. So the resulting set S2 is the set of maximally specific generalization consistent with the two observed positive instances of form. And of course you should also observe here like in the examples for depth first, that the every instance is an unordered pair of objects, so the order of the two parts within the instant doesn't matter. So the third data item is a negative training instance, in this case one of the members of S2 was found to match the negative instance you can see that if you study the previous slide and therefore need needed to be removed from the right set of S3. So when we look then the resulting was in S3 there is no possibility of finding an acceptable specialization of the discourage generalization and no more specific irritation is consistent with the observed positive instances. At the same time no further generalization is acceptable, since this will also match the new negative instances. So all of these things have to be taken care by the algorithm.
Let us now turn to the third strategy the so-called version space strategy. Actually the version based strategy is an extension of the breadth-first search approach and it's a combined specific to general, in general to specific approach. So in addition to the set S that we used in the breadth-first case, we define a set G, Center members of G are as general as possible. So G is all hypotheses consistent with the observed instances such that there is no generalization which is both more general term G and consistent with instance. The set S are handled exactly as it was in in the breadth first case together the sets S and J precisely delimit what we call than the version space. So generalization X is contained in this version space represented or bounded by S and G if and only if **x** is more specific than or equal to some

member of G and also more general than or equal to some member of S. So the advantage of the version space tragedy lies in the fact that the set J summarizes the information implicit in the negative instances that bounds the acceptable level of generality of hypothesis, while the set S summarizes the information from the positive instances that limits the acceptable level of specialization of hypothesis.

So this can then be depicted in a simple slide like this, where you can see S and J and you can see the area of S and J and between them which are the consist of generalization. So upwards more general that much more specific. So and exactly what happens is that when we go through the algorithm iteration for iteration positive examples tend to move the S set upwards a negative example tend to move the J level down, which means that step by step the possible hypothesis are squeezed in between S and G where the gap becomes more and more narrow and of course to make in the end there should only remain one optimal hypothesis.

Some more comments on the version space strategy then before we go to the example. So testing whether a given generalization is consistent with all the observed instances is logically equivalent to testing for it lies between the sets S and G in the partial ordering generalizations. The version space method is assured to find all generalizations within the given generalization language, that are consistent with the observed training instances independent of the order of presentation of training instances. As you may know has many machine learning algorithms may depend in the order in which we look at training instances but in this case it doesn't matter. Also the sets S and G represent the version space in an efficient manner summarizing the information from the observed training so that no training instances need to be stored for later reconsideration. As you remember from the depth first and breadth first, there is a need of different kind to store the training instances for further inspection. However there are few restraints that should be noted for this technique, so as for the breadth-first search this technique will only work when the more specific than operator and what general an operator can be computed by direct examination or hypothesis because we just said that we don't need to store the instances and so therefore all operations have to rely on this explicit hypothesis space that we construct. In addition this technique assumes existence of a most general and most specific generalization. And it may be the case that these hypotheses do not exist.

Let's now look at the pseudo code for the outline of the version space strategy and the important parts here are the two blocks. So there one block for actions for in the case that we encounter negative instance and one black collections and we encounter a positive instance. And as you see here there is a nice symmetry between these blocks with respect to how the two sets S and G are handled. So if we look at the first part of each block so you can see that what we need to secure when we see a negative instance is that we have to retain in S only those generalizations which do not match G because we cannot allow for any hypothesis that match a negative instance okay. So symmetrically then when we see a positive instance we have to see to that in G we only have such generalizations that match G because we don't know what generalization that do not cover one of our positive examples. Then there is a second case you need block of actions. So and for this a case for the negative case we look at generalizations of G that match **i** more specific only to the extent required, so essentially what we do when we see a negative instance we make generalizations of hypotheses in G but we make it in a conservative way. In the same fashion as we see positive instances we generalize members of S that do not match G but also conservatively only to the extent required to allow them to match i and only in such ways that each you may more specific than some generalization in G. And finally there is a third case in both blocks because afterwards all of this is done one and two we then have to see to that we don't have unnecessary elements in G and S, so in the first block band when you move from the any element that is much less perfect specific than some other elements in G, because we only and only allow G to go down so to say in the level from you're not to specific if this is motivated by the instance encountered and then in the same way remove S any elementary is more general than some other elements in S because you also are conservative in the way we generalize S .So this is the essence of this algorithm.

Let's look at the example again now for the version space. So we have the same example you can see three instances, what happened now is that if we look at the hypothesis space the set S is the most specific generalizations are exactly the same as for the breadth-first situation because essentially as I already said the version space approach is actually an extended double version of breadth-first, where we both look at the most specific and the most general hypothesis at the same time. So what actually the only difference now with this trace is what's happened to the to the set G. So the situation is very similar but we have to consider the set G. And what we choose to do here is to set G to the most generalization describable within the given language and that's all question marks and that matches every possible instance, because it's consistent with two positive training examples shown in this figure, the G is unaltered in the first two iterations. As was already said the set S is revised as in the breadth-first search in all the three iterations. So the difference here is but in the third iteration G2 is

revised since the negative instance reveals the current member of G2 is a version the generalization in G2 specialized along the possible branches of the partial ordering that leads

came up somehow down towards a member of S3. Along each such branch its specialized only to the extent required so that generalization no longer matches the new negative instances so in done in a conservative manner. The version space at this point contains the members S3 and G3, as well as all generalization that lie between these two sets in the partially for the hypothesis space. Subsequent positive trainings and may force S to become more general while subsequent negative training in may force G to become more specific. Given enough additional training instances, S and G may eventually converge to sets containing the same description. At this point the system will have converged to the only consistent generalization within the given generalization language.
I hope now you've got a feeling for the behaviour of these kinds of algorithms. I will end this lecture by assigning a short comment on performance in general we don't focus much on performance on this course but I want to mention one thing because it's also anyway repeat something already said which has some importance. So if you look at this little table on the slide and you look specifically at the storage space, you can see that for the version space strategy when you look at the bottom right corner you see that the order of the space needed is only proportional to the size the number of elements in the hypothesis space more specifically of the order of the number of elements in S and elements in G, because only those elements only defines the abstractions needed. If you move one row up you can see that for the breath first you need to store of course the most specific generalizations in that case but actually also have to store all the negative cases because in every step you have to check that what the generalizations you and you proposed it need to be consistent with the negative. While in the depth-first search you have to really store all instances, because in every step you have to release both the positive and negatives. So this performance issue in a way repeats some of the differences between the two the three algorithms so actually this was the end of the second lecture of the fourth week and so the next lecture 4.3 will be on the topic decision free learning algorithms so thank you and goodbye.