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 8
Feature related issues

Welcome to this third lecture of the second week of this course on machine learning. This lecture will focus on features and issues related to features as you may already have understood in the earlier lectures, Feature engineering that means the way to decide on the appropriate set of Features to characterize objects within the Data set is a crucial issue for machine learning. So the idea here is to try to characterize the concept learning tasks, in terms of the relevant Features, Feature vectors and what one can call the object or Feature space. So the classical way of viewing a
scenario for a learning task is to define an appropriate set of Features, view each Data item in this data set as a Feature vector consider the feature or Object space spanned by the features, populate the feature space with the Feature vectors or Data items, find optimal multi-dimensional surfaces,
Hyperplanes in the object space that circumscribe the extensions of all the cost concepts involved. The engineering of features is crucial for the complexity of the object space and as a consequence also crucial for the complexity of their learning problem. We want to distinguish three basic cases
for Feature engineering : In Case 1 we have a reasonably well composed set of Features given based on domain theoretic considerations. In Case 2 we have a huge set of possible Features available, but these features have to somehow be reduced to a manageable size that can ensure an efficient learning process. The third case is that we have data items that are of non-digital nature it could be images, it can be sound, it could be other forms of representation and in that case the relevant features need to be extracted from the primary form of the data items as a separate process, typically this need to be done in a case-to-case fashion depending on the nature of the primary form and the character of the application. So now we will discuss the first and third case shortly, and then
more or less focus on the second case for the rest of this lecture. So now I'm going to discuss an example to illustrate the character of the first case and this example is fetched from a systematic example introduced in this week, the zoo dataset. So the problem in this case is neither a volume problem due to a large ungraspable set of possible features, because we talk about a limited set of features neither it is a representation problem caused by data items in non-digital form, because we
assume that we can define the features along we can have digital values of those features. So then the question is what are the problems here so, so if you look at the two columns in this

example you can see the feature that is the original Features of the data set from the standard repository fetch from the standard repository. In the second column you can see number of features that are inferred from the conventional zoological taxonomy used to characterize or classify animals. And the set of features indicated in the second column is derived more or less exactly from the complex taxonomy ranging from animal down to a specific kind of buffalo. So as you may see in two cases there is a correspondence, so the same the same features occur, in other cases you can see that there are differences. Obviously there is a reason for differences because in the second column the taxonomy is the taxonomy for a specific line in the animal taxonomy leading down to buffaloes and related animal and of course the optimal set of features related to that line is necessarily not optimal for all kinds of animals like fish and reptiles and birds and so on. But it's not a trivial thing to say whether the left column is the optimal one for the problem at hand or something like the right column may be more correct. So the bottom line here is for this kind of case where we have a reasonable amount features, for a reasonably well-known domain we still need domain based sanity check of the features in such a case. It's also very important that we have a terminological consistency here, that we use the same term it cannot be the case that we use one term like in in in one of the columns here and another in the other and make a mis-judgment with these are the same or different and so on. So the same characteristics need to be referred to in the same way in in any proposal competing proposal for the correct feature set this is more like common sense. But also it's very important that there is a clear feature definition for every candidate feature, so for example we have a splendid example of that I mean in in the first column in the bottom we have something like cats size which is absolutely not self-evident was it means actually looking at the data set cat's size means actually that this kind of animal have the size of the cat for larger, so this is a Boolean feature that'll be the very one if that kind of animal is considered being equal to or larger than a cat in size. But it's just an example that just by ranking a symbol like cats size is not evident what it means so homework here in any feature engineering task is, first of all see to that the terminology is so crystal clear I don't know it was the same term for the same kind of feature, and also that every feature is well defined, that's a starting point and still of course be many other issues to consider at this point I just want to mention these few. So the third case is the case where the data items of our data set are of heterogeneous and non-digital nature and we need a separate pre-processing to go from the initial form and map that form into relevant digital features. And this can be done in in several ways, so just look at the example here there are four images, so either you can have a fully manual process where a person looks at each image and infer a feature set for that image, so for the first the inference is that we talk about birds, in the second image it is a fishes, in the third there are mammals and in the fourth there are insects and when you go further there are so many birds of a certain size there are so many parts of another size for the Fishes we can do the same, so we can do a manual analysis of each image and from that image decide upon a description of that image in a digital form in terms of a manually produced features. The other extreme is that we have a totally automated process where a computer program a computer vision enabled program manages these images and automatically infer the most likely set of important feature characterizing that image And that could be things in between, that could be automated but still need to be some human intervention in this process of course in the end we want a description of these

images in something equivalent to the kind of feature set up we really looked at for the case number one. For sure every non digital form of representation demands its own analysis here, we cannot believe that we can use the same techniques for any form so for images we need certain techniques for sound we need other technique and so on in order to enable some degree of automation for this kind of case. So now we turn to case two, Dimensionality reduction or Feature Reduction so in this case we have a large set of possible features and we want to reduce them to a manageable size, so in most realistic cases the amount of possibly available features can be used to characterize data items is overwhelmingly large, in general we want to reduce the number of considered features the ground for removing the specific feature is, that it may be either redundant or an irrelevant and if so can be removed without causing loss of information the goal is obtained is to obtain an adequate set of informative relevant and non-redundant features still being able to describe the available dataset. The underlying motivations for dimensionality reduction can be summarized as follows: At 1 making models easier to interpret by humans, the smaller set of features is more easy to grasp for a human when looking at them all.
Avoiding the curse of dimensionality and we will come back to what that could mean. Third reducing the risk of overfitting and we will also go further into that and finally in general shorting the computation times for learning processes. Naively you can say that the more features we have to consider, the more costly the computation will be, so the term Curse of dimensionality refers to various phenomena that arise when analyzing and organizing data in high dimensional spaces, we talked about hundreds or thousands of dimensions problems that do not occur in low dimensional settings such as the three dimensional physical space and others. The term was coined initially by Richard E Bellman when he worked with problems in Dynamic Optimization. The common theme of the problematic phenomena is that when dimensionality increases the volume of the space increases so fast that available data perhaps sparse. This sparsity is problematic for any method that requires statistical significance, as the amount of data needed to support a result often grows exponentially with the dimensionality. Also organizing and searching data often relies on detecting areas where objects from groups with similar properties unite, however in high dimensional cases all objects appear to be spares and is similar in many ways, which prevents efficient data organizations. So now we turn to what is termed Overfitting vs. Underfitting. This case is being two of the phenomena most frequently affected by wrong or inadequate selection of features. So Over-fitting is the production of a model that corresponds to closely or to exactly to a particular data set and may therefore fail to fit additional data or predict future observations reliably. Typically an over-fitted model is a model that contains more features than can be justified by the data set and by the existence of all these features the current set of data is to exactly fit it. In contrast under-fitting occurs when a set of feature cannot adequately capture the available data set, typically an under-fitted model is a model where some features that would normally appear in a correctly specified model are missing, and as for a over-fitted case such a model will also tend to have a poor predictive performance. So in the example above, you can see two dimensional examples how it could look like but as you understand as for many other examples were given and the same can occur phenomena can occur in multi-dimensional cases. So finally we turn to the two concepts of Feature Selection and Feature Attraction. The concept of feature selection is pretty straightforward by being the process of selecting a

subset of relevant features from the original set and discarding the rest. The three main criteria for selection of a feature are: One, how informative is the feature, The second: how relevant is the feature and thirdly is the feature non-redundant. Actually relevance and redundancy are could be thought of as equivalent but not so because it may be so that two relevant feature are so similar or overlapping that one can consider one of them as redundant. I mean in general what we want to achieve is to have a few features as possible as long as we can discriminate in a good way among more data items in the data set and the categories in question. So features extraction is a little more complex it's rather the process, it's not the process of selecting anything or throwing something away, it's rather the process of deriving new features could either be as a simple combination of the original ones or it could be as a more complex mapping from the original set to a new set. When we started to talk here about always reducing the number of features but actually it could be so that without really reducing the number, one could create a set of more suitable features that simplifies the learning tasks so feature extraction in that way kind of more general because it captures all kinds of mappings from one original set of features to a new one, given of course that the new set is more useful for the learning task, so in both cases the learning touch is supposed to be more tractable in the resulting feature space than in the original independently if you use a selection procedure or use some extraction process. So this is the end of this lecture I want to run summarize by saying that I hope now you understood that feature engineering is the key ingredients in the area of machine learning, and that all the cases mentioned in this lecture are relevant, both the case where you have non-digital data items that have to be transformed into some digital form with a discrete set of features also the case where you already have a reasonable number of features but where they have to be judged in terms of domain relevance and in the light of a domain theory, and thirdly when you have a huge set of discrete features that have to be reused in order to be more optimal for the performance of the learning algorithms. So by this I want to thank you for your attention the next lecture will be on the topic of scenarios for concept learning so thanks and good bye.