[Music]

Welcome to the fourth lecture of the fourth week of the course in machine learning. In this lecture we will talk about instance-based learning, for practical reasons this lecture is divided into two videos and this is part one. The subtopics of the this lecture are as follows, first I will talk in in general about instance-based learning what it is, and say a few words also about the character of the so called instance space, then we look in more detail on a particular algorithm called the K-Nearest Neighbor algorithm and in conjunction with that discuss an important aspect of this kind of systems, the Distance and Similarity matrices. And then we will expand and look at a more general Neighbor K nearest neighbor algorithm called the Weighted Nearest Neighbor, after that we will go into a kind of different area and we will talk about support vector machines which is a kind of binary linear classifier ystem and we will I will talk about something called Kernel Methods where Kernel Methods is useful and in order to make machine learning algorithms that in basically handles the linear case also being able to have a nonlinear cases.

Instance Based learning is a family of learning algorithms that instead of performing explicit generalization, compares new problem instances with instances seen in training which have been stored in memory. As a consequence of that this kind of technique is also often called Memory based learning. The reason for the name instance based learning is because this kind of technique evaluate is cause it directly based on the training instances themselves, and not explicitly towards to build up hypothesis. One can say that instance based learning is a kind of lazy learning where the evaluation is only approximated locally and all computation is deferred until classification. tTere is no computation needed to build a hypothesis obviously. And in the worst case a hypothesis is the list of n training items, so therefore the computational complexity of classifying a single new items is at least of the order of n. One advantage that instance-based learning has over other methods machine learning is its flexibility to adapt its model to previously unseen data because actually the character of its classification change with a built up of the memory of instances. And an instance based learning may store new instances on or throw away instances, it depends in of course of the details of the algorithms. In many machine learning approaches the internal structure of the instance space is not explicitly considered. However the character means and space is always important it is always implicitly influenced the performance of learning algorithms, even if the character of the instance space is not explicitly considered in the algorithm design. But here can contrast for instance based learning the character of the instance base is of key

importance. In earlier lectures a few crucial aspects of the instance base have been mentioned, the number of it features, the value set of features, instances of special status, we talked about prototypes, outliers and near misses which are all structural aspects. And also we mentioned similarity distance measures which is a key issue here but we will also now talk about structural properties of the whole space such as spareness density etc. These aspects will now all come into play. I will now concentrate the discussion on one particular kind of instance-based learning algorithm all the k nearest neighbor algorithm KNN. In this algorithm, the analysis is based on the k closes training examples in the instance space, what I mean by closest is closest to the query item. k is a predefined positive integer that have to be set before you use this kind of method normally issues k to be small and odd. Potentially an optimal k can be calculated by special technique hyper parameter optimization techniques. So as you may have understood by now many of these parameters can also be learned that they the parameters control a learning technique but they the families themselves can be learned so special techniques. So this is a second-order kind of learning that is possible. The typical representation of any instance in this case it's a feature vector with a number attributes, or features plus labelled or a number which constitutes the target function we still here talked about supervised learning. So we still suppose that all our instances are labelled. The training phase is simply the storage of the feature vectors of all training in a data structure nothing more happens at that point. For this kind of approach we only is a distance metric that means a metric that that measures the distance between the instances, the default metric is Euclidean distance the well-known distance it's the distance between two elements in a Euclidean space, the square root of the sum of the differences between the Euclidian coordinates of the two elements. Normally also you select and define a metric like this one but also as for the number k, the metric can also be learned hypothetically. The k-NN can be used both for classification and regression and in the kNN classification case the output is of course a class membership. Here a query instance is assigned the class label, most common among its k nearest neighbors. If k is equal to one, of course then it's which is a special case then the instances assigned in the class of that singles nearest neighbour. And one can say in a way using an analogy the that all the k-nearest neighbors vote for the choice of the class label in the case of k being larger than one. And in k-NN regression, the output is the property value for the query instance, and this one is the average of the values of its k nearest neighbors. For example for practical reasons we will use simplest case which is binary classification in a feature space of two dimension. So to the right you can see just a summary of the stages in this kind of algorithm, you look at the look at the data, as per in the space, calculated

distances, find the appropriate number of neighbors and given that you have chosen the neighbors, you calculate the average of the outcome and take that as the class label in this particular case of making a classification task.

From this slide you can see our very simple illustration, over classification application for the k nearest neighbor algorithm. It illustrates the outcome of the algorithm for different values of k in this case value 1, 3 & 5. So as you can see for the inner circle there is own and you only look at the closest a neighbour, which in this case happened to be an instance classified as blue. And when you select the number three, you take consider three neighbors and then it happens to be that two more ready instance is coming to play, so the outcome the algorithm will be will be red. While if you consider five instances two more blue come into play, so then suddenly it walked over again towards the blue case of the outcome of the algorithm it will be blue. So this shows that the result outcome of this algorithm will clearly change depending on how many instances you view include in the process. It's obvious that in the case of instance-based learning there is only one explicit space, the instance space. As been said a hypothesis space is never explicitly built up. One can say that the hypothesis are implicit in the structure of the instance space as defined by the chosen matrix. One form of illustration for these implicit representations is the co-called Voronoi diagram. A Voronoial diagram is a partioning of the decision surface in two convex, polyhedral surroundings of the training instances. Each polyhedron covers the potential query instances possibly determined by training instance. And query point are also specifically is closer to another training instance and it's of course also included in some polyhedral. And this kind of diagram is not explicitly used for the processing on the algorithm, but it shows an important purpose in being able to illustrate for the user of the system, how the situation looks like. We exemplify here all the time by two dimensional cases, but as you understand it's natural to extend this kind of technique to dimensions larger than two, so you can see at the bottom to the right an illustration of that case.

One important feature of our aspect of instance based learning, is the existence of distance measure, similarity measure, approximate measure and it you have always to define this kind of measure for all instances in the instance space. And as you can see from the figure to do that to the right here, this means that that the instance space always have to be a metric space, this means a space with a matrix. And however the most typical case is that our instances are feature vectors, it doesn't necessarily have to be so because an instance can could be anything and you can also think of other kind of entities, on which you hypothyse a matrix, but the

very very common case is the feature vector. So therefore I want to talk a little on this slide, about two kinds of vector spaces that are well known from mathematics it's called a Normed space and an Inner Product Space. So the more general category is a kind of vector space that has something called a norm. So a norm is a real valued function that intuitively couple to a length or a distance. And a norm satisfies the number of key properties, one of them is number 4 here is a triangular inequality, this means that the distance is the norm of two vectors is always less or equal to the sum of the norms for each one of them. And we have a special notation for the very special case of a Euclidean norm, Euclidean norm as it is actually a norm, the traditional norm we know from a Cartesian space where the distance is the square root of the square root of the sum of the arguments or features of the vector. Obviously a distance here naturally is the norm of one vector minus the other. So this is this is normed space and then we can go further and we can look at normed space whether also exist an operation called the Inner product. And as you can see down there and then an inner product or dot product, in your Euclidean case they are equivalent, is first of all a scalar value and also the classical definition is that the inner product of two vectors are the product of the norm multiplied by the cosine of the angle between the vectors. And so on as a special case when the angle is 90 degrees, the cosine becomes zero, so then inner becomes the products as a consequence and we call this in this case the two vectors are orthogonal that's the terminology. So we need these concepts because when we define distance measures, we need both the cause of them norm and we also in some case the concept of inner product. And you can see in the picture to the right, inner product space it is a subset of the Normed vector spaces and Normed vector spaces is a subset of the Metric spaces. Let's turn then into Distances and Similarity matrices. So a distance metric, this is measure, function are synonyms, is typically a real valued function that quantifies a distance between two objects. So this is between a point and itself are zeros, of course all other distances are larger than zero, distances are symmetric and then we have a triangle inequality which informally means that detours cannot shorten the distance, so yeah it's always longer to go from one point to another by another point. So Distance and similarity metrics have been developed more or less independently so in certain cases, it's more practical to talk about the distance between one vector and another or a one point with another, in other cases is more natural to talk about how similar are things. So therefore depending on the context and these people have developed on one side distance metrics, on one side similarity metrics but obviously they are always connected intuitively. And obviously they are somehow inverses of each other and can be transformed into each other, but you will find and maybe it's a little confusing that's

sometime talk about this distance metrics separately and similarly define them separately, and but of course that can always be a transformation between them. So typically similarity metrics takes values in the range of -1 to 1, so they are always normed somehow, and where one means that the objects are regarded as identical and -1 means is the maximum distance considered by the corresponding distance metric. So if you have a same similarity of -1 there your are as far as you can, think of in that context. While distance metric often take values from 0 to infinity or anything, but of course there is always done some transforms and normalization to make distance a similarity matrix comparable. We will always here exemplify by using metrics in a normed Euclidean vector space matrices based on overlapping elements. So those are the two cases we will focus.

Now I want to give you some examples of distance measures so first we start with the category which is called the Normed and Inner product vector spaces. So these are kind of classical measures suitable for instance spaces where we have normal feature vectors. And now there's something Minlovsky distance which is maybe not so well known to you, but one can say that the Minlovsky distance is defined in such a way that it is becoming a umbrella concept for three more well-known the types of distances. So actually Minlovsky the distance is defined as you take the sum of the differences between arguments for two vectors, you raise each sum to the degree of k, you make the summation and then you take the kth root out of that sum. When one that looks at different values k, you can see that if you take the value k you get your Euklidean distance, so then you get simply the sum of the squares of the differences and then you take the square root out of that sum and the classical theorem of Pythagoras as a distance. Then so that's well known if you take k us number one, you get something else that is also pretty well known call the Manhattan the distance. Actually the Manhattan distance is then the sum of the absolute differences of the argument differences.And it's called Manhattan distance because if you consider you are moving around from one office in a row in Manhattan the most important thing is not the distance on the ground it's also the distance in the houses. So you go from the hundredth floor in one building to the 250 floor in another it's more important with the vertical distance and the horizontal, so that's a joke called as the Manhattan distance. Then we have a third case this is also rather well known that if we choose the value k to be infinity and then the formula deteriorates are actually ending in this limit of infinity we get actually the definition of the distance as the maximum value of the differences in our arguments. This distance is also called the chess board distance because actually this is equivalent to the to the minimum

number of moves the King can make on between two chess pawns. A special measure I want to mention that not only needs norms but also means inner product, is the cosine similarity measure, so actually the cosine similarity measure is equal to the cosine of the angle between the two factors and that can also be calculated as an equation in terms of square roots. It's interesting because in some cases when we compare feature vectors, this size, the length of the vectors are not necessarily important but the rather the angle between them, of course it's a little abstract here because if we have a feature vector and another feature vector but what does an angle mean, but there are obviously some domains where it makes sense primarily to look at the difference in angle and that's exactly what this measure measures.

So obviously this algorithm gets very different property depending on which similarity metrics you applied. So if you have a nice system where you can simulate this and you can easily produce Voronoi diagrams visualization, then you can then easily see that when you plug in one matrix you get one result and when you change you get a difference, so here you can see an illustration how you can how the Voronoi diagrams looks when you apply in a Euclidean matrix on an application and then what you get when you change to Manhattan distance for example.

Some of the similarity measures are pretty straightforward and easy to understand, but maybe the use of the cosine similarity is a little more unintuitive, so to give you an illustration of the use of the cosine similarity, I included an example here which I find pretty neat. It also shows actually it's a simple example that in many applications you have to map, you have to map properties of instances in a domain that is a different rather different form into a kind of feature vector. So this is an example of a domain where you want to look at medical texts or pharmaceutical tags or medical texts and you want to measure the frequency of certain kinds of terms. So the trick here is you take each text fragment or page or whatever is the unit of analysis and then you count the number of the key terms you want to measure, and then you choose as features or dimensions of your instance these terms, and the values of each such term becomes the frequency of that term in that particular text. So this it shows how you can take like a normal text and the page is of course the kind of raw instance data, and then you transform that text into this feature vector that just keeps the key terms and the frequencies. So then when you transformed the instances of that domain into a feature space and then you can apply these kind of methods, and then also intuitively it makes sense that the it's the relation here of frequencies it's not so important with the actual number,it's more important to see is it one term that dominates over the other, so intutively I hope you can understand that

that the difference is that the distance between these are more of the angle between the vectors because the angle is influenced by the relative size in the different dimensions, so therefore this is a case where the cosine matrix makes sense.

Metrics for feature vector spaces are important but not necessarily we have to be tied to that so sometimes it could be so that our instances are something else, so our instances could be sets, binary strings, texts and so on that's possible. So therefore I included here three examples of similarity measures that works on that kind of instances. So if you look at texts or words, we have something called the Levenshtein Distance which actually which actually embodies the idea of changes or substitutions that have to be made of single characters in the text or wording in order to create the other. So Levenshtein Distance is an example how your view can define a similarity matrix or distance matrix between text. Similarly when if the instances are sets and we have something called the Jaccard Index which measures actually then similarity between finite sets and it's defined as the intersection of the two sets divided by the union. So the final example is something called Hamming distance which is a classical distance from information theory, where you actually it's very similar of course to the language time distance but it's defined for binary strings. So actually the Hamming distance is the number of positions at which corresponding symbols are different. So it measures the number of minimum substitution required to make one the two strings equivalent.

Going back to the k nearest neighbor algorithm is automatic (32:35) before we leave it some issues that one have to consider for this kind of algorithm. So actually the theoretically the number of k can number k can be anything but typically one should avoid an odd number to avoid tied votes, very practical comment. And either you define k yourself or you can try to learn it and in the next slide we exemplify a kind of method it is also a kind of learning method how one can define or infer a suitable value of k through something called the bootstrap method. And another comment is that this kind of majority voting for deciding that can be problematic when the distribution is skewed, so of course this method is also dependent on on how the instances are distributed in the instance space. Also if you have a very large number of examples of one class and very few or another, it also can create an issue because of the density or of the dominating class can happen to be high in the context of a new example, so this can also create anomalies in the analysis. Also if you have a large feature set and where you haven't really done a thorough feature engineering many of these features can be are irrelevant and therefor those instance space expressed in these features can have strange properties of balance the relevant histories may be in one corner, the irrelevant

in another. So this may also cause problems and relating to that you can actually get a very spares instance space and it may be so that this idea to have exactly the same distant metric for every case may may have negative. So of course there are positive things with k2 neighbor but as you have understood from this line there are also a number of issues that can occur because the structural properties of the instance space can be skewed in many ways.

So last time I mentioned on method to mainly define optimal values of some of these parameters or hype parameter needed to be set for an algorithm like a k-near neighbor one. One such method is called the Bootstrap method and this method works in such a way that one takes small more samples out of the original dataset and you draw observations from the large sample one at a time, returning them of course because you won't disturb the dataset after this process the dataset have to be as it was so. This method allows you to use observations for a pre study and then you can as if nothing (35:54) happened you can then continue in the end with the original analysis is called sampling with replacement. So the booster method can be used to estimate the quantity of the population so you repeatedly take small samples, calculate whatever statistics you want ,so in y our case last slide you try to calculate prognosis for good k, then take the average of the calculated statistics.

So to summarize in the boot strap method, you choose a number of such samples you choose the size of the sample and the draw a sample with a replacement, you can latest statistics and then you take the calculated mean of the statistics, of course you know you have to be as precise this is general method so you have to define of course precise measures for calculating the statistics you're interested in, but it's a general scheme for making pre studies so to say of a data size, the data set in order to get a good prognosis for the setting of a hyper parameter. For practical reason not to make the video too long we make a break here, and we continue to talk about instance-based learning in part two. thank you bye