

Database Management System
Dr. S. Srinath
Department of Computer Science & Engineering
Indian Institute of Technology, Madras
Lecture No. # 35

Data Mining & Knowledge Discovery
Part II

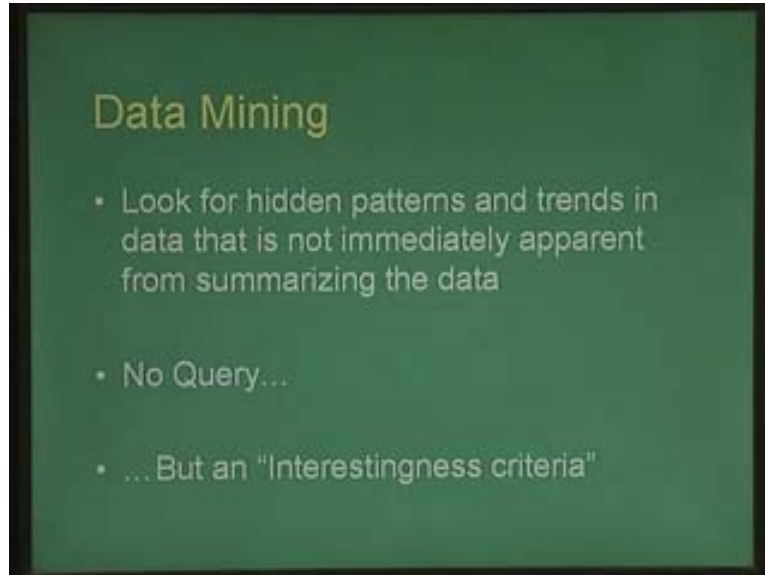
Hello and welcome to this second session in data mining. In the previous session we saw what this concept of data mining was all about and we saw some very fundamental concepts of item sets and association rules and how do you discover particular patterns in an item set. That is how do you discover something that you don't know from a data set using the concept of support and confidence and so on.

(Refer Slide Time: 02:06)



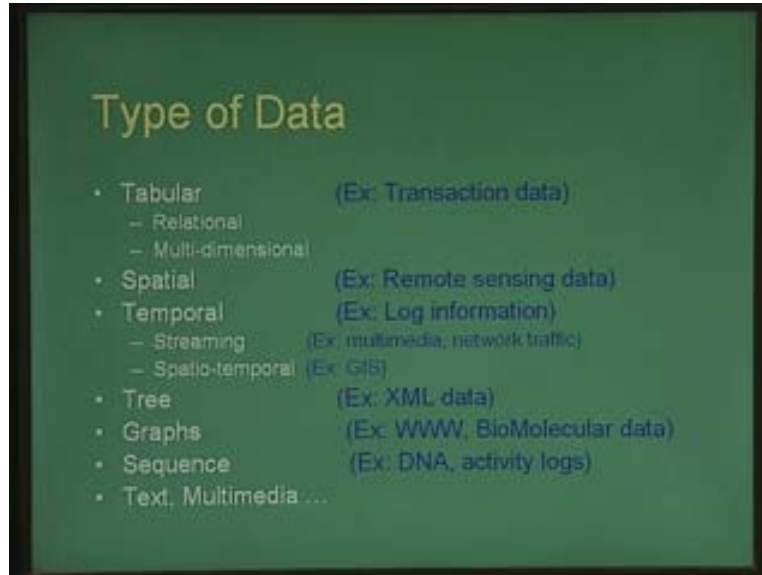
So, essentially you give a particular interestingness criteria and then you start distilling out certain patterns from the data set. Let us move on further in this session where we will briefly look into some fundamental algorithms or some very simple algorithms on different kinds of data mining activities namely in discovering classification trees or discovering clusters of properties of data and mining sequence data, the data of different sequences or stream data mining and so on. Let us briefly summarize what data mining was all about.

(Refer Slide Time: 02:23)



Data mining essentially is the concept of or is the idea of looking for hidden patterns and trends in data that's not immediately apparent by just summarizing the data. So when we say hidden patterns, its essentially means that something that we don't know about. There is nothing hidden if you already knew such a pattern existed in the data base. So in a data mining setting there is no query but we use the concept of an interestingness criteria. That is we use let us say frequency or consistency or rarity or whatever be the interestingness criteria and certain parameters define each of these interestingness criteria like frequencies is parameterized by support and confidence for association rules and just support for item sets and so on. And again there are different kinds of data we can think of tabular data, spatial data, temporal data, tree data, graph data and so on and so forth.

(Refer Slide Time: 03:24)



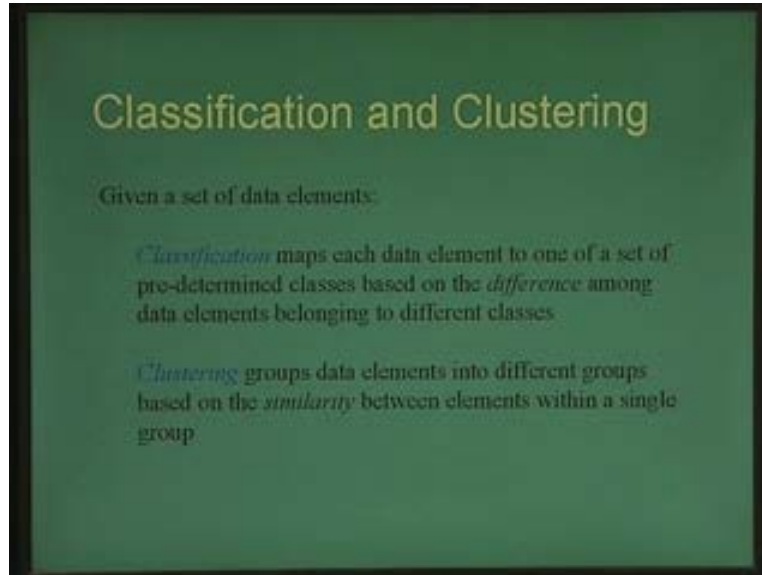
So today or in this session we shall look at specifically at sequence data mining and streaming or mining streaming data and in addition to other mining algorithms. And of course type of interestingness itself could be varied that we could talk of frequency as frequent patterns as being interesting or rare patterns being interesting and so on.

(Refer Slide Time: 03:48)



Now let us move further from here and look at the concept of classification and clustering that is discovering classification tree and discovering clusters within a given data set.

(Refer Slide Time: 04:03)



Now what is the difference between classification and clustering? Intuitively they both seem to do the same thing. That is, when you classify a given data set into different classes or when you cluster a given data set into different clusters, essentially few observe. Closely, classification maps data elements to one of a different set of pre-determined classes based on the differences between data elements. That is, if data element *a* and data element *b* belong to different classes if they are different enough.

On the other hand, clustering groups data elements into different groups based on similarity between elements within a single group, and sometimes it's also the case that in a classification, we know the classes a priori. We know what are all the different classes into which data can be classified, and sometimes in clustering, we don't know how many clusters we are going to get before the clustering process begins. Let us look at mining in relation to classification techniques rather; we are not interested here in the idea of classification itself, but we are interested in the idea of discovering classification. What is it meant by discovering classification? Discovering a decision tree or which decides how to classify data sets into different classes.

(Refer Slide Time: 05:19)

Classification Techniques

Decision Tree Identification

Outlook	Temp	Play?
Sunny	30	Yes
Overcast	15	No
Sunny	16	Yes
Cloudy	27	Yes
Overcast	25	Yes
Overcast	17	No
Cloudy	17	No
Cloudy	35	Yes

Classification problem

Weather → Play(Yes,No)

Let us take a small example. Discovering this algorithm is best represented by an example. So let us take a small example and see how we can discover a classification tree. Let us say that we have data about different cricket matches that have been played over the last several years. Now we have a, let us say in a given city. Now the question is this city is notorious **for it rains** for its rains and its unpredictable weather. Now in the past several times, play had to be abandoned that is play were to be continued or was abandoned and so on.

Now we have data like this from different data sets. When it was sunny and the temperature was 30 degrees, play was continued. When it was overcast and the temperature was 15 degrees play wasn't continued, when it was sunny and temperature was 16 degrees play was still continued and so on. So in some times play was continued and sometimes play was discontinued, its no. Now **what is** the classification problem is can I classify weather conditions which is a combination of the outlook and the temperature into one of two classification classes that is whether we are going to play or play is going to be discontinued. That is what is the criteria, when play was discontinued and what was the weather criteria when play was continued.

(Refer Slide Time: 07:32)

Classification Techniques

Hunt's method for decision tree identification:

Given N element types and m decision classes:

1. For $i \leftarrow 1$ to N do
 1. Add element i to the $i-1$ element item-sets from the previous iteration.
 2. Identify the set of decision classes for each item-set
 3. If an item-set has only one decision class, then that item-set is done, remove that item-set from subsequent iterations.
2. done

So there is a well-known algorithm called Hunt's method for identification of decision trees and like before let us first look at an example of how we identify a decision tree before looking at the algorithm itself.

(Refer Slide Time: 07:53)

Classification Techniques

Decision Tree Identification Example

Outlook	Temp	Play?
Sunny	Warm	Yes
Overcast	Chilly	No
Sunny	Chilly	Yes
Cloudy	Pleasant	Yes
Overcast	Pleasant	Yes
Overcast	Chilly	No
Cloudy	Chilly	No
Cloudy	Warm	Yes

```
graph LR; Sunny --> Yes; Cloudy --> Unknown1[ ]; Overcast --> Unknown2[ ]
```

The way of identifying decision tree is quite simple. First of all because this temperature field here (Refer Slide Time: 08:04) is a numeric value, it could take several different values and which might be of no interest to us. So let us perform a hand classification of this numerical values into different classes.

(Refer Slide Time: 08:19)

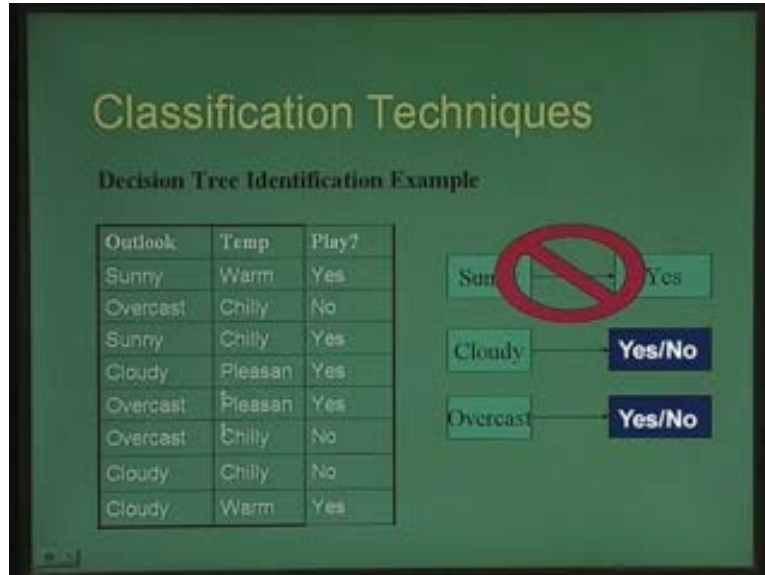


So what we have done here is that temperature is now classified into three different classes warm, chilly and pleasant. So whether the temperature was warm whether the temperature was chilly or whether the temperature was pleasant based on dividing the set of temperatures into different classes. Now first of all because there are two values here that is there are two fields here outlook and temperature, **both of them** both of them will affect the decision on whether we are going to play.

So how do we know what is the best or **how do each** how do each parameter affects the decision whether to play or not. Let us start by looking at one parameter after another. First let us look at sunny. Now if you see here that whenever the outlook was sunny, the cricket match was played it was not abandoned. It is sunny only twice here and in both cases cricket matches played. Therefore we can directly conclude that if the weather is sunny regardless of whether the temperature is warm or **whether the temperature** whether the temperature is chilly or whatever, we can conclude that play will continue, the play is not going to be stop.

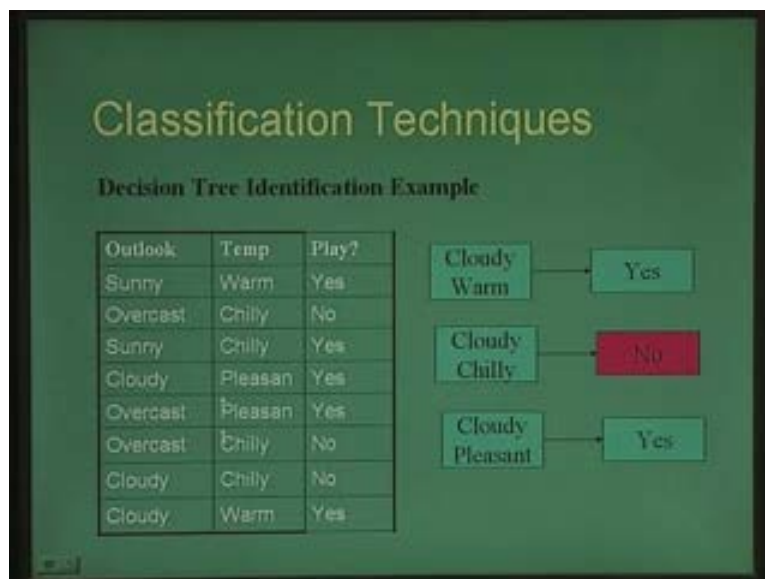
On the other hand let us look at cloudy here. Now when it is cloudy here play was continued in one case or rather in two cases and when it was cloudy here, play was discontinued in one case. So from cloudy we are still in a what is called as a bivalent state that is it is still yes or no may be or whatever, may be yes may be no, we still don't know. Similarly when the outlook was overcast, let us say here it was overcast and they didn't play. Here once when it was overcast they actually played and once more, when it was overcast then they didn't played. So, from overcast we still say yes or no, we don't know whether they are going to continue play or not.

(Refer Slide Time: 10:38)



So what we can do now is we can safely remove the first rule from our process that is this is a rule that we have already discovered that is when it is sunny they are going to play. So now let us remove this rule **from our** from consideration and take these two rules. Now because from cloudy and over cast, we are still in a bivalence state we have to ultimately reach to a state where we can remove this bivalence that is we can either conclude yes or no conclusively. So we will try to **we will try to** now introduce the second parameter temperature into this state here to see whether we can remove this uncertainty about yes or no. The first case the uncertainty is already removed, so there is nothing we need to do any more.

(Refer Slide Time: 11:32)



So we have introduced let us say here (Refer Slide Time: 11:41) for cloudy, we have introduced all three possible cases warm, chilly and pleasant, similarly for overcast warm, chilly and pleasant. So let us take cloudy and warm. So, whenever it was cloudy and warm there is only one case here play was continued, yes. So basically we have removed the bivalency that is we have conclusively stated that whenever it is cloudy but the temperature is warm, play is going to continue, we are not going to abandon play.

On the other hand whenever it was cloudy and chilly, there is only one case here where play was discontinued. So again there is the bivalency is removed that is cloudy and chilly means no. So we can again conclusively state that the play is going to be abandoned if the outlook is cloudy and the temperature is chilly.

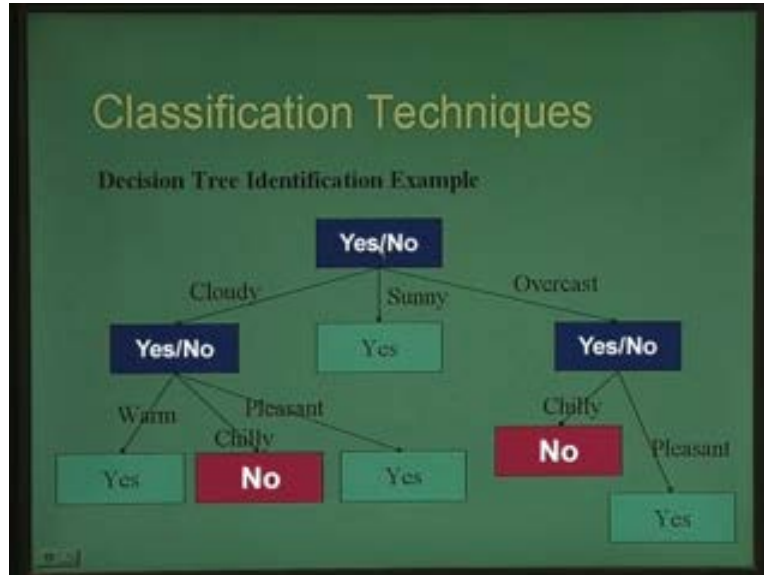
Similarly when it is cloudy and pleasant, cloudy and pleasant is here and there is only one case here, cloudy and pleasant is yes. So when the outlook is cloudy but the temperature is pleasant, we can still conclude that they are going to continue play. Similarly overcast and warm there is no entry at all, so we don't know there we can't decide anything. So overcast and warm remains as it is and overcast and chilly gives us no, that is play is going to be abandon. Similarly, overcast and pleasant gives us yes.

(Refer Slide Time: 12:53)



So effectively we have removed this bivalency that existed here (Refer Slide Time: 13:20) when it was cloudy and overcast and decided or came to know when, under what conditions play is going to be continued when it is cloudy and under what conditions play is going to be discontinued when it is cloudy and the same thing for overcast. So therefore what we have actually done is we have discovered this decision tree. So initially we were in a bivalent state that is we don't know play is going to be continued or discontinued. Now in this bivalent state we were told that the outlook is sunny then we can immediately conclude yes we are going to play today.

(Refer Slide Time: 13:53)



On the other hand if you are in this bivalent state here, if you are told that the outlook is cloudy, we will still be in a bivalent state we still don't know whether they are going to, whether the play is going to be continued or not. So we ask for more information and then when you find out that the temperature is pleasant, let us say for example then we say that yes the play is going to continue. On the other hand if the temperature is chilly then we have reasons to believe that play is not continued that is the data set tells us that play is going to be abandon and so on.

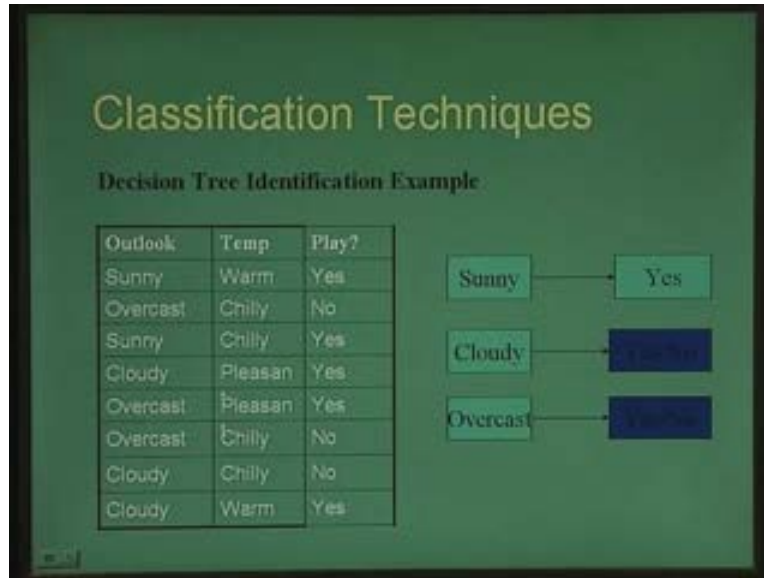
So what we have got here is a tree data structure where from a bivalent state, we eventually go into a univalent state that is a state were the uncertainty is removed and then we have concluded or we have classified his this play into two different classes that is yes or no that is play is going to be continued or play is going to be abandon. So let us look back (Refer Slide Time: 15:11) at the algorithm little bit as how to go about this. Suppose we are given n different elements.

In our case in the example that we right now saw, n was equal to 2 that is outlook and temperature. so suppose we are given n different element types and m different decision classes, in this case again m was two that is yes and no. so what we do in this loop here, for each of the different element types we keep progressively adding element i to the i minus oneth element item sets from the previous iteration. And then whenever and then we see whether we can decide, identify the set of all decision classes for each such item set.

If the item set has only one decision class that means we have already decided. so this is done, removed that item set from subsequent iterations otherwise keep continuing until you finish all your element types.

And of course it could well be the case that even after finishing all my n different item sets, I may not be able to reach a conclusive decision.

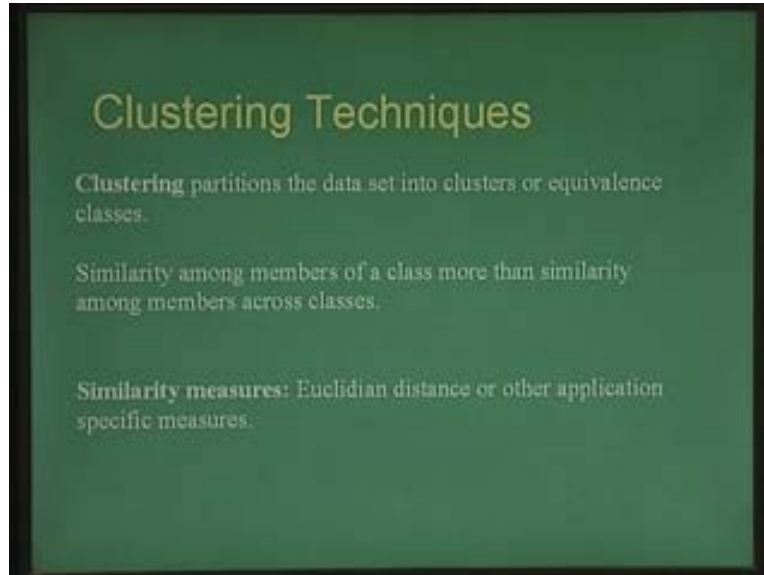
(Refer Slide Time: 16:07)



So it might well be the case that when it is over cast and chilly. Sometimes they actually play and sometimes they didn't play and so on. so that again, there are several methods to deal with such kinds of indecisiveness for example to use probabilities that is this is going to or some kind of fuzzy classification where we say that outlook is overcast and temperature is pleasant then they are going to play with a probability of 90% or something like that.

So let us look further into what are some clustering techniques (Refer Slide Time: 17:03). Now what is meant by clustering or how does it differ from classification? We saw earlier that there is a philosophical difference between classification and clustering, probably not in the n result but philosophically there is a difference. Of course even in the end result there are differences but the most marked difference is philosophically. That is classification is based on amplifying the differences between different elements so as to make them belong to different classes.

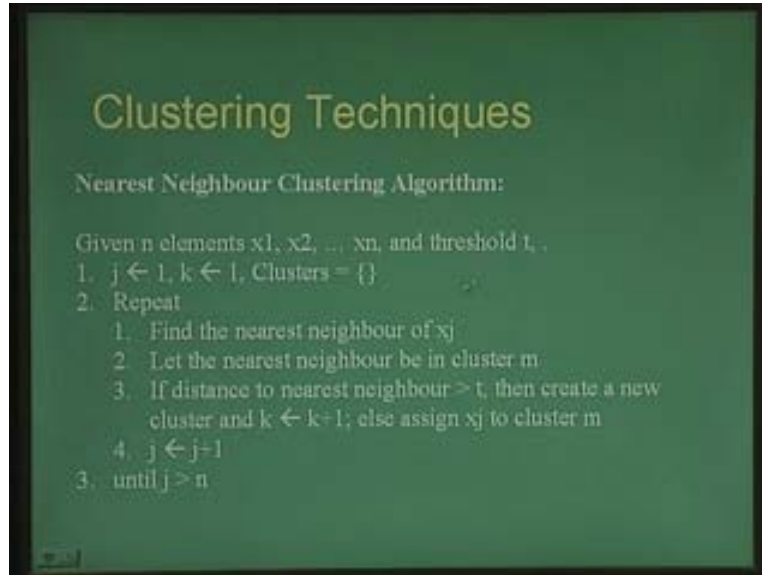
(Refer Slide Time: 17:48)



On the other hand clustering is based on amplifying the similarities between elements so as to form them into different clusters. So clustering essentially partitions the data sets into several clusters one or more clusters or equivalence classes. And what is the property of a cluster or an equivalence class? Essentially the property here is that the similarity among members of a given class in a cluster is much more than similarity among members across clusters.

So members belonging to the same cluster are much more similar to one another than they are to some members belonging to some other clusters. And there are several measures of similarities and most of which are reduced to geometric similarity by projecting these data sets into hyper cubes or n dimensional spaces and then use some kind of Euclidian distance or other kinds of distance measures like Manhattan distance and so on and several distance measures to compute the similarity.

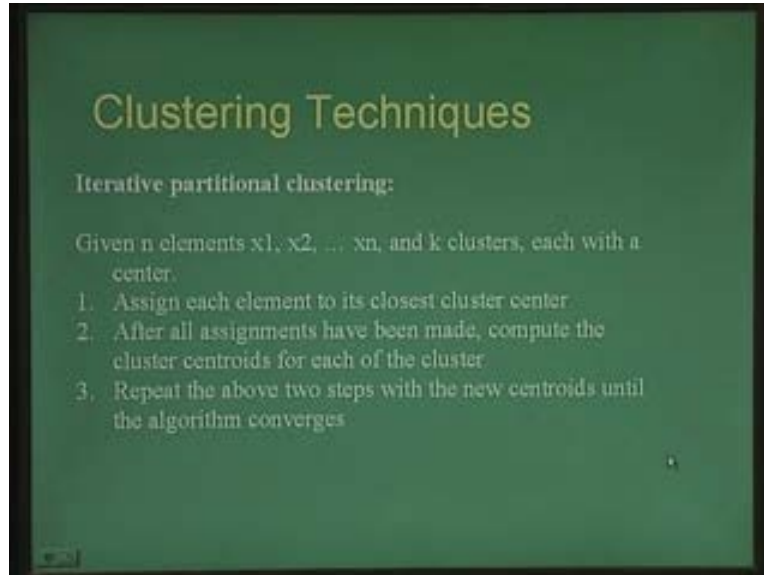
(Refer Slide Time: 18:58)



Let us look at the first kind of clustering algorithm which is called the nearest neighbor clustering algorithm. This is quite simple that is this clustering algorithm takes a parameter called threshold or the minimum distance or the maximum distance t between members of a given cluster. So given n elements that is x_1, x_2 to x_n and given a threshold t which is a maximum distance that can exist between elements of a cluster, we can find clusters in a very simple process. Initially the set of clusters is a null set. Then for each element let us say j equal to 1 here and j goes to, until j plus one here for each element find the nearest neighbor of x_j .

Now let the nearest neighbor be in some cluster if it is already in a cluster, if it is not in a cluster then fine you can just create another cluster by yourself. So suppose the nearest neighbor is in cluster m . now if the distance to nearest neighbor is greater than t that is if it is greater than threshold then we know that there is no other element that is nearer to me with a distance less than t . therefore I should belong to a new cluster so then create a new cluster and increment the number of clusters else assign it to the cluster m were the nearest neighbor of it existed. So, as simple as that. That is given a small threshold, you basically start partitioning your set of elements into different clusters based on which is the nearest neighbor to a given element. If the nearest neighbor is within this threshold distance then I join the cluster, otherwise I belong to a new cluster.

(Refer Slide Time: 21:03)



There is another kind of clustering techniques which is again quite popular which is called as the iterative partitional clustering. This is another clustering technique where this differs from the nearest neighbor technique in the sense that here the number of clusters are fixed apriori. In the nearest neighbor technique or in the nearest neighbor clustering techniques, the number of clusters are not fixed apriori that means you don't know how many clusters you are going to get, given a particular threshold and a data set.

So this is very much unlike classification where we know the classification, where we know the classes under which data can be classified into. In iterative partitional clustering, the number of clusters are already known apriori and then we are trying to rearrange the clusters that is **but that is** we don't know how many or what elements belong to which clusters. So, given n different elements and k different clusters, each with a center. What do we mean by a center here? It's the centroid in the statistical sense, for example it could be the first centroid. That means if a cluster has several features, the average of all these features along all different dimensions will form the centroid of a given data set.

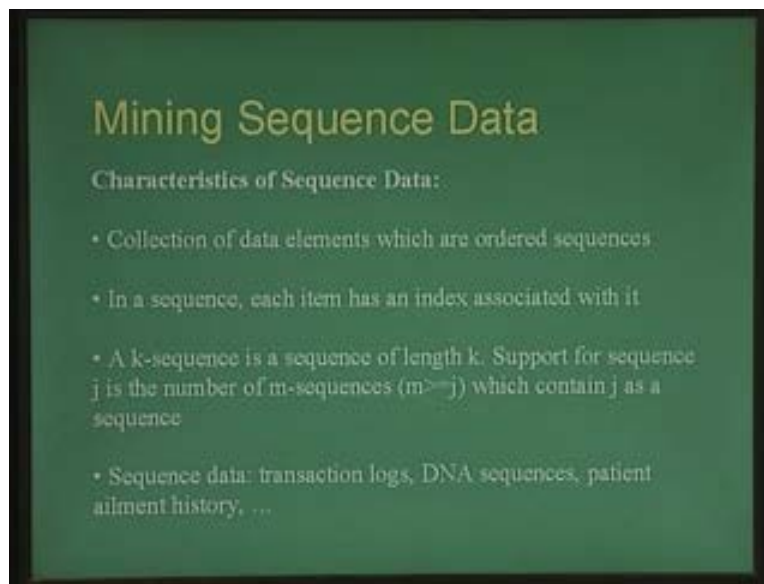
So let us say we have k clusters each with a center. Now assign for each element, assign it to the closest cluster center. So each clusters has a cluster or a centroid. For each element, find out which is its closest cluster center and assign it to that cluster. After all assignments have been made, compute the cluster centroids for each of the cluster. That is compute the average of all the points that made up this cluster and possibly this will shift the centroid to a different to a different location. So once this centroid is shifted to a different location, the nearest centroid or the nearest cluster center will now differ for each element.

Therefore we keep repeating these two steps, until the new centroid I mean with a new centroids that are formed until the algorithm converges. That is until the algorithm

stabilizes so that the centroids will stop shifting and then we know that we have found the exact or we have found the best centroids for each of the clusters, each of the k clusters. so iterative partitional clustering essentially is a technique were something like saying, suppose I have a data set and I say that suppose I want to create 10 different clusters out of this data set, where would these clusters lie and so on.

On the other hand, a nearest neighbor clustering technique would say suppose I have this data set and suppose I have a maximum distance, a threshold distance of 5 between elements that can lie within a data set then how many clusters will I find. whereas in the in the iterative clustering algorithm, we are interested in where the clusters are going to be, where are the cluster centroids of these 10 different clusters that are going to be formed.

(Refer Slide Time: 24:52)



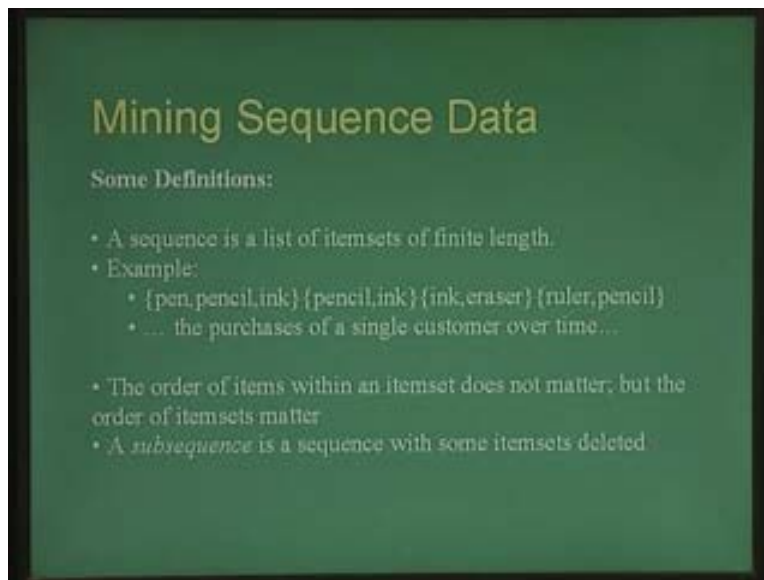
Let us now move on further and look at different other kinds of data sets. We have been looking into, until now we have been looking into let us say the tabular data as in apripr or association rule mining or some kind of multi-dimensional data. Tabular data can be treated as multi-dimensional data as long as they belong to certain ordinal classes which is of course beyond the scope of this session here that is how do we convert a tabular data into multi-dimensional data. But any way as long as the data can be converted to multi-dimensional form, we can use clustering techniques for clustering them into different clusters.

Similarly tabular data can be used to also infer classification trees. Let us now move on to different kind of data what is called as sequence data. What do we understand by the term sequence? Sequence is essentially a collection of data elements wherein it's not just the collection, it's an ordered collection that is where in the ordering matters.

That is in a sequence each item in a sequence has an index associated with it. That is some kind of a subscripted element, each element is a subscripted element. So this is the first element, this is the second element and so on. So when we say we have a k sequence, it means that we have a sequence of length k that is there are k different elements in a particular order in this.

there are different kinds of sequence data like for example any kind of transaction log over a period of time or let us say some kind of web browsing logs, http logs or DNA sequences or the patient history, the medical history of a patient over time that is how is the history changing or what kinds of events happened and so on. So all of these are sequence data.

(Refer Slide Time: 27:13)



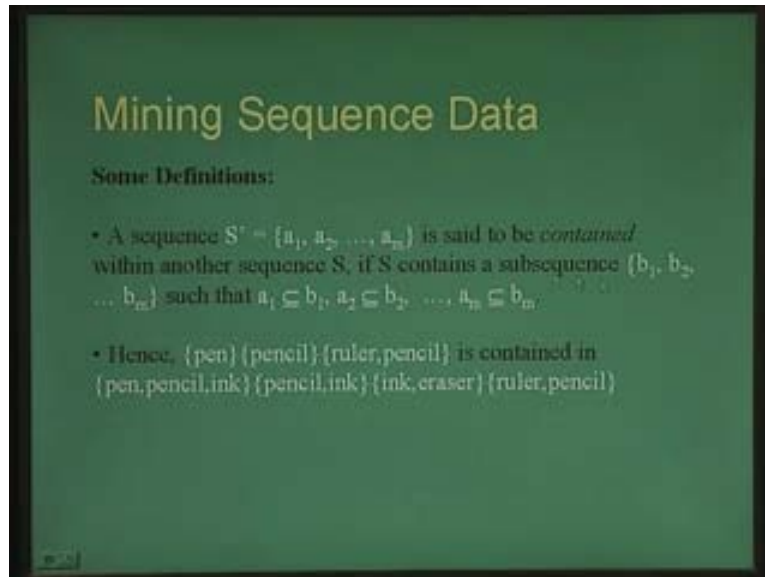
So let us look at some definitions in mining sequence data and which help us in formulating algorithm for looking at patterns in sequence data. First of all when we talk of a sequence, a sequence is essentially a list of item sets of finite length that is each element in a sequence need not be atomic, it could actually be a set, it could actually be a different set of items. So for example this is the sequence. The first element here is pencil, pen, ink or pen, pencil, ink. The second element here is pencil, ink. The third element is eraser, ink and so on and the fourth element is ruler, pencil and so on.

So this sequence essentially for example could be denoting the purchases of single customer over time in this particular store or whatever. So let us say the customer came in the first month and purchase these three things, the second month you purchase these two and the third month you purchase these two and so on in some stationary store.

Now the order of items within an item set here does not matter but the order of item sets itself matters. That is this is the first month, this is the second month, this is the third month, so the position of this item set matters but the position of items within an item set

doesn't matter. So whether I read this as pencil, ink or ink, pencil it doesn't matter. And we define the term sub sequence, as any sequence with some item sets deleted from it. So, some more definitions. Suppose I take a sequence a_1, a_2 until a_m , this is actually a sequence it's not a set, so this curly braces should actually be a, it should not be there.

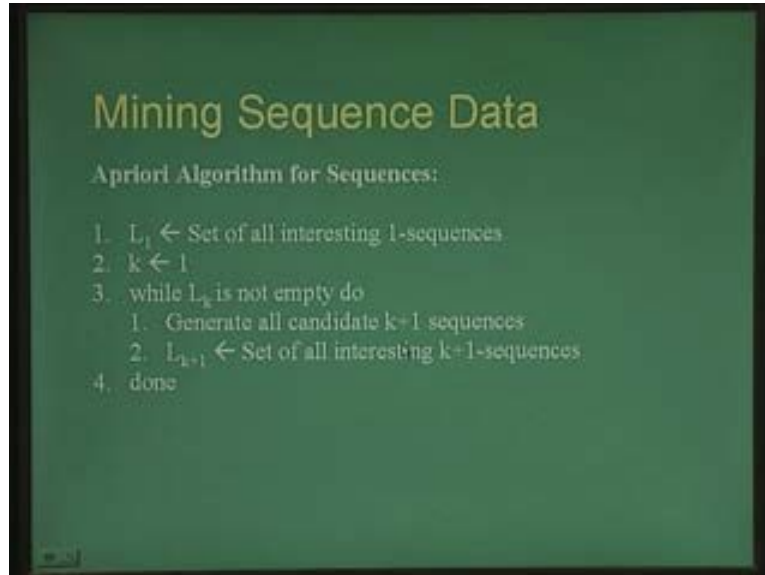
(Refer Slide Time: 28:49)



So suppose I take a sequence s prime $a_1 a_2$ until a_m . we say that s prime is set to be contained within another sequence s , if s contains a sub sequence of the form $b_1 b_2$ etc b_m that is m different elements such that each corresponding element is a subset, a_1 subset of b_1 subset equal to rather and a_2 subset equal to b_2 and so on. So, hence for example this sequence pen, pencil and ruler pencil is contained in this sequence. That is pen is a subset of this, pencil is a subset of this and suppose you take this out and create this sub sequence pen, these three as a subsequence then ruler pencil is a subset of this one.

So, let us look at the apriori algorithm. I think called the apriori gen algorithm or whatever apriori all algorithm where it is applied for sequence data rather than item sets or association rules.

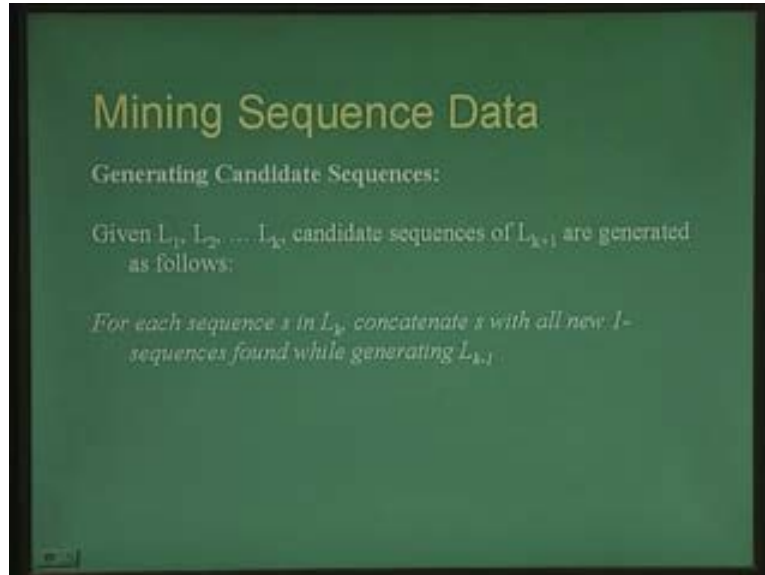
(Refer Slide Time: 30:06)



The apriori algorithm for sequences looks very similar to the apriori algorithm for item sets as well. How does the apriori algorithm look? First of all we set, we generate L_1 that is the set of all interesting one sequences. What is the one sequence? A sequence containing just one element. And then when L_k is not empty when k equal to 1, we generate all candidate k plus 1 sequences and out of these, we take only the set of all interesting k plus 1 sequences.

What is interesting k plus 1 sequence here? It is simply the set of all k plus 1 sequence which have at least the minimum support that we have specified and so on. Now the main question here lies in this statement here 3.1, that is how do we generate or what is the candidate generation algorithm? How do we generate all candidate k plus 1 sequences?

(Refer Slide Time: 31:14)



Mining Sequence Data

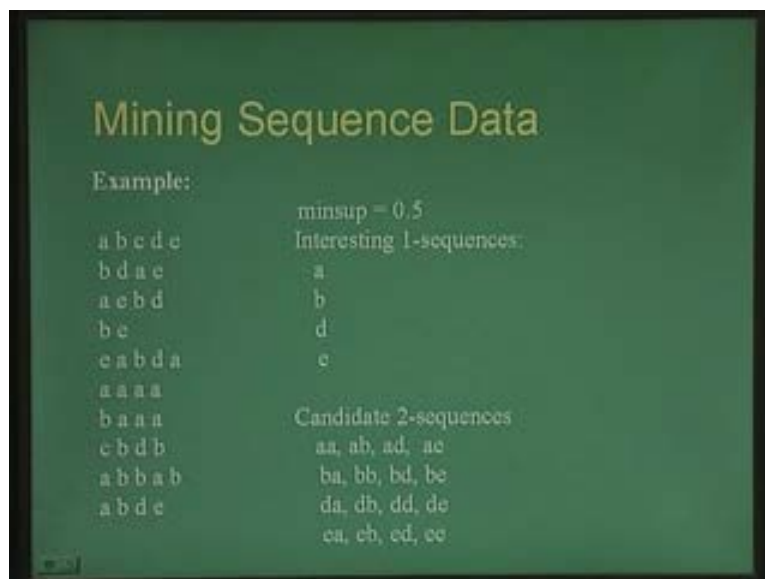
Generating Candidate Sequences:

Given L_1, L_2, \dots, L_k , candidate sequences of L_{k+1} are generated as follows:

For each sequence s in L_k , concatenate s with all new 1-sequences found while generating L_{k+1} .

So how do we generate all candidate algorithms? Now given let us say different interesting sequences that is L_1, L_2 until L_k , candidate sequences of L_{k+1} are generated simply by concatenating all sequences in L_k with all new one sequences found while generating L_{k+1} . What is this mean? Let us illustrate this with an example.

(Refer Slide Time: 31:46)



Mining Sequence Data

Example:

a b c d e	minsup = 0.5
b d a e	Interesting 1-sequences:
a e b d	a
b e	b
e a b d a	d
a a a a	e
b a a a	Candidate 2-sequences
c b d b	aa, ab, ad, ae
a b b a b	ba, bb, bd, be
a b d e	da, db, dd, de
	ea, eb, ed, ee

Let us say this is my data set and this data set let us say denotes, let us say I have a website and this data set denotes which are all the different pages that have been visited by users in different usage sessions. So one user a went from, one user went from page a

to b to c to d to e and so on. Another user came from b and went to d and a and e and so on like this.

So we have different sequences and of course as you can see here that an element can repeat in a sequence that is this user has requested for the page a 4 times one after the other and same thing here (Refer Slide Time: 32:55) that is after b, a is requested three times and so on for whatever reason. Now from here in order to look at, in order to mind for all interesting sub sequences that is what will be visited before what in this data set, let us start with the set of all interesting one sequences. Now we have set a minsub as 0.5 that is at least 50% of support. Now let us look at the set of all interesting one sequences. What is it mean to say interesting one sequences? Essentially it means that which all sequence of length one have appeared at least 5 times or more. So a has appeared 1 2 3 4 5 6 7 8 times in 8 different sequences, b has appeared 1 2 3 4 5 6 7 8 9 different times and so on. So a b d and e are interesting one sequences, c for example has appeared just once here, so therefore it is not interesting at all as a one sequence.

Now we generate all possible candidate two sequences that is it is now rather than a combination, it's a permutation that is where the order matters. So aa and rather it's not a permutation, it is a concatenation rather that is concatenation of all possible concatenations that are possible between elements of this one. So ab is different from ba and ad is different from da and so on. So these are the set of all candidate two sequences. Now we just see which of these candidate two sequences have minimum support.

(Refer Slide Time: 34:34)

Mining Sequence Data

Example:

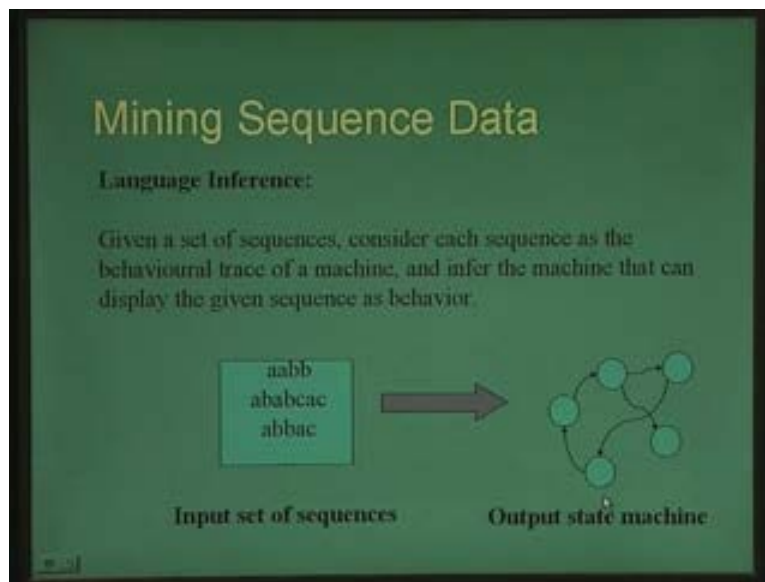
a b c d e	minsup = 0.5
b d a e	Interesting 2-sequences:
a e b d	ab, bd
b e	Candidate 2-sequences:
e a b d a	aba, abb, abd, abe,
a a a a	aab, bab, dab, eab,
b a a a	bda, bdb, bdd, bde,
c b d b	bbd, dbd, ebd.
a b b a b	Interesting 3-sequences = {}
a b d e	

Now among these you see that only ab and bd have a minimum support of 0.5. That is all others aa for example has the minimum support of 1 2 3 that's it, not 0.5. That is one is here rather 4, 1 2 3 and 4, ab also has minimum support less than 5 and so on.

So the only set of interesting two sequences are ab and bd in this case. So we have got the set of all interesting two sequences. Now how do we generate the set of all interesting three sequences that is candidate three sequences? We concatenate ab and bd with all the interesting one sequences found in the previous iteration. So the previous iteration here is still the one sequence here ab d and e. therefore we concatenate both of this with a b d and e like this and then we see that there are no interesting three sequences at all and then the process stops.

Otherwise we would have filtered out few more elements here and then out of these, again we would have concatenated with all possible interesting one sequences that we found in the previous iteration. So here the interesting one sequences that we have found in the second iterations are a b and d. So for level 4 there is no need to concatenate it with let us say e, so it's enough if we just concatenate with a b and d.

(Refer Slide Time: 36:34)

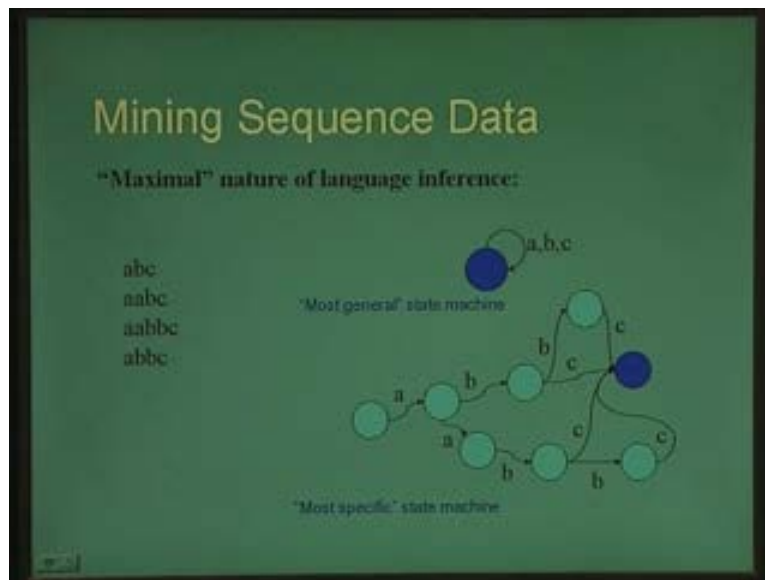


With sequence data there is an other kind of interesting mining problem that occurs, when we look at a sequence data as a behavioral pattern. See for example when we say this is the way that users behave in a data, user behave in a website. The user here comes to page a then goes to page b then goes to page c, d and e and so on. Now we are encountered with a question as to can we model the behavior of the user. What would be a model that would explain me how users behave on my website?

So what this means is that we have to find out, suppose these are all the different strings of a given hypothetical machine, we have to find out some machine which can generate all of these strings and possibly other strings that belong to the same class in whatever sense that is. So the question here is that given different sequences, treat this different sequences as strings that are generated by a particular machine. The simplest kind of machine that we can generate is the state machine or the deterministic finite automate or the finite state machine or whatever.

Now but that doesn't mean that everything can be modeled by a finite state machine but it's purely because of complexity considerations or practical considerations that we assume that the model representing user behavior is given by a finite state machine. So given a set of input sequences, we have to find out what is the finite state machine that recognizes this class of input sequences. This also called as language inference that is given the strings of a language, you are trying to infer the grammar of the language or you are trying to infer the structure of the language. Now what is the problem in language inference? **What is the big**, where is the trickiest problem that occurs in language inference?

(Refer Slide Time: 38:39)



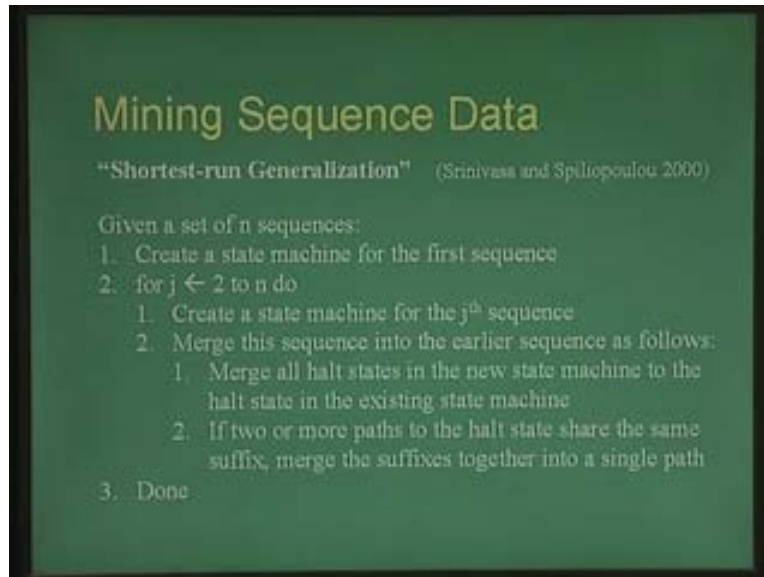
Take a look at these strings. Let us say I have these four strings abc, aabc, aabbc, abbc so on. Now if I want to give you these four strings and tell you that create a state machine that will recognize these four strings. It is quite obvious that one would come out with the state machine like this **which says** which accepts these four strings and exactly these four strings, so abbc, abc and aabbc and so on. So which accepts exactly these four strings.

On the other hand, one can also write a machine like this comprising of a single state which leads on to itself and accepts all strings like this. So this is a most general state machine that is this state machine is also correct in a sense that it accepts these four strings but it also accepts anything else made of a b and c in addition to these four strings, while this is a most specific state machine. That is this is a state machine that accepts these four strings and these four strings only and nothing else.

Now the challenge or now the trickiest problem in language inference is to find the right kind of generalization. That is if we make something into a most specific state machine, it will be of no use, while we make something into a most general state machine, it will be useless as well.

So when we discover or when we try to discover a model of user behavior, we should discover a model which is not too specific and is neither too general, it has to have the right kind of generalization. How do we do that? There are several different algorithms that try to generalize a little bit and not too much and not be too specific and so on.

(Refer Slide Time: 40:34)



Mining Sequence Data

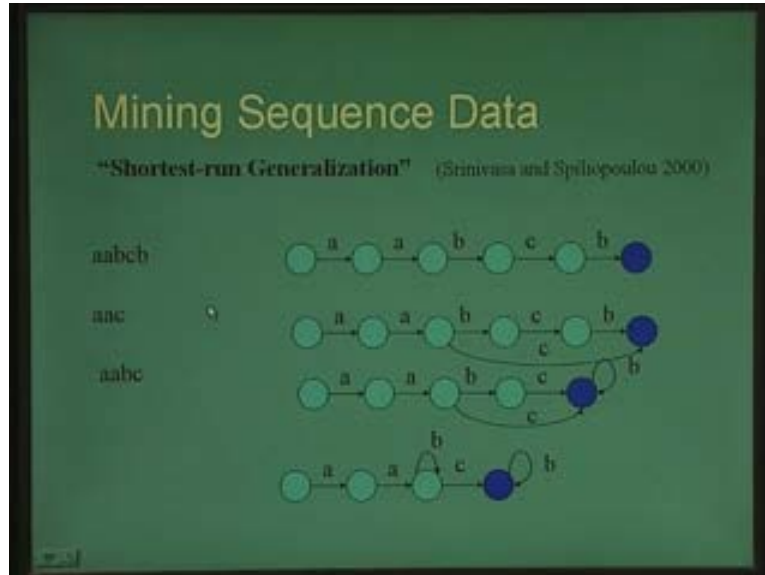
"Shortest-run Generalization" (Srinivasa and Spiliopoulou 2000)

Given a set of n sequences:

1. Create a state machine for the first sequence
2. for $j \leftarrow 2$ to n do
 1. Create a state machine for the j^{th} sequence
 2. Merge this sequence into the earlier sequence as follows:
 1. Merge all halt states in the new state machine to the halt state in the existing state machine
 2. If two or more paths to the halt state share the same suffix, merge the suffixes together into a single path
3. Done

We will just look at one specific algorithm which might be termed as the shortest run generalization that is generalize based on behaviors by using what is called as a shortest run technique of this thing. Now as we did for the previous algorithms, let us first look at the example and then come back to the algorithm.

(Refer Slide Time: 41:21)



Now the way shortest run generalization works is shown in this state machine here. Now let us say that we encountered different strings. Now let us say this is the first string that we encounter aabcb. Now there is no other string therefore we just build a state machine like this which accepts only aabcb and we haven't seen anything else, so we can't generalize anything else. Now second we encounter the string aac. So what this means is this state machine should accept not only aabcb but also accept aac. What does this mean? This means that start from aa and after aa if I get a c I can go directly to the end state, so it has to accept not just aabcb but also aac.

Now let us say that I get the third string, even here I won't be able to generalize anything. This is the state machine that accepts aabcb or aac, so we still haven't generalize anything. Now let us say I encounter one more string of the form aabc. Now what is this mean? This means that aabc that is this string, that is this is a prefix of this thing. That is this is the substring of this thing, this is the prefix of string of the first one. So aabc this state itself should be an end state. So basically we come like this here and aabc this becomes the end state.

Now what we do is we merge both of these end states, so b comes back like this. When we merge these end states, note that we have performed a specific particular generalization here. Now what is this machine recognize? This machine recognizes aabc b* that means any number of b's after aabc. So essentially what it sees is that or any number of b's after aac as well. That means it has seen a b appear after aac that is this substring aa and c with or without b included, it has seen that b may appear or not appear. And it generalized to the fact that any number of b's may appear, including 0 number of b's which may or may not be right that means to say that there might be an implicit, there might be some more hidden variables that says that at most 3 b's can appear let us say 0 1 2 or 3 b's can appear not 4 b's but we don't have that information here as such.

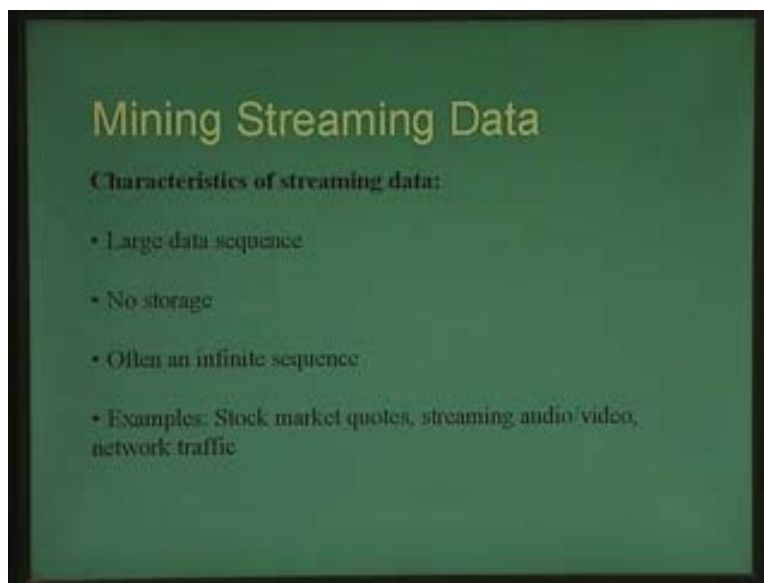
So basically the state machine generalize to the fact that after aabc or after aac zero or more b's can appear and we still lie in the end state but then we also see that when we look at the end state here, we look at the tails of all the edges coming into the end state. so there is a tail here which says c and there is a tail here which says c. now whenever from the end state it finds that there are two or more tails having the same suffix, these two the corresponding states are also merged.

So what we finally get is $aa b^* c b^*$ so that means what the machine **generally is** actually saying is that this language has to have two a's to begin with, so it has two a's and it can have 0 or more b's following two a's and then it should have a c and then it can have 0 or more b's and so on. So because it has found 0 or 1 b's between a and c and it has found 0 or 1 b's after this c, it has performed this generalization. So this is one way of performing or trying to discover the behavior that is exemplified by a set of sequences.

Let us look at the last kind of data set for this session namely streaming data. Streaming data has been of relatively newer interest among the data mining community and especially since the streaming data or mining on streaming data has several interesting applications.

Now what is the characteristic of streaming data, what you understand by streaming data? You have let us say streaming audio, streaming video, network traffic and sever several other such data sets which are essentially large data sequences possibly infinite data sequences. in practice of course there are finite but possibly infinite data sequences and there is no or very little storage that is it is not practical to say that I am going to store the entire streaming data into a file and then start mining the file.

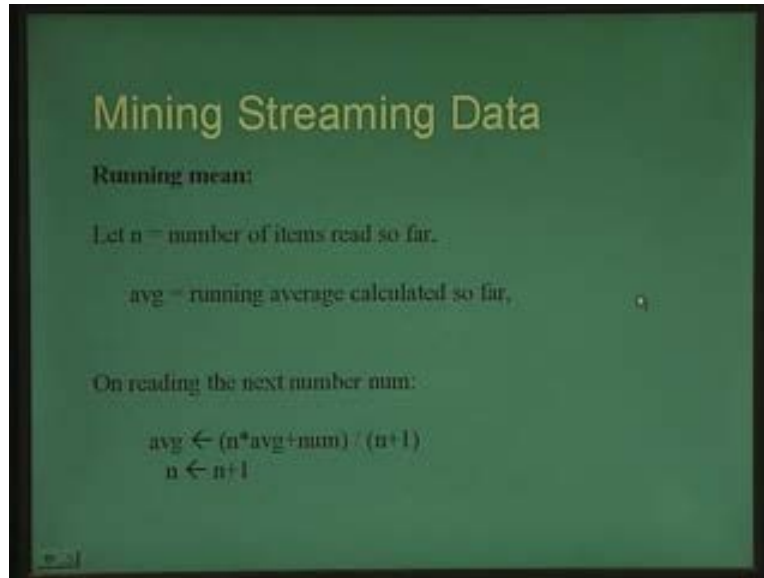
(Refer Slide Time: 46:53)



Because this if it is infinite or if it is extremely large, it will be impractical, it could be tera bytes or even more bytes of data that could eventually accumulate into the file. So

some examples are stock market quotes or streaming audio or video or network traffic and so on.

(Refer Slide Time: 47:12)



Mining Streaming Data

Running mean:

Let n = number of items read so far.

avg = running average calculated so far,

On reading the next number num :

$$avg \leftarrow (n * avg + num) / (n + 1)$$
$$n \leftarrow n + 1$$

So in order to mine streaming data or rather even in order to let us say query streaming data, there is a notion of what is called as running queries or also what are called as standing queries. That means in a traditional database the data is standing, the data is there and the query actually slides through the data set in order to return you the answer. But in a streaming data set it is the query that is standing and the data streams through the query and then the query keeps returning you answers as and when the data streams through it.

So how do we write some standing queries or how do we find some aggregate behaviors based on some standing queries? Let us look at some simple standing queries, computing the running mean of a data stream. That is suppose I am getting a stream of different numbers and I have to calculate the average of these numbers as and when I read a new numbers, so it's a running mean. So a simple way to calculate this running mean is like this, let us say I just need to maintain two variables here. One is the number of items that I have read so far or the number of numbers that I have read so far and the running average that I have calculated so far.

So whenever I read the next number, all I need to do is first compute n times average that is average times the number of numbers that I have read so far, add number to it and divide it by n plus 1 and then increment the number of numbers that you have read or the number of items that you have read that is n equal to n plus 1, so as simple as that. That is as soon as a new number comes, you generate the sum, see n times average is basically the sum of all the numbers that have come so far. So generate the sum here, add the new number and divide it by the new that is number plus 1, n plus 1 as the new set of numbers that have come and then increment your set of numbers.

Similarly this slide shows how to write a running query that computes the running variance. Variance as you know is the square of the standard deviation of a given data set. How do you compute standard deviation? That is it is for every element x , compute x minus \bar{x} that is number minus average whole square and compute the sigma or compute the sum over all of them, all of these differences, so mean square distances essentially.

(Refer Slide Time: 49:34)

Mining Streaming Data

Running variance:

$$\text{var} = \sum(\text{num} - \text{avg})^2$$

$$= \sum \text{num}^2 - 2 * \sum \text{num} * \text{avg} + \sum \text{avg}^2$$

Let $A = \sum \text{num}^2$ of all numbers read so far
 $B = 2 * \sum \text{num} * \text{avg}$ of all numbers read so far
 $C = \sum \text{avg}^2$ of all numbers read so far
 $\text{avg} =$ average of numbers read so far
 $n =$ number of numbers read so far

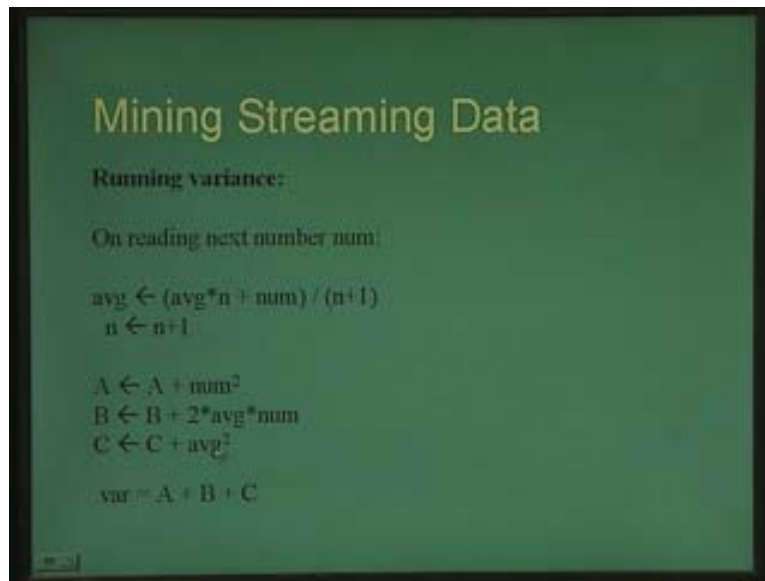
So in order to compute the running variance, we look at this formula little more carefully. Variance equal to sigma of number minus average whole square where number ranges from i equal to 1 to n or whatever. Now, when you expand this, you can expand this into number square minus 2 times number times average plus average square. So essentially what this means is we have to maintain certain variables, one is sigma of number square. So, every time you read a number, square the number and add it to the previous sum that you have maintained. Of course you also have to maintain the number of numbers that have been read so far. Then you also have to maintain two times number star average of all numbers that have been read so far.

So you know how to compute the running average, so every time you get new number compute the running average that is we saw how to compute the running average in the previous slide and then compute two times number times average and add it to this. So essentially you can take out average out of this and sigma of number or two times average out of this and you just basically have to maintain sigma of numbers. That is the sum of all the numbers that we have calculated until now and multiplied to the new average that we have found.

And then we have to maintain, there is no sigma that is necessary here because average is a single number and we have to just maintain the square of the average of all the numbers that we have read so far and we know how to maintain the average. Now by maintaining

all this, we can easily calculate the running variance that is you just compute each of them, put each of them in their corresponding places and compute the running variance. Therefore even if I have a long, let us say stock quotes from the stock market giving me how the quotes of, how the stock price of a particular stock is changing I can maintain what is the mean stock price that it has recorded so far and what has been the variance and I can easily calculate standard deviation at any point in time by computing the square root of the variance. So I know how much it has varied over time and what has been the mean behavior of this stock over the entire time that I have read so far.

(Refer Slide Time: 53:10)



The slide is titled "Mining Streaming Data" and contains the following text and formulas:

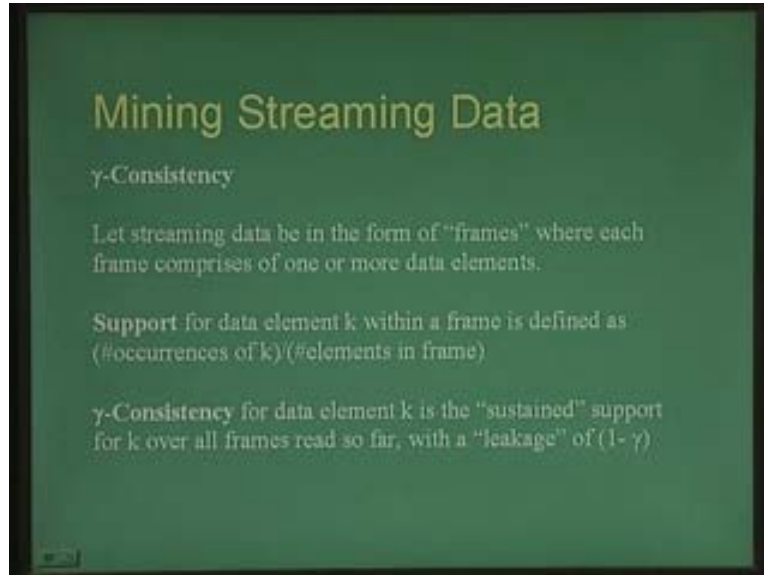
Running variance:

On reading next number num:

$$\text{avg} \leftarrow (\text{avg} * n + \text{num}) / (n + 1)$$
$$n \leftarrow n + 1$$
$$A \leftarrow A + \text{num}^2$$
$$B \leftarrow B + 2 * \text{avg} * \text{num}$$
$$C \leftarrow C + \text{avg}^2$$
$$\text{var} = A + B + C$$

So this slide essentially shows how you can calculate the running variance that is whenever you read the next number first compute the average, we know how to compute the average then each of these is computed like this. That is A equal to A plus n square B equal to B plus two times average star n and C equal to C plus average square and variance is A plus B plus C. We shall also look at one more algorithm for streaming data essentially what is called as a gamma consistency or looking for events that have what are called as gamma consistency.

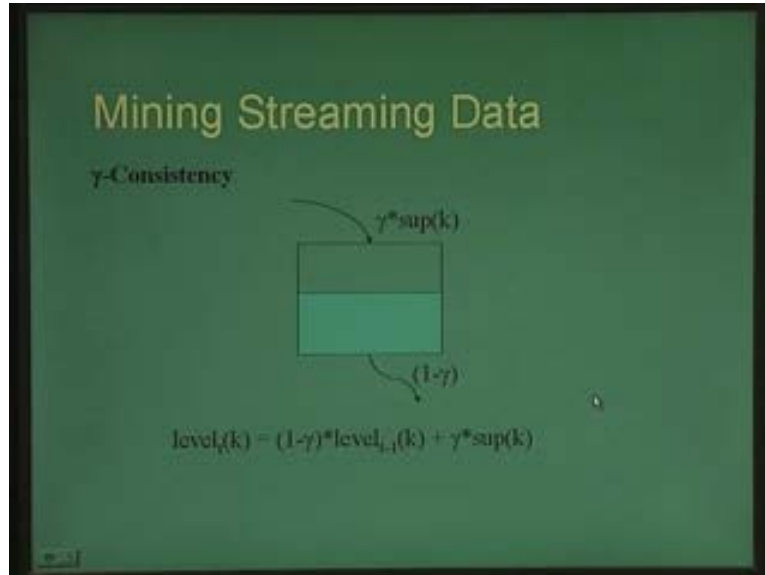
(Refer Slide Time: 53:49)



What is meant by this gamma consistency? Essentially the idea behind this is as follows. Suppose an event happens at some point in time. The interestingness of that event will be high in the vicinity of the event that is right after the event happens, let us say stock market crashes. The interest in that event will be high in the next few days but over a period of time, the interest that event starts going down unless of course the stock market crashes again. So that is the essential idea behind gamma consistency. That is first consider this streaming data to be in the form of frames where each frame comprises of one or more data elements.

Then we look for some interesting events within a frame essentially let us say support based interestingness. So by let us say number of occurrences of k divided by number of elements in frame and then we see which of these events have sustained support over all frames rate so far with a leakage of 1 minus γ . That means in every frame let us say every day or every week or whatever, we look at events that are interesting with a support of k .

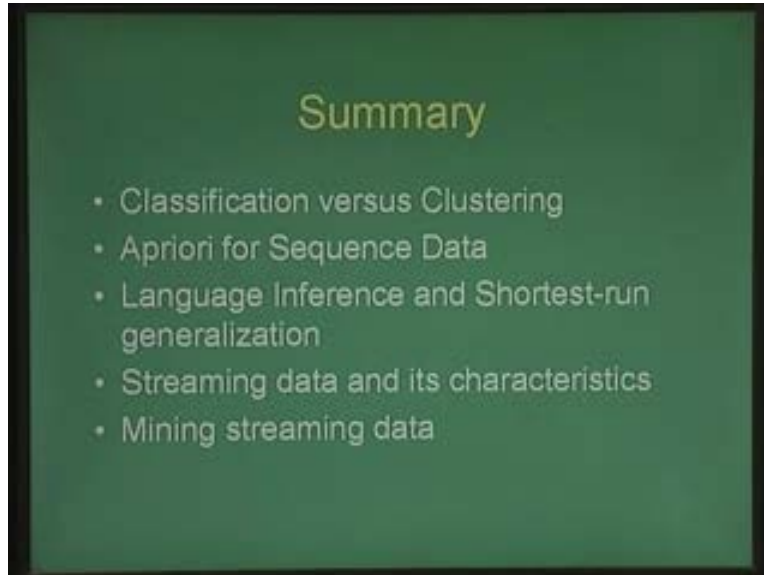
(Refer Slide Time: 55:02)



And if this event keeps on occurring with at least this much support then you can consider this to be some kind of beaker where you are pouring in the events which are coming in with some kind of support and this beaker has a small hole underneath where in it leaks at a rate of 1 minus gamma. So over a period of time if you take it over a period of time, if and only if this event has a sustained support over time this beaker is going to be full or this beaker is going to have a particular level. And if the event does not sustain over time eventually, the beaker is going to empty itself.

So the level in this beaker is an indication of two things. One is how sustained is the support for this event and second could also be how recent was this event. So the more recent the event is the higher the level is going to be, similarly the more sustained the support for an event is again the higher the level is going to be. So you can calculate the level like this and then you can again put a threshold for this level and look at all events which have a particular level or so or level are higher at any given point in time.

(Refer Slide Time: 57:04)



So we now come to the end of this second session on data mining. We have just scratched the surface of what is a vast area of knowledge discovery from databases and we have kind of scratched it in a breadth first fashion that is we looked at several representative algorithms for different kinds of data mining problems whether it was a apriori or whether it was classification or clustering or sequence data or something like language inference and streaming data and so on. But this is just still the tip of the iceberg. So anyway that brings us to the end of this session.