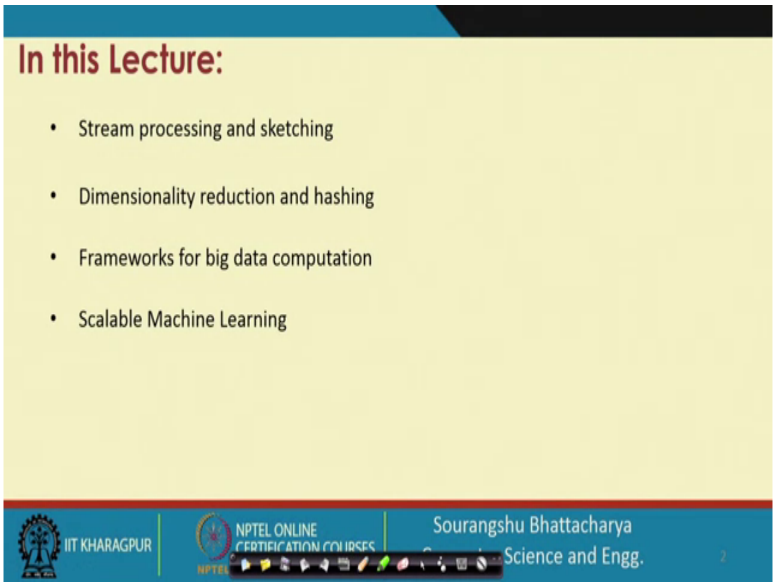
Scalable Data Science

Prof. Sourangshu Bhattacharya
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 01
Introduction

Hello everyone. Welcome, to the first lecture of scalar of NPTEL course on Scalable Data Science. Today we will introduce the course. So, today's lecture is an introduction and I am Professor Sourangshu Bhattacharya from Computer Science and Engineering Department at IIT, Kharagpur.

(Refer Slide Time: 00:37)



In this Lecture:

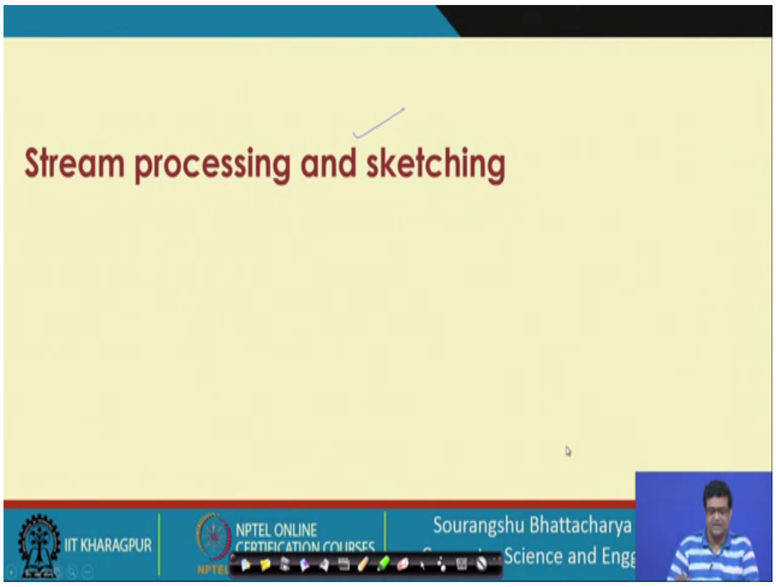
- Stream processing and sketching
- Dimensionality reduction and hashing
- Frameworks for big data computation
- Scalable Machine Learning

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Sourangshu Bhattacharya
Science and Engg.

So, in this lecture we will be. So, we will be discussing the main topics that we will cover in this course, ok. So, broadly we can say that the main topics that will be covered in this course are divided into these four categories.

So, the four categories are stream processing and sketching. So, we will see we will have an introduction to what is stream processing and sketching. And we will see small toy problem or rather introductory problem which is called the problem of reservoir sampling, this is to give you a feel of the kind of things that you will be seeing in this course, and then the second group of topics is around dimensionality reduction and hashing. So, we will introduce the problems and we will describe what the problems are and what is the area. The third theme is frameworks for big data computation again we will describe problems and the fourth broad theme is scalable machine learning.

(Refer Slide Time: 02:12)

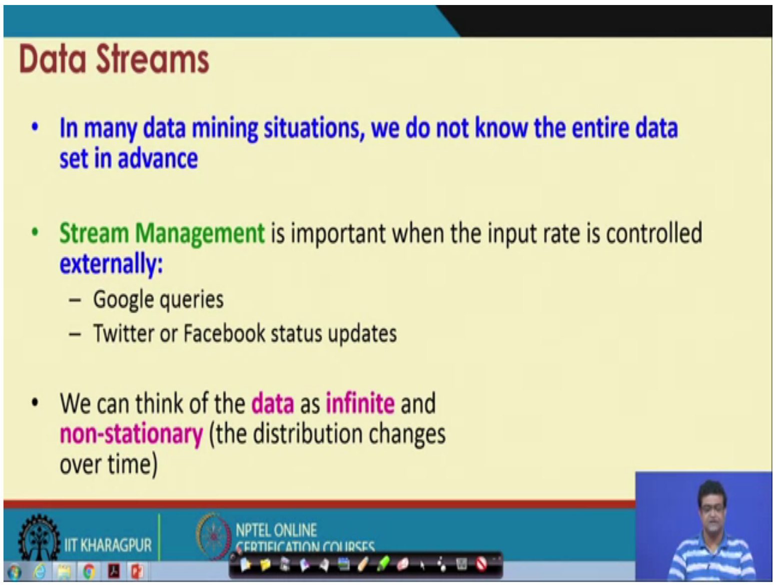


Stream processing and sketching

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Sourangshu Bhattacharya Computer Science and Engg

So, stream processing and sketching.

(Refer Slide Time: 02:15)



Data Streams

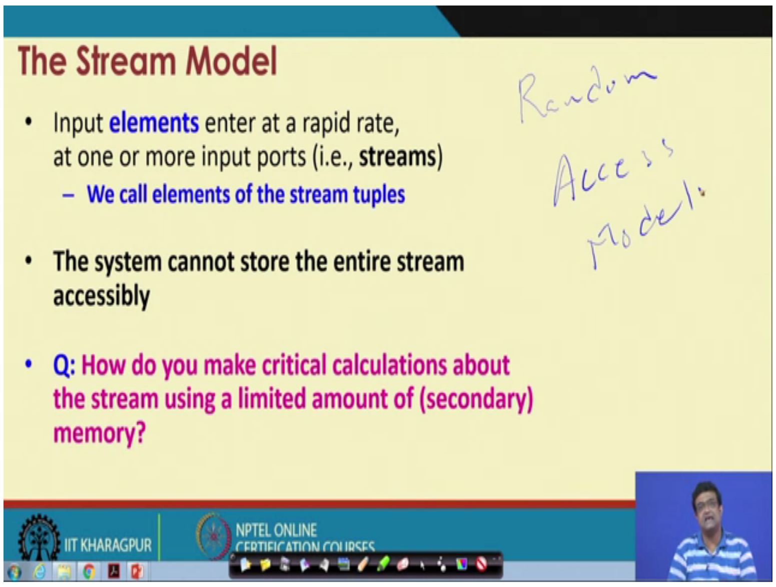
- In many data mining situations, we do not know the entire data set in advance
- Stream Management is important when the input rate is controlled externally:
  - Google queries
  - Twitter or Facebook status updates
- We can think of the data as infinite and non-stationary (the distribution changes over time)

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, what is stream processing or rather what is data streams. So, in many data mining situations we do not know the entire data in advance, ok. So, example can be like Google's queries, ok. So, as Google gets to know the queries as people type the queries or from all over the world. So, you can think like from the point of view of Google the queries come in a stream. So, the queries arrives at the Google servers in a stream. So, whatever algorithm is processing these queries.

So, for example, there could be algorithms which are trying to find out the most frequently searched query. Then this algorithm has to operate in a setting which is the stream processing setting. So, stream management is an important when the input rate is controlled externally, ok. So, another way of thinking about the same thing is that the data is infinite and also the distribution of the data changes over time, ok.

(Refer Slide Time: 03:46)



The Stream Model

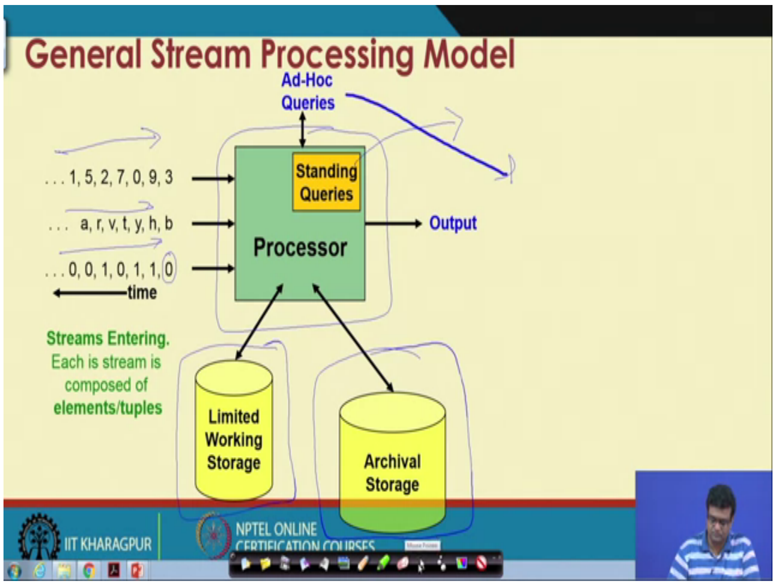
- Input **elements** enter at a rapid rate, at one or more input ports (i.e., **streams**)
  - **We call elements of the stream tuples**
- **The system cannot store the entire stream accessibly**
- **Q: How do you make critical calculations about the stream using a limited amount of (secondary) memory?**

Random Access Model.

So, the input elements in a stream typically enter at very rapid rate, ok. So, these are sometimes called either input records or input tupelos or just input data or input elements. So, the system cannot store this entire stream in an accessible manner. So, the big question is how to how do you make critical calculations about this stream using a limited amount of memory, ok. So, that is the main question which is asked in this stream processing model. So, this is very different from if you see the existing algorithm the existing algorithm typically your algorithm will be given a certain data sets and then. So, these are also sometimes called random access model.

So, random access model random ok: so in this case you get to access the data randomly whereas, in streams you get to access the data one after the other and typically you cannot go back to a previous data point that you have accessed, ok.

(Refer Slide Time: 05:29)


General Stream Processing Model

So, this is the graphic situation. So, you have stream of data. So, this is stream 1. So, there could be multiple streams. So, this is stream 1, this is stream 2, this stream 3. So, time is going backward. So, basically so, this data item arrives or rather time is going this way. So, this is the first data item that arrives, this is the second data item that arrives and so on and so forth, ok.

And, this is the processor. So, the processor will process the data in these streams now broadly there can be two types of queries, ok. One is called the standing queries that is you know beforehand the question that is asked or the query that is asked. So, for example: you can think of these kind of queries as reports some kind of report. So, say for example, you think like. So, the queries are arriving at Google and every 5 minutes or every 10 minutes you have to report that what was the most asked query in the last 5 minutes, or what was the most accessed website in last a 5 minutes ok. So, these are called standing queries.

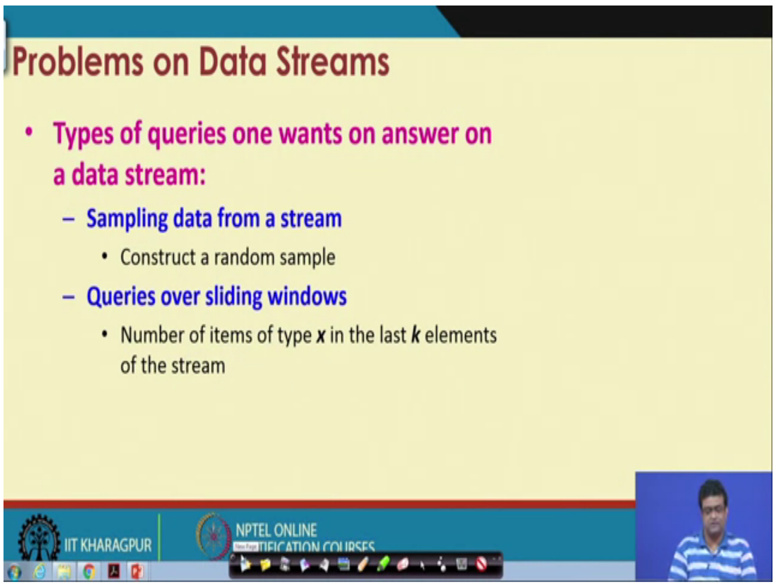
So, the so, the other set of queries is something like the ad-hoc queries ok. So, ad-hoc queries are queries which are asked on the fly, ok. So, there is there are multiple streams coming in here and at some point in time you may be asked that. So, you may be interested in knowing that you know let us say how many which is the most accessed websites from India or something like which is the you know which is the most searched keyword by a group of teenagers. So, people aged between let us say 13 to 19 years 13 to

20 years. So, these kinds of queries are not asked in advance they are asked one time and then you may have to answer this kind of queries, ok.

Now, the thing that you have in at disposal is typically you will have a limited amount of working memory. So, mething like so, these this processor the stream processor will run on a server. So, this server will have a certain restriction for the hardware. So, let us say it has a certain number of gigabytes of RAM and maybe much of the stream not always the whole stream, but much of the stream can be stored in archrivals storage. So, these are the resources that are available to you.

So, so, maybe you can go back to the stream a little bit, but definitely not often because it is in a archival storage and this is very slow access, ok.

(Refer Slide Time: 09:17)



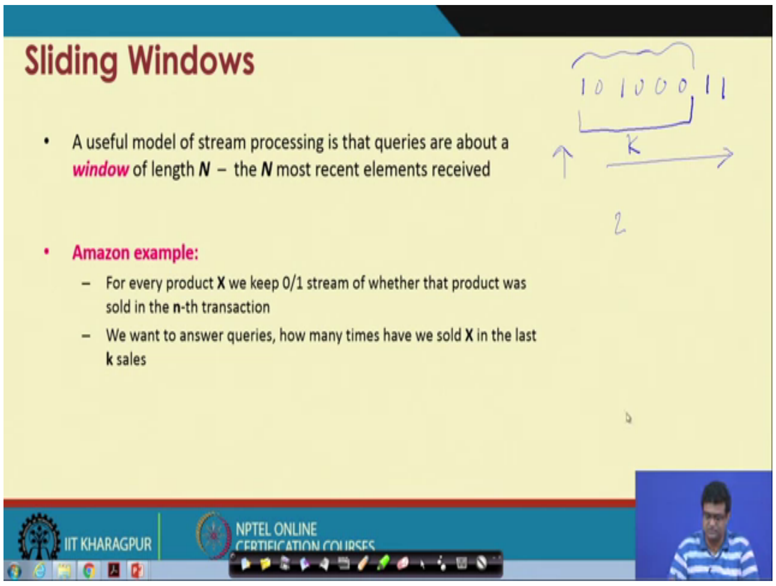
Problems on Data Streams

- Types of queries one wants on answer on a data stream:
  - Sampling data from a stream
    - Construct a random sample
  - Queries over sliding windows
    - Number of items of type *x* in the last *k* elements of the stream

So, again one important class of queries that one might want to ask is to construct a random sample from the data stream, ok. Another important query can be that people ask some queries over sliding windows ok. So for example: the number of items of type x in the last k elements of the stream, ok. So, this is one type of query, ok.

Now, we will see a small problem regarding a constructing samples of fixed size from a stream ok.

(Refer Slide Time: 10:13)



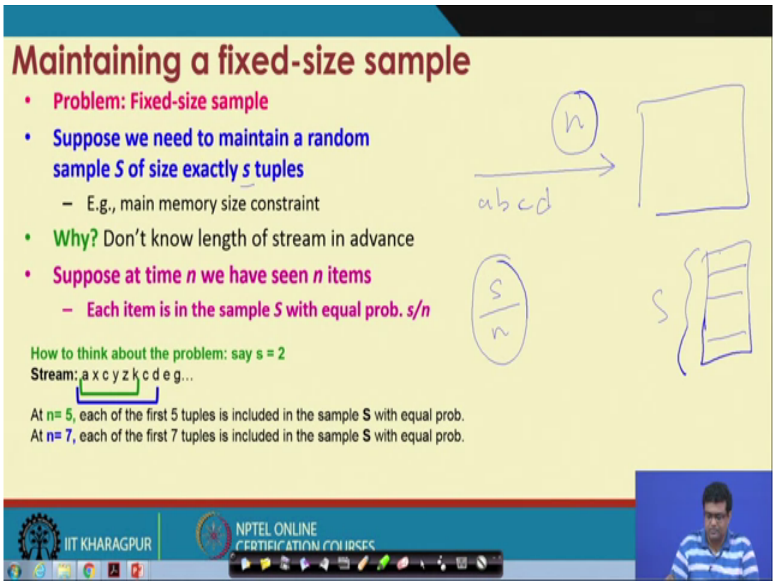
Sliding Windows

- A useful model of stream processing is that queries are about a *window* of length ***N*** – the ***N*** most recent elements received
- **Amazon example:**
  - For every product **X** we keep 0/1 stream of whether that product was sold in the **n**-th transaction
  - We want to answer queries, how many times have we sold **X** in the last **k** sales

So, before that so, what are some examples of you know constructing queries that you have to process over sliding windows. So, for example, for every product X so, suppose you are Amazon you are working for the Amazon or any other retail company internet retail company and then for every product x we keep a 0 1 stream of whether the product was sold in the nth transaction or not ok. So, the query that you may want to answer is how many times have you know have you sold the item X in the last k sales, ok.

So, so, your query is sold sorry query is sold, not sold, sold, not sold, not sold, not sold, sold, sold and then you want to ask in last k sales let say this is the start of the stream. So, this is the current position of the stream and this is where time goes back, ok. So, and this number is k. So, how many times was this item sold? So, me body is browsing this item and how many times has it been sold. So, it has been sold two times in this case, ok. So, this is one practical application of why you would want to answer query ad-hoc queries on sliding windows ok.

(Refer Slide Time: 12:24)



Maintaining a fixed-size sample

- Problem: Fixed-size sample
- Suppose we need to maintain a random sample *S* of size exactly *s* tuples
  - E.g., main memory size constraint
- Why? Don't know length of stream in advance
- Suppose at time *n* we have seen *n* items
  - Each item is in the sample *S* with equal prob. *s/n*

How to think about the problem: say s = 2

Stream: a x c y z k c d e g…

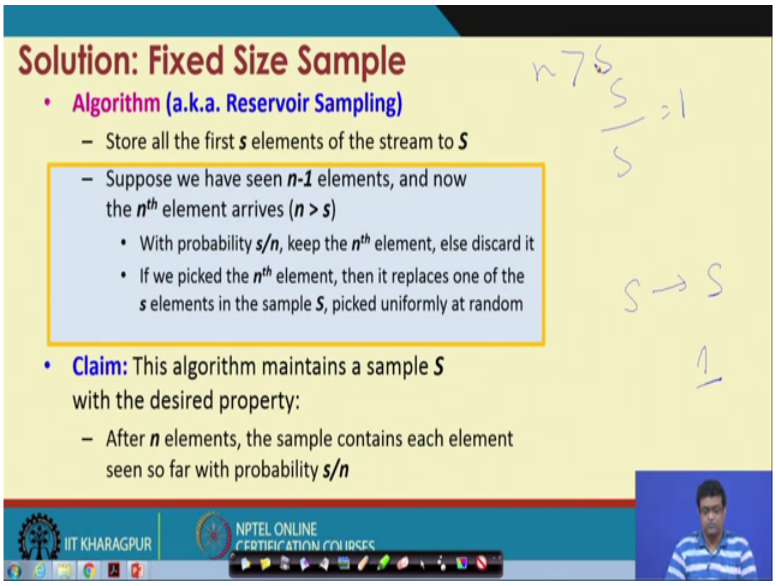
At n= 5, each of the first 5 tuples is included in the sample **S** with equal prob.

At n= 7, each of the first 7 tuples is included in the sample **S** with equal prob.

So, the important problem which we will discuss today is maintaining a fixed sample, ok. So, what is the problem? So, suppose we need so, suppose we have a stream a large stream infinite stream let us say, ok. So, this is an example of the stream. And then at any point in time let us say you have seen n elements, and now you have to maintain a sample of size s, ok. So, you have to. So, there is a stream coming and you have to store. So, abcd something is coming in the stream and you have to store exactly a sample of size s.

So, there should be s entries in this memory that you should store, and the when we say sample the key property which should be preserved is whenever you have seen this n element ok, the probability that let us say each of these n elements was stored has to be s by n that is each element should be stored with equal probability. And since you have to store s out of n elements you have to store s element out of n elements that you have seen. So, the probability should be s by n, ok. So, this is the problem.

(Refer Slide Time: 14:29)



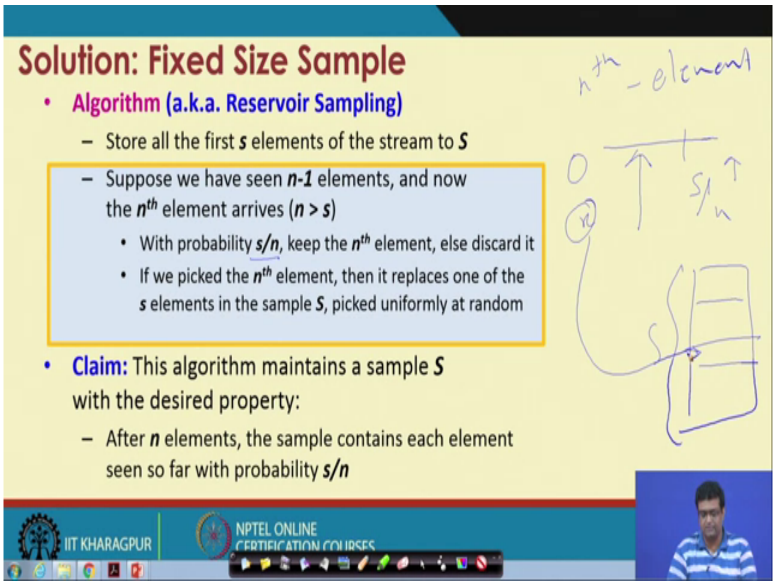
Solution: Fixed Size Sample

- Algorithm (a.k.a. Reservoir Sampling)
  - Store all the first *s* elements of the stream to *S*
  - Suppose we have seen *n-1* elements, and now the n^{th} element arrives ($n > s$)
    - With probability s/n, keep the n^{th} element, else discard it
    - If we picked the n^{th} element, then it replaces one of the *s* elements in the sample *S*, picked uniformly at random
- Claim: This algorithm maintains a sample *S* with the desired property:
  - After *n* elements, the sample contains each element seen so far with probability s/n

So, how to solve this problem? So, we will describe an algorithm which is called the reservoir sampling algorithm. So, the algorithm proceeds like this. So, whenever you so, you start with the starting of the stream. So, for the first s elements you store all the s elements, ok. So, the probability that you store s elements when you have seen exactly s elements is exactly 1, because the probability is s by s which is 1 ok. So, it makes sense only when n is greater than s, ok. So, this kind of algorithm makes sense only when s is greater n is greater than s.

Now, when you have seen let us say n elements, where n is let us say greater than s ok, then what you can do is. So, you do the following steps ok,so with probability s by n, ok.

(Refer Slide Time: 15:59)



Solution: Fixed Size Sample

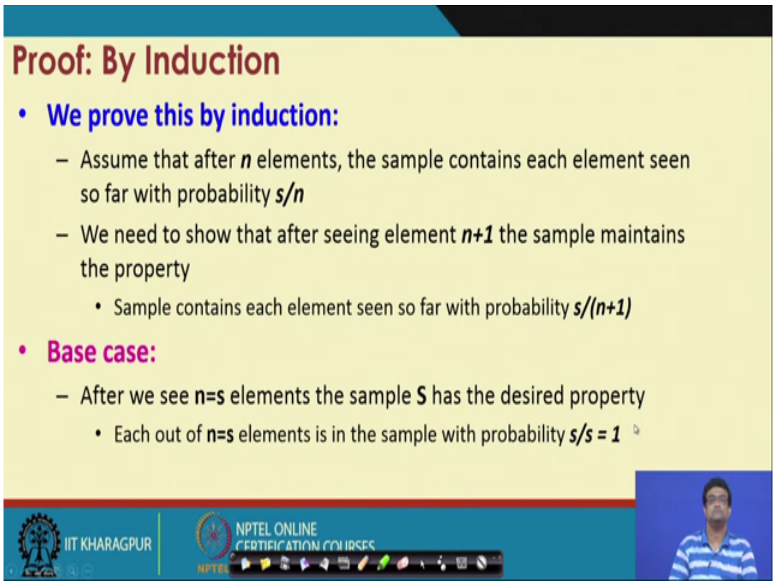
- Algorithm (a.k.a. Reservoir Sampling)
  - Store all the first *s* elements of the stream to *S*
  - Suppose we have seen *n-1* elements, and now the n^{th} element arrives ($n > s$)
    - With probability *s/n*, keep the n^{th} element, else discard it
    - If we picked the n^{th} element, then it replaces one of the *s* elements in the sample *S*, picked uniformly at random
- Claim: This algorithm maintains a sample *S* with the desired property:
  - After *n* elements, the sample contains each element seen so far with probability *s/n*

So, sorry you have already seen n minus 1 element and you are now seeing the n-th element, ok. So, you are now seeing the n-th element and. So, with probability s by n you keep the current element else you discard it, ok. So, you toss a coin and let us say biased coin uniformly biased coin between 0 and 1 and if this is greater than s by n you discard the you discard the element, if it is less than s by n you keep the element.

Now, if you have decided to keep the element that is your coin has landed in this portion of the of 0 1, or you have generated a random number which lies in this portion of the number line then you already have a set of n element a set of s elements that you had stored in this memory, you pick any one of this s elements at random, and you replace that with the current element let us call this x the current element. So, you pick any one of these whatever was there you replace that with x. So, that is the algorithm and the claim is that this algorithm satisfies this property that at any stage n, you will you will stored the elements each element with probability exactly s by n, ok.

So, how will you prove this? Ok.

(Refer Slide Time: 17:58)



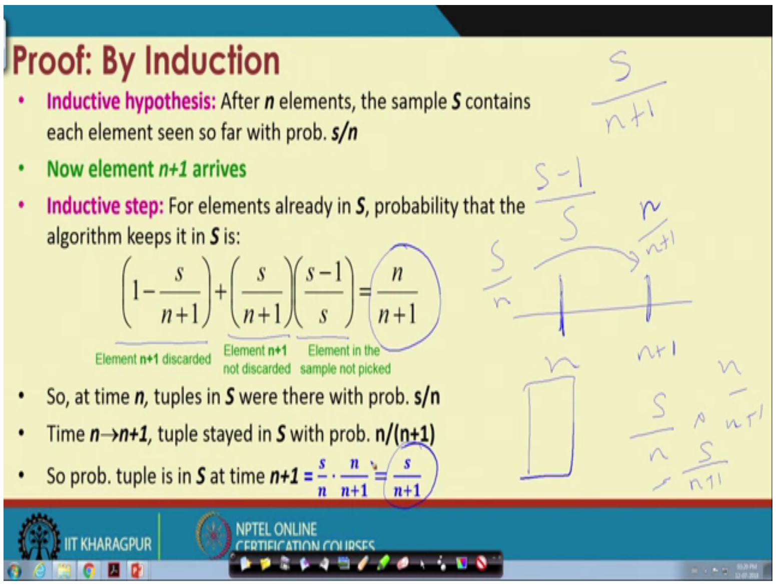
Proof: By Induction

- **We prove this by induction:**
  - Assume that after *n* elements, the sample contains each element seen so far with probability *s/n*
  - We need to show that after seeing element *n+1* the sample maintains the property
    - Sample contains each element seen so far with probability *s/(n+1)*
- **Base case:**
  - After we see **n=s** elements the sample **S** has the desired property
    - Each out of **n=s** elements is in the sample with probability *s/s = 1*

So, we prove this by induction. So, what do we need to do we need to assume that after n elements the sample contains each element seen so far with probability s by n we need to show that after seeing n plus 1 element the sample maintains these properties that is now the sample contents each element seen so far with probability s by n plus 1.

Now, the base case is when as we have already discussed when n is equal to s, ok. So, if n is equal to s we know that we have stored each element with probability 1. So, the base case hold ok. So, because we need to store each element with probability s by n, but s is equal to n or rather n is equal to s. So, you have with probability 1 you store each element.

(Refer Slide Time: 19:20)



Proof: By Induction

- **Inductive hypothesis:** After n elements, the sample S contains each element seen so far with prob. s/n
- **Now element $n+1$ arrives**
- **Inductive step:** For elements already in S, probability that the algorithm keeps it in S is:

$$\underbrace{\left(1-\frac{s}{n+1}\right)}_{\text{Element n+1 discarded}}+\underbrace{\left(\frac{s}{n+1}\right)}_{\text{Element n+1 not discarded}}\underbrace{\left(\frac{s-1}{s}\right)}_{\text{Element in the sample not picked}}=\frac{n}{n+1}$$

- So, at time n, tuples in S were there with prob. s/n
- Time $n\to n+1$, tuple stayed in S with prob. $n/(n+1)$
- So prob. tuple is in S at time $n+1 = \frac{s}{n}\cdot\frac{n}{n+1}=\frac{s}{n+1}$

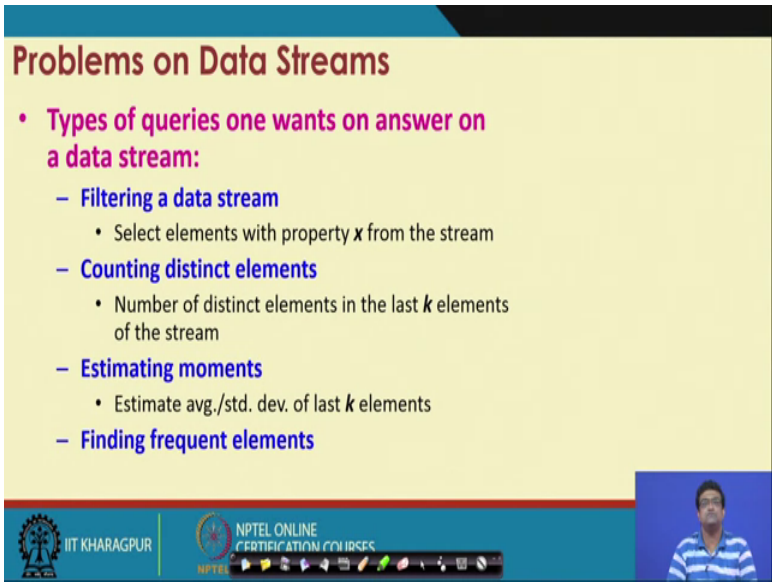
Now, we also know that the inductive hypothesis is that if we have already seen n elements then we know that the sample contains each element with probability s by n. Now, the n plus oneth element arrives, ok. So, what happens what happens to elements which are already in s, ok. So, first let us see the element. So, with probability; so, you know that. So, with probability 1 minus oh sorry with probability 1 minus s by n plus 1 the current element is discarded ok, that is because the algorithm says that when it is seeing n plus oneth element with probability s by n plus 1 it will keep the current element. So, with probability 1 minus that it will discard the current element, ok.

So, this is one possibility, ok. The other possibility is that. So, in this case with probability 1, all the existing elements are kept, ok. So, in the first case the current element is discarded and with probability 1 all the existing elements are kept. The second possibility is that the current element is not discarded, and the chance of that is s by n plus 1. So, this is the chance that the current element is not discarded. In that case with probability s minus 1 by s and element existing element is kept because you have a total of s samples now and you will take one of them and discard and put in the current element, ok.

So, the probability: so only s minus 1 of them we will actually survive ok, if this case that is the current element is kept then only s minus 1 of them will survive. So, if you do this calculation you will see that this probability comes out to be n by n plus 1 that is the probability that the algorithm keeps one of the existing elements turns out to be n by n plus 1. So now, let us see at time n plus 1 again what happens, ok. So, we know that an element was kept in the till time n. So, this is time n and this is time n plus 1. So, an element was kept till time n with probability s by n ok.

Now, from here to here the probability that an element will be kept is n by n plus 1, ok. So, the chance that after n plus 1 and element will be kept is s by n times n by n plus 1 which is s by n plus 1, ok. So, the probability that an existing element will be kept is s by n plus 1 and the probability that the new element is selected is also s by n plus 1. So, all elements in this sample or all elements that you have seen till now are selected with probability s by n plus 1, ok. So, this concludes the proof.

(Refer Slide Time: 24:03)



Problems on Data Streams

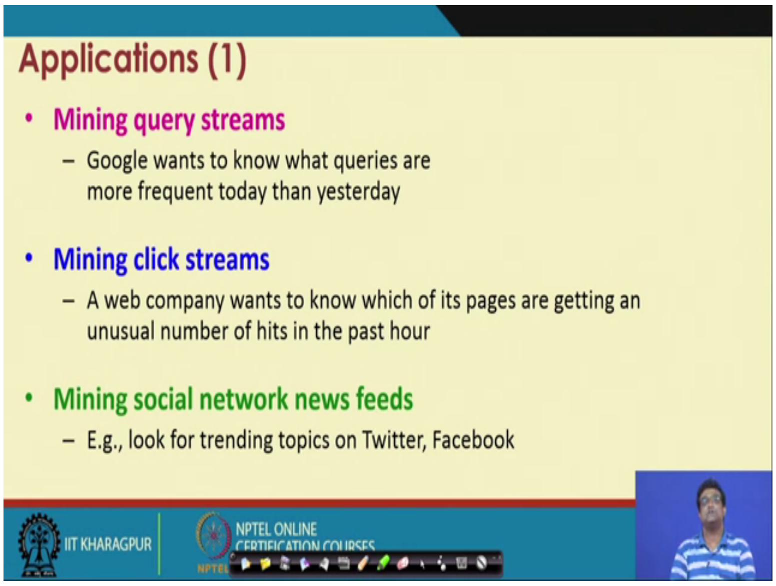
- Types of queries one wants on answer on a data stream:
  - Filtering a data stream
    - Select elements with property *x* from the stream
  - Counting distinct elements
    - Number of distinct elements in the last *k* elements of the stream
  - Estimating moments
    - Estimate avg./std. dev. of last *k* elements
  - Finding frequent elements

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

So, what are the other problems on data streams that we will be covering the types. So, we will be trying to answer many different types of questions on data streams, ok. So, the first type of question is filtering question. So, filtering so, you may want to select elements from a data stream with a certain property x, ok. So, that is the first type of question. Second is the counting count distinct question which is you have to count the number of distinct elements in a certain data stream the third question is what is called the estimating moments question. So, you may want to estimate the average or standard deviation of counts of last k elements and finally, you may want to find the most frequent elements in a stream ok.

So, you have to build data structures for answering all these questions in a streaming model ok, where so, in the last example what you saw is that at any point in time you were only maintaining a small set of sample of size s, ok. So, a similar computation has to be done in this particular case, ok.

(Refer Slide Time: 25:44)



Applications (1)

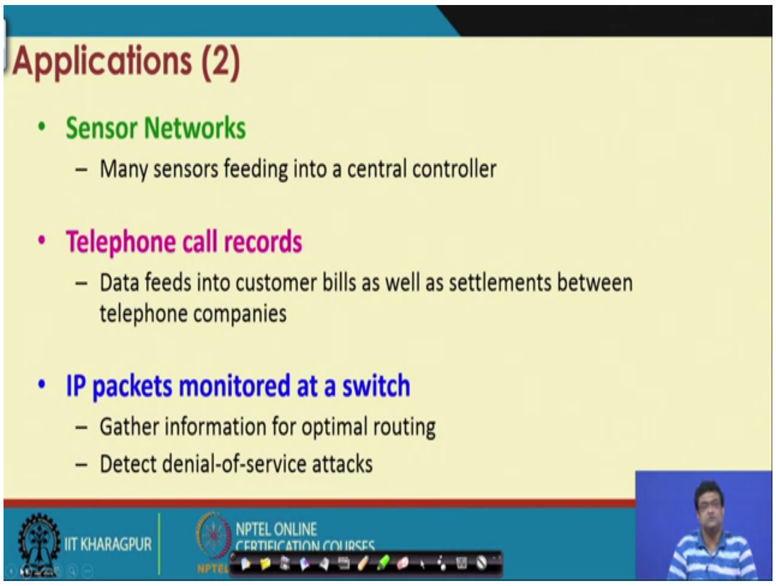
- Mining query streams
  - Google wants to know what queries are more frequent today than yesterday
- Mining click streams
  - A web company wants to know which of its pages are getting an unusual number of hits in the past hour
- Mining social network news feeds
  - E.g., look for trending topics on Twitter, Facebook

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

So, what are the applications? So, for example, you may want to mind query teams query streams. So, like Google wants to know what queries are more frequent today than yesterday. Some such question may be asked or you may want to mind a click stream. So, a web company wants to know which of it is pages are getting an unusual number of hits in the past hour, ok. So, this can be something like trending pages, ok.

Then mining social network feeds: so looking for trending topics on Twitter, Facebook etcetera.

(Refer Slide Time: 26:33)


Applications (2)

- Sensor Networks
  - Many sensors feeding into a central controller
- Telephone call records
  - Data feeds into customer bills as well as settlements between telephone companies
- IP packets monitored at a switch
  - Gather information for optimal routing
  - Detect denial-of-service attacks

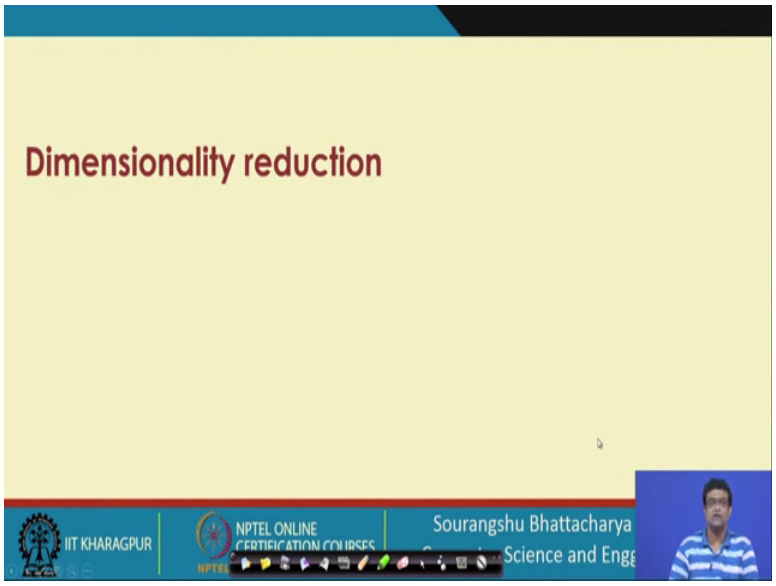
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

So, then there are applications in sensor networks. So, for example, you may have a set of sensors and the sensors may be feeding in streams of number something like temperature or you know other properties that they are measuring in a stream of numbers then you may have telephone call records. So, data feeds into customer bills and then these call records will have to be. So, you may have to answer questions like how much time did the customer talk in past 30 days or something like that, ok.

Then IP packets monitored at a switch. So, many times what happens is there are huge switches which basically power the internet. And then you may have to each of these switches maintain a certain kind of routing tables. So, that you know if the destination packet is certain IP then which of the links should I send it to forward it to, ok. So, for that it may need to maintain what kind of packets are coming from which link.

So, a switch will basically have a certain number of ports maybe 32 ports or 62 ports or something like that and which of these 64 ports a particular IP address is coming IP range is coming. So, this kind of routing table has to be maintained. Then sometimes you have to prevent denial of service attacks, ok. So, all these are applications of this.

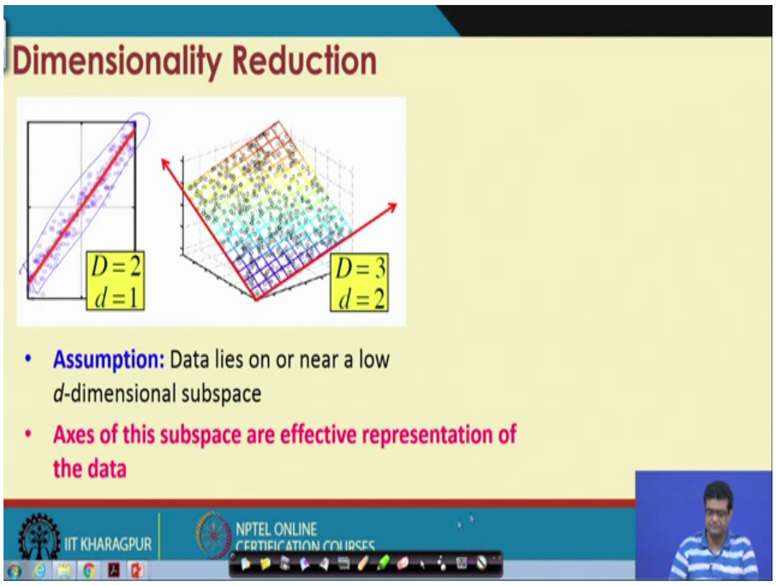
(Refer Slide Time: 28:32)



Dimensionality reduction

Now, the second topic that we will be covering here is dimensionality reduction.

(Refer Slide Time: 28:45)

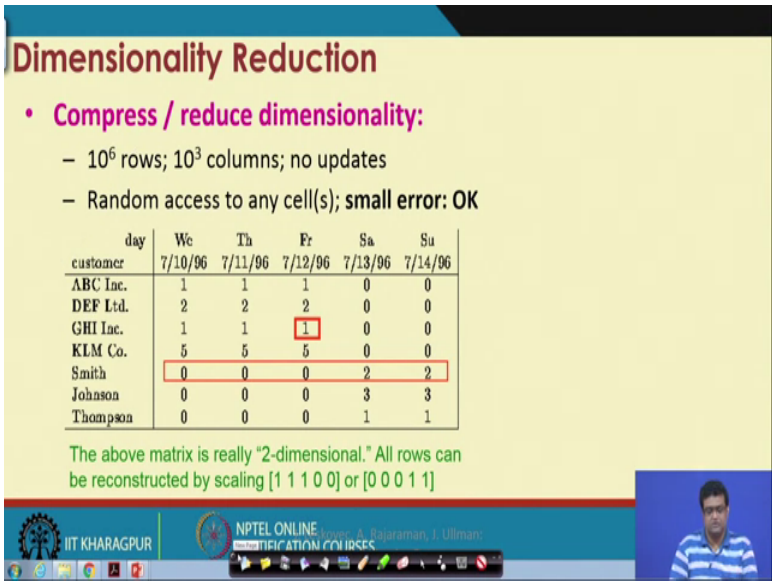


Dimensionality Reduction

- **Assumption:** Data lies on or near a low *d*-dimensional subspace
- **Axes of this subspace are effective representation of the data**

So, what is dimensionality reduction problem? Dimensionality reduction problem is that many a times we have to process a lot of data and we assume that the data lies in a 2-dimensional subspace. So, for example, in this case your data is actually 2-dimensional, but as you can see most of the data is around this line which is a 1-dimensional subspace or in this case most of the data is around this plane which is a 2-dimensional subspace of the 3-dimensional space.

(Refer Slide Time: 29:19)



Dimensionality Reduction

- **Compress / reduce dimensionality:**
  - 10^6 rows; 10^3 columns; no updates
  - Random access to any cell(s); **small error: OK**

customer \ day	We 7/10/96	Th 7/11/96	Fr 7/12/96	Sa 7/13/96	Su 7/14/96
ABC Inc.	1	1	1	0	0
DEF Ltd.	2	2	2	0	0
GHI Inc.	1	1	1	0	0
KLM Co.	5	5	5	0	0
Smith	0	0	0	2	2
Johnson	0	0	0	3	3
Thompson	0	0	0	1	1

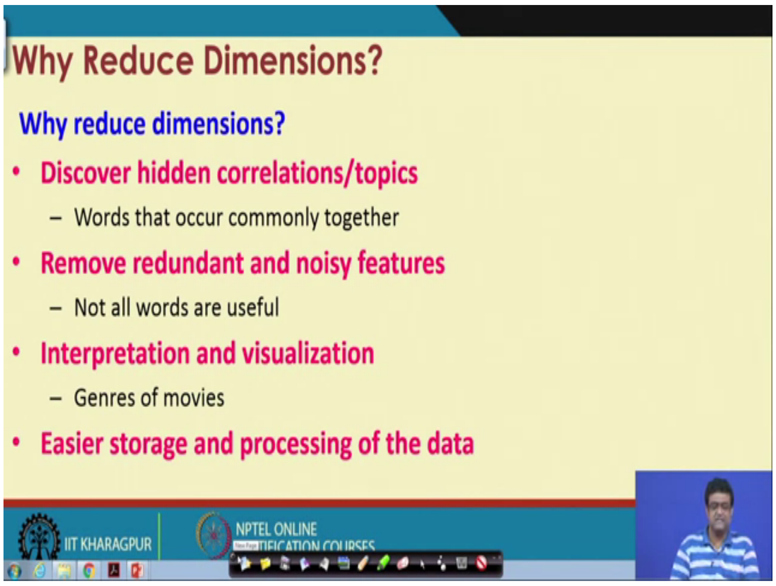
The above matrix is really "2-dimensional." All rows can be reconstructed by scaling [1 1 1 0 0] or [0 0 0 1 1]

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Leskovec, A. Rajaraman, J. Ullman:

So, here for example, you can see that. So, for example, let us say this is this is the number of times a company has accessed or a customer has accessed a certain entity or certain database or something like that and now. So, you have this data forward as a Thursday, Friday, Saturday and Sunday etcetera, but what you can see is that essentially this data is coming from two different types one is the customer who accesses something on Wednesday also accesses on Thursday and Friday, but not on Saturday and Sunday.

And, the second is whosoever accesses on Wednesday, Thursday, Friday, or does not access on Wednesday, Thursday, Friday accesses it on Saturday and Sunday, ok. So, you want to so, even though this is a 5-dimensional data essentially it is a 2-dimensional data which is customers accessing it on weekdays and weekends, something like this. So, one may have to determine this.

(Refer Slide Time: 30:44)



Why Reduce Dimensions?

Why reduce dimensions?

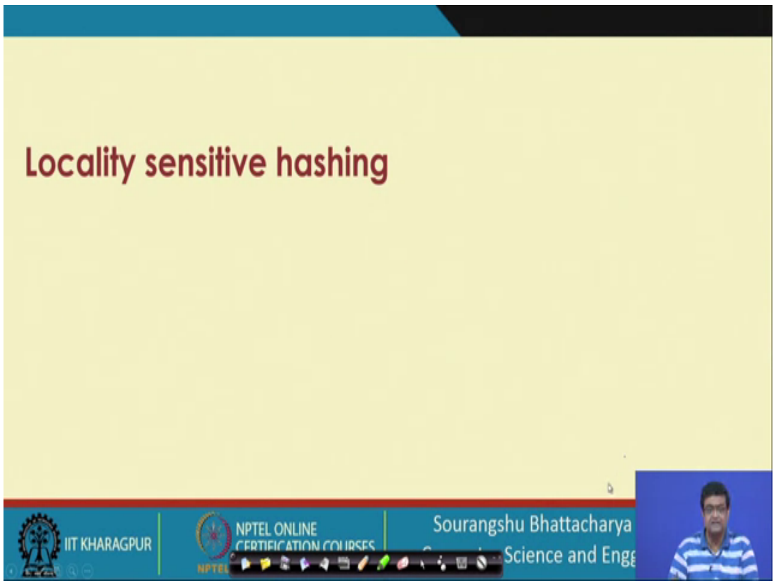
- Discover hidden correlations/topics
  - Words that occur commonly together
- Remove redundant and noisy features
  - Not all words are useful
- Interpretation and visualization
  - Genres of movies
- Easier storage and processing of the data

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

So, basically applications include let us say you want to know which words or topics occur common occur commonly occur together, ok. So, this is like discovered hidden topics hidden correlations. Another important application is to remove redundant or noisy features, ok. So, for example, you are trying to do some text classification you are trying to determine the whether a particular text belongs to sports or music or which category. And then you do not want confusing words you do not want to use confusing words. So, then also you may want to remove the confusing words as dimensions, ok.

Another application is interpretation and visualizations. For example, if you have data about which users like which movies. So, from that you may be able to figure out the different zoners of movies, because users tend to like similar zoners of movies. So, you may be able to find out hidden features or implicit latent features like zoners of movies. And finally, of course the application we are most interested in is if you are able to reduce the number of dimensions then you will be able to more easily store and process the data. So, with less effort you will be able to process the data.

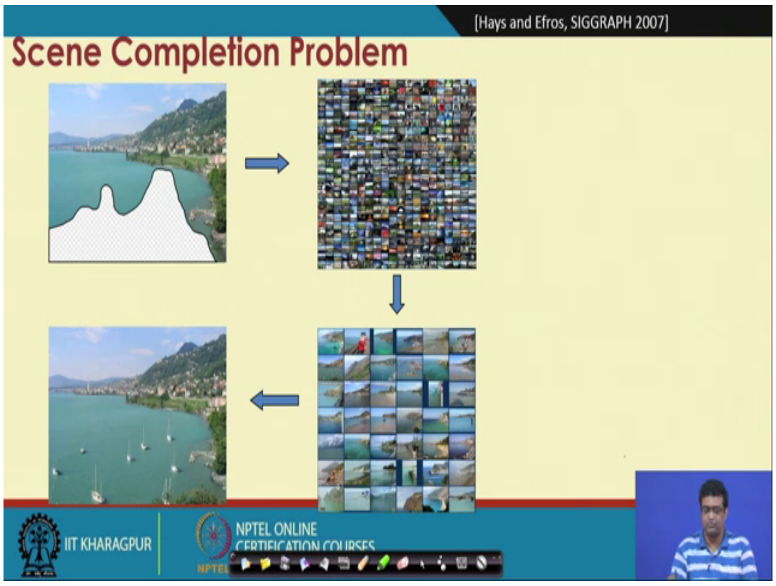
(Refer Slide Time: 32:36)



Locality sensitive hashing

Another important application that we will be discussing is the locality sensitive hashing.

(Refer Slide Time: 32:43)

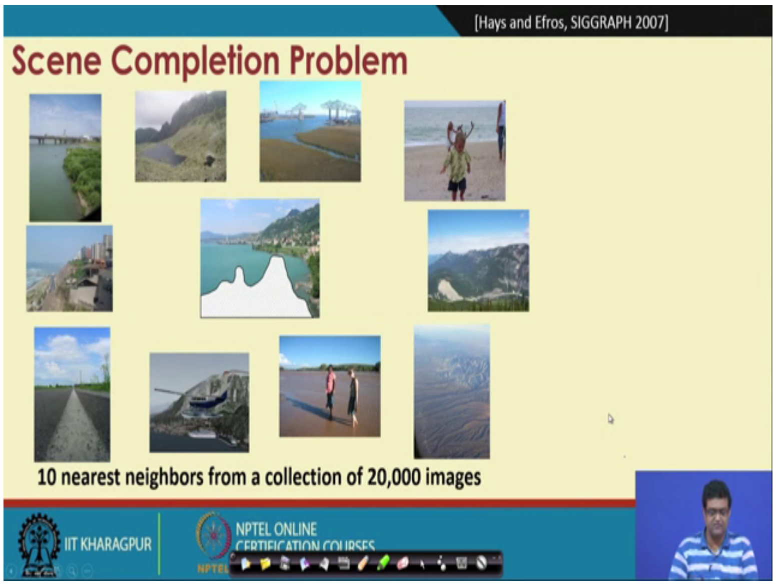


[Hays and Efros, SIGGRAPH 2007]

Scene Completion Problem

So, consider this scene completion problem, ok. So, consider that you have the following scene and you want to remove this house, and now the way to do it is that you have a large database of now images and you want to hit that large database and find some related scenes and then you want to reconstruct a scene something like this, ok. So, this is your problem, ok.

(Refer Slide Time: 33:17)

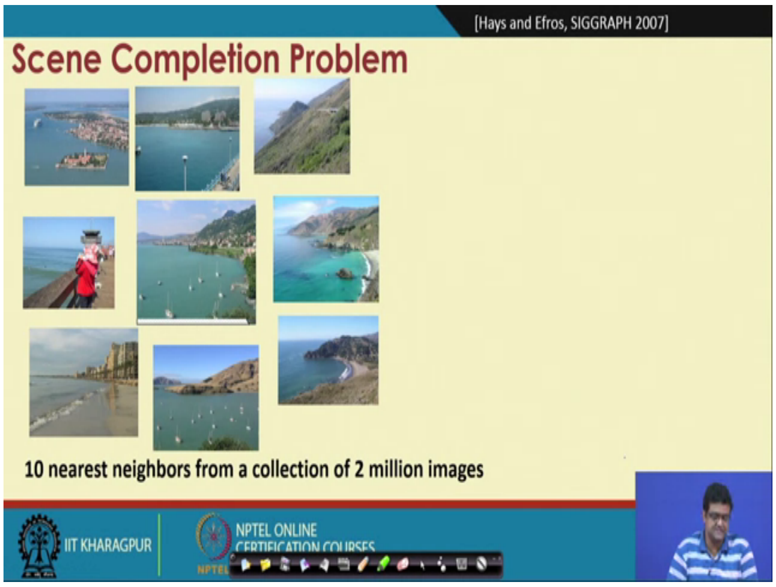

[Hays and Efros, SIGGRAPH 2007]

Scene Completion Problem

10 nearest neighbors from a collection of 20,000 images

So, for this what you have to do is you have to find let say something like 10 nearest neighbors from the collection, ok. So, this is like 10 nearest neighbors from collection of 20000 images these slides are taken from the MMDS book which I will refer to at the end.

(Refer Slide Time: 33:40)

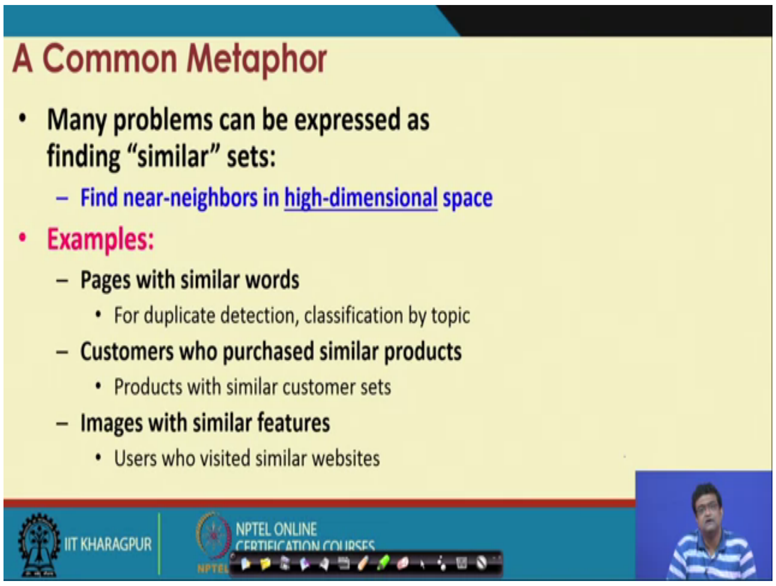

[Hays and Efros, SIGGRAPH 2007]

Scene Completion Problem

10 nearest neighbors from a collection of 2 million images

And, these are the 10 nearest neighbors from like a large image collection which is like 20 million images. And then you can basically just use one of the existing images to complete the scene of this image.

(Refer Slide Time: 34:00)



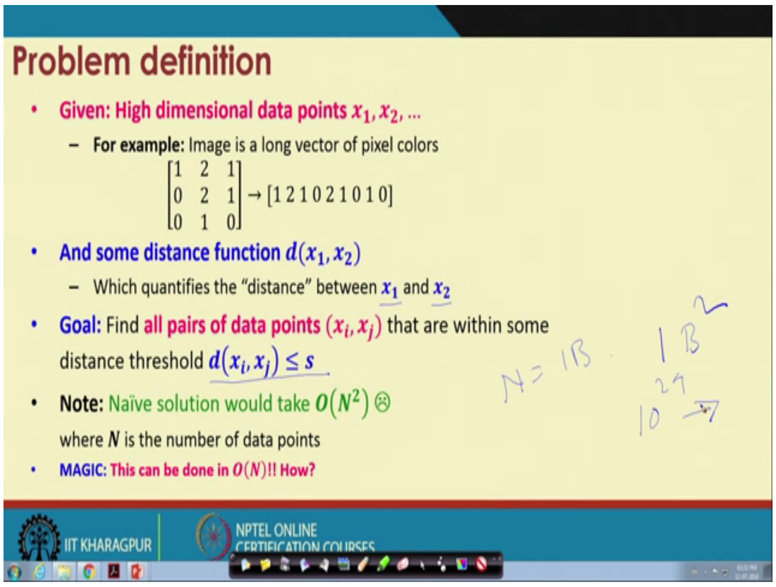
A Common Metaphor

- Many problems can be expressed as finding "similar" sets:
  - Find near-neighbors in high-dimensional space
- Examples:
  - Pages with similar words
    - For duplicate detection, classification by topic
  - Customers who purchased similar products
    - Products with similar customer sets
  - Images with similar features
    - Users who visited similar websites

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

So, there are many such examples where you may want to find the so, given a particular data point you may want to find the nearest neighbors of these data points, ok. So, for example, you may want to find pages with similar words you may want to this may be used for duplicate detection or classification by topic, you may want to find customer to purchase similar products, you may want to find images with similar features, ok.

(Refer Slide Time: 34:36)



Problem definition

- Given: High dimensional data points $x_1, x_2, \ldots$
  - For example: Image is a long vector of pixel colors
    $$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 2 & 1 \\ 0 & 1 & 0 \end{bmatrix} \rightarrow [1\,2\,1\,0\,2\,1\,0\,1\,0]$$
- And some distance function $d(x_1, x_2)$
  - Which quantifies the "distance" between x_1 and x_2
- Goal: Find all pairs of data points (x_i, x_j) that are within some distance threshold $d(x_i, x_j) \leq s$
- Note: Naïve solution would take $O(N^2)$ ☹ where N is the number of data points
- MAGIC: This can be done in $O(N)$!! How?

$N = 1B$

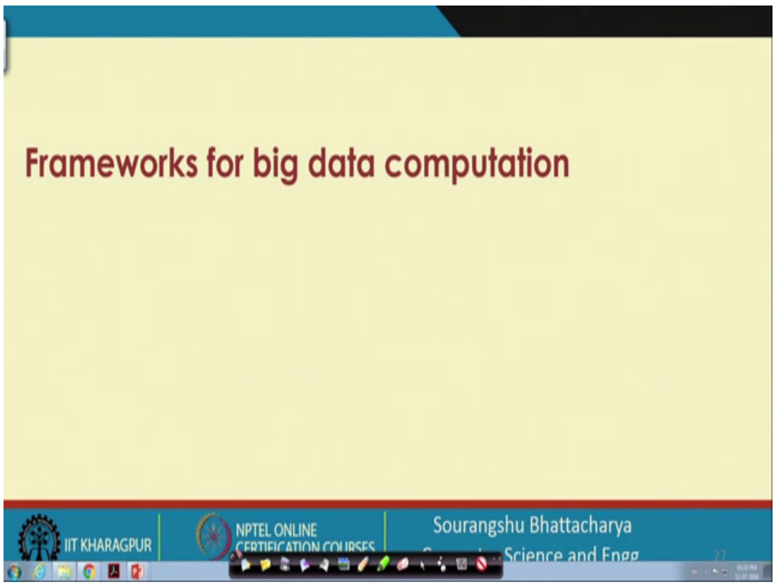
$1B^2$

10^{24}

So, what is the problem definition? So, the problem definition is something like this that you do not want to find an exact duplicate, you want to find a nearest neighbor, ok. So, you want to first define some sort of a distance between any two data points, ok. So, given two data points x 1 and x 2 you want to define a distance between these two data points and then the goal is to retrieve all pairs of data points x i and x j that are within this distance threshold, and as you can understand this is an order n square problem, ok. So, in the case that you are for example, if you want to do a duplicate detection and in the case that you are n is actually 1 billion which is one billion web pages then you have this problem which is not solvable because order n square is actually one billion squared which is like 10 to the power 24. So, this is a very high number, ok.

So, you have to do it in order n time and you can use the trick called locality sensitive hashing to achieve this. You will learn about this trick in this course.

(Refer Slide Time: 36:04)

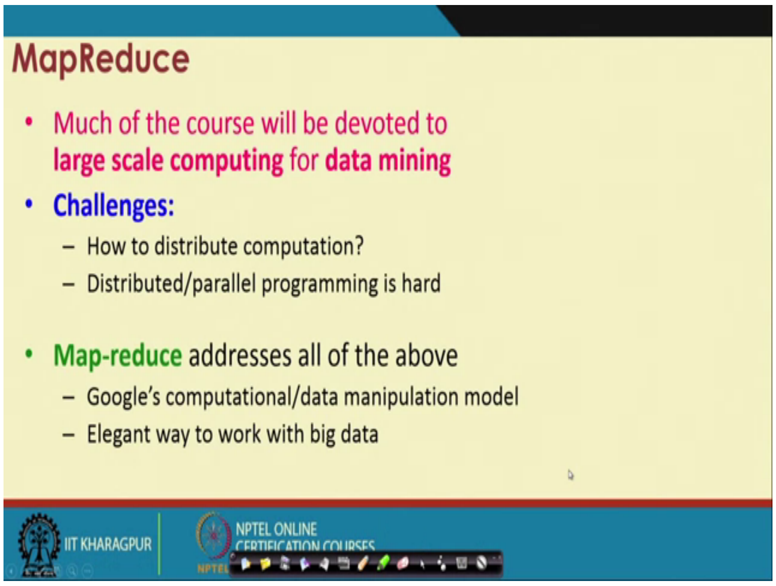


Frameworks for big data computation

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Sourangshu Bhattacharya
Computer Science and Engg.

The third important area is frameworks for big data computation.

(Refer Slide Time: 36:16)



MapReduce

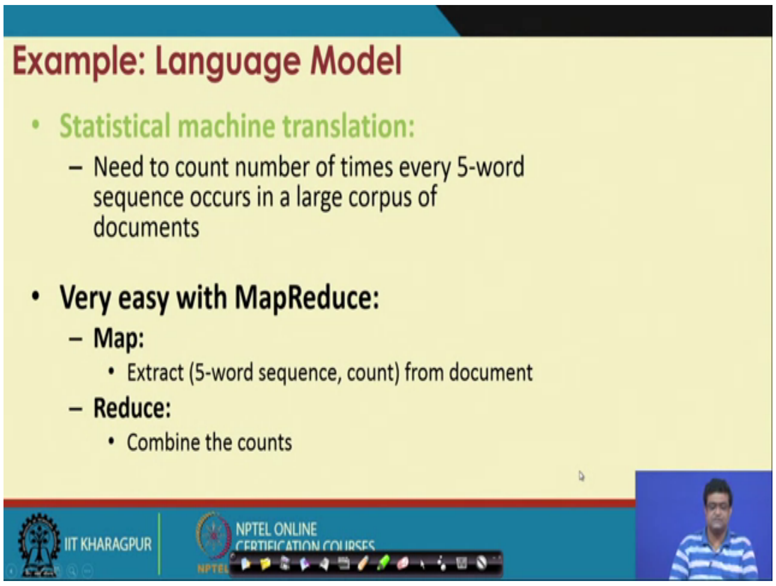
- Much of the course will be devoted to **large scale computing** for **data mining**
- **Challenges:**
  - How to distribute computation?
  - Distributed/parallel programming is hard
- Map-reduce addresses all of the above
  - Google's computational/data manipulation model
  - Elegant way to work with big data

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

So, much of the course will be devoted to large scale computing for data mining, ok. So, in this case the challenges the main challenges are first how to distribute the computation. So, what is the best way of distributing the computation, what is the best way of storing data, what is the best way of how much data should be transferred on which server. So, if you have a large class of servers then which servers should the data be processed and so on and so forth. So, this is the problem of how to distribute the computation and what if you know thus some of the servers fail, hm.

And, the second is actually writing the program for distributed or parallel a computation. So, this itself is a very hard task, ok. So, we will look at frameworks like the map reduce framework or this part framework which make this easy and elegant, ok. So, this is Google's this is a computational framework that came from Google and which is a very easy way it is called big data computation framework also and it is an elegant and easy way to do programming with big data.

(Refer Slide Time: 37:44)



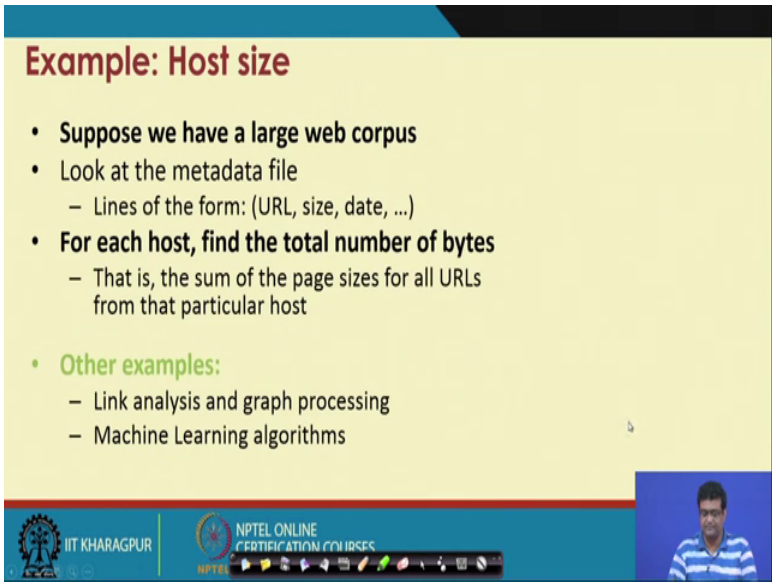
Example: Language Model

- Statistical machine translation:
  - Need to count number of times every 5-word sequence occurs in a large corpus of documents
- Very easy with MapReduce:
  - Map:
    - Extract (5-word sequence, count) from document
  - Reduce:
    - Combine the counts

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

So, examples let say a use you need to count the number of times every 5-word sequence also called 5 gram in general k gram, every k word sequence that occurs in a large corpus of documents. And you can write a map reduce program where you write a mapper and a reducer and you can achieve this task in a distributed manner very easily.

(Refer Slide Time: 38:18)



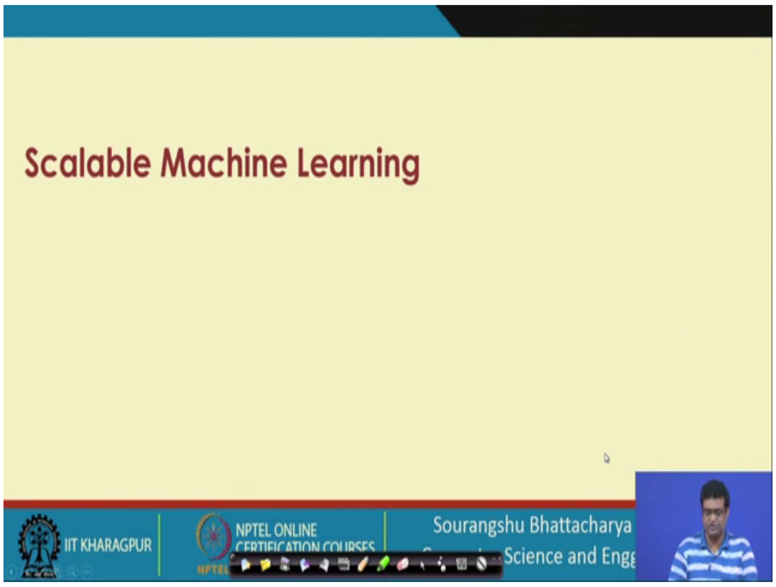
Example: Host size

- Suppose we have a large web corpus
- Look at the metadata file
  - Lines of the form: (URL, size, date, …)
- For each host, find the total number of bytes
  - That is, the sum of the page sizes for all URLs from that particular host
- Other examples:
  - Link analysis and graph processing
  - Machine Learning algorithms

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

Similarly, you may want to you may have a large corpus of files where you have the URL and then the size and the date. And then you want to know the total number of bytes of pages received from a particular domain or host, and many other such computation.

(Refer Slide Time: 38:46)

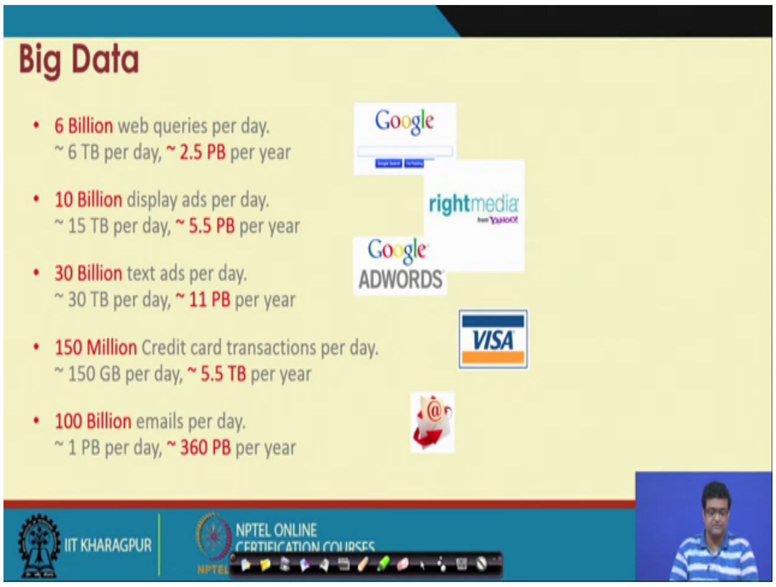


Scalable Machine Learning

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Sourangshu Bhattacharya Science and Engg

The last important area that we will be covering in this is the scalable machine learning area.

(Refer Slide Time: 38:58)



Big Data

- 6 Billion web queries per day.
  ~ 6 TB per day, ~ 2.5 PB per year
- 10 Billion display ads per day.
  ~ 15 TB per day, ~ 5.5 PB per year
- 30 Billion text ads per day.
  ~ 30 TB per day, ~ 11 PB per year
- 150 Million Credit card transactions per day.
  ~ 150 GB per day, ~ 5.5 TB per year
- 100 Billion emails per day.
  ~ 1 PB per day, ~ 360 PB per year

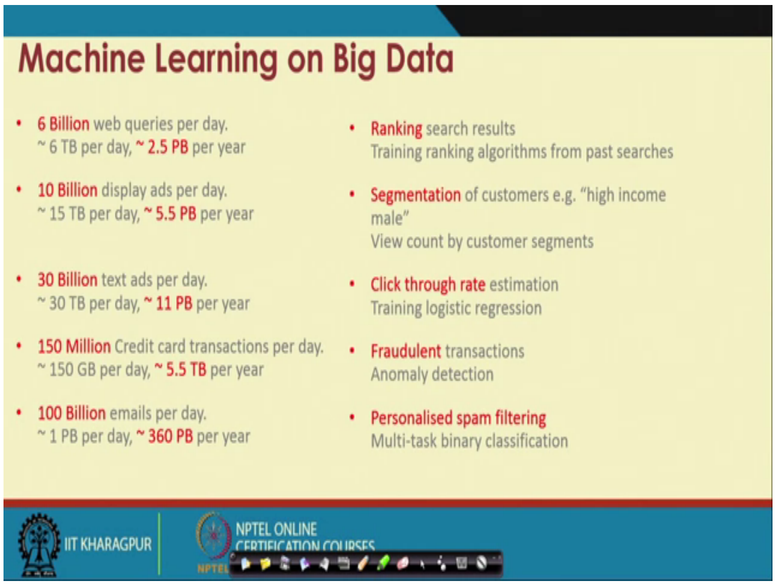
Google
rightmedia from Yahoo!
Google ADWORDS
VISA

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, consider this situation that you have 6 billion web queries per day this is from Google. So, you have 6 terabytes of data and 2.5 petabytes per year. Similarly, you have 10 billion display ads which are shown every day, this is the right media exchange 30 billion text ads per year, 150 million credit card transactions per day 100 billion emails are exchanged per day. So, these are some really big numbers, ok. So, we know that we live in the era of big data.

Now, the point is that many of this data are used to train machine learning models, ok.

(Refer Slide Time: 39:55)



Machine Learning on Big Data

- 6 Billion web queries per day.
  ~ 6 TB per day, ~ 2.5 PB per year
- 10 Billion display ads per day.
  ~ 15 TB per day, ~ 5.5 PB per year
- 30 Billion text ads per day.
  ~ 30 TB per day, ~ 11 PB per year
- 150 Million Credit card transactions per day.
  ~ 150 GB per day, ~ 5.5 TB per year
- 100 Billion emails per day.
  ~ 1 PB per day, ~ 360 PB per year

- Ranking search results
  Training ranking algorithms from past searches
- Segmentation of customers e.g. "high income male"
  View count by customer segments
- Click through rate estimation
  Training logistic regression
- Fraudulent transactions
  Anomaly detection
- Personalised spam filtering
  Multi-task binary classification

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

So, for example; with the web queries you may want to rank the search results. So, you take every web query and it is click feedback and you train the ranking algorithms from the past searches.

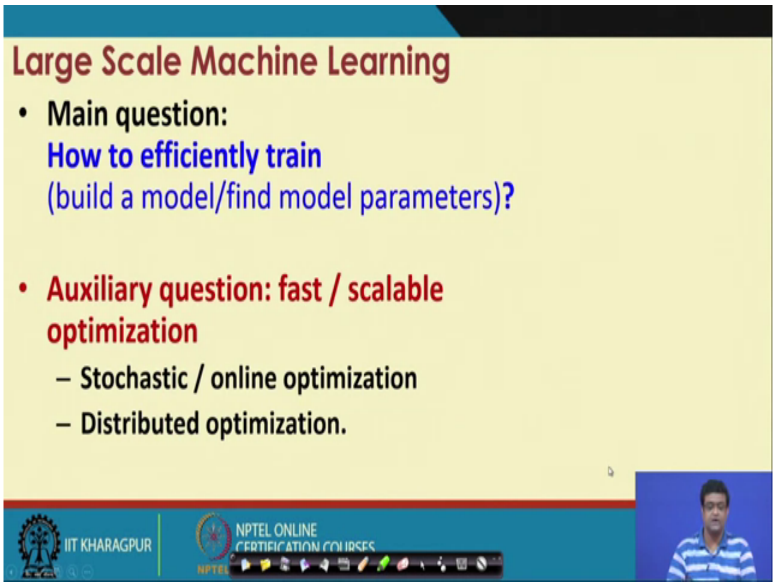
So, this is a machine learning task, ok. So, since the data is huge you will have to learn or you have to run the training algorithm for example, for ranking on a huge data set, ok. So, similarly you may want to do a segmentation of customers who are shown a display ads. So, for example, you may want to characterize the customers into high income male may be low income male and so on and so forth, ok. And then you may want to show to the advertisers how many of your display ads where shown to high income males, ok. So, this is called the problem of customer segmentation.

So, this is again a machine learning problem and it has to be done on huge amounts of data because of the nature of the problem. Similarly, for text ads many a times people pay per click. So, even though even if you show many times, but there is no there is no click on the text add then you do not get paid. So, people try to estimate what is called the click through rate for text ads, and ha and then the estimation of click through rate again is a machine learning problem. So, maybe you use logistic regression or some other model for training the click through rate, ok.

Then you may want to detect fraudulent transactions among the credit card transactions that happen every day. So, this is again anomaly detection problem which is a machine learning problem. And then finally, you may want to you may want to solve a personalized spam filtering problem on all the emails that are exchanged, ok. So, this is multi task binary classification problem which is also again a machine learning problem.

So, for all these cases you need the bottom line is that you need tools which scale which are able to train machine learning models different types of machine learning model, it could be ranking, it could be some segmentation model like supervised or unsupervised models it could be regression something like, click through rate prediction problem or it could be anomaly detection problem or it could be a multicast learning problem all these problems need to be solved on large scale data, ok. So, we will discuss techniques of how you can do this.

(Refer Slide Time: 43:12)



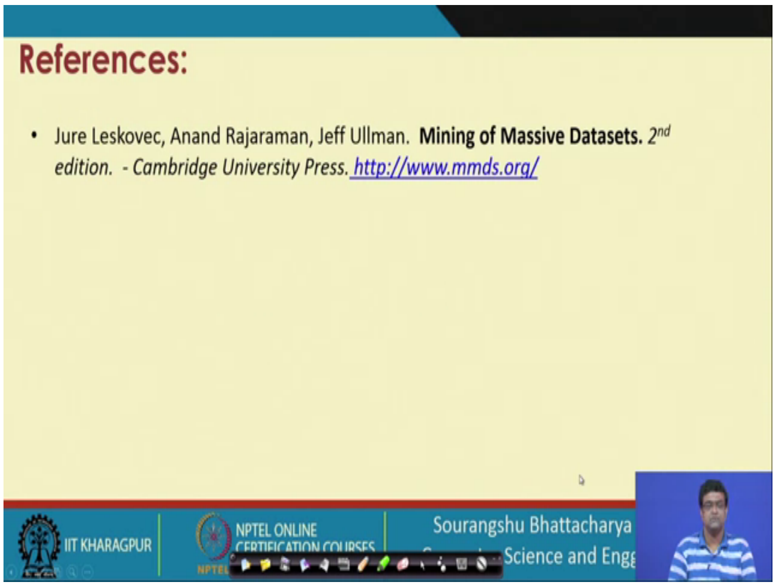
Large Scale Machine Learning

- Main question:
  How to efficiently train
  (build a model/find model parameters)?
- Auxiliary question: fast / scalable optimization
  - Stochastic / online optimization
  - Distributed optimization.

IIT KHARAGPUR
NPTEL ONLINE
CERTIFICATION COURSES

So, the main question is how to efficiently train or build a model or find model parameters, and many a times training models in machine learning boils down to solving an optimization problem. So, then the question becomes how to solve the optimization problem in a in a scalable or fast manner. And we will look at two broad classes; one is the stochastic or online optimization and the other is the distributed optimization.

(Refer Slide Time: 43:52)



References:

- Jure Leskovec, Anand Rajaraman, Jeff Ullman. **Mining of Massive Datasets.** *2nd edition. - Cambridge University Press.* http://www.mmds.org/

So, many of the slides used here and many of the topics are taken from the Mining of Massive Datasets book and course.

Thank you.