IIT Madras

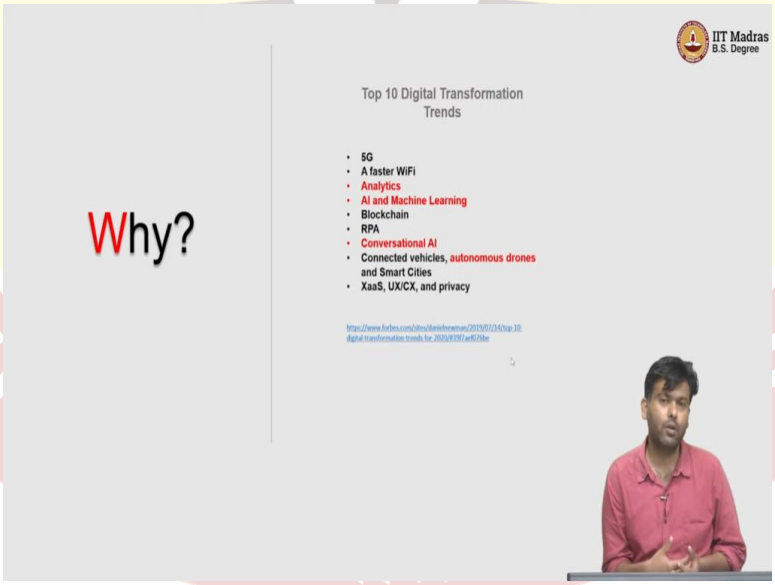
ONLINE DEGREE

Machine Learning Techniques
Professor Arun Rajkumar
Department of Computer Science and Engineering
Indian Institute of Technology, Madras
Introduction to Machine Learning

Hello everyone and welcome to this Machine Learning Techniques course of the online B.Sc. in Programming and Data Science Program. My name is Arun Rajkumar and I will be the instructor for this course. So, what we are going to do now is start with the very high level introduction to what machine learning is. Of course, if you had taken a previous prerequisite course, which is the machine learning foundations course.

You will have a fair enough idea of what machine learning is, what kind of problems are being solved using machine learning. Nevertheless, to keep this course a little bit self-contained what I thought was I will just go through a very high level introduction to try and give a flavor for what is going to be covered in this course, what types of problems we will try to solve and things like that; so we will get started.

(Refer Slide Time: 01:01)



The first question we'll ask is why are you trying to take this course, what is the reason? There could be several reasons, but one of the very compelling reasons is that the there are several applications of what you might see in this course. So, for example, the top ten digital transformation trends as listed by Forbes magazine a couple of years back, listed a couple of things which are highlighted here in red, including analytics, AI and machine learning, conversational AI, autonomous drones, and so on.

All of these things depend on one way or the other machine learning algorithms. As you can see, machine learning is in some sense, fundamental to several applications, practical applications; and you will see many more applications as we go along in this course. So, that is the why part.

(Refer Slide Time: 01:56)

So, now you can also ask how popular is machine learning? I do not have to really answer that question, if you have been following the technical digital trends and things like that. Nevertheless, what I did was, I did today, a search for the term machine learning on Google, Google Trends. And then I also added, I am also showing you the trends for the word search for the keyword deep learning and for the keyword artificial intelligence.

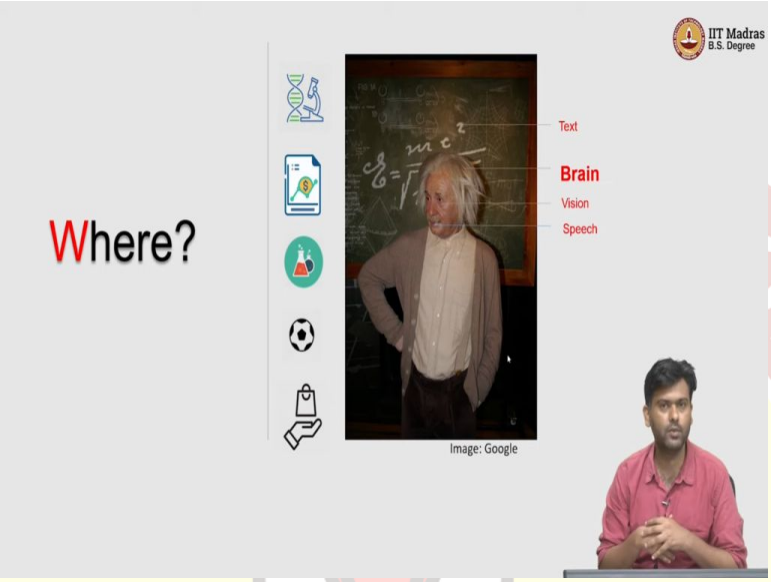
The one in machine learning is blue, deep learning is in red and artificial intelligence is in yellow. And as you can see in the last couple of years, let us say from 2016 onwards, there has been a boom in these in these keyword searches. So, just indicating the amount of popularity and the amount of applications that algorithms and machine learning have had an impact on.

So, of course, deep learning is a subset of machine learning, which will also be covered in one of the elective courses in this program. Artificial intelligence was originally the idea of making a computer trying to, in some sense mimic a human; so you would see a lot of artificial intelligence based searches in the early 2000s and so on, as the yellow curve indicates here.

But, then there as machine learning kind of took over, so the broader term artificial intelligence had gained lesser and lesser popularity. But, now recently with more recent advent of several

new ideas in artificial intelligence, including causality and things like that, so that also has seemed to have picked up a lot.

(Refer Slide Time: 03:39)



So, now let us ask the question, where can we apply machine learning algorithms? So, for this I would like to imagine a prototypical human being, and the type of activities a human being does. If you really want machines to learn and mimic humans, we should first understand what are the things that come naturally to humans; and see how we can make a machine mimic these.

So, for this, one of the main or one of the major areas of areas where machine learning is applied is in vision, what is called as a computer vision, which just tries to understand how you can make a computer see. So, how can you provide the ability of vision to a computer if you will, which means to understand a very very complicated organ like the eye which has evolved over several millions of years.

The human eye, where it can take in information, process it using the brain, and then take decisions based on that. So, now if you want to mimic that, so there are lot of challenges. So, for example, how does the human eye kind of look at a picture and then say, hey, this is a picture of a tree. So, now if you want the computer to do that, to do things like object recognition, so now we need to develop algorithms, which more or less try to mimic the human eye.

So, that is a broader area, where you might see machine learning algorithms often being applied. Speech is another thing where, speech is in some sense an unstructured data; where different

people say the same thing in different ways, and yet as humans we make meaning out of this. So, very easily naturally over time, we have learned to understand human speech.

Now, , how can we make a computer do the same, that would be learning to understand natural language for instance, that would be the speech part of it. Text is another important source of inputs; documents, there are billions and billions of documents, I mean, millions generated as we speak on the internet; and all of these are unstructured.

Again, people write in different ways the same thing, but then, can we somehow understand the semantics of text. So, from just written text or it could be even simpler problem like, can you look at a text and then look at the handwriting; and then figure out what is being written in this and so on. So, there are a broad range of applications there as well.

Of course, the holy grail is to crack the human brain. How does the brain work? So, brain is such a complicated organ with millions and millions of neurons interconnected in complicated fashions, and then produces decisions. How can we understand that from a more abstract computational point of view; so that is how can you learn, like how our human brain learns. So, that is another major area of course, the holy grail of machine learning.

We are nowhere close to doing that, but then we are making slow baby steps progresses. Of course, these are the human sensory applications, but then there are several others. So, you can think of, any experiment that generates a lot of data is amenable to analysis. In fact, you can learn and predict from data; so that is the whole idea of machine learning.

Stock markets, financial sector is another major area, where machine learning algorithms are typically used to predict whether a stock will go up or not; you have to see how the stock has performed over time, and then make models and then use that to predict and so on. Experiments, again biological experiments, chemistry experiments, and so many other materials engineering for instance; so, there are innumerable number of applications in science and engineering.

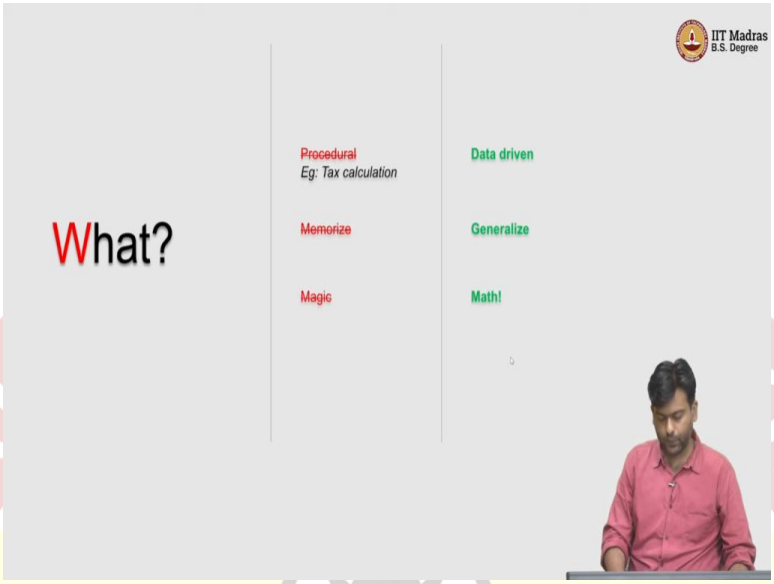
Two other examples that I just want to highlight are one in sports. So, sports analytics is a big area in itself, where you want to predict which team would perhaps win the next IPL match, would it be Chennai or would it be Bangalore? So, of course being a Chennai fan, I would say that perhaps Chennai would win, and maybe many of you might agree with that.

Nevertheless, if you want to confirm that with the computers' prediction; you can take a lot of data and then run it using algorithms; and then it would give you a prediction as well. And finally, not finally, one other example to just give you a flavor is e-commerce. We all use e-commerce websites on a perhaps on a daily basis; your Amazons and Flipkarts and what not, where you purchase a lot of, you go there to purchase some productand then immediately you get a recommendation saying that if you like this product, maybe you should buy this product as well. So, which means that there is some machine learning algorithm under the hoods, which is trying to figure out, this person is trying to buy this product and there are these people who are similar to this person, who have bought similar products.

So, why not recommend that product to this person? So, there is this algorithm at the back which decides, which kind of predicts, which is the product that should be shown to this person, which will perhaps maximize his chance of buying it; so that the company can make money. So, these are some applications of machine learning; there are many more.

And you can think of, I mean, you can try to think of different applications not listed here, by just observing your workplace or your areas of interest. I am sure there will be some application of machine learning, whenever there is a lot of data involved.

(Refer Slide Time: 09:27)



So, now let us ask the question, what is machine learning? So, we know why we are learning this topic; and we kind of know, have a high level view of where, where it is appliedbut then we want to understand what it is? So, the way I want to think about this is first, let me tell you what is not machine learning; and then we will try to compare and contrast with what is machine learning.

So, what is not machine learning? Machine learning is not a procedural approach to doing things. For those who have a computer science background you must be very used to an algorithmic way of thinking about things. You have a problem, you write an algorithm and the algorithm solves the problem right.

So, a simple example would be tax calculation, give the input as my salary, my savings; and then there is a set of rules that you apply; and then outcomes the tax that you need to pay. So, there is no learning happening here per se. So, this is a procedure that you follow, which takes as input, does some series of steps in an algorithmic fashion, and outcomes an output. So, there is nothing that is being learnt here.

It is just a procedure, which maps an input to the output; that is not machine learning. So, there has to be a learning component and will see what this learning component is as we go along in this course. An important and perhaps the most important point about what is not machine learning is that machine learning is not memorization.

To give an example, let us say we want we want to compute the; I mean we want to perform the task of recognizing a tree in a picture, let us say. So, now let us say you have thousand pictures of trees and thousand pictures of which does not contain a tree; so now you have 2000 pictures. You can look at these 2000 pictures, memorize which ones have trees and which ones do not have trees. Now, does that mean you have understood the concept of a tree?

How do you for instance, will you be able to predict whether a picture has a tree or not? In a given picture, which is not in these 2000 pictures that have been shown to you? Perhaps not, because what you have done is just memorizing. And then if you are shown a picture, which is already there at the set from which you have memorized, perhaps you would be able to answer well. So, on the other hand, if you are shown something which which has a treebut then, it was never a part of your data that was shown to you. Unless you understood what it means to be a tree, what what does it mean for a picture to contain a tree; you would not really be able to answer that question. So, it is another example; think of it as if you are being taught a lot of things in this course and you solve all your assignment problems.

And if your final exam is going to just test you only on the problems that you solved during your assignments; then of course, it is not going to be too hard. So, you can simply memorize all the steps for all the problems in the assignment, and then you would score the highest amount of marks. But, that does not mean necessarily that you have learned the subject.

So, this memorization does not imply your learning; that is the point I am trying to push here. We will see what it actually then means to learn, in a minute. Of course, another popular misconception is that, we see machine learning as a buzzword that has been thrown around in different contexts all around the place; artificial intelligence, deep learning, machine learning, you keep hearing these buzzwords, it feels like magic.

So, there are so many things that these complicated algorithms have achieved today that it feels like magic; whereas, of course it is not. But, then if it is not magic, what is it? So, let us try to go over these three points and then try to actually see what they actually mean in a machine learning context. If machine learning is not procedural than what is it?

What is it? So, there is a secret sauce which does not make it procedural and what is that? Well, that is data; so all your machine learning algorithms are going to work on some data. It is going

to learn from this data, and then use this learning to make predictions, or whatever the task that you want to solve on a test data. So, the secret sauce for machine learning is data; so it is of course algorithmic.

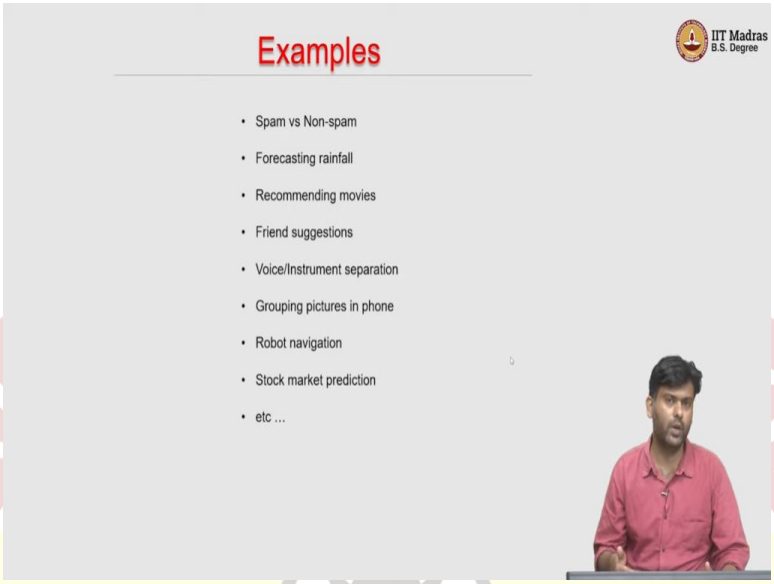
So, we are going to see algorithms and so on. But, then how does it differentiate from a standard computer science algorithm is that there is this extra bit which is the data. We will see more about, what does it mean to say you are learning from data and so on? I mean, that is the point of this course anyway. If it does not memorize, then what does it do?

We want machine learning algorithms to generalize; so, that is the technical term that we use in the machine learning literature. What does it mean to generalize? In layman's term, it means that you actually learn, not just memorize; so that you re able to predict, you do well on an unseen data. That is you are able to generalize your understanding from the data that you see to unseen data.

So, you are not like specific to the data that you have seen only which would be easy to do, which you can simply memorize; whereas, if you do not want to be specific, you must be general enough. When can you be general enough? When you actually learn. So, we want our algorithms to learn. We want our algorithms to generalize, to do well on unseen data. How do you do that? That is what this course is about. If it is not magic, what is it then?

It is basically math. So, it just uses a lot of mathematical ideas and will see some of the prerequisites. Of course, if you have taken a machine learning foundation course in this in online B.Sc. course you would already know what these prerequisites are, and I mean we are going to cover them. But then these are basically mathematical ideas that we need to do well in machine learning; I mean to understand and appreciate the machine learning algorithms.

(Refer Slide Time: 15:38)



So, with that short introduction, let me give you some concrete examples of machine learning problems. And then will quickly try to understand what are the broader paradigms of machine learning; so here are some problems. One of the problems, very popular problem is to predict whether a given email or let us say even an SMS is a spam email or a non-spam e-mail.

So, basically your spam filters in your email inbox, most email providers have a spam filter; which means that automatically you do not do anything. Automatically the email is tagged as spam or not and then it automatically goes to a spam filter. Of course, it makes mistakes sometimes, but most of the times it gets it right.

So, how does this happen? The spam filter itself is an algorithm which has had access to a lot of data; a lot of which are spam, a lot of which are not spam, from which it has learned to differentiate a spam email from a non-spam email. So, that is an example. Forecasting rainfall, if you have data about how much did it rain in the last 15 years, let us say.

And you want to forecast whether it will rain tomorrow or not. Of course, you might have a lot of other features associated with this. For instance, you might know the precipitation levels, the atmospheric pressure, humidity whatnot, about every single day in the in the last 10 or 15 years that you have. And you have to use that to somehow figure out whether on this particular day, which is tomorrow let us say, will it rain or not.

Even if you know these other parameters, let us say precipitation or pressure temperature and so on, can you forecast rainfall? Another example is recommending movies of course. If you use an OTT platform; let us say and you login to it, you are going to be shown a list of movies. And then these might be categorized as new arrivals, whatnot and so on. But then you might also see a topic which says that movies that you might like or recommended for you.

So, that is a set of movies, which are very very specific to you, as opposed to your friend who might also have the same account. I mean, who might have an account in the same OTT platform, but then might get a different set of recommendations. So, which means that depending on what kind of movies you watch, what kind of genres you like, so you get a list of tailor-made suggestions for you; that might be one again, machine learning application.

Friend's suggestions: so if you are in social media, or even in a professional social network, such as such as LinkedIn, let us say. So, you might, you might often see people you may know, or people you may want to be friends with kind of list that that gets automatically generated. So, these are people with whom you are not friends or connections, a priori, but then the system suggests you these things.

So, which means there is an algorithm which uses a lot of data to figure out what kind of people are more likely to be friends; and then uses that to give you a list of suggestions, an ordered list of suggestions. As a slightly different example, you can look at voice/instrument separation; let say you have an mp3 of a beautiful song, sung by a favorite singer.

And for whatever reasons, you may want to separate the voice, the singer's voice from the instrument. So, now, how can you do this automatically? So, there is some structure in how voice signals are, some structure in how instruments signals are. But then what you have in an mp3 is a combined effect of these two.

But, how can we actually separate these out, let us say if I give you a lot of voice, only mp3s , lot of instrument only mp3s somehow can we learn how to separate voice from instrument in a different mp3 file which has both overlaid over each other. That might be a machine learning problem too.

Grouping pictures and phone, so you have your favorite smartphone, where you take a lot of pictures; but then you do not tell who is in which picture. But the system automatically says, hey

these pictures, it looks like they all have the same person. These pictures might have been taken in different angles, might have been taken in different backgrounds, different locations, different time of the day. So still, the algorithm in your smartphone is able to figure out that it looks like these pictures contain the same person.

It may not know who that person is, and that is why it is asking you to tag that person with a name. But then it knows by just looking at these face similarity that well, all these seem to have the same person; so that might be a machine learning problem. It means that you need to figure out using a lot of face data, how our faces related to each other.

So I mean, what distinguishes one face from the other, so things like that. That is a slightly more complicated example. You might want to think about things like robot navigation, maybe there is a robot that is in one end of the room; and then the goal of the robot is to reach the other end, and then there is a lot of obstacles in between.

Now, from the standard machine learning problem, this might be slightly different; because here, the robot has to make a decision to move either forward, left, right, back. And then after it makes a decision, it is given some feedback. So, and meaning if it if it moves forward, maybe goes and hits against a wall; and that is a feedback to the robot that this is the wrong direction to move. But, moving forward might be a good direction in a different place.

So, it is not like moving forward is always a bad direction, bad decision to take. So, it has to learn where to take which decision by making mistakes; so that is that is again a learning problem, it is learning by experience. So that is that is the standard example or a little bit complicated example than the algorithms that we will see in this course, nevertheless, a machine learning problem.

Stock market prediction, as I said you would want to predict whether a a particular stock will go up or not on a day; of course you do not know that. So, you make the prediction in the morning, and then in the evening you observe how much did the stock go up or down. So, based on that, maybe you change and then make a prediction for the next day; and then again, only in the evening, you get the feedback whether the stock went up or not.

So, the over time, I mean, what is a good metric to say that, is your algorithm successful or not? Maybe how much money did you make? How many missed chances were there and things like

that? Nevertheless, the main point is you are trying to learn over the time. So, this is also a learning problem. So, you may not learn in one shot, but then you may want to learn over the time. So, that is an example of a stock market prediction problem.

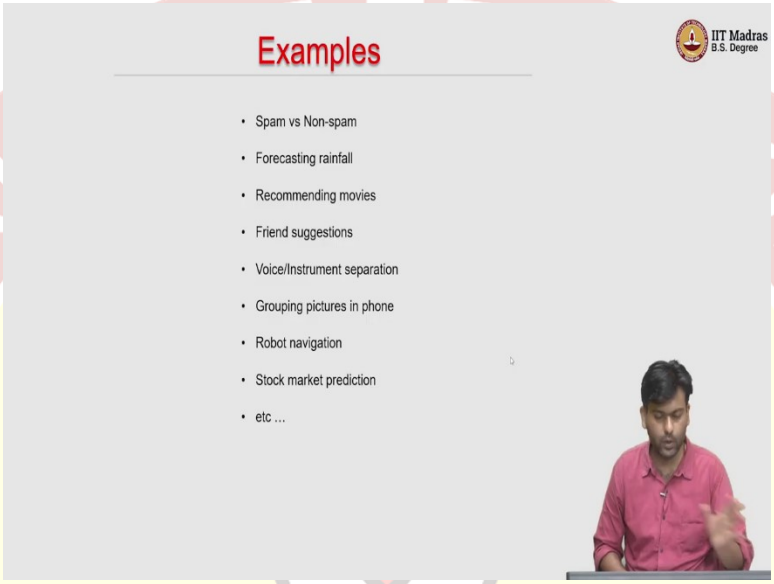
So, these are some examples; of course you can think of, I mean your own examples of where machine learning problem can be applied.

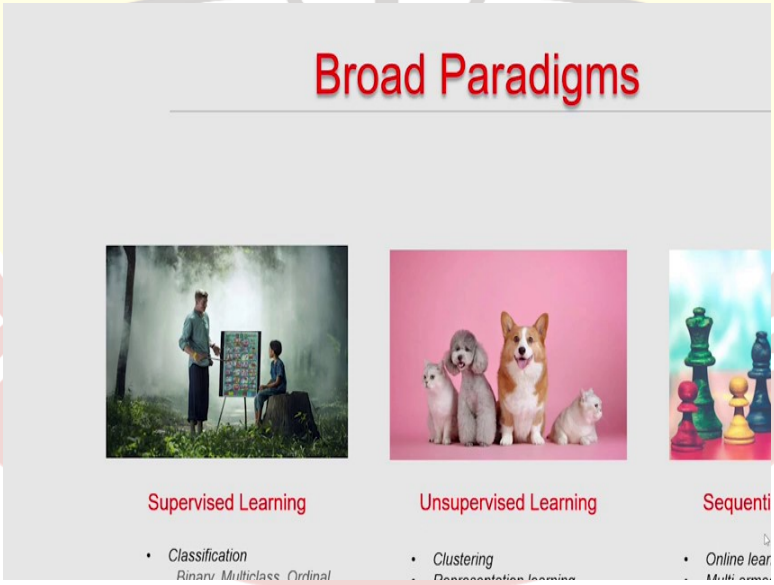
IIT Madras

ONLINE DEGREE

Machine Learning Techniques
Professor Arun Raj Kumar
Department of Computer Science and Engineering
Indian Institute of Technology Madras
Paradigms of Machine Learning

(Refer Slide Time: 00:13)





So, now these are examples. But now what I want to say is give a high level view of what are the broader paradigms of machine learning? And, then how do these examples fit into these paradigms. So, there are three major paradigms. The first one is called supervised learning, where in addition to data, you also have some kind of supervision associated with the data.

For example, you might have a lot of emails with you, but then that is just data that does not have any supervision. But then some teacher or labeler, might come and say that, hey, these are the emails, which are spamming, which the labeler labels, that these emails are spam, and then a bunch of other emails as non-spam emails. Now, you not only have data, you also have the labels associated with them.

And now your problem would be to learn how to differentiate one label from the other. In this particular case, how to differentiate spam from the non-spam? So, using the data that is available to you. So, that is what is called as a supervised learning problem where not only the data or the features are available. In addition, you also have the labels, associated labels.

On the other hand, you might also have an unsupervised learning problem where you are just given a bunch of dataset, data points, and then you are asked to make sense of that. For example, if you have a lot of pictures, like the one that I am showing here, but they do not tell you, what are the labels for these pictures? Now, what would we do? So, it does not seem like a very well posed problem at this point.

For instance, you can group animals versus background that could be one way to group pictures. I mean pictures that have animals in the foreground versus pictures that do not have these animals. For instance, or it could be, pictures which have dogs can all be together versus pictures which all have cats. Maybe, pictures which have four animals versus pictures which have six animals.

Maybe, pictures which have animals, which are white in color versus pictures which have animals, which are black in color or brown in color. So, whatever it is, or animals which are dogs versus animals which are cats. So, there are multiple ways to group things. I mean, basically what I am trying to get at is, there may be multiple notions of similarity that you might want to use to group things together.

So, once your notion of similarity is fixed, then, so let us say we fix cats versus dogs, then you can kind of try to group pictures, which have only cats, only dogs, both cats and dogs. So, these could be three different groups and for instance, this picture would fall into third category and so on.

So, this way of automatically figuring out groups from data is one example of an unsupervised learning problem. We will see more examples, but this is one, just one example. And the third broad paradigm is what is called a sequential learning; we would not really touch upon this. In this course, nevertheless, I just wanted to put it out there so that we are aware of, what it is that we would not be also seeing?

In this course, sequential learning is where as the name suggests, you do not learn in one shot. But then you learn in a sequential fashion, meaning you make a decision, you observe a feedback, and then based on the feedback, you make the next decision, you observe the next feedback and so on. So, it is not as if all the data along with all the supervision is given to you.

It is as if you are given the data and asked to make a prediction, you make the prediction, and then the supervision comes, that is the feedback. The feedback is a form of supervision. And now you look at the feedback and see your own predictions and see what went wrong or what went well. And based on that, you update your model. So, that you make better and better predictions over time. So, that is the problem.

So sequential learning, we would not really see that in this course. Nevertheless, it is a good thing to know at this point. So, each of these three broad paradigms have their subcategories. Supervised learning, for instance, has classification, which is where the labels that are given to you are just a finite set of things. For example, spam versus non spam, it is a classification problem.

You just have two labels, spam as a label, non-spam as a label. Maybe, your number recognition would be a problem with ten labels. So, 0, 1, 2, 3…. 9. But still, it is finite, it is not a binary classification problem. It is not two labels, but then it is still a finite set of labels. On the other hand, you might have a regression problem where you do not have a finite label, but then you want to predict, a continuous variable as output.

So, we will see again, some examples of this as we go along. There are other methods of supervised learning like ranking, where you want to rank a set of objects, instead of just prediction, you want a ranking, or structured learning where the labels themselves might have some structure. Maybe, there are some trees that you wish to predict, and so on.

We would not really be talking too much about the other problems like ranking and structure learning in this course. Whereas we will concentrate more on the basic fundamental ideas, including classification and regression in this course, with respect to unsupervised learning. Well, grouping objects is something that we spoke about, which comes under the topic of clustering.

We will also talk about what is called as representation learning, which is, how do I represent a data point in the best possible way. So that something becomes easier, whatever the something is, depends on what your task is and so on. For example, if it is an image, so image can be represented as just a bunch of pixel values, maybe that is not the best way to represent it. So, maybe, can we learn better representation?

So, that recognizing, let us say, a tree in an image, might become an easier task. So, how do we do this representation learning? Well, it can also be done in a supervised fashion. But then there is also a notion of unsupervised representation learning which we will talk about in this course. Sequential learning, I am not really going to dwell deeper in this topic in this course, whereas I just want to let you know that there are, again, multiple types of sequential learning problems, depending on what kind of feedback that we get.

Let us say you have a group of experts, from whom you take advice, whether I should invest in a stock or not, and all these experts give you some advice and based on that advice, you make your own decision, let us say, go ahead and invest in a stock. And then, you get a feedback, saying that, well, your investment was wrong. So, meaning the stock actually went down on that particular day.

Now, that is a feedback that you get, but then that is not a feedback just for you, but also to each of the experts who made a prediction. So, some experts made up invest prediction, some experts said, do not invest as their prediction. Now, depending on how the market reacts, you are getting feedback, not just for your algorithm, but also for these, all these experts. That is what is called as a full information setting.

And, how do you develop algorithms in this setting to learn in a sequential fashion, is what is called as online learning. On the other hand, you might have a problem where you are a medical

doctor, let us say and the patient with certain symptoms comes in. And then, your algorithm says that, well, there are four different treatments that you can give for this particular patient. And your algorithm says, hey I prefer treatment number two, so depending on the symptoms as a doctor, maybe you take the advice of the algorithm, and then you go ahead and prescribe medication for treatment number two, and then maybe after a week, you know that the patient became healthy or not healthy. So, his health deteriorated, his or her health deteriorated. Now, that is a feedback that you get, but remember, now, this feedback is not the same as the previous feedback that we discussed here. You do get a feedback, but only for the treatment that you gave, whereas you do not know what would have happened had you prescribed a different treatment.

So, this is called as a partial feedback setting, you get feedback only for the action that you perform. This is slightly harder problem in some sense, as you can imagine. And how do you solve this problem? Studying algorithms for this problem is what is called as a multi armed bandits problem and multi armed bandits algorithms who would typically solve this problem. Again, we would not really cover that in this course.

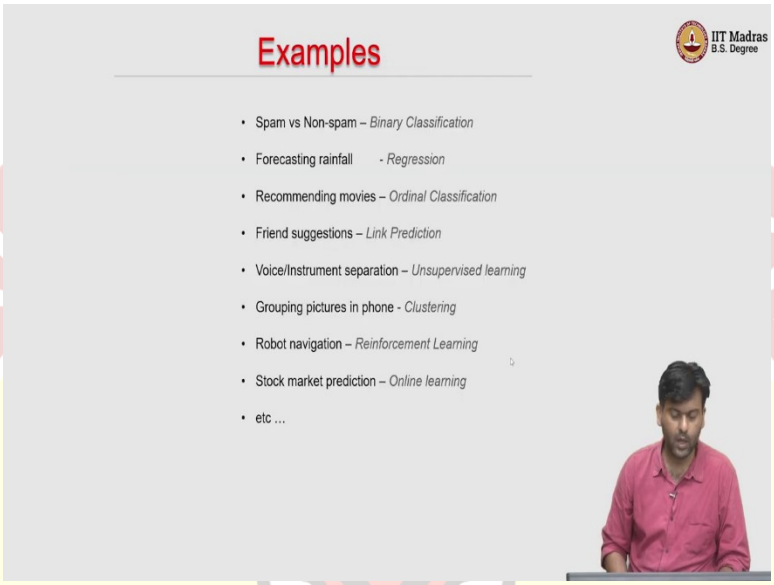
And the final one, which we will also not cover but it would be good to know is what is called as reinforcement learning. Here, as I said typical example, prototypical example is the robot navigation problem, where at every point in the space the robot has four possible decisions. Go front, left, right or back and the robot has to make a decision based on where it is placed. And it has to learn to make good decisions.

So, it has to learn a mapping from its location to a decision, which is typically called as a mapping from a state to an action. State meaning every possible location is a state and the actions in this case are just the four actions that are listed before. And this mapping from state to action is what is called as a policy.

So, and the goal of the robot is then to learn a good policy that will, eventually take it from one end of the room to the other end of the room. So, how can it learn a good policy over time by receiving feedback from the environment? Because, if nobody tells the robot anything about the environment, it kind of learns everything by doing.

So, such algorithms are called as reinforcement learning algorithms and the setup itself is called reinforcement learning. Again, we would not discuss that as I mentioned in this course.

(Refer Slide Time: 10:25)



So, now quickly, let us try to put back, the examples that we saw earlier into buckets that we have discussed. Now, the spam versus non spam problem is a classical standard problem, which comes under the topic of binary classification. It is supervised learning problem, because you have supervision in terms of labels, the labels are either spam or non-spam, there are only two possible labels.

So, it is a binary problem. And then, because if you are trying to classify whether it is spam or not, it is a binary classification problem. On the other hand, if you look at forecasting rainfall, here, you have a bunch of features, and then you are trying to predict how much will it rain, and this how much is a continuous number. So, it can rain 2.23 centimeters, or maybe 100.82 centimeters, depending on which location you are in, and so on.

So, that is a continuous number and that is, what is called as a regression problem. So, now there is also this problem of ordinal classification or ranking type of a problem where you not only have to recommend, decide whether a movie has to be recommended to a person or not, you also has to have to decide in which order they should be recommended.

So, again, you imagine your OTT platform, there is only so much space that the platform can actually use to recommend new movies, maybe five movies, which are shown to you at a time. And, it is critical for the platform that it shows you the right five movies in the right order. So, that it maximizes the chance of you clicking on the first five movies.

So, if the movie that you like, has been found by the algorithm, but then it is somewhere in the 100th place that is not so useful. So, it also not only has to make the prediction well, it also has to order these things carefully. So that is an ordinal classification problem. The friends suggestion problem is a link prediction problem. Imagine a huge network with nodes and edges where nodes represent people and then edges represent connections or friendship, relation between people.

And now, there are some links or some edges which are missing. Now, the question is, can that edge actually exist in this network? Meaning, those people on either ends of this edge, what is the chance that they might actually become friends if a suggestion is given? That is what the problem that the algorithm will try to solve, such a problem is called as a link prediction problem.

Now, the voice/instrument separation is an unsupervised learning problem simply because nobody is going to tell us saying that looking at an mp3 at this second, you have a voice playing, at this second you have an instrument playing. For two reasons, one, it is too tedious to do that for every second of the mp3, the second reason, being that on several cases, it simply might not be possible because the voice and the instrument might be playing together. So, now, how would you say whether that second a voice is second or an instrument second? Typically, that might be hard. So, can we do this in an automatic fashion? Would that be a question that one can ask?

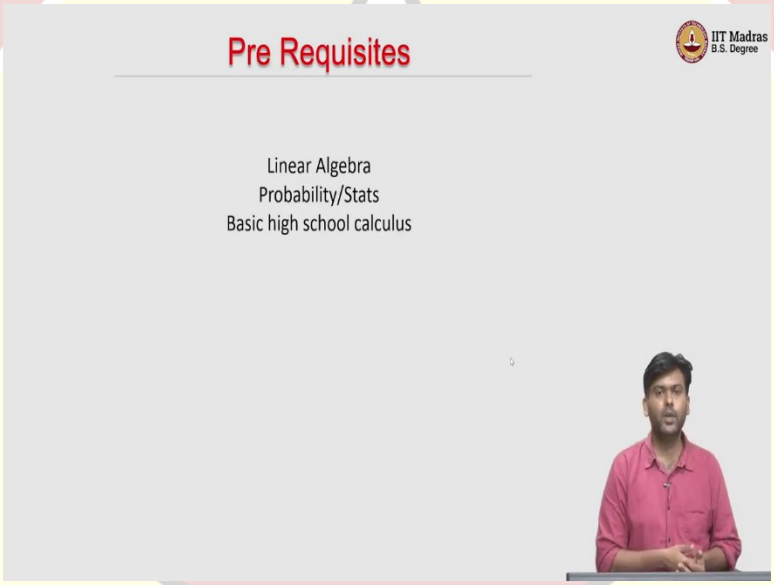
Grouping pictures and phone is a typical example of a clustering problem, or a grouping problem where you take a lot of pictures, but then you do not supervise telling that these pictures are of me, these pictures are of a dog, and these pictures are of a tree or things like that. So, the system automatically figures out that while these pictures look similar, it would not know that these pictures are of a human face, or it would not know that these pictures are of a dog.

But then it knows that well, these pictures where this human face looks similar, more similar to each other than the pictures where there is a dog. So, that is a problem of clustering in an

unsupervised fashion. Robot navigation we have discussed, is a reinforcement learning problem. Stock market prediction is an online learning problem because you might have a bunch of friends whose advice you are taking and then you are making a decision.

And then, the stock market reacts at the end of the day, you have a feedback not just for you, but for how all your friends perform. And then you want to learn over time based on that. Of course, as I said, the last two are not something that we are going to, you know, think about in this course.

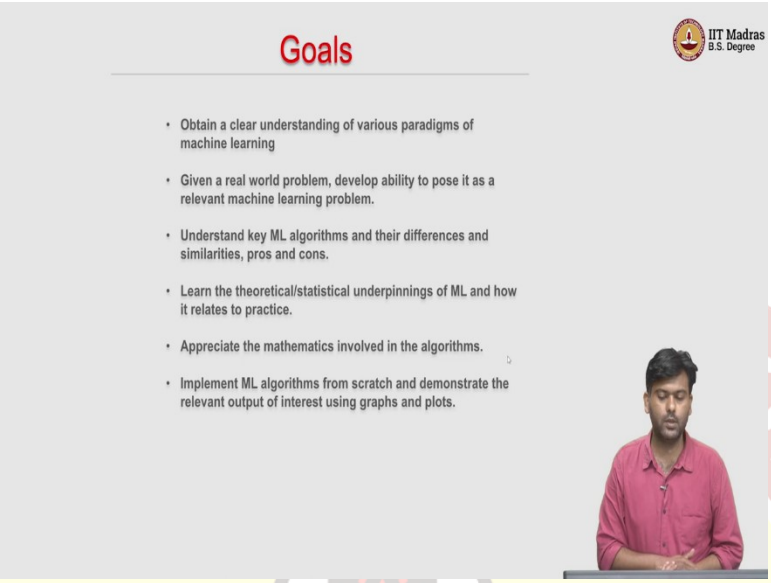
(Refer Slide Time: 14:47)



So, now let us talk a bit about pre-requisites. The main pre requisites are those that are typically covered in your machine learning foundations course, which are linear algebra, probability, statistics, and basic high school calculus, a little bit of optimization would really come in handy. So, if you have done the ML foundations course already, and then you are coming to MLT, then you are pretty much, you have the pre-requisites, the necessary pre-requisites.

But then if you are watching this without having done an ML foundations course, but then if you know, this pre-requisite material, then you can still most likely follow most of the lectures. The idea is to keep the lecture self-contained as much as possible. I might use some results that you might have seen in a linear algebra class earlier, but then I will try to keep them as self-contained as possible. That will be the goal of this course as well.

(Refer Slide Time: 15:50)



To specifically state the goal of this course would be to put them down in points, mainly, what we want to do is, at a basic level, get a very high-level clear understanding of what various paradigms of machine learning are the ones that we saw, just a while ago. But then in much more detail, in terms of algorithms and so on. Again, given a real-world problem, we should develop an ability to pose it as a relevant machine learning problem.

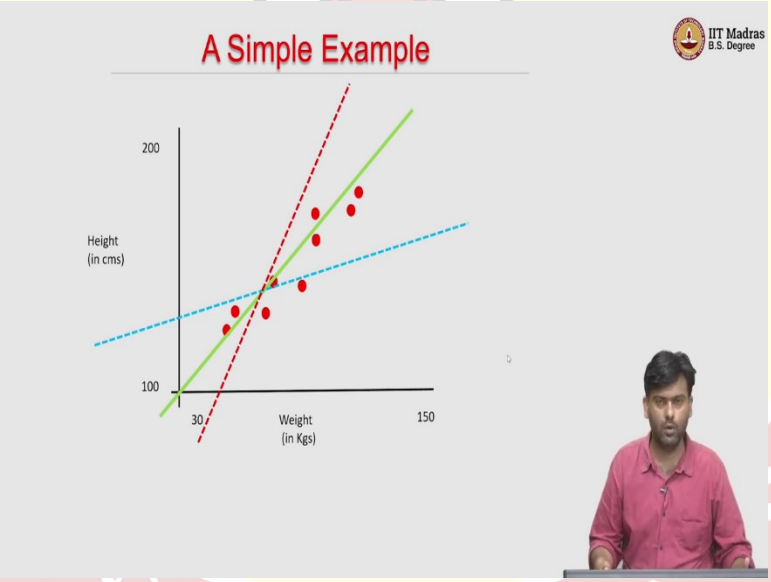
So, because the applications are endless, so now, you might end up in a situation where you have, well, you might encounter a problem, which is not the standard machine learning problem that we discuss in this course. Nevertheless, the ideas in this course should help you, pose the problem as a relevant machine learning problem, abstract out things, post it in as a supervised learning problem or an unsupervised learning problem and things like that, and then try to see which algorithms that we develop in this course, are best suited, and so on.

Of course, one of the major goals would be to understand key machine learning algorithms, their differences, when what works and why their pros and cons, in situations where one might be better than the other, all these are things that we will discuss in this course. So, we will try to cover in this course; we will lay as solid a foundation as possible. So, there are going to be, theoretical/statistical ideas that we will discuss as much as possible, it will be self-contained.

But then the goal is to make sure that we understand the inner workings of the ML algorithms are not just at a, like a black box level. Of course, we want to appreciate the mathematics involved in these algorithms. And if you have the prerequisites that are just listed before, we typically would be good to understand them. We also want you to develop the ability to implement these ML algorithms from scratch, and demonstrate the relevant output of interest using graphs and plots.

Of course, as part of this course, we hope to give you a lot of non-graded assignments where you can write code to develop these algorithms from scratch, and then, get some relevant results and then try to derive insights from them, and so on, so, of course, the discussion board would always be open. So, the more you discuss in the discussion forums, and so on, the more insight you would gain about these algorithms. So, I strongly encourage that as part of this course as well.

(Refer Slide Time: 18:31)

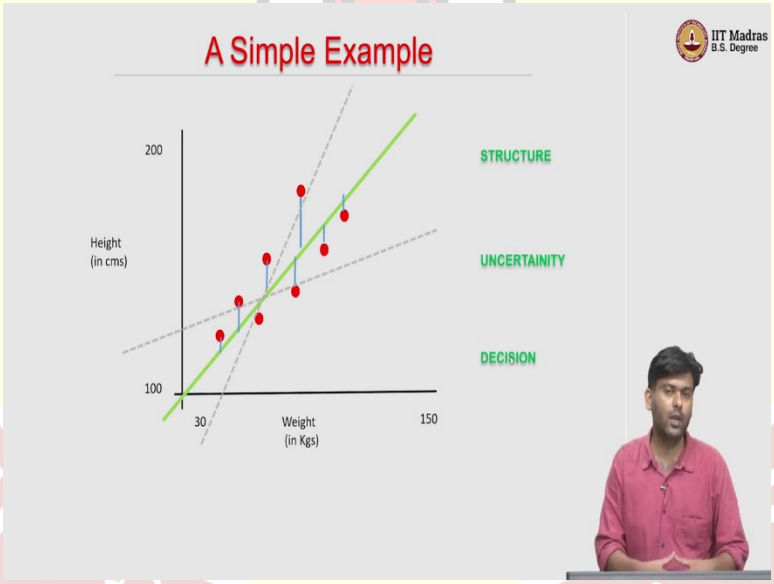


What I am going to do is give a very, very simple example, something that you might have already seen if you're taking an MLF course. Nevertheless, we will quickly go through this example just to illustrate why these prerequisites are relevant. A simple example would be to take two values from a set of 100 people, one would be their height, and one would be their weight, and then plot it on a two-dimensional plane, you might get a set of points like this.

So, now, let us say your goal is to predict the height of a person, given just a weight, you do not know the height of that person, you want to make a prediction, but you only have the weight as input. Now, how can we use this data to learn a mapping from weight to height? But one simple observation here is that well, on an average, if the weight increases, the height also tends to increase. So, in a linear fashion, so, there is a linear relationship between weight and height.

So, we might want to find out that best linear relationship that maps the weight to the height, right. So, in some sense, we want to find the line and then now once you have the line, if I just tell you the weight, you can use this line to make the prediction for the height. But the problem is there is not a single line, there are infinite possible lines on the two-dimensional plane. Even if you look at only lines passing through the origins, there are still an infinite number of them. How do I find which is the best line?

(Refer Slide Time: 19:56)



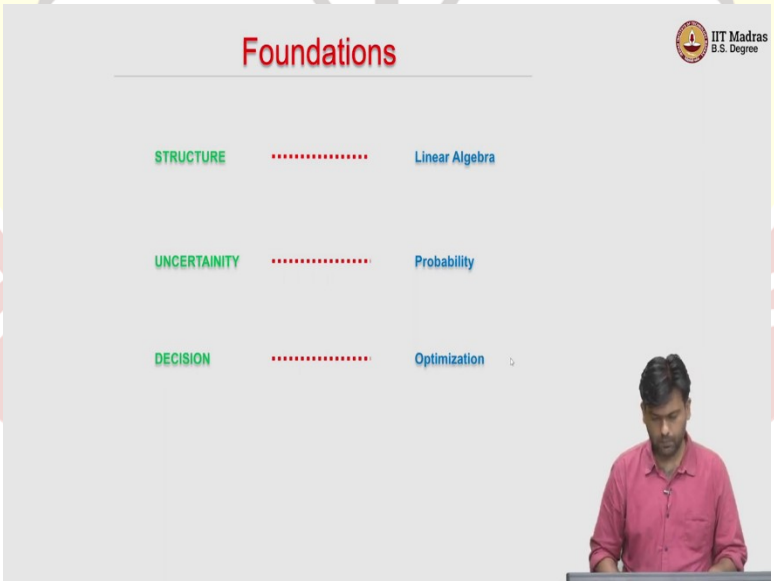
Well, we somehow have to find what is the best line using some criteria. And using that criteria, we will then find the best line and then use that best line to make predictions. So, essentially what we have done is, we are saying that, well, first thing we said is that the weight and height are linearly related, which means that the structure that relates the weight and height is a linear structure, which means we are looking for lines.

But as you can see, in this data, none of the data points pass through the line, which we think is the best line, yet we think that is the best line. So, one of the major things that would happen when you collect data from a large scale is that there is going to be noise. So, there is going to be some kind of uncertainty associated with the data, either because of the nature of data collection or because some equipment was faulty, some data was missing.

There could be different reasons why uncertainty could arise. Nevertheless, uncertainty is an essential part that you need to deal with, if you are doing data science. And, that is one other thing to keep in mind. And finally, well, you have data, you somehow have figured out structure, in this case line, somehow you have figured out a way to deal with uncertainty.

And now how do you convert this data into decision? So, how can you figure out that this is the best client? So, what is best? In what sense is it best, these are the things that we need to talk about. So, once we have decided a notion of performance of a line, given a data set, then we can pick that line which performs the best. So, that, so basically, then we can convert your data into decisions.

(Refer Slide Time: 21:37)

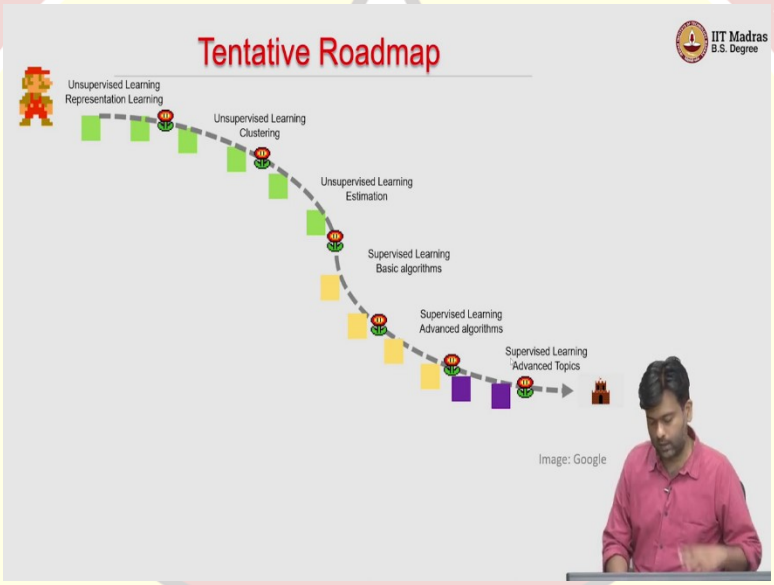


So, all these three are essential ingredients of a machine learning to develop a machine learning algorithm. And this is why these three become major pre requisites for this course, to understand structure, we need to understand basic structure, which is a linear structure, which will help us

develop algorithms to understand nonlinear structures later on. But to understand linear structure, we need to understand lines and the calculus of lines, which is linear algebra.

To deal with uncertainty, our mathematical language to deal with uncertainty is probability. So, we would need probability. And finally, to convert data to decisions, you need to solve a problem that maximizes something or minimizes something. And then you would have to convert data to decisions, which is where your optimization comes into picture. So, these three become key prerequisites for this course.

(Refer Slide Time: 22:44)



So, this is a tentative roadmap for this course, most likely, I will try to follow this roadmap. So, basically, you are at the beginning of this course. So, let us assume this is you. And then you have a long way to go before you finish this course. And then, mount your flag on the castle of victory, whatever you want to call it.

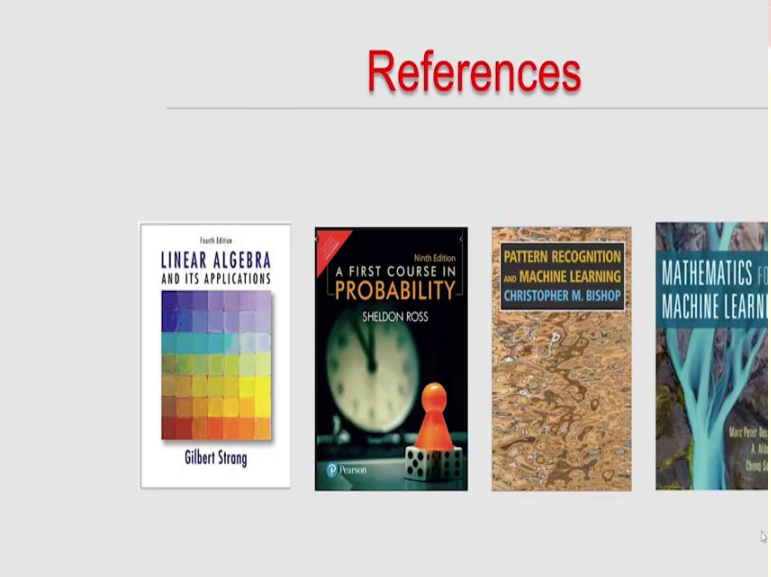
So, on the way, you are going to see a lot of levels. So, the first one would be unsupervised learning, which includes representation learning, and then we will talk a bit about clustering in unsupervised learning. And then we will talk about estimation methods specifically tailored for unsupervised learning.

Then we will talk about supervised learning in the second part of the course, some basic algorithms for supervised learning, and some advanced algorithms for supervised learning.

Again, I would not talk about the exact algorithms. Now, as we go along, we will see what these algorithms are. And finally, we will talk about some advanced topics in supervised learning.

So, this would be the high level, roadmap for this course from, where you are right now to where you will be after you finish this course. Of course, there are going to be all these assignments in between, well, it is important that, we test ourselves where we stand, and so on. So, all these assignments will actually help you, understand these ideas that we go over in the course very well. So, that is the overall plan.

(Refer Slide Time: 24:16)



The high-level references, when I say high level, because I am not going to follow a single textbook for this course. But these might be useful references. So, for linear algebra, if you want to brush up linear algebra, you can look at the book by Professor Gilbert Strang on linear algebra and applications. For probability, a first course on probability should suffice by Sheldon Ross, which is a great book.

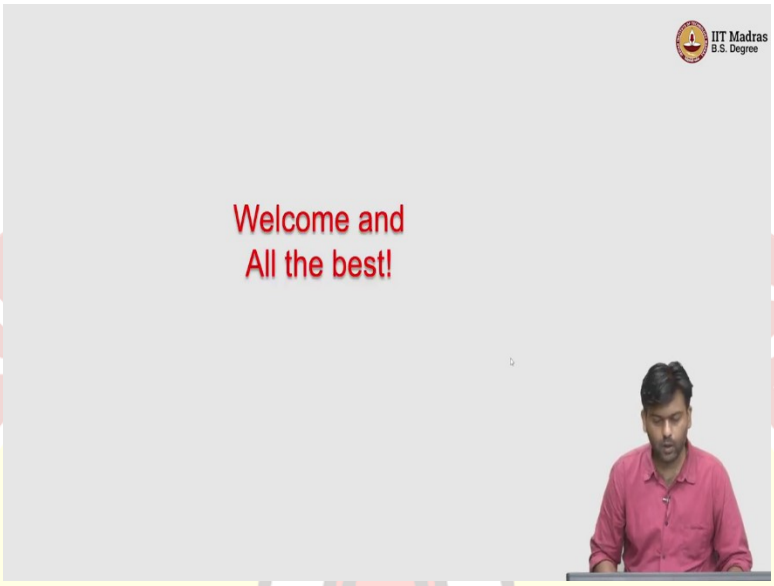
The go to reference, would for this course, would be the pattern recognition and machine learning book by Christopher M. Bishop. Though, we may not follow the notation of that book, we may not follow the order in which ideas are presented as that book, but then whatever we see in this book has a relevant topic in that in the VRML book by Christopher Bishop as well. So, you would be able to do that mapping yourself.

If you just search for things that are being taught in the class in that book, the book is, I think one of the editions of the book is freely available, as well. Another interesting book, which might be very useful for this course, is the mathematics for machine learning book. Again, a soft copy is available, which may available by authors for free, you can look at that.

So, this would try to, combine whatever you might have seen in your machine learning foundations course, or pre-requisite courses that you might have taken, and then put them in a machine learning context. So, very nice book, definitely recommend you to take a look at that as well. Again, as I said, these are references.

These are not, we are not going to be following strictly in a textbook for this course, the class, the lecture notes should suffice with respect to solving the assignments or solving the exam problems for this course. But if you want to go over and beyond that, and learn a little bit more deeper about some of the ideas that are being taught in this class, you can always refer to these books. So, some of these, for instance, the PRML book would have a super set of ideas that are being covered in this course.

(Refer Slide Time: 26:27)



So, with that we will stop the high-level introduction part of the course. So, I welcome you all to the machine learning techniques course and wish you all the best and hopefully see you for the next lecture. Thank you.

IIT Madras

ONLINE DEGREE

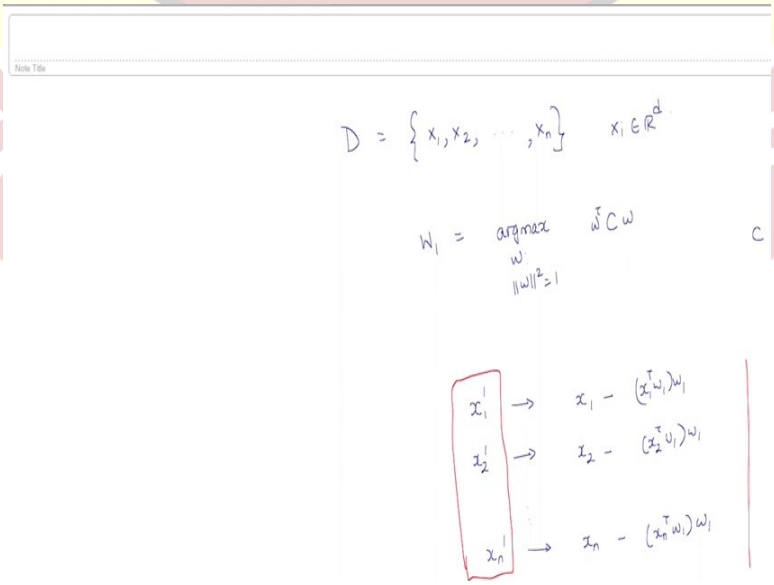
Machine Learning Techniques
Professor Arun Rajkumar
Department of Computer Science & Engineering
Indian Institute of Technology Madras
Principal Component Analysis Part - 1

Hello and welcome back. So, we will, in this video continue our discussion of this new algorithm that we are trying to develop for unsupervised learning, where we were trying to find a line that given a data set minimizes the sum of the lengths of the projections onto this line. But when we did that, we came up with a lot of questions. For example, we started with an error minimization problem and we post it as an equivalent maximization problem of a particular objective.

We want to understand that in detail. So, the second question is, we came up with a possible algorithm where we find iteratively multiple lines. And then we wanted to know, how many rounds should we run this procedure for? When should we stopp this procedure, that was the second question.

And more fundamentally, we were trying to understand where is really the compression happening? And what are the representations that we are really learning in this problem? What we will do in this video is try to answer some of these questions. So that the algorithm that we are trying to develop kind of solidifies and then we have a solid algorithm in place.

(Refer Slide Time: 01:31)



So, we will start today with where we were last time. So, we had a data set. Let us call this $\{x_1, x_2 \ldots . x_n\}$ as usual, the data points are in d dimension and we have a vector w_1 that we are able to find for this data set, which is argmax over w, such that $\|w\|^2=1, w^T C w$. When you say argmax, it is the argument that maximizes the subject to function and which means specifically, that w that maximizes the subject to function, C here was just $\frac{1}{n}\sum_{i=1}^{n}(x_i x_i^T)$.

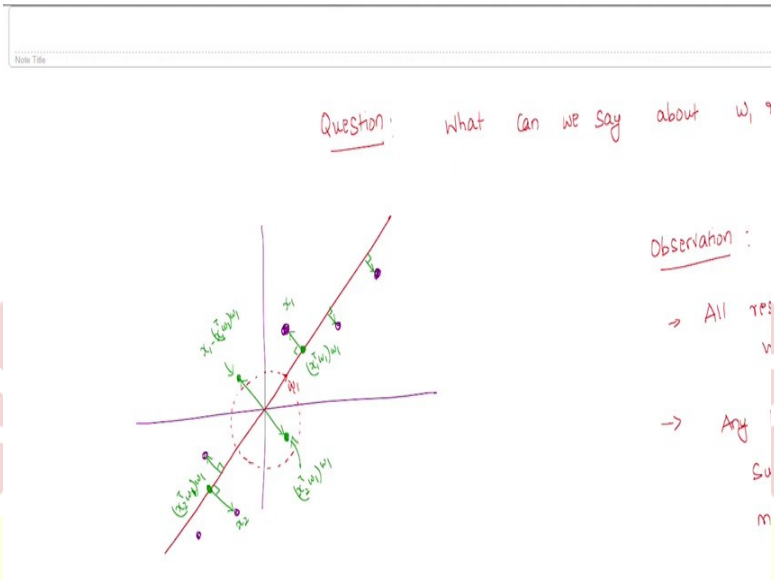
So, once you find this, w by the best line that kind of minimizes the error or maximizes this quantity, what we then said was, we would compute this new data set where the point x_1 becomes x_1' , which is just $x_1 - (x_1^T w)w$

x_2' would be $x_2 - (x_2^T w)w$ and so on and x_n'.

So, I mean, I am just calling it tilde it should be primewhich is $x_n - (x_n^T w)w$. So, we have this as a new data set so, these set of points for our new dataset now. Remember, all these points are still in R^d, just that they are the error vectors in $d$dimension. So, now what would our w_2 be our w_2 would be the argmax over w such $\|w\|^2=1$, $w^T C' w$, where C' is now the matrix dx$d$matrix that you compute out of these error vectors, which would simply be $\frac{1}{n}\sum_{i=1}^{n}(x_i' x_i'^T)$.

So, this would be the new matrix. So, now you have two lines, so, the first line was w_1, which was the best line that minimize the error with respect to the data set now, you took the error vectors and formed the second line, which is w_2.

(Refer Slide Time: 04:06)



Now, a natural question arises, we have two lines now, can we say anything about the relation between these two lines? So, natural question is the following. What can we say about w_1 and w_2? So, in particular, is there any specific relationship that exists between these two directions? So, let us let us look at it geometrically.

So, we have a bunch of data points. As usual, the data set let us say centered and now let us say this was our erase this point, it can fall on the line does not matter. But let us say to show it clearly, so let us say this was the best line w_1. So, w_1 is somewhere here. To remember, so w_1 is unit length. So, it would be the direction that would point (along the) in the unit circle. Now, what are the error vectors?

If this is x_1, this is our error vector, this point is $(x_1{}^T w_1)w_1$, which means this error vector, which is actually somewhere pointing this is the remaining piece, which is $x_1 - (x_1{}^T w_1)w_1$. So, it is pointing in this direction. Let us take one more error vector, which is perhaps this error vector, which is this vector, maybe this point is x_2 this point is $(x_2{}^T w_1)w_1$. So, this error vector is here. And this point is $(x_2{}^T w_1)w_1$ and so on. So, now, as you can observe that these error vectors all line up in the same line.

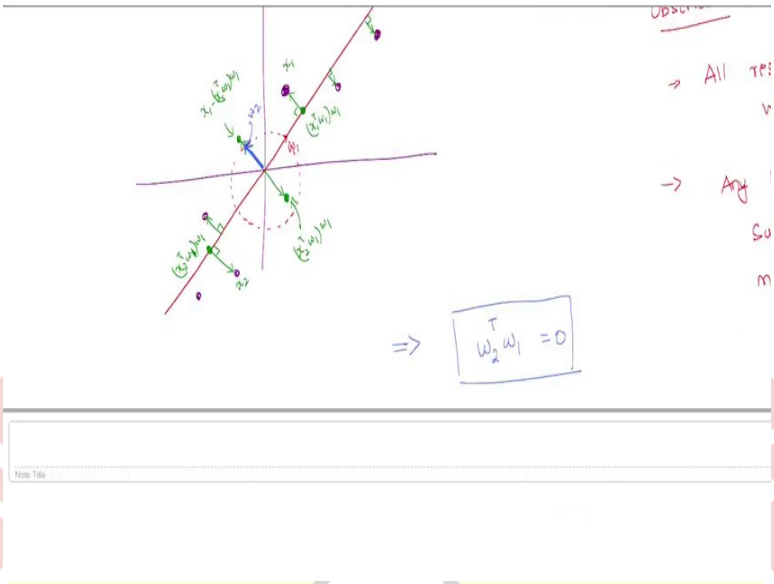
Now, why should that be the case that should simply be the case, because all these error vectors are perpendicular to the original line that we created that is because of the nature of projections here. So, the projections are all perpendicular. These are all perpendicular. And so, if we just create a data set which only has the error vectors that will be (per on the) that will be on the line perpendicular to the original line that we found so, in in 2 dimensions.

In higher dimensions, it will be in some hyperplane, which is perpendicular to the original line that we found. But in 2 dimension there are only two perpendicular lines possible. So, it will be on a line. So, now, if we find w_2 which best fits these error vectors, and because of the fact that the error vectors are all lining up along the line, the best line would exactly be the line where they line up along.

So, because if you pick any other line, I am going to suffer some extra error, which I can avoid by picking w_2 as the line which is exactly the line where these error vectors are falling on, which is the line perpendicular to w_1. So, which means the observation again, I did not prove this, but then the observation is that all errors or let us call them residues residues are orthogonal to w_1.

And this implies, any line which minimizes sum of errors with respect to residues must also be orthogonal to w_1, this needs proof, it is not too hard. I am not going to do this, but then try to argue why this is true yourself. If you are not already seeing it, this is a good place to pause and think about why this must be true. So, we won't prove this formally, but then it is not too hard to do this.

(Refer Slide Time: 08:34)

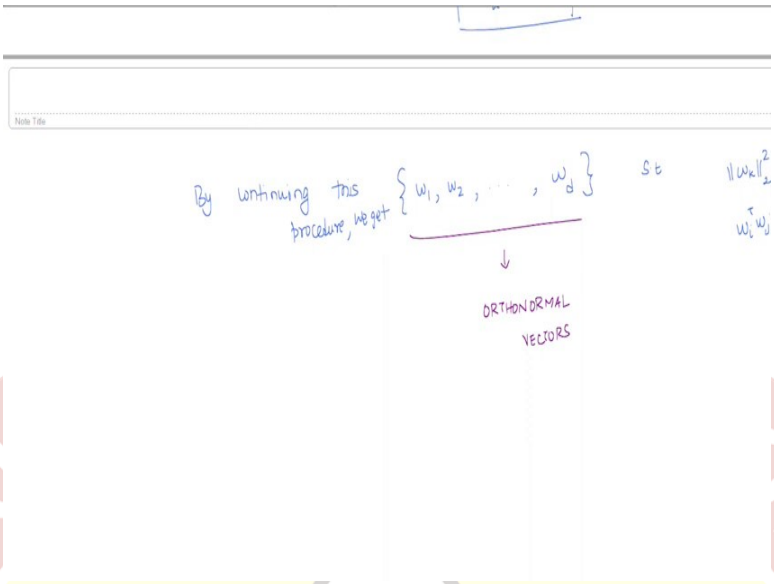


So, what are we essentially saying? So, the implication is that the conclusion is that $w_2{}^T w_1$=0, so that is what it means to say that, w_2 is orthogonal to w_1, which means are my w_2 is actually going to be just, something this way? Not that vector, it has to be, by definition, it has to be unit length, so it would be this way, this would be w_2, well, it could be in the opposite direction also, that is also fine. So, that would also be a valid unit vector.

But then nevertheless, that does not violate the orthogonality property at least so the orthogonality still holds for w_1 and w_2. So, now what happens if we continue this procedure, so in 2 dimension, you have to stop here because after 2 lines, you kind of found 2 lines such that the residue of the residue is just the 0 vector.

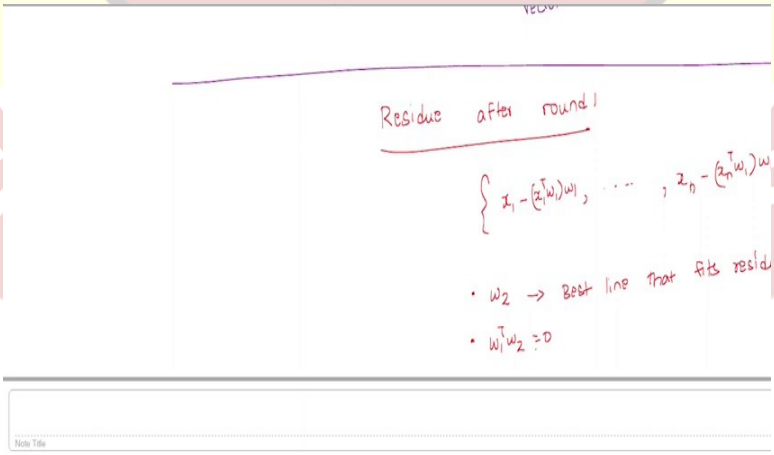
So, because this line exactly fits the data, so there is no residue anymore. But then if your data point was in 100 dimensions, then you do not have to necessarily stop after 2 rounds you can still continue this procedure and you might get residue of the residue on so on.

(Refer Slide Time: 09:55)



And as we keep fitting lines, what we would observe is that we would get a bunch of lines, $\{w_1, w_2 \ldots . w_d\}$. So, in R^d by continuing this procedure we get w_1 to w_d with the property that $\|w_k\|^2=1$ for all k and $w_i{}^T w_j=0 \forall\ i \neq j$, when basically you get w_3, which is orthogonal to both w_1 and w_2, w_4, which is orthogonal to all the previous one and so on. So, basically, what we have ended up getting is a set of orthonormal vectors, orthonormal is just a set of vectors which are mutually perpendicular and of length 1, so, that is the orthonormal vectors. So, this is good.

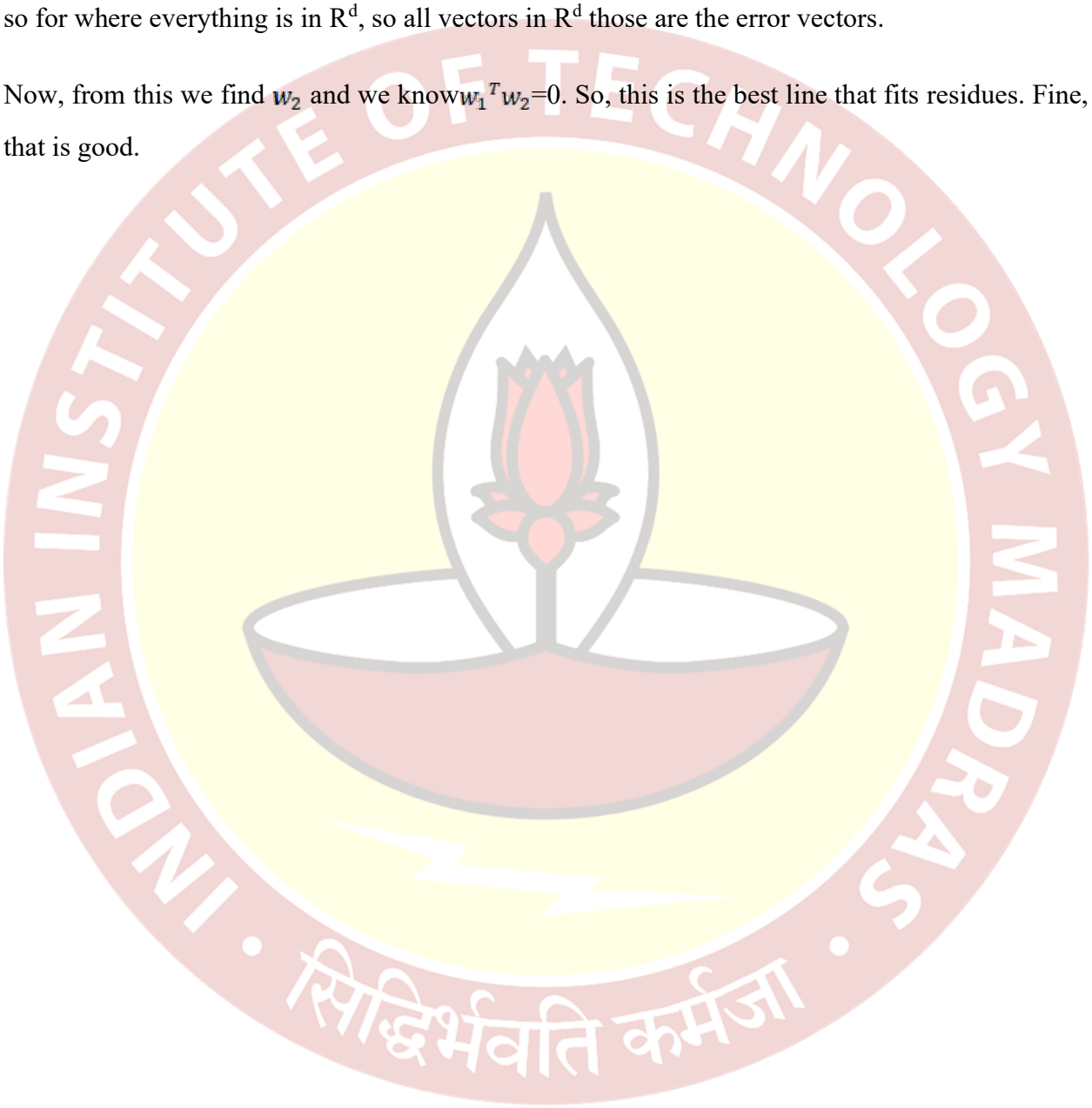
(Refer Slide Time: 11:09)



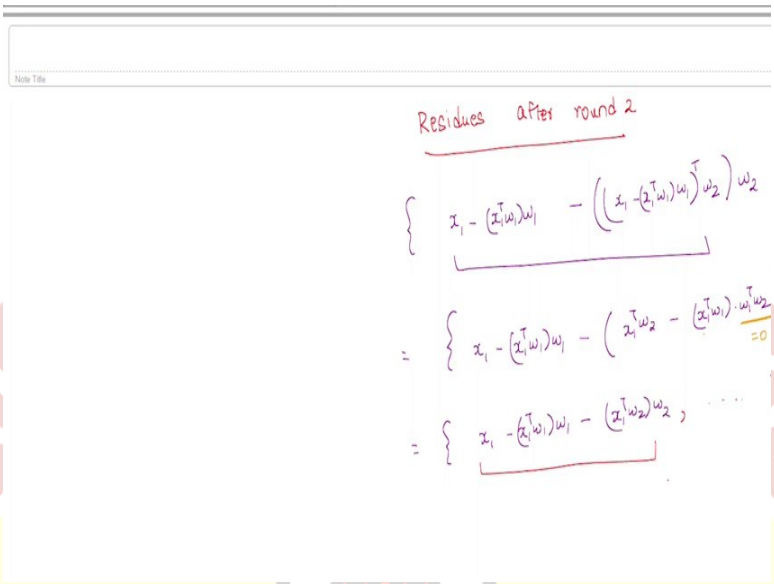
Now, just to understand what is the use of this orthonormal vectors, let us, let us understand this a little bit in a different way to explicitly computing the residue after round 1. So, we know the residue after round one is the set or the data set that is used to compute w_2 is of this form. So, $x_1 - (x_1{}^T w_1)w_1$,.....$x_n - (x_n{}^T w_1)w_1$

so for where everything is in R^d, so all vectors in R^d those are the error vectors.

Now, from this we find w_2 and we know $w_1{}^T w_2$=0. So, this is the best line that fits residues. Fine, that is good.



(Refer Slide Time: 12:13)

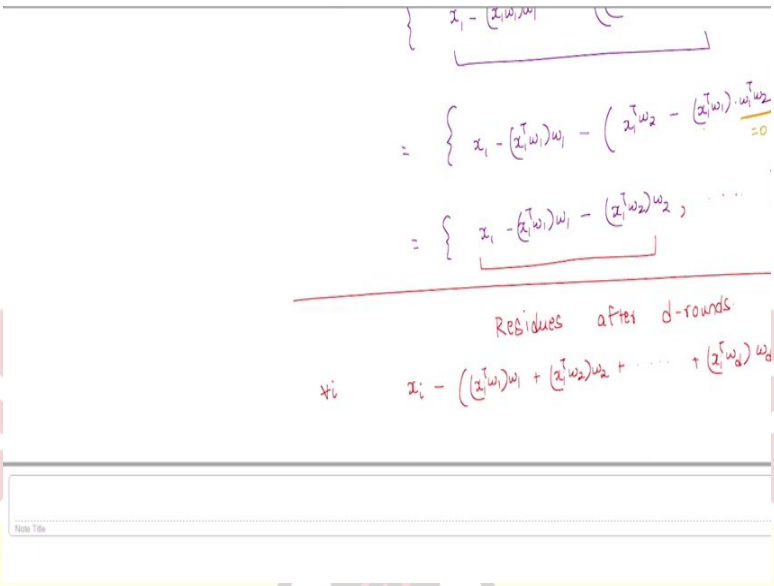


Now, we have two w's. Now, let us say we go ahead and compute the residues after round 2, just to see what is happening. So, it is maybe some of you are already seeing it. But let us try to compute this explicitly. Now, what would be the residues after round 2? From round 1 data set, which was this dataset, we compute a w_2.

And then we take the residue of each of this data point with respect to w_2, which means that we have to take this data point subtract it out from its dot product with respect to w_2. So, which means the first data point would be, this was the data point, which was $x_1 - (x_1{}^T w_1)w_1$, now, this was my original data point, I subtract out its dot product with w_2, the same vector transpose w_2, this is the scaling factor with respect to w_2.

So, this is what I subtract out and then I do this for every vector, let us focus on one point to see what comes out. Now, what is this so, this is just let me try to simplify this $\{ x_1 - (x_1{}^T w_1)w_1$- $((x_1{}^T w_2) - (x_1{}^T w_1)w_1{}^T w_2)w_2, \ldots\ldots\}$ which is exactly $\{x_1 - (x_1{}^T w_1)w_1 - (x_1{}^T w_2)w_2 \ldots..\}$ The first term is $(x_1{}^T w_2)w_2$. , but now, we notice that because of the fact whatever we argued earlier that w_1 and w_2 are orthogonal, this value is 0, which means that the second term is 0. So, which means this would be our first data point and so on. So, after round 2, we observed that the residue is x_1 minus x_1's projection onto w_1 subtract out the x_1's projection onto w_2.

(Refer Slide Time: 14:14)



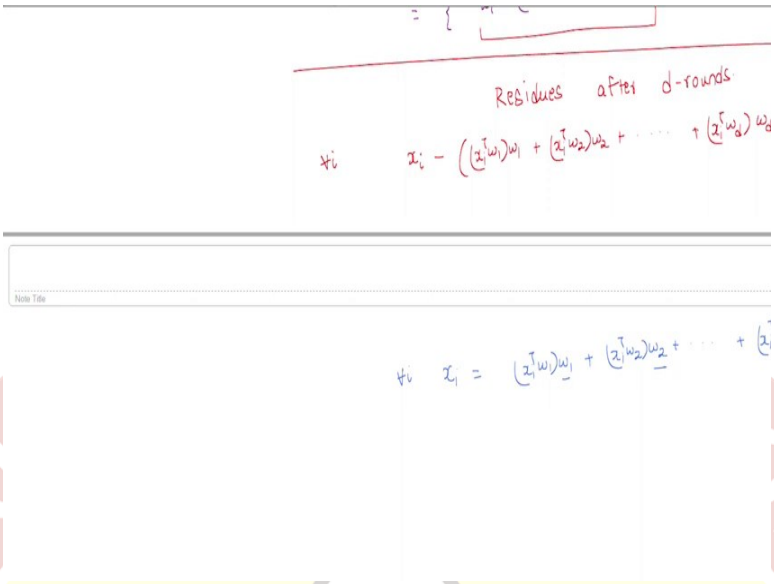
Now, if we continue this after d rounds again this needs an argument but then it should not be too hard to see that residues after d rounds what would be the residue? It would be for all i the residue x_i would be$x_i - (x_i{}^T w_1)w_1 - (x_i{}^T w_2)w_2 \ldots\ldots (x_i{}^T w_d)w_d$.

Now, what do you think should be the residue after d rounds so I am in D dimension all the data points are in D dimension and I run this algorithm d times and then I look at the residue What do you think would be the residue after d rounds? In 2 dimensional case it was very obvious.

So, you had an initial set of data points you found a line, formed the projections and the error vectors with respect to the best line all the error vectors lined up so, the second line exactly fit the data set. So, the residue after round 2 would just be set of all 0 vectors because the second vector exactly fit the data points, all the vectors are lined up.

Now, the same thing would happen in D dimension if you run it long enough. So, which means if you run it for d rounds, so after d rounds, all the error vectors would simply becomes 0 vectors. So, because you have kind of removed all the components of this error vector, and there can only be d independent components, which are w_1 to $w_d$in this case.

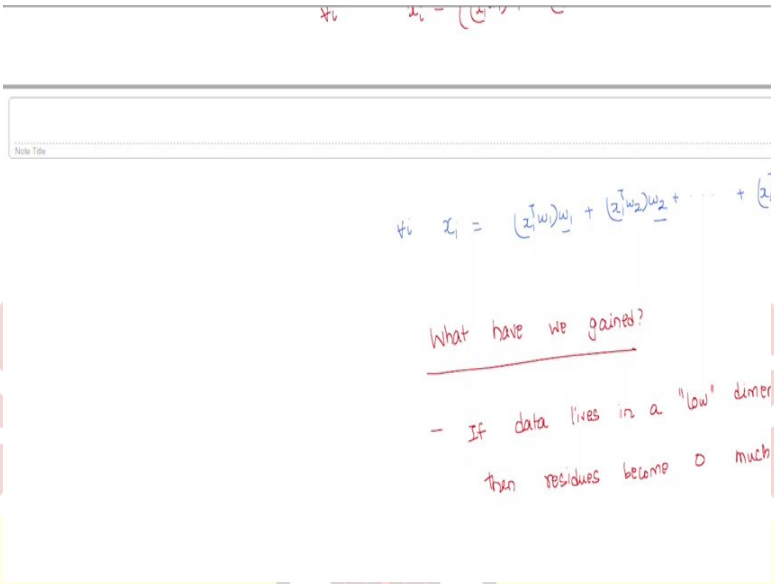
(Refer Slide Time: 15:50)



So, which means all these vectors are actually 0 vectors after d rounds so, in R^d. Now, that tells us that x_i itself is just $x_i - (x_i{}^T w_1)w_1 - (x_i{}^T w_2)w_2 \ldots\ldots (x_i{}^T w_d)w_d$. So, basically, what we have done is again, if you know a bit of linear algebra, you can already see what is happening is that we have expressed our data set, every point in the data set in a different basis in a different orthonormal basis.

So, the $\{x_1, x_2 \ldots .x_n\}$ were expressed in standard basis. Now, you are expressing the same data set in the basis of w_1 to w_d. So, basically, all we have done is doing a change of basis. So, all these data points can be expressed in any basis whatsoever. So, there is nothing, the question is what have we gained by doing this. So, we could have had the standard basis itself. But then now we are choosing to express all the data points on a different basis, which is the w_1 to w_d basis.

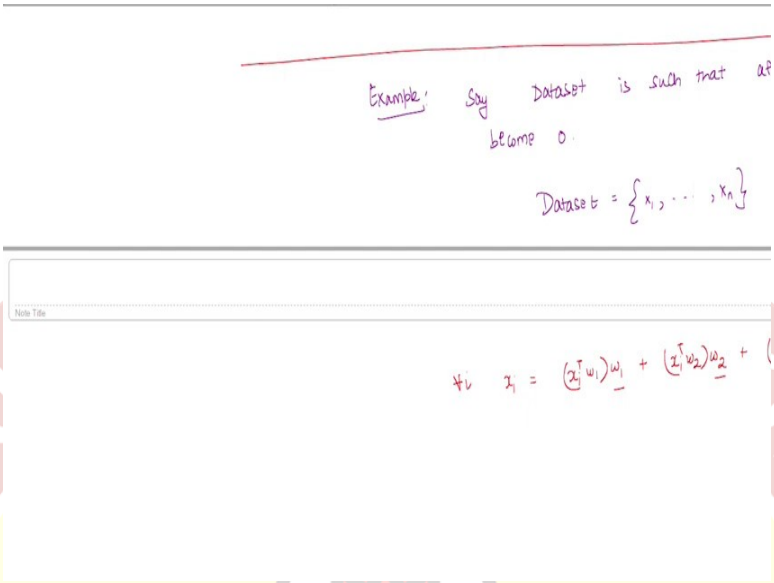
(Refer Slide Time: 17:07)



So, what, so the question is, what have we gained? What have we gained? The first thing we are gaining is the following. So, if our data lives in a low dimensional linear space or subspace, then we do not have to run d rounds. So, the residues become 0 much earlier than d rounds. So, so, we keep running this algorithm iteratively and at some point, we realized that the residues have all become 0, then there is no point in continuing further and finding a different line, so because that is not going to add anything more to our information, because all the vectors is 0.

So, what is this telling us that if it so happens that your data was in low dimensional space, then your algorithm would stop much earlier than running for d rounds.

(Refer Slide Time: 18:29)

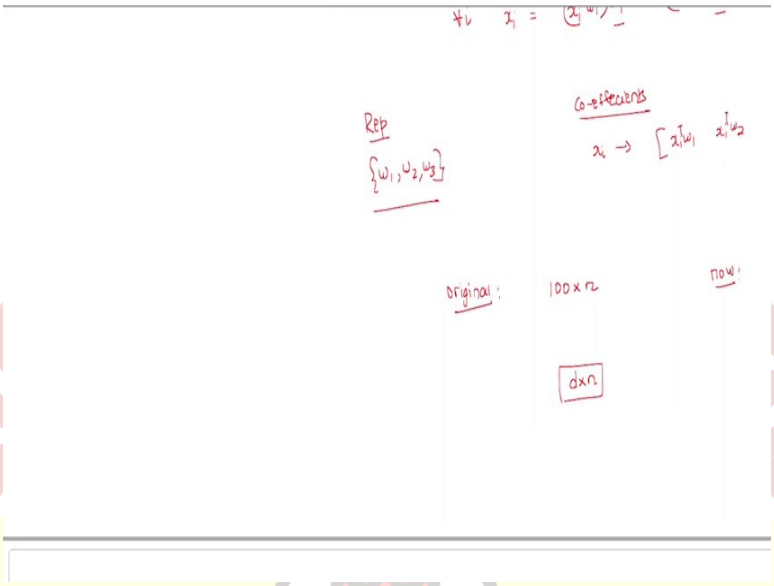


So, why is this a useful thing now? So, this is useful, because let us take an example and see why this is an useful thing. So, for example, say data set that we have such that after 3 rounds, the residues become 0 so, you run 3 rounds, and then all the error vectors residues become 0. So, residues become 0. Now, what does that tell us?

So, the data set itself initially was in some D dimensional data set that is the usual data set that we have $\{x_1, x_2 \ldots .x_n\}$ in data points all x_i are in R^d. Now, what does what does it mean to say the residues become 0 after 3 rounds, it means that there exists we have found w_1, w_2, w_3, such that for all i x_i equals $x_i - (x_i{}^T w_1)w_1 - (x_i{}^T w_2)w_2 - (x_i{}^T w_3)w_3$. Now, where are these w_1, w_2, w_3? w_1, w_2, w_3 are the lines that we find for our datasets.

So, they are all in R^{100}, , so let us say this is 100 for the moment to make it to give some numbers. Let us see the dataset this is on 100 dimensional. So, all these three w_1, w_2, w_3 are in 100 dimension. That is good.

(Refer Slide Time: 20:12)



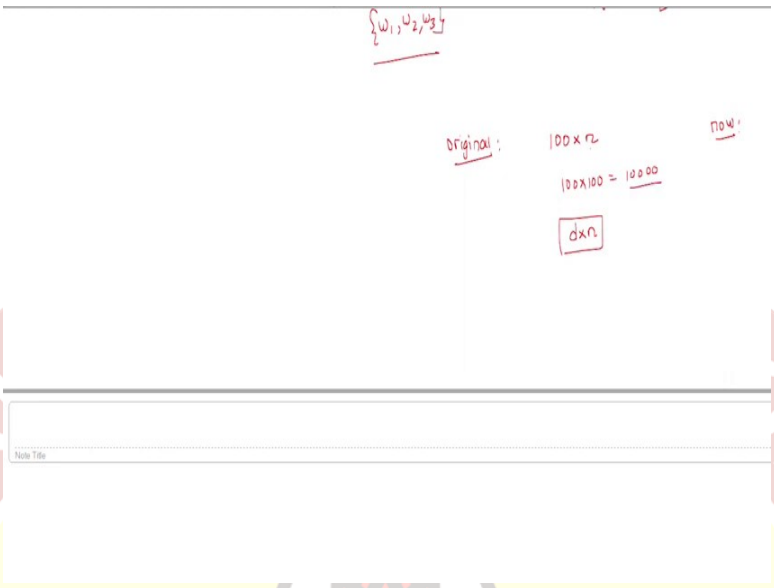
But what are we saying? We are saying that now if we have the representatives and the coefficients, so the representatives, of course, will be w_1, w_2, w_3, which will all three lines are in 100 dimension, but the coefficients now, to reconstruct x_i, we do not need 100 numbers, so we need we need only three numbers per data point.

So, coefficient for the ith data point, so R just $x_i^T w_1$, $x_i^T w_2$, $x_i^T w_3$, it is just the dot product of the data point with respect to each of these directions that we have found so, which is in R^3, just 3 numbers per data point. So, initially, we had 100 dimensional data points n of them.

So we had to do, the naive way to store these would be to say that we would have 100 into n numbers that that are needed to store. Now, because of the observation that after 3 rounds, the residues all become 0, then it means that for the representative, you need 3 into 100, 3 into 100 for w_1, w_2, $w_3$plus, for each data point, you only need 3 points.

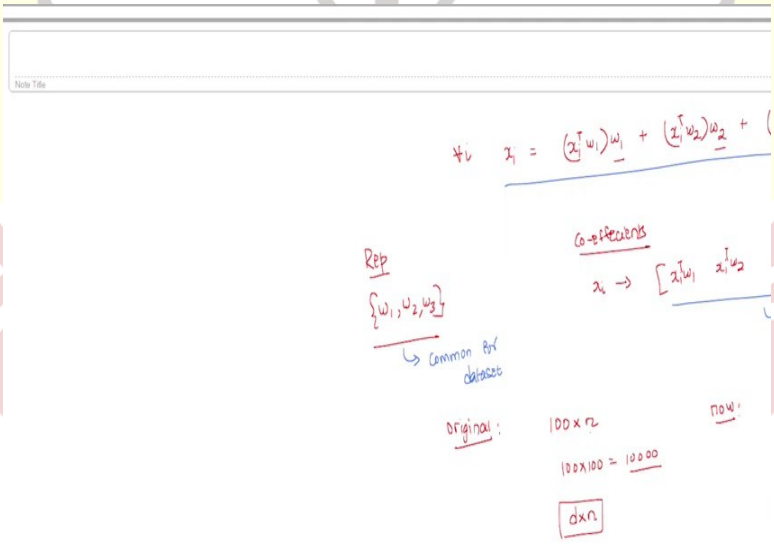
So, which would be just $3n$. So, this is, the number of this is the total number of numbers that you need to store or values that you need to store. So, in general, instead of d xn, which you may have to store now you have to store$d$ xk, where k is the the number of rounds after which the residues become 0, plus kxn x for each data point we only have k numbers instead of d numbers, so, d xn could be much, much larger than d xk + kxn , especially if k is small.

(Refer Slide Time: 22:12)



So, think of n as 100 so, then this means that this is 100 x 100, that would be 10,000 whereas this is 3 x100 + 3 x 100, which is just 600. So, instead of 10,000, we have only managed to store only 600 numbers. Of course, this is assuming that you have the residues become 0 after 3 rounds.

(Refer Slide Time: 22:37)

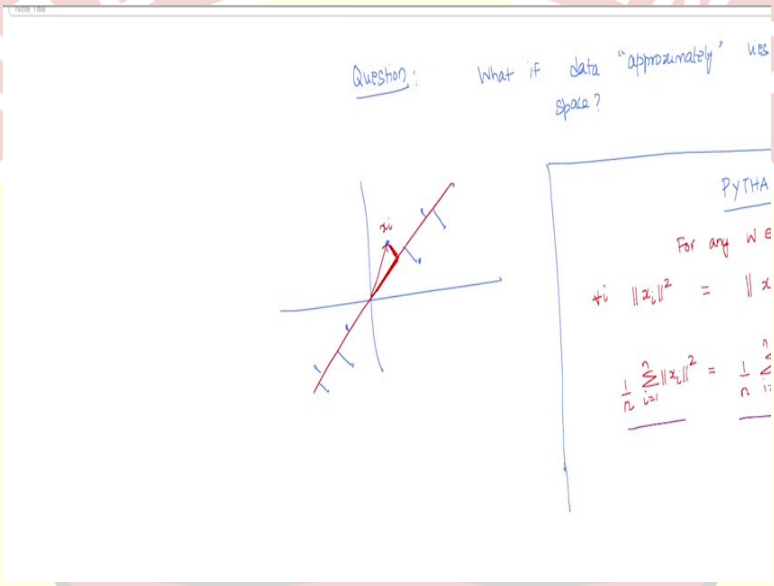


Now, what just to be very sure, so this, this is data points specific, so every data point, the numbers that we are storing, will depend on what the original data point itself, so this is data points specific. The w_1, w_2 is common for the entire data set. So, that is why they are the

representative and then this representation for this data point depends on the representative but then specific.

Now, if we want to reconstruct the data set, we could do that using the representative on coefficients by taking the linear combination of the representatives where the coefficients are specified by these, so that would give us exact reconstruction in this case. So, now, this is good.

(Refer Slide Time: 23:32)



Now, the question then is, if it so happens that after 3 rounds, the residues become exactly 0, were good, fine. But this may or may not happen in practice. And the relevant question that we should answer in practice is, what if data, approximately, lies in a low dimensional space. What does that mean?

That means that, for instance, the simplest example is the 2 dimensional space again, we might have data like this. There is no single line that fits the data so, there is no single 1 dimensional space where the data actually lies in, but still we are kind of happy with this line 1 dimensional representation, which means that we are saying we can tolerate some amount of error. So, that number has to be fixed by us how much tolerance we need, should be fixed by us.

So, how can we fix this tolerance? What should we look at to decide this tolerance? There are different things that you can look at. And the simplest way to decide on a tolerance would be to

do the following. So, here, let us look at the Pythagoras theorem. Let us take any data point. So, let us say look at any data point and look at its length.

So, maybe some data point x_i and then its length. By Pythagoras this length is the sum of the lengths of its projection onto any line not just the base line, any line plus the error vectors length so, length squared sorry length squared of x_i is the sum of the length squared of the projection onto any fixed line plus the error vector that it suffers with respect to that line.

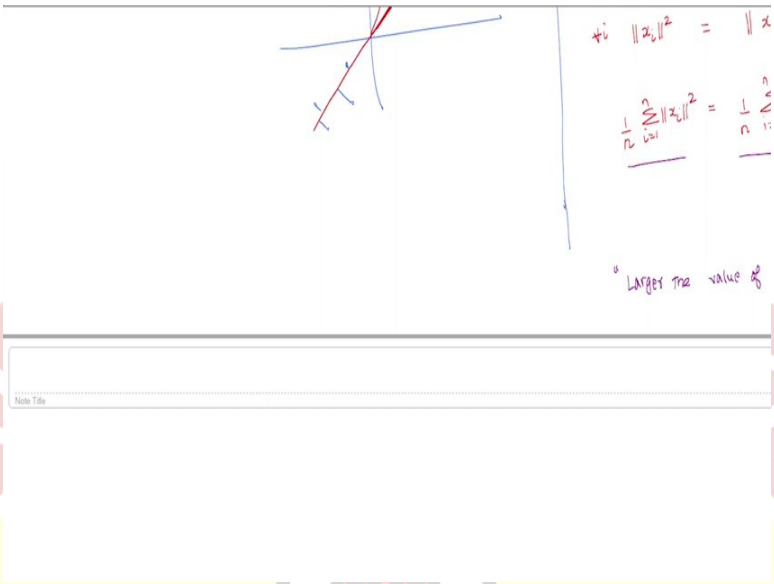
So, which means this is $\|x_i - (x_i^T w)w\|^2 + \|(x_i^T w)w\|^2$, so, this is true for any data point. Now, this is true for all i which means so, if I just average this left hand side over all i this is so for any w, so, in R^d this is true, where x_i is all in R^d. If I do the average in here, I get this $\frac{1}{n}\sum_{i=1}^{n}\|x - (x^T w)w\|^2$

$+\frac{1}{n}\sum_{i=1}^{n}\|(x_i^T w)w\|^2$.

For any w in R^d, but then we typically care about w such that w's length is 1, the fact is true for any w, if you have any w then you will have to divide this particular length of w so let us fix the length of w to be 1 so, that this is true always. So, now what are we saying we are saying that for any w the average length of the data set is equal to the average error that the data set incurs with respect to the w that we have plus this quantity.

So, which means we want that w such that this quantity is as large as possible, the error should be as small as possible, or this quantity should be as large as possible which means we can look at this value after we find a w and see what happens to that value.

(Refer Slide Time: 27:20)



So, basically, what we can do is use, we will the fact that larger the value of the second term, which is $\frac{1}{n}\sum_{i=1}^{n}\|(x_i{}^T w)w\|^2$so, this is a constant that that square of it will come out $\|w\|^2$is anyway 1 so, (the) this is essentially$\frac{1}{n}\sum_{i=1}^{n}(x_i{}^T w)^2$, the better the fit, so this is exactly same as this, both of these are same.

(Refer Slide Time: 28:00)

So, now what we can do is do the following, so we will do the following, where we will look at rounds, as the number of rounds proceeds, let us call that round k, we will look at the average, that we will suffer with respect to w_k, the kth rounds value that we find out.

So, now, this is going to be as large as possible for w_1, this is for round 1, because you are essentially maximizing this for round 1 and now whatever you find for round 2 is going to be lesser than so if it is greater than this, we would have found that in round 1 itself.

So, this value is going to go down, so, this is going to fall down. And let us say it falls down like this so, now if it falls down like this, there might be an elbow point, which is some k, after which these numbers are very, very small. So, what does that essentially tell us at a high level, that tells us that the top k, w's that were found are actually the signal directions.

So, that is where the most of the information is and we will talk about what it means to say information signal and so on a little bit later but for now, we understand that, if it kind of drops down and becomes really small after point, these can be thought of as errors.

So, error directions, so, what we can do is we can kind of use a rule of thumb to say that after a particular value, if the sum of these values is divided by the total sum is greater than sum quantity threshold, let us say 95% of the total sum is kind of explained by just the top k then we will pick only the top k.

We will revisit this in a slightly different notation later that why I am not writing this time but the idea should be clear, so as the rounds progresses, we are trying to find the find the quantity and then cutting it out at some point where we think enough amount of information has been gained. Of course, that is just a rule of thumb.

So, we are not like (saying) arguing that this particular k is the best k unless we make more assumptions for both the data, we cannot really do that, but the rule of thumb would be to say that, this captures 95% of the total mass that you will have where you sum this upon all d rounds so after d rounds, this guy is going to become 0 anyway.

So, so, so now let us we have understood the algorithm, so the algorithm is kind of complete now. So, you will start with some data set, and then you can center this data set, and then start

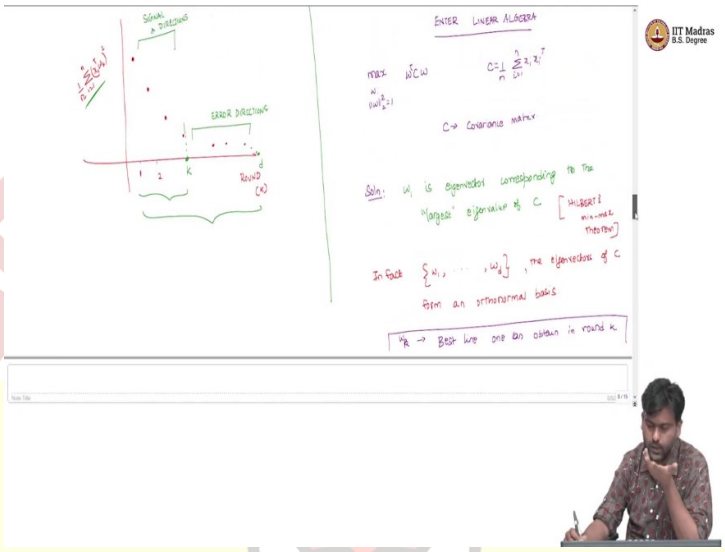
doing this iterative procedure to get the w's and after some point, if you feel that the error vectors, so this quantity that you are trying to maximize is too small or with respect to the total quantity that you have so far managed to achieve. Then you stopped there.

IIT Madras

ONLINE DEGREE

Machine Learning Techniques
Professor Arun Raj Kumar
Department of Computer Science and Engineering
Indian Institute of Technology Madras
Principal Component Analysis: Part 2

(Refer Slide Time: 00:14)



So this is one way to understand this algorithm. Here is another way. So we will look at some linear algebra. So let us say, enter linear algebra now, for those who are familiar with linear algebra, we are already seeing these connections. Nevertheless, I will try to explain it now. So, now, we did talk about this briefly last time, but let us make this more precise now.

So, this is the problem that we are trying to solve, where C is $\frac{1}{n}\sum_{i=1}^{n}(x_i x_i^T)$ and this C is called the covariance matrix with some caveats with covariance matrix in general, and when you have a matrix like this, the solution to this problem is nothing but the Eigen vector corresponding to the largest Eigen value of this matrix.

So, the solution to this problem turns out to be w_1 which is the Eigen vector corresponding to the largest eigenvalue of C. So, this comes from a general theorem, which is called the Hilbert min-max theorem. I am sure if you had taken a linear algebra course, the foundational course so, you would have encountered this at some point a variant of this theorem.

So, this is from that. In fact, more is known. So, Hilbert theorem says more. So, in fact, not just w_1 you can go ahead and find out w_1 to w_d. So, the Eigen vectors of C form orthonormal basis. So, the orthonormal basis is of interest for us which does the error minimization of

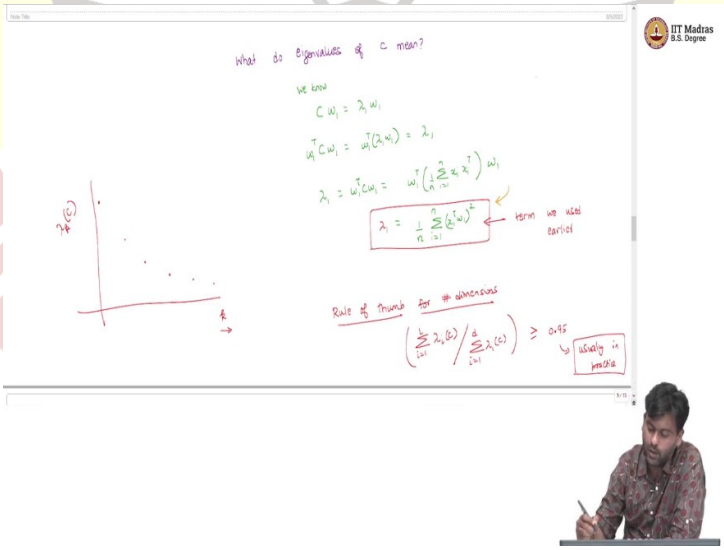
centred data, happens to be just the Eigen vector basis of the covariance matrix. That is essentially the takeaway from the linear algebra point of view.

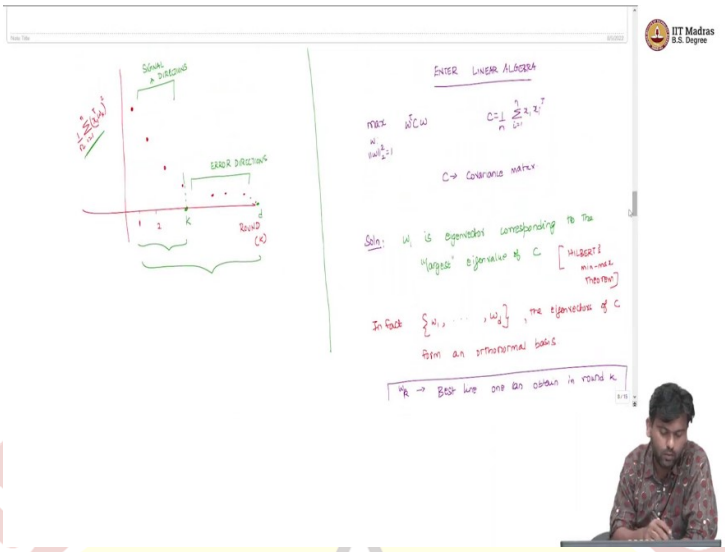
Now, this w_k, the kth Eigen vector or the Eigen vector corresponding to the kth largest Eigen value is exactly the best line one obtains after round k, in round k rather. So, the previous procedure that we said, where we go one step at a time to get these w's, in the kth round, you will get w_k and that w_k is exactly same as the w_k that you would get if you did this.

So, this is again a conclusion that you can derive from Hilbert min-max theorem, that if you try to maximize this same quantity, well, you want w to be norm 1, but then w should be orthogonal to an already existing subspace, well, then, that subspace is formed with the top k-1 Eigen vectors, then the w_k would be the kth Eigen vector.

That is also Hilbert min-max theorem, but we can either understand it that way or we can simply say that this is the line that you get in the round k. Remember, the line that we got in round k will by definition is orthogonal to the previous lines, because the data set that we used were all orthogonal to the to the eigenvectors, to the lines that we derived in the previous rounds by construction.

(Refer Slide Time: 04:09)





So, this is fine. So, we can think of this as eigenvectors and eigenvalues of the covariance matrix, but what does it even mean. So, what do the Eigen values mean? What do Eigen values of C mean? Can we say, can we interpret this slightly differently? Or we should just be happy to say that these are Eigen values of C so that that is fine, but then that does not really tell us anything more than the fact that it is a known, well studied object.

So, can we say something more? Can we think of this in a more geometric point of view? Well, let us try. So, we know $Cw_1 = \lambda_1 w_1$, where Cw_1 is the Eigen vector and λ_1 is, let us say the largest Eigen value. So, corresponding to Eigen vector $w_1$or in this, this works for you in general w_k, where λ_1 will be replaced by λ_k. But for now let us start with w_1. Now what${w_1}^T C w_1$?

Well, that would be${w_1}^T \lambda_1 w_1$, that would simply be λ_1. Because ${w_1}^T w_1$ is one by our definition of looking for lines through unit length. So λ_1=${w_1}^T C w_1$, which equals${w_1}^T \left(\frac{1}{n}\sum_{i=1}^{n}({x_i}^T x_i)\right) w_1$

, that is your C times w_1, which is $\frac{1}{n}\sum_{i=1}^{n}({x_i}^T w_1)^2$. So, now, this term must be, this is λ_1 and this term must be familiar.

So, this is exactly the term we used earlier. So, when did we use it when we used it to find out how many rounds we should run this algorithm for? So, this is basically this term is exactly the Eigen value the highest Eigen value. So, now, basically then we can do the following I

mean whatever we did last time, I mean before can also be interpreted like this. So, you have k which is the number of rounds, or the kth Eigen value and small k.

And now you have $\lambda_k$of the covariance matrix. And if you plot that, that those guys are going to fall down. So, our λ_{ik} of C is the kth largest Eigen value of C because this is exactly the, you know the term that we used earlier too. So now the rule of thumb, so, rule of thumb for how many dimensions, for number of dimensionslast time, like how we just said like a few minutes back would be to look at the following quantities $\frac{\sum_{i=1}^{l}\lambda_i(c)}{\sum_{i=1}^{d}\lambda_i(c)}$, if this ratio is greater than or equal to 0.95. Usually, in practice, this is a thumb rule, usually in practice, then we are saying that while we are retaining some 95% of the information in this dataset, so, that is the general idea.

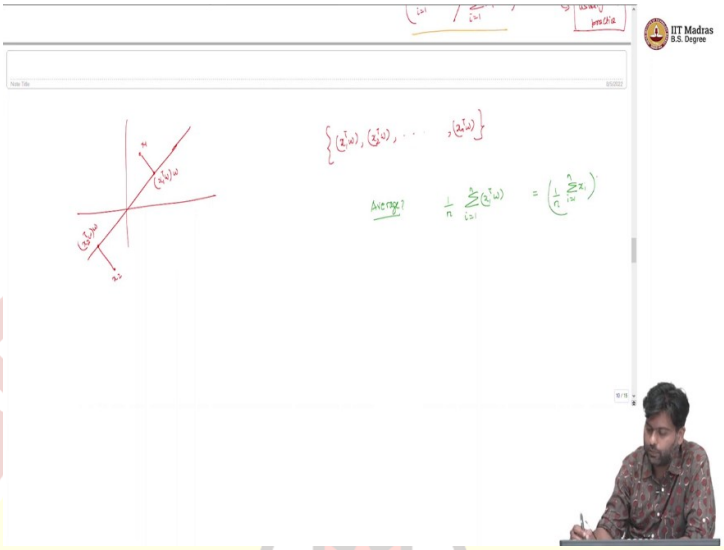
Now, for this quantity to make sense, so, summing up the top k, top l and then dividing it by the entire thing to be greater than 0.95, all of these have to be, positive values, or at least non negative values,? But then that is true, because of the fact that you know, your $\lambda_1$in general, λ_k will be $\frac{1}{n}\sum_{i=1}^{n}(x_i{}^T w_k)^2$e. So this will be just the average of a bunch of squared terms. So all of this will be positive.

So in fact, this covariance matrix, this kind of tells us that the covariance matrix has non negative Eigen values, which means the covariance matrix is for people who have seen this is a positive semi definite matrix, So this is also a proof of that. Anyway, so the main point is that this quantity makes sense. So because these are positive numbers, so you can look at the ratio and see when it crosses 0.95, and then use that as a as a value that you want.

So what we are basically doing is that we are saying that we are maximizing the quantity that we are maximizing is actually the Eigen value of the dataset that we saw of the covariance matrix of the dataset. But that still does not tell us what this quantity is. So it just explains us the quantity equation in terms of Eigen values.

So but that does not really tell us what is this quantity about? So can we understand this quantity in a more simpler way? Specifically, this quantity? I mean, of course, this is, it happens to be the Eigen value of the covariance matrix. That is well and good. But what is it? So can we say anything about this term?

(Refer Slide Time: 09:26)

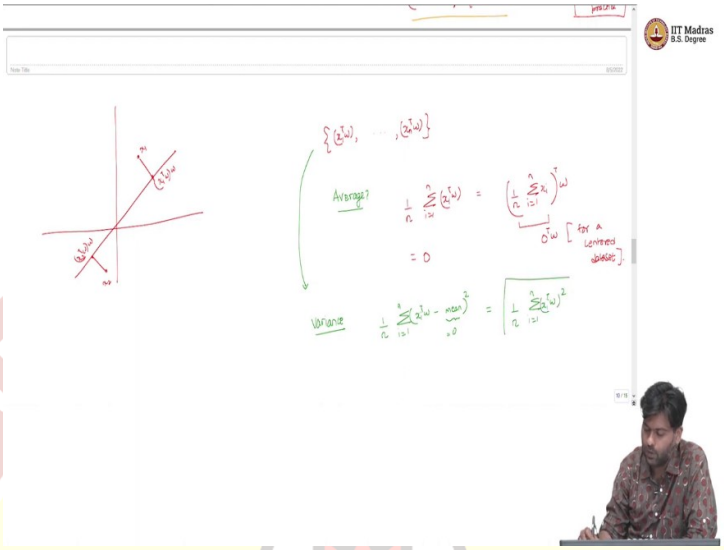


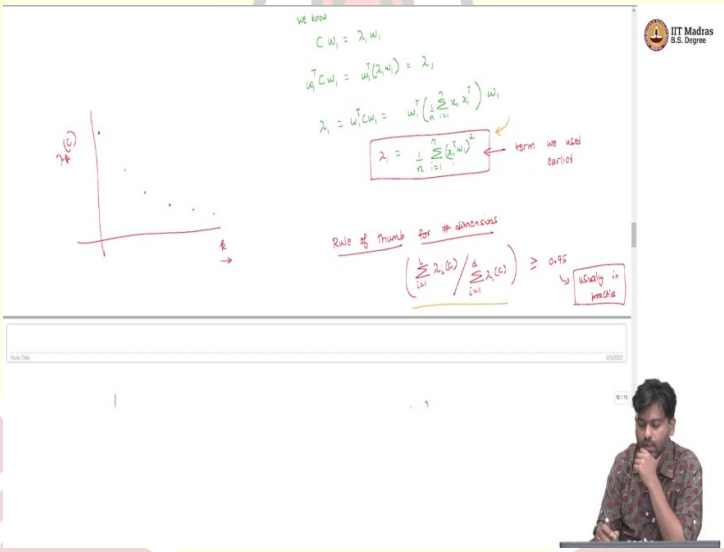
So now, let us say we take, again, a bunch of data points on the line. So maybe there is an x_1, and then this point becomes$(x_1{}^T w)w$, so on, maybe there is an $x_2$becomes $(x_2{}^T w)w$, and so on. Now, let us say we fix w we collect these values $(x_1{}^T w)$, $(x_2{}^T w)$……$(x_n{}^T w)$.

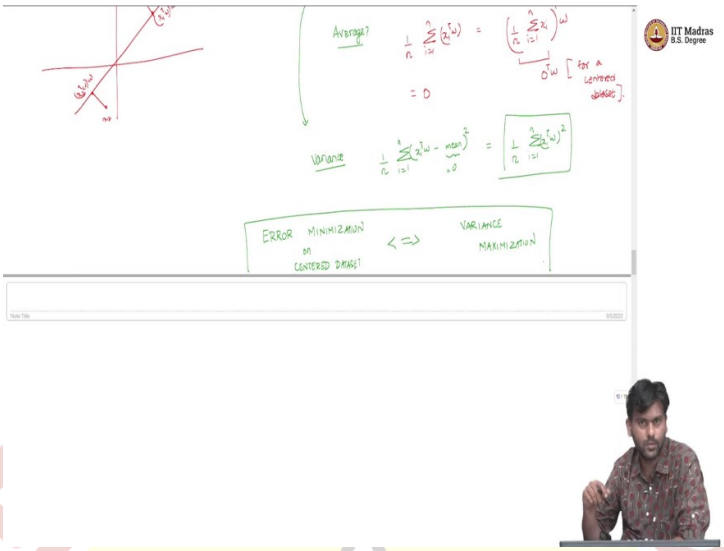
Let us say we collect these values. So these are like the, in some sense the, if you square this, they will be the length squared of the projection, the proxy for each of these data points on the line w. So I am not saying the line w is the best line or anything, it could be any line. And then we are collecting this bunch of data points.

Now, the first question is, what is the average of these values? Average? What would this be $\frac{1}{n}\sum_{i=1}^{n}(x_i{}^T w)$. I am not saying anything about w being the best line or anything, pick some arbitrary line and then project onto that line, compute these $(x^T w)$values and then compute their averages. So what is this value going to be? So you may want to pause and think about this before I tell you the answer now.

(Refer Slide Time: 11:00)







So this value is just $\frac{1}{n}\sum_{i=1}^{n}(x_i{}^T w)$. Now this value for a centre data set, this value is just the mean, but then the mean is 0 for a centred data set. So, we are assuming that the data set is centred. So this is a centred data set. Now, what does that mean? That means that the average is actually 0 for a centre data set. So this is it is a number, it is 0. So now, what can we say about the same set of values?

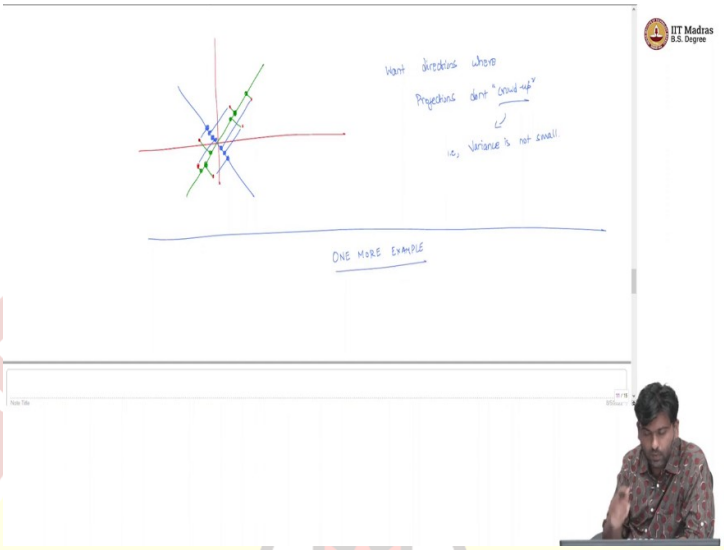
But then let us look at the variance. So, how much spread? Is there in this bunch of values? Now, how would we compute the variance? Well, we can do the $\frac{1}{n}\sum_{i=1}^{n}(x_i{}^T w - mean)^2$, where the mean or the average is the average of this bunch of numbers. But we know that the mean is 0.

That is what we just looked at. So, which means this value is just going to be $\frac{1}{n}\sum_{i=1}^{n}(x_i{}^T w)^2$. So what is this telling us, this is telling us that you pick an arbitrary w and then project all your centre data points on to this w and then just compute the variance of the projected values. Now that variance is exactly the term that we were using earlier. So it is exactly this term that we are trying to maximize.

And it is exactly the term which is equal to the Eigen value. So now, this is interesting. So now at least we know, you know, what are we doing when we are doing this maximization. So basically, then we can conclude the following. So, error minimization, which is what we started this algorithm with, on centred data, on the centering is important because only then the equivalence that we are putting out now holds is equivalent to variance maximization.

So if you find the line where the error is minimum, then it also means that it is the same line where the variance of these projected values is as high as possible.

(Refer Slide Time: 14:02)



So to see why this has to happen geometrically. Of course, algebraically. We have seen this but then geometrically, also, you can kind of convince yourself that maybe you have a bunch of points, like this. And maybe the green line is the best line, maybe there is also this, maybe there is also this blue line. Now you can project it onto either of these lines. If we projected onto the green line. I am going to get the proxies as follows.

The green dots are my proxies. Whereas if I project it onto the blue line, the blue dots are the proxies with the red points. Now, if you focus on the green points, you can see that they are kind of spread out then the blue points which are clouding, So now the problem is if data points all crowded up, in fact, if they go to the same point, then there is no way you can distinguish this one data point from the other.

Whereas, if they are spread out, then this distinguishing capability is still present in this data points. You do want compression, but you also want to be able to distinguish one data point from another because in downstream, you are going to use this compressed representation for some other task, let us say supervised learning, then you will still want to say one data point from the another.

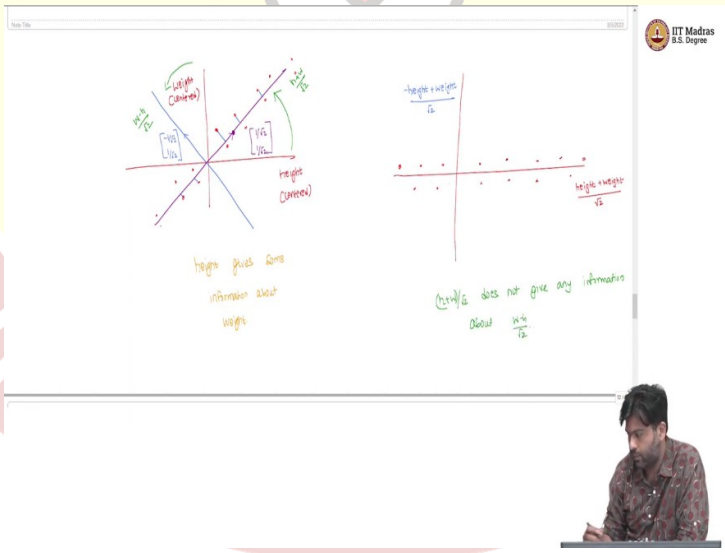
So the distinguishing capability is still necessary, which is lost if you project it along a wrong line. So the wrong line where all the data points crowd closer to each other, which means if the variants of the projector data points is small, then that is a problem. In fact, if, if all data

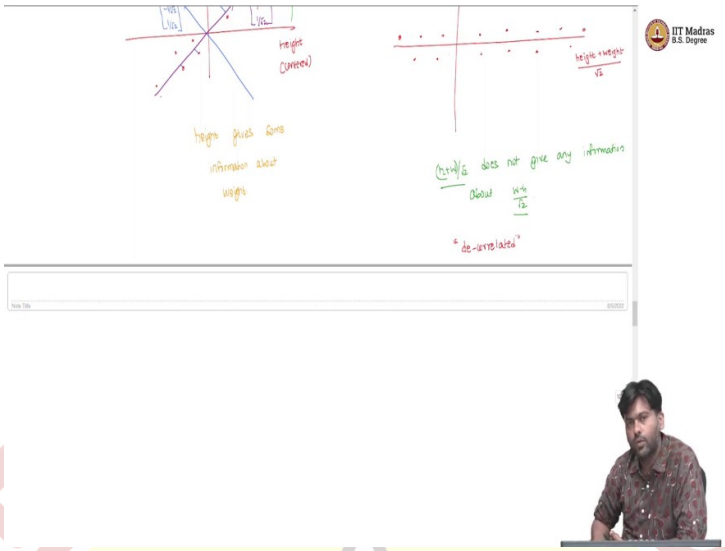
points actually fell on the same line, now, if you look at the perpendicular line, and then think of that line as your actual line, then what would happen is all the data points would reject onto the 0 vector, which means that there is no way to distinguish one data point from another. And that is a bad thing. So because you cannot really use this representation to do much later. So what you really want to do is you want to find directions where the projections do not crowd.

So we want directions where projections do not crowd. So I am using this crowd up in a loose fashion. But for us, the definition of crowd up is that variance implies that this variance is not small. So we want variance to be as high as possible. And in that case, we would have found a line where the spread is as high as possible.

In other words, the information retained is as high as possible. So that is one other way to think about PCA itself. So this algorithm itself, of course so this is one way to think about it. And there is one more example that we will see, and then we will conclude this initial algorithm that we are starting to look at, probably use it this.

(Refer Slide Time: 17:33)





Here is one way to understand what is happening. Let us say we are given, again, a bunch of data points centred, and so on. And let us say this is the height, and this is the weight, they are centred. So that is why you see negative values. So let me put it centred, centred, and so on. Now, if the main thing that we are observing is that height gives some information about weight, so if I tell you what the height is, if the height increases, the weight also tends to increase.

So that is the main observation here. And that is why we started with fitting lines and so on. But the more importantly, height and weight are not completely, loosely saying, independent of each other. So I should be careful when I use the word independent. But let us say intuitively, height and weight are not independent of each other, because height gives me some information about the weight.

So if I choose to represent these data points as height and weight as 2 numbers, height and weight, then 1 number gives me some information about the other number. Now, if I did this algorithm, where do we find these lines? So let us say we found this line, which is the best line and this line, $\boldsymbol{w}$ is going to be [1 1].

Well, of course, it cannot be [1 1], because it has to be of length 1, let us say I normalize this and the length is $\left[\frac{1}{\sqrt{2}}\frac{1}{\sqrt{2}}\right]$, then it means that the first so if I am allowed to use only one number to represent each data point, then that number would be the dot product of each data point along this direction. Well, the data point itself is just height and weight.

Now this direction is $\left[\frac{1}{\sqrt{2}}\frac{1}{\sqrt{2}}\right]$, which means that what I would actually be storing is the value height x $\frac{1}{\sqrt{2}}$+ weight x $\frac{1}{\sqrt{2}}$ which is (height+weight)/$\sqrt{2}$. Now, for whatever reasons, if I went ahead and found the second line, so which is, which would fit the error vectors, it would be along the perpendicular direction,we know that and that vector would simply be $\left[-\frac{1}{\sqrt{2}} \quad -\frac{1}{\sqrt{2}}\right]$. Now that axis, so that value projection along that value would be (-height+weight)/$\sqrt{2}$. Also, if I think of that vectors, you know, $\frac{1}{\sqrt{2}}$, -$\frac{1}{\sqrt{2}}$does not really matter. So it can, it can be either.

So if I, now thought of plotting the same data set, but then by thinking of these 2 axis values, so instead of height and weight, I am going to compute 2 numbers (height + weight)/ $\sqrt{2}$, and (weight- height)/ $\sqrt{2}$, and then plot the same data set, how do you think that dataset would look like? This is a good place to pause and think, let me tell you how the data set would look like. The data set would look something like this.

So basically, what has happened is, you can think of it as if this high axis gotten transformed, rotated, in this case by 45 degrees to get this(height + weight)/ $\sqrt{2}$, because there is a scaling happening, which I am not indicating here. But yes, you can think you can imagine that it is on the weight axis has gotten rotated this way to become (weight- height)/ $\sqrt{2}$. Now, if I think of looking at the same data set in this direction, well, that would exactly look like this. If we plot now what is the point here.

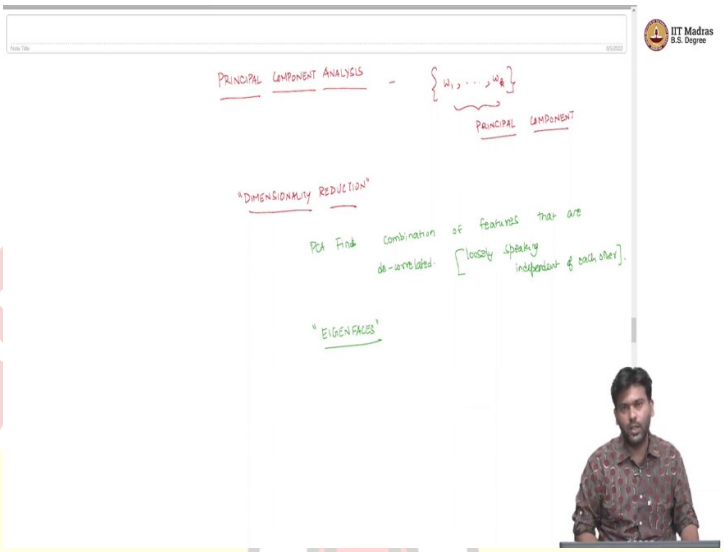
So, what is the point of plotting this well, this is telling us that well, now if I told you the x axis value, which is (height + weight)/ $\sqrt{2}$ value, there is nothing that you can really say about height (weight- height)/ $\sqrt{2}$, if (height + weight)/ $\sqrt{2}$ increases along this direction, there is nothing really happening for (weight- height)/ $\sqrt{2}$, it can either be above the axis or below the axis, but then there is no real pattern there.

Whereas in the previous case, as height increase the weight tend to increase and height decrease the weight also tend to decrease here nothing happened, like that is happening. So in other words, (height + weight)/ $\sqrt{2}$ does not give any information about (weight- height)/ $\sqrt{2}$. So well, in other words, these two directions, so the w_1 and w_2 directions, and then the values that we project on to these directions.

So the right, or the terminology is that these directions are de-correlate for this data set. So the directions where every direction gives you some new information that the previous

directions are not giving you, you are trying to find those directions. So that is what our algorithm is essentially doing.

(Refer Slide Time: 23:06)



And this algorithm well is called the principal component algorithm. You might have come across this algorithm already, if you are taken MLF course. Nevertheless, we thought it would be good to start with this algorithm, because some of the ideas that we will see now, which you may not have seen, the MLF of course, would lead us to something more fundamental in machine learning.

And the w_1 to w_k that we have managed to find these vectors are called as the principalcomponents. And what our example, now say is that these principal directions are in fact, you know, de-correlated. So, the projections along these directions kind of for de-correlated.

So, essentially, the algorithm is given the data set, find the covariance matrix, find the top k Eigen vector and Eigen values, project your data set along the Eigen vector directions, and then that would give you compressed representation, because we are going from R^d to R^k, where k might be much, much smaller than d. This is also a dimensionality reduction technique.

So, our essential dimension that we believe for this data set is only k, It is the Top k Eigen vectors which constitute to 95% of the variance. It can explain 95% of the variance in our dataset. That is why the 0.95 that we used, where it said its information is essentially the

amount of variance explained by the top k Eigen vectors. So, so we should think of each of these directions as a dimension, essentially.

And what PCA does, essentially is now finding new axis, so finding PCA finds a combination of features. So by when I say $x_i^T w_1$, it means that you are weighing these features in x_i using the weights given by these Eigen vectors. So it finds a combination of features that are de-correlated loosely speaking, independent of each other.

So, this is a very fundamental algorithm in data science in dimensionality reduction and so on. And one of the basic algorithms that we will start with for representation learning. So, next time what we will see is one simple example where this algorithm can be applied. And this is this example is called as the Eigen faces example.

So, a face recognition type of a system that you can build using the principal component analysis algorithm will demonstrate that, at least show you some pictures. That is the first thing. And then what we will do is we will identify some potential issues with PCA, and places where it may not necessarily work that well.

And once we have identified those, we will try to fix them and as a matter of fixing them, that will reveal to us some interesting insights about how to look at a dataset, what is important in a data set and so on, which will lead us to more foundational ideas in machine learning. So, all that will come in the next few lectures. For now, we have seen PCA and I hope you are able to appreciate the foundational fundamental nature of this idea and this algorithm. Thank you.

IIT Madras

ONLINE DEGREE

Machine Learning Techniques
Professor Arun Rajkumar
Department of Computer Science and Engineering
Indian Institute of Technology Madras
Representation Learning: Part 1

Hello, everyone, welcome back. Today, we are going to start the proper mathematical foundations of this course, Machine Learning Techniques. And as I said, we are going to cover different aspects of machine learning techniques in this course, which include unsupervised learning, supervised learning. And, specifically, we will look at algorithms for both of these. So, what we will start with first is unsupervised learning. So, the first part of this course will cover different types of unsupervised learning.

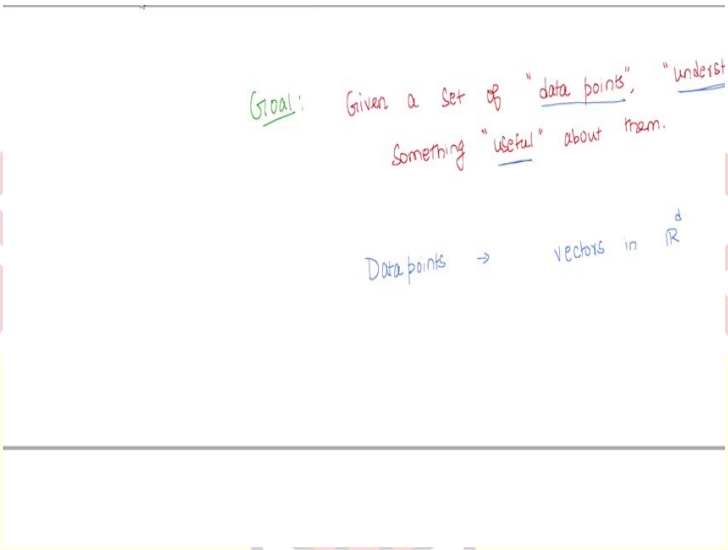
(Refer Slide Time: 0:48)

So that is what we will get started with - unsupervised learning. In particular, we will start with one subcategory of unsupervised learning, which is called as representation learning. So, what is the goal of unsupervised learning? So, what is unsupervised learning? Well, we are given a set of data points. So, we are given a set of data points, our goal is to understand something useful about the set of data points.

So, this is a very-very vague, broad goal that we want to start with. So, of course, it is vague and broad because of three different things, one is, we have to define what do you mean by given a set of data points. So, what are data points? More importantly, we need to understand, well, what does understand mean? And what does usefulness mean?

So, remember, in unsupervised learning, we are just given a bunch of data points, we are not given any supervision, we saw this before. But now if you are given just a bunch of data points, what can you understand what does it mean to say we are understanding something and more importantly, something useful about them?

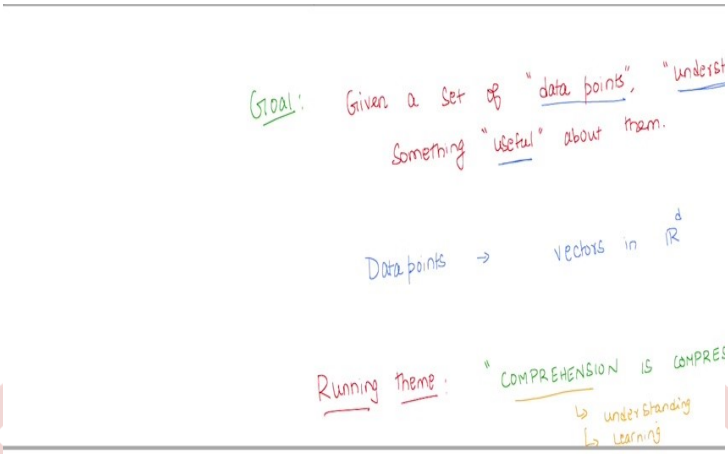
(Refer Slide Time: 2:27)



So, the easier question to answer here is the question of what are data points. Well, data points, as you may already have guessed, we are going to think of these as vectors in some D dimension. So typically, real numbers, D real numbers, these are features. For example, I might collect the height, weight, and age from 100 different people, so each person becomes a three-dimensional point, where each of the coordinates correspond to height, weight and age, in that particular order.

So, data points are clear. So, we are going to think about data points is just a bunch of vectors in the dimension. But the more important question is, what does it mean to say we understand something about this data point and what does it mean to say, we understand something useful? So, those once we fix that, then we will be able to do something useful. So, towards this, of course, there are multiple ways to approach this, we are going to start with one running theme, which I will put down and then explain what it is.

(Refer Slide Time: 3:35)



So, this running theme of what does it mean to understand will come in very handy not just for the representation learning part that we are looking at today, but in general throughout this course. So, what is the running theme that we are going to look at? Well, this is based on a very nice quote by a famous computer scientist and a philosopher George Chaitin. And he says, comprehension and it is worth writing it down, "Comprehension is compression" that is a very interesting statement and this is by George Chaitin.

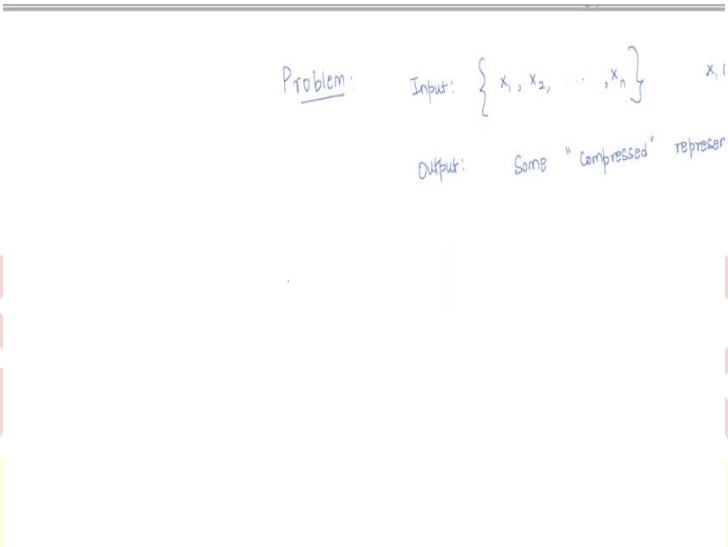
So, we want to comprehend. So, what does comprehend mean? In our context, we can think of comprehension as well understanding, so or even learning. So, you can comprehend, which means you can understand something you can learn something all these are synonymous in our context. But what does it mean to say you can comprehend something, it means that you are able to compress.

So, for instance, if you are able to compress information such that you retain only the important part of the information that can be explained to somebody else, then it means that you have necessarily understood or learned from the data. So, this is a very high-level running theme that we are going to use not just for the algorithm that we will see for unsupervised learning now, but then in some sense throughout this course.

So I may not repeat this every time, but whenever we see a new algorithm, I think there is merit in asking the question, well, what does it mean to say, we are learning from in this particular algorithm? In other words, where is the compression really happening? Ask yourself that question every time you encounter an algorithm in this course and I think that

will give you some new insights as to how these algorithms are designed and why these algorithms act in a particular way and so on.

(Refer Slide Time: 5:47)

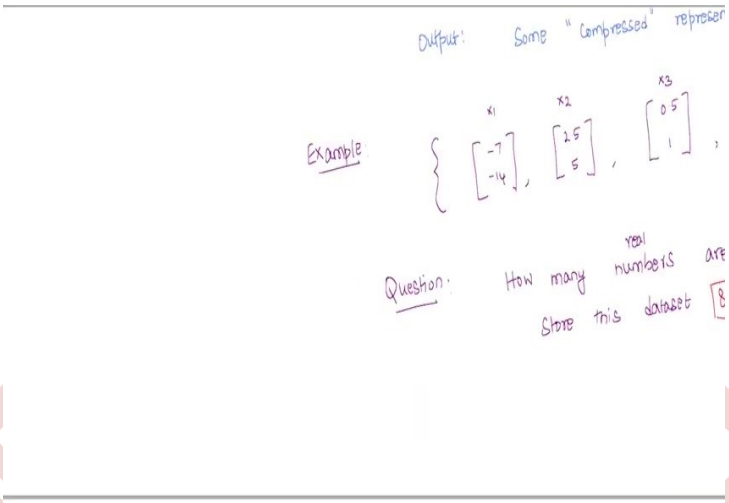


For now, we are going to start with the running theme for the problem of unsupervised learning. So, let us make that problem a little bit more precise. Let me not use yellow, let me use blue. So, here is the problem that we want to think about now. So, you have an input. Now we have decided the input is a bunch of data points and these data points can be thought of as a bunch of vectors in R^d.

Let us say we have n such vectors, each x_i is in R^d which means there are n people. From each person, let us say we collect d different numbers or these are in general data points, maybe these are n images, whatever it could be, but then we just have n different data points, which we are abstracting it out as a bunch of vectors in R^d. So, this d can be thought of as features, the number of features.

So, what is the output that we want? Well, from this data point, because we are going to think of comprehension as compression, learning as compression, we want some compressed representation of the data set. Now, what does it mean to say we can output a compressed representation of the data set, so that is the next question we have to ask ourselves. And for this, let us start with a simple example.

(Refer Slide Time: 7:08)



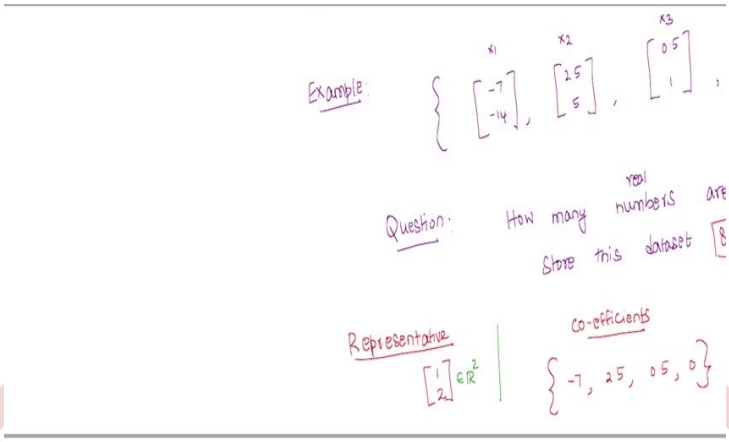
So, here is an example. Let us say I give you a data set, which looks like the following. So, maybe you have [-7 -14], this is x_1, x_2, let us say is [2.5, 5], x_3 is let us say [0.5, 1], x_4 is let us say [0, 0]. Let us say you have these four data points, I am going to ask you a question and then maybe you can pause and think about this question a bit and we will talk about what is the answer to this question in the way that we are thinking about.

So, how many numbers are needed, when you say numbers, real numbers. So, how many real numbers are needed to store this data set, let us say on a computer? So, there are 4 data points, each data point has 2 coordinates. So, how many numbers do you think are needed to store this data set? Pause and think about this, I will tell you the answer.

The naive answer to this question is that well, 4 data points to features per number, so 4 x 2, 8, so, you would need 8 numbers, you can store 8 numbers on a computer, and then well good, so that you can retrieve the data set exactly as you are seeing it here, which is good. But then is this the best that we can do, do we really have to store 8 numbers in this case or can we do better?

So, if you think about it, now look at the data set and see if there are some relationships between the features that can be exploited, that will allow us to store lesser number of values, and still be able to reconstruct this dataset exactly, pause and think about it. And now, let me tell you the answer to this, it is not too hard to see that there is a relationship between the first coordinate and the second coordinate. In particular, the first coordinate is always half the second coordinate. So in this particular data set for all the four data points.

(Refer Slide Time: 9:26)



So, which means one way you could store this data set on a computer is going to be as follows. We will store a representative point, this is one type of representing this dataset, not the only type, here is one way. So, the representative in this case, let us say is [1 2]. So, here is a single representative for the entire data set, which is [1 2], which kind of tells us that if the first feature is 1, the second feature is twice the first feature.

That is what this representative says. And now for each data point we will store what are called as coefficients, and the score sufficient for the data point 1 is going to be -7, for data point 2 is going to be 2.5, for x_3 is going to be 0.5 and x_4 is going to be 0. Now, suddenly you see that well, if you exploit the fact that the first two coordinates are related.

And if you want a representation where you have a representative, which is one vector in R^2, for the entire data set, and coefficients, which is one coefficient per representative, then suddenly we just need only 6 numbers to be stored. Now, one might ask, well is this a big savings, how much of a savings can we really achieve? So, we just went from 8 to 6, but that does not seem like a big savings.

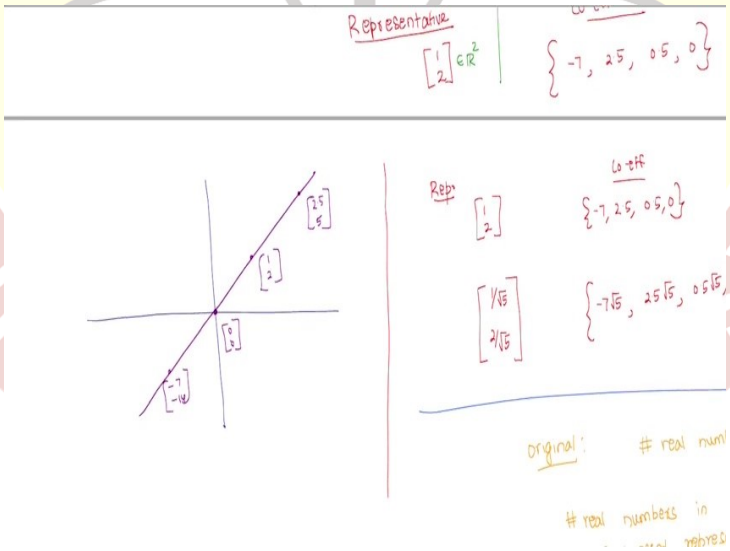
Now, if you think about it, it is going from 8 to 6 only because it is just 4 data points, but if you had a billion data points, instead of storing 2 billion numbers, now how many numbers would you have to store if you had this way of representing things? Again, you would just have one representative, which needs two numbers, one vector in R^2, and then for every data point, you just need to store one number instead of two.

So, you would have 1 billion + 2 numbers to store, + 2, because for the representative, and 1 billion, because one number per data point, as opposed to 2 billion numbers where there are two numbers per data point. So that is like a 50% almost 50% compression rate. So, which means we are able to compress this dataset by exploiting this relationship between the first and the second coordinate.

Now, we will talk more about this but then at this point, I want you to make a note of this point that using this representative, one thing that we can achieve is while using representative and the coefficients, one thing that we are able to achieve is we can reconstruct the data set exactly. Well, that might not seem like a big deal at this point, of course, that is what we can do. So, how do you reconstruct the data set?

So, if I asked you what is x_2, but then if I give you only the representatives and coefficients, how would you reconstruct x_2? You would simply take the representative and multiply it by the coefficient, in this case, 2.5 times to vector [1 2], which would give me [2.5 5], which is x_2. So, and you can do this for each of the data points, and you will get back the exact data set, so there is an exact reconstruction that is possible, using this way of looking at things.

(Refer Slide Time: 12:54)



So, now one can also think of this in a geometric fashion, I look at the same data set, but then let us say I try to plot these points, these 4 points on the plane, where would these 4 points lie, again, you should be already able to see it, if not, just pause and think about it. While I plot where these points lie. So, these points are going to lie along the line along the line where

these points would be [0 0], [-7 -14], [2.5 5], and I think maybe this is [1 2], and you have [2.5 5].

Now, all of these lies along this line, and that is not too hard to see, because the relationship is that the y coordinate is two times the x coordinate. So, y equals $2x$, is this line. So, all the points in this line. Now, even if you had other points in this line, well, you could still you could have used a representative and then reconstructed it the way we did. Now, one point to note here, again, a simple point, but then it will come out very useful as we go along.

Instead, well, remember what we did was the representative that we choose was [1 2], and the coefficients were -7, 2.5, 0.5 and 0. Now, there is nothing sacrosanct about this representative [1 2], I could have chosen a different representative and achieved the same exact reconstruction. For instance, I could have chosen any point along this line as a representative, of course, except the [0 0], point.

Any other point along this line could have been chosen as a representative. For instance, I could have chosen $[1/\sqrt{5} \quad 2/\sqrt{5}]$. For whatever reason I want to let us say choose this as my representative, it is still a valid represented because there are coefficients that I can use, which are $-7\sqrt{5}$, $2.5\sqrt{5}$, $0.5\sqrt{5}$, 0, and I still will be able to access exactly the constructor dataset.

So, the exact choice of representative is immaterial as long as the representative lies along this line. So let me make a note of that point and then again, it might seem a simple silly point at this point, but we will see the use of this as we go along. So, any vector along the line, purple line that I have drawn here, can be chosen as a representative, of course, except [0 0], which we will not allow, because if it is [0 0], then we will not be able to reconstruct anything.

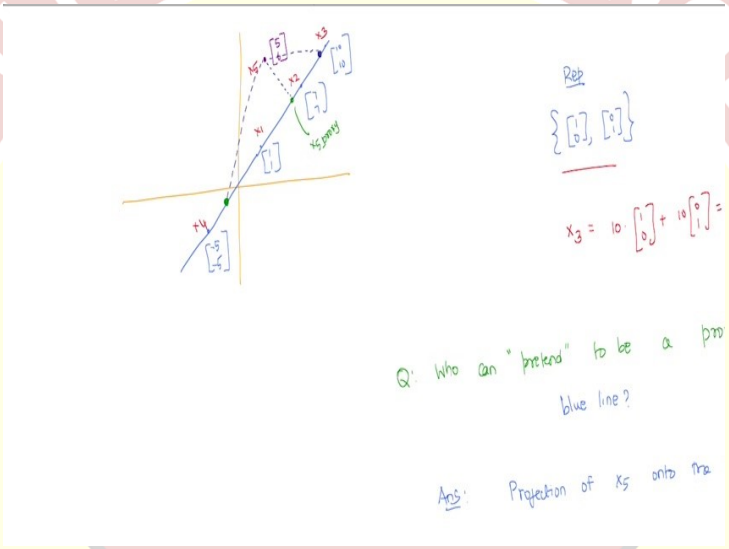
So, originally, if we did not have this representative coefficient view of things, then the number of real numbers that you would have needed to store let us say, n data points in two dimension would have been 2 x n. So, in d dimension, it would have been d xn. Now, the number of real numbers in the compressed representation is much smaller. And how many are there? Well, you should have been, you should be able to guess it now.

So, there are 2 for a single representative plus 1 for each data point. So, that is 2 + n. In general, this will be d x n. And then this would be $d + n$. So, this d is for the coefficient, if

you were thinking about in d dimension, if every other coordinate is some multiple of the first coordinate, let us say, then you just need d numbers for the representative and one number for each of the data points, so it would be interesting.

So, now that that is a huge compression that you might potentially achieve. But now, if you have been thinking about this, an obvious question that comes that should come to your mind is, well, it is fine that we are able to compress this if the data points all lied along this line. But now let us say that was not the case.

(Refer Slide Time: 17:37)



Let us say we had a situation like this. So, which is not a very nice situation for our data set here. Let us say we had this case, let us make it even easier, we have the line $y=x$ where we have a bunch of data points [1 1], I do not know, [7 7], [10 10], maybe, now this can still be compressed, 50% compression is still possible for these four data points from 8, you can go to 2 + 4, 6, all that is good.

But let us say our data set was not like this, maybe our data set had this extra point, which is [5 6], and now, the question is, can we do the compression in the way that we have done already? Specifically, the question that I am asking is, well, now if you can you represent all these data points as a single representative, and one coefficient per data point?

Well, of course you cannot, so, because all these data points do not fall on the same line, there is one data point, which is kind of, in some sense, making things harder for us. But then

if you want to exactly reconstruct this data set, well, then you have to give up the notion that can be a single representative one coefficient per data point.

Now, what else can we do? Well, one thing you can do is, instead of a single representative, you can use two representatives, and then you can think of each data point as a linear combination of these two representatives. For example, I can think of the representatives now as not just a single representative, but a set of representatives. Let us say, {[1 0], [0 1]}, are my representatives, imagine. And my coefficients would be what?

Would be to reconstruct the point [-5 -5], the coefficients would be -5, -5. To reconstruct [1 1], the coefficients will be 1, 1. To reconstruct [7 7], it would be 7, 7, [5 6] would be 5, 6, and [10 10] would be 10, 10. So, for example, how would I reconstruct the x? Let us say this is x_1, x_2, x_3, x_4, x_5. If I had to reconstruct x_3, how would I do that? Well, I look at the coefficients that are two coefficients.

One corresponding to each of the representatives. So, this would be 10 [1 0]+10 [0 1], that gives me [10 10]. Now we can check, you can do this for every data point. But now, the question, is this a great idea? Perhaps not, because what we have done is that we have increased the number of data points that we need number of coefficients that we need to store per data point from 1 to 2.

So, which means if I had to store these numbers, I would anyway store two number per data point, in addition I will also store four numbers for the representative. So initially, if it was n numbers, we would have stored 2n numbers, but now we are going to store $2n+4$, that is not really compression at all. So, we are not really compressing we are in fact, increasing the number of data points.

Now, that means that this is also not a good idea. So, increasing the number of representatives for this data set does not seem like a great idea, either. But then if you do not increase the number of representative for this dataset, then we know that you cannot exactly reconstruct this dataset. So, you have to give up one of these, what are these? One is exact reconstruction, the other is increasing the number of representatives.

So well, if you need exact reconstruction, it looks like you need to increase the number of representatives. So, the only way you can possibly do compression here, then it looks like is that you need to give up the notion that you need exact reconstruction. So, let us say we do

that. So now, I do not really need exact reconstruction, which means that I know all these four points are all on this nice line, and this x_5 is not on this line.

If I do not have to reconstruct this exactly, which means what does that equivalently mean? It means that I need to somehow find a proxy for this $x_5$along this line along this blue line. So, if I find a proxy for this x_5, let us say at some point here is a proxy for this line, maybe this is x_5 proxy, then I can imagine as if my data set only had x_5 proxy, along with x_1, x_2,x_3,x_4 and I can forget about x_5.

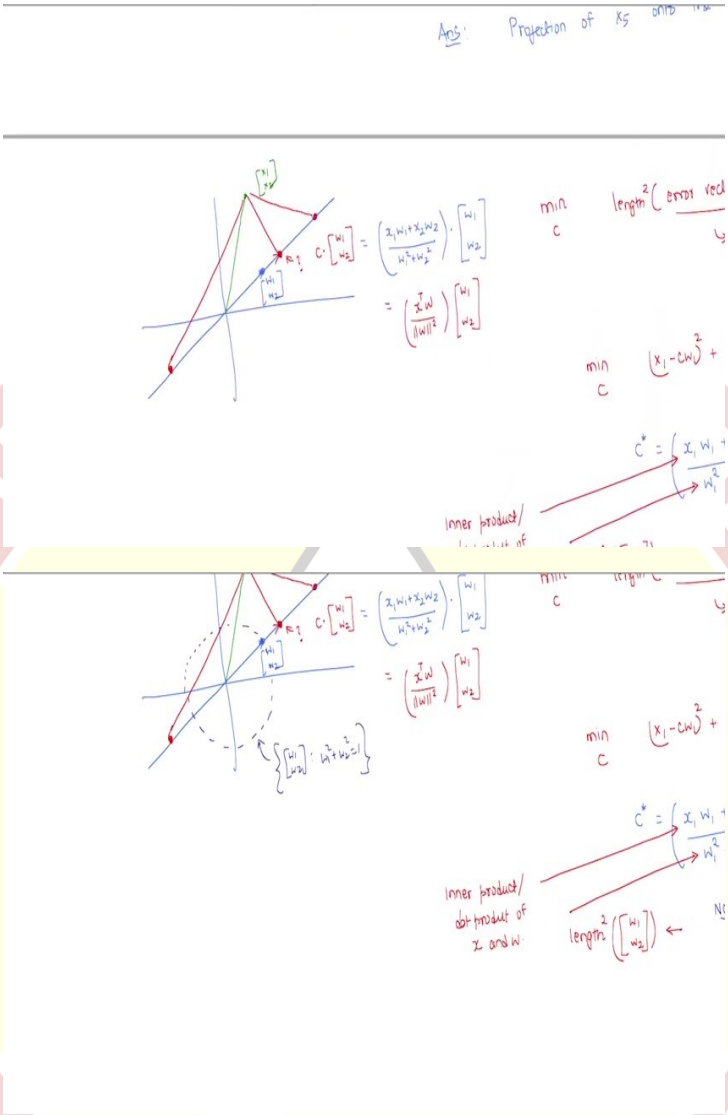
Now, because all these points lie along the line, I can use a single representative and under coefficients per data point, and compression is still possible. Now, that means we need to find a proxy for this data point along this blue line. So, how can we find this proxy? Well, if you have looked at linear algebra, before you already, perhaps know the answer to this.

Well, what could be a good proxy, a proxy the best proxy for x_5 along the blue line would be that point which for which we lose the least. So, in other words, so if I choose for instance this point as the proxy, then what do I lose? Well, I lose this bit. So, whereas if I choose this point as the proxy, I lose this bit. Now if I choose the green point as a proxy, I lose this bit.

Now, the proxy should be chosen such that I lose the least. So, which means that the length of the vector, which should be added to this proxy to get the original point should be as small as possible. Now, how do we choose that? Well, again, if you have seen linear algebra, this is just projection of x_5 on to this blue line. So, now the question that we are asking is, who can pretend to be a proxy for x_5 along the blue line?

Of course, the answer to this is projection of x_5 on to the blue line. Well, projection is just finding that point which is closest to x_5 along the blue line. So, how do we find that point? Well, of course, for people who are conversant, linear algebra might already know the answer to this. Nevertheless, just to keep it self-contained, I will go with this. So basically, this is what we want to do.

(Refer Slide Time: 24:45)



So, you have a line. Let us say we have the vector w_1, w_2, which we think of as the representative for this line, which means this line is just the set of all vectors which are scalar multiples ofw_1,w_2, $w_1$and w_2 is both not equal to 0. And now you have a point, which is, let us say x_1, x_2. Now the question we are asking is, well, what is the projection of x_1,x_2, along the line given by w_1, w_2, what is this?

Well, because of the fact that this red point is along the blue line, it has to be some constant times w_1, w_2. Now, which is that constant as you vary this constant, this point is going to move along this blue line. And for some value of this constant, we will find that this error piece is as small as possible, the length of the error is as small as possible, which means we

can set this as an optimization problem where we want to minimise with respect to c the length, or the length square of the error vector.

But what is the error vector? If the point that you want to project on this x_1, x_2, on the line that you want to project along is w_1, w_2. And if the point is c times w_1, w_2, the error vector itself is just what should be added to c, w_1, w_2, to get x_1, x_2, well, what should you add, you should add ( ) $(x - cw_1),(x - cw_2)$ And the length square of this is just $(x - cw_1)^2$ $+(x - cw_2)^2$.

Now, you want to find the c such that this value is as small as possible. So, as I move c, if I choose a different c, I might get this point where then where I am kind of measuring this line, if I choose a different c, negative c, I might get this point where the length that I am considering caring about is this length. So now, I want to choose a c such that the length is as small as possible, which means I need to minimise this function.

Now, here is an exercise, take the take the derivative of this function with respect to c, equate it to 0 and see what we get, I am not going to do that derivation, it is a very simple derivation, I will leave that as an exercise. But then if you do that, you will observe that c^* will have the following form $\frac{(xw_1+x_2w_2)}{(w_1^2+w_2^2)}$. So, remember this is a scalar.

So, it just says that what should I scale my vector w_1, w_2 by to get to a point along the line, blue line, which is closest to the point x_1, x_2. And that scalar, of course, has to depend both on w_1, w_2, and it also has to depend on the original point x_1, x_2, and a sanity check, you will see that it depends on both. So, basically what is this c?

This is c times w_1, w_2 is actually from our derivation $\frac{(xw_1+x_2w_2)}{(w_1^2+w_2^2)}$, this is a vector this is a scalar. So, you are multiplying that scalar for both the both the numerator and the denominator. So, it is worthwhile to notice the numerator and the denominator separately. So, what are these, so, how do we interpret this numerator and denominator.

Now, the numerator or let us first look at the denominator, the denominator is $(w_1^2 + w_2^2)$, but what is that we know that that is just length square of the vector w_1, w_2. So, this length is just the length of a vector is just the norm. So, w_1, w_2, is by Pythagoras theorem is $\sqrt{w_1^2 + w_2^2}$ and the length square will be $(w_1^2 + w_2^2)$.

And we also should be able to recognize what the numerator is? The numerator is just the inner product or the dot product or dot product of x and w. So, this is what we have done the derivation is for two dimensions, but then the same would go through for higher dimension as well. So, for higher dimension we can say in general that c^* is just put it here only.

So, c^* can be written as $\frac{x^T w}{\|w\|^2}[w_1 w_2]$

, $x^T w$ of course, is just a notation for $x_1 w_1 + x_2 w_2$ and $\|w\|$ is just the definition for length. This is length square of w_2, of course, this is $l2$ norm but then I am not really going to explicitly say that here. Okay, good. So, now, the c^* itself looks a bit messy, it has a numerator and a denominator.

We can perhaps simplify this a little bit more by making use of the observation that we did earlier that if you want to do this compression business, then you can pick any vector along the blue line to represent the blue line, we picked some w_1 and w_2. Now, we could we could have picked any vector along this w_1 along this blue line and our c will adjust accordingly. So, because the c depends on w_1, w_2 andx_1, x_2, of course, now, if you pick a different vector, the c^* that you will get will depend on w accordingly.

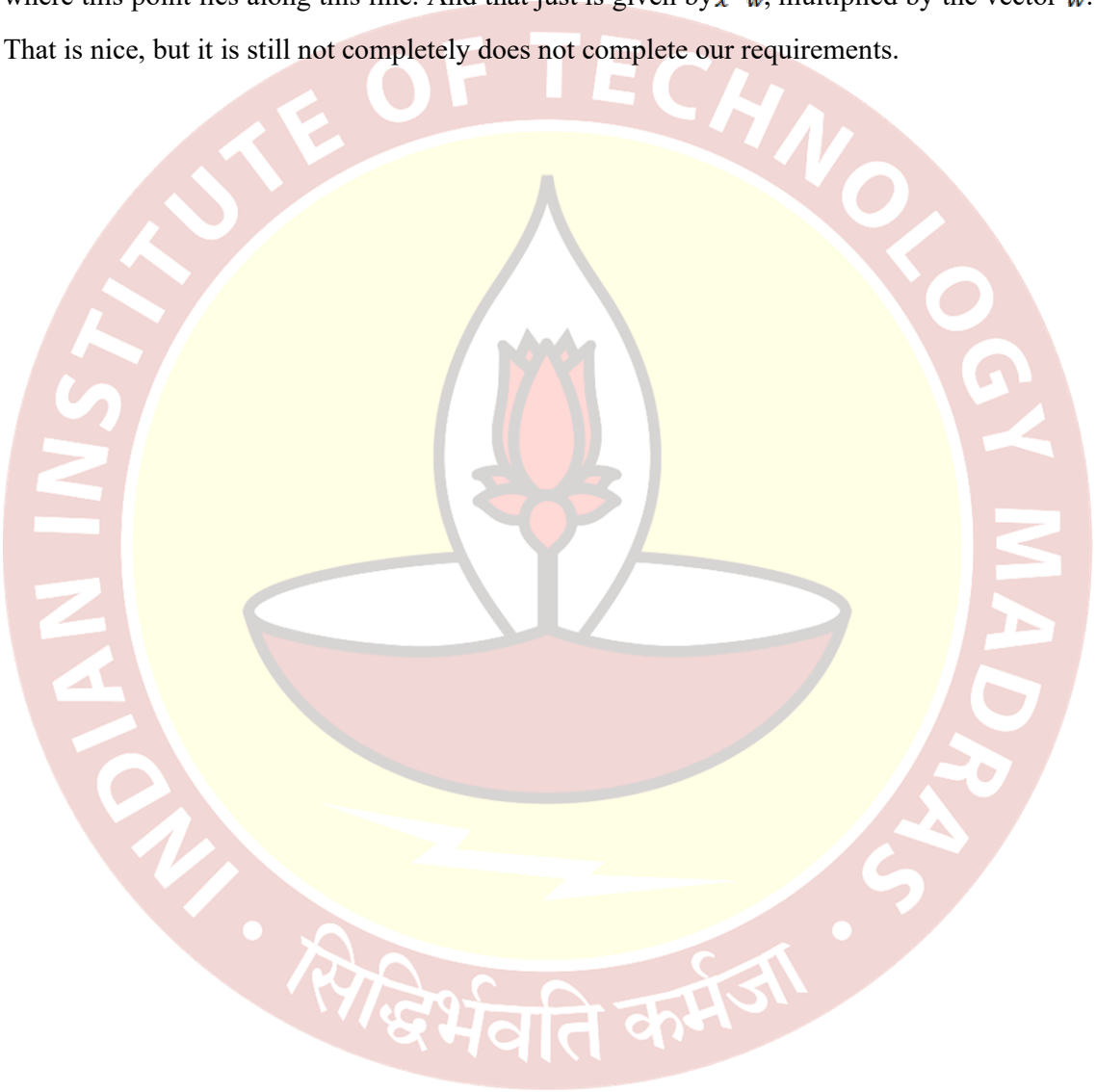
Now, because of this, what we are going to do and what is what might be easier is that we can always pick let me make this as a note can always pick w_1, w_2, such that length of $[w_1\ w_2]$ equals 1. So, we could have take this w_1, $w_2$such that this lies on the unit circle. Well, what is what does it mean to say it lies on the unit circle? This is just a set ofw_1, w_2, such that$w_1^2 + w_2^2$=1 the length is 1.

Now, what is the advantage of doing that? Well, the advantage of doing that, is that, so, which implies the c^* is just $x^T w[w_1 w_2]$. So, the denominator this 1. So, because we choose a representative to have length 1, now, c^* would not have would the denominator of c^* is 1. So, we do not have to keep track of the length of w there, because we know our understanding is that we are going to represent lines as using representatives which have length 1.

So, then c^* is just$(x^T w)w$, well, w itself is a vector, which is$[w_1\ w_2]$. Okay, good. So that is good. So basically, what we have done so far is the following. So, we said initially, that if

you had a bunch of data points, such that all of them lined up along the line, then you can choose a representative and a coefficient for each of these data point.

But now we then said that, hey, only four points lie along the line, the first point does not lie along the line, then we looked for a proxy on this line and we said that well, you can just project this point onto this line, and then you are done. Which means that we need to know where this point lies along this line. And that just is given by $x^T w$, multiplied by the vector w. That is nice, but it is still not completely does not complete our requirements.

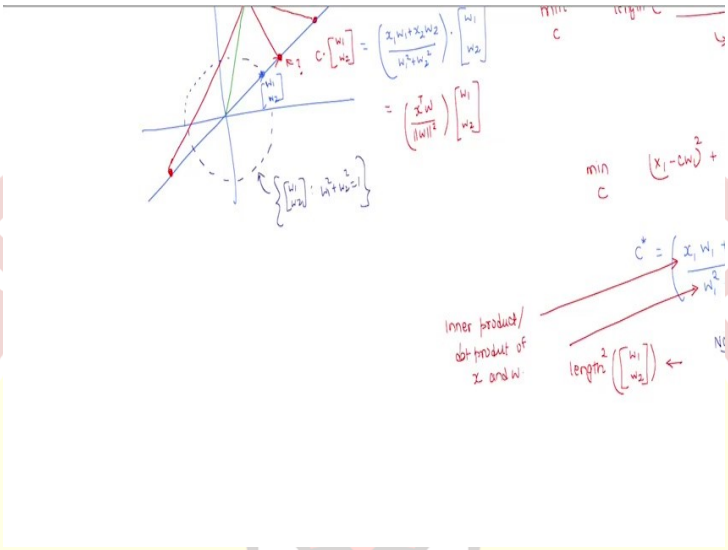


IIT Madras

ONLINE DEGREE

Machine Learning Techniques
Professor Arun Rajkumar
Department of Computer Science & Engineering
Indian Institute of Technology, Madras
Representation Learning: Part 2

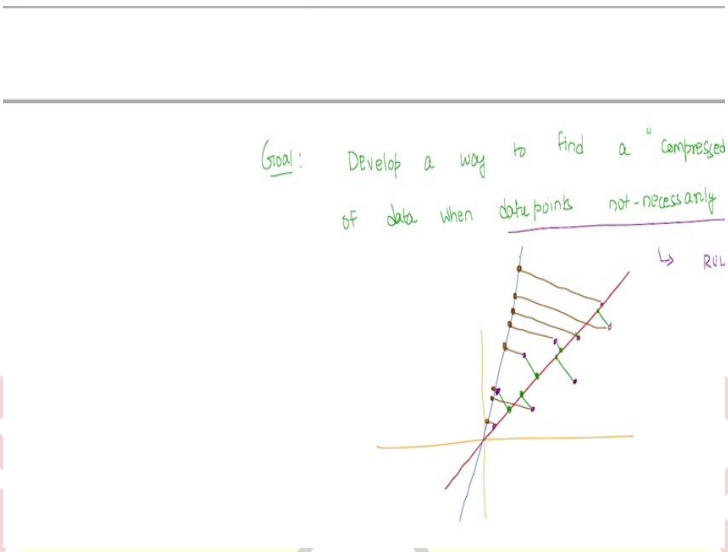
(Refer Slide Time: 0:13)



Here we are seeing if we know that, well, four points are on a line. And the fifth point is not on the line, then how can we find the proxy for the fifth point along the line that has the four points. But in practice, nobody is going to come and tell us that, hey, these points are on the line, these points are not on the line. So, now you can find the proxies for the points not on the line along the line, nobody tells us that.

We are just given a dataset, which has a bunch of points. So, which means that we need to find that line on to which you need to project these data points as well. So, that is also something that we need to find. So, now there are multiple lines. So, we need to somehow find that line, right. So, that is what we are going to do next, we are going to use the fact that given a line, we know how to find the proxy for a data point on this line, to actually find that line itself. Let us see how we can do that.

(Refer Slide Time: 1:15)



So, now just to make this goal a little bit more precise. So, this is our goal now. So, we want to develop a way to find of course, a compressed representation of data, when data points do not necessarily fall along the line. So, we are not going to assume that all the data points are along the line, just that one or two points are not on the line.

So, this is not an exception. So, this is almost a rule. So, it is never going to be the case that we are going to find almost all points on a line, but then just one or two points not on the line, is more like a rule not really an exception. What do I mean by that? Let us say if I if there was a class of 100 people, and then I measured the height and weight and try to plot them, we might get some values like this. So, of course, as the height increases, the weight increases.

Now, we may want to represent this dataset using this line, but as you can see, I mean, only one or two points actually perhaps fall on this line, maybe none of the points fall on this line, yet we want to represent this data points using this line. So, which means that the fact that there is an exact linear relationship between the features is a myth. So, that is never going to happen. That is, very, very rare.

So, physical loss might satisfy something like that. But then your real world data will perhaps never satisfy that. So, which means that we are always going to find proxies for data points along the line. So, you will have to find these proxies. Now, if I know that the red line is the best line, then I know how to find proxy for each of these data points. But nobody gives us the red line. So maybe it is not the red line, maybe it is the blue line.

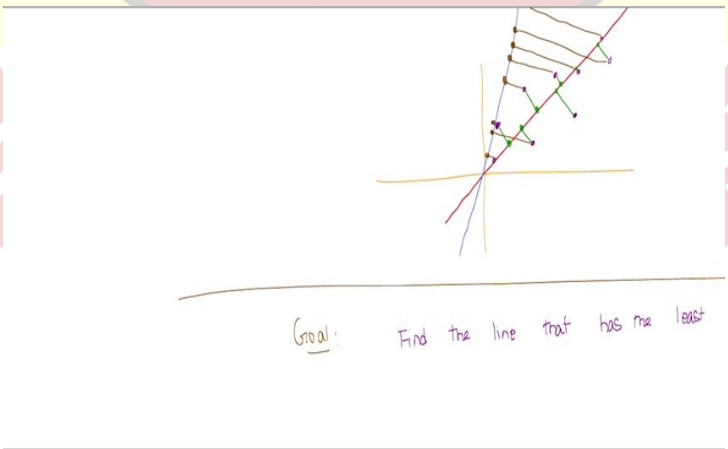
So, now how do we know? So, because for given a line, I can always find a proxy along that line for any data point so, I could have found proxies for all these purple data points along the blue line as well. So, by finding, by projecting them along the blue line, now the brown points would act as the proxies, and I can still compress them.

Now both are going to give me the same amount of compression. There is no doubt about that, because once I found proxies along the line, I can find the representative for this line and one coefficient for each of the data points both are good ideas. But one line is still better than the other, not in terms of the amount of compression you achieved.

But in terms of what, think about that. Let me tell you what it is. So, it is obvious that why the red line is better than the blue line, though both give us 50 percent compression, the red line is a better line because if you had to reconstruct this data set, then the amount of error that you suffer with respect to the original data set with respect to the red line is much lesser than the blue line.

So, the reconstruction error is lesser for the red line, not the compression ratio, the reconstruction error is less for the red line than the blue line. So, which means we have to find that line to represent our data that gives us the least reconstruction error. Now how do we find that is the question.

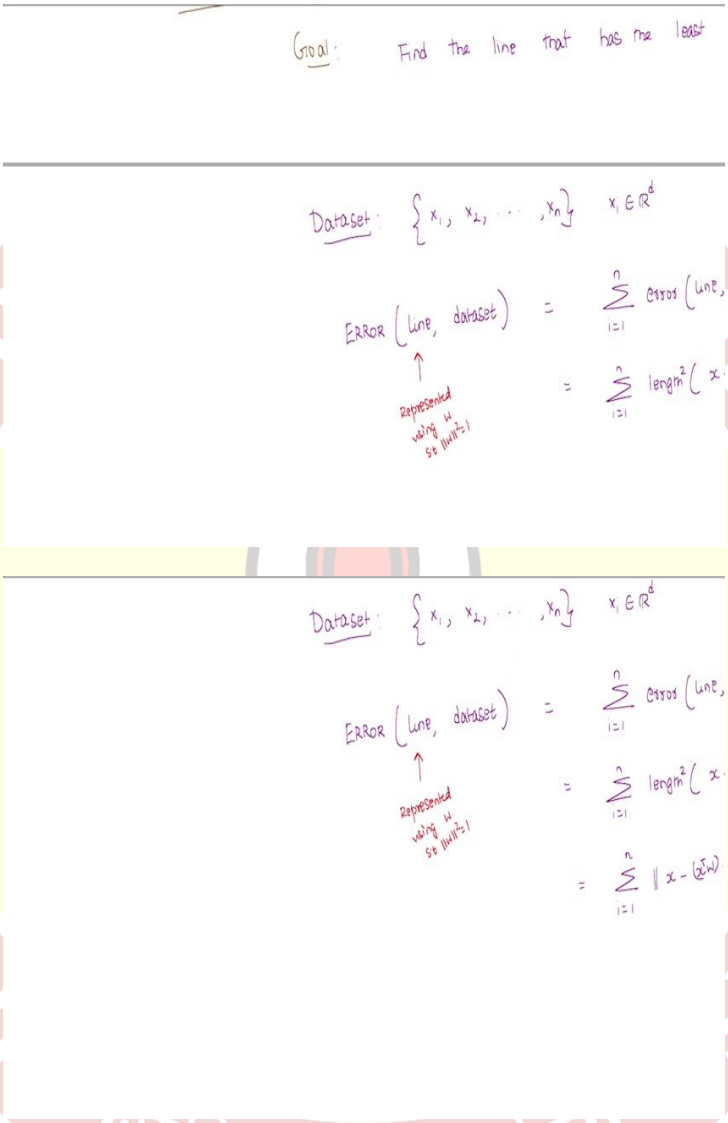
(Refer Slide Time: 5:09)

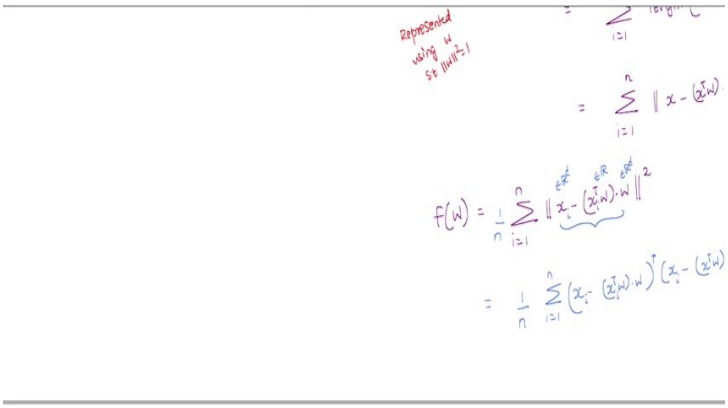


So, now we have a very, very precise goal to tackle. So, now our modified goal, or rather not a modified goal, our specific goal now is find the line, given a data set, of course, I am not

writing that again, find the line that has the least reconstruction error. This is what we want to find.

(Refer Slide Time: 5:40)





So, Let us do this. Now, Let us say you have a data set. As usual, we have unsupervised data set, which is $\{x_1, x_2 \ldots . x_n\}$. Where $x_i \in R^d$. Now, how do I define the error for a given line with respect to the data set? Well, basically, what you need to do, you need to kind of sum up the errors for each of the data points. So, your data set has a bunch of N data points.

So, you sum up the error incurred by each of these data points along the line. So, this is for a given line, if I give you a line, you can measure the performance of the line as the error that the dataset incurs on this line. So, by counting, by summing up the error for each of the data points, but what is the error of a particular data point on a line?

Well, that is just the length squared of this line, which we are going to think of as being represented by some w, which is just we know $x - (x^T w)w$. Now, remember, when I say line, I am going to represent using, this line is going to be represented using w such that the length of w is 1, So, $w^T w$ or $\|w\|^2$ is 1.

That is how we are going to think of lines, so among all possible lines represented by vectors in the unit circle, I want to find the best line so, given a line, which means I give you a w. And I asked what is the error, and that would be the sum of the errors for all the data points along the line. Now, what is the length squared?

Well, we know the length is just $\|w\|^2$. For any vector, let us say we take a vector, the $\|w\|^2 = z^T z$, so dot product of a vector with itself is just its length squared. Now if you do that, so then this is going to be just $\sum_{i=1}^{n} \|x - (x^T w)w\|^2$

. This is what we want to minimize.

And if you want to minimize this, we can well, this is the error of a given line given w with respect to the data set. And our goal is to find that w that minimizes this, so we can think of this as some function of w, which is just the sum of the errors,$x - (x^T w)w$. This, so now you want to minimize this function.

Now, a couple of things that we will do here, so we will think of this as the average error instead of the sum of their that does not really change the w. So, if some w minimizes the sum of the errors, it is the same w that will also minimize the average error. So, I can think of this as function as that w that minimizes the average error as well.

We will see why we can think, I mean, the whole derivation would go through even if you think of the sum. But in terms of interpretability, we will see why the average is a better thing to look at later, so now this we know, the length squared is just of a vector. Remember, this is a vector x is in R^d, w is in R^d. This $x^T w$is in R. So, this is a scalar times w, this is the proxy for x along the line w.

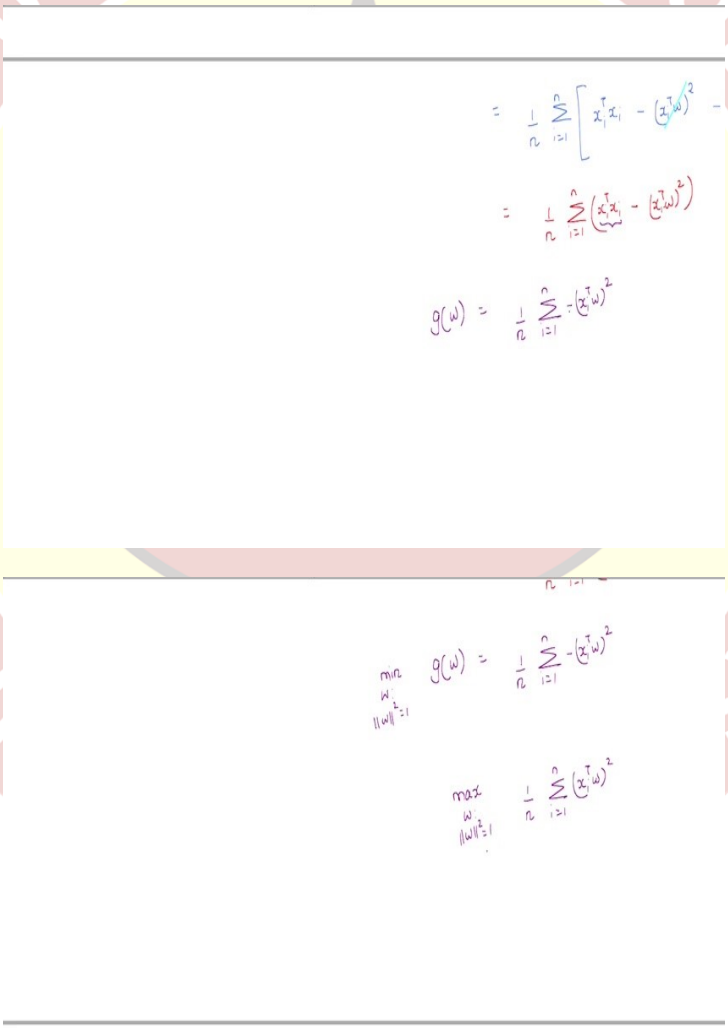
And then you are looking at the difference vectors. Difference is a vector and we are looking at its length. And because is a vector, the length squared is just the transpose of the vector with itself. So, I can write this as $[x_i - (x_i{}^T w)w]^T[x_i - (x_i{}^T w)w]$. That is still. Well, I should write x_i everywhere I should use i. Sorry about that. Because this is the ith data point, x_i.

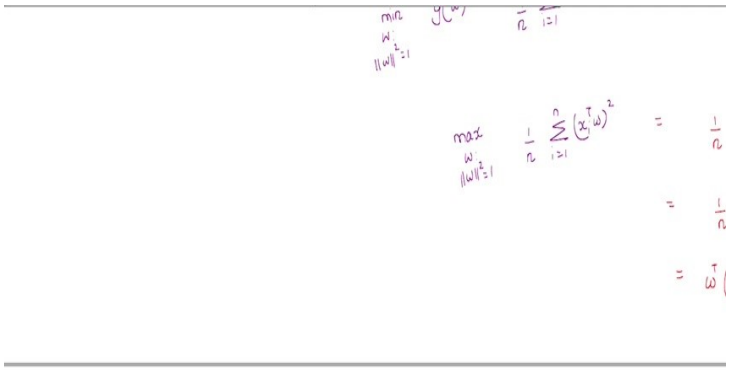
(Refer Slide Time: 9:55)

So, now I can expand this. I am sure. I mean, anybody who has done an MLF for linear algebra course will know how to do this expansion, I will do this nevertheless. So, this is $x_i^T x_i$ minus $x_i^T w$ times $x_i^T w$, that is a square minus, again, $x_i^T w$ into $w^T x_i$, which is same as.

$(x_i^T w)^2$. And the last term is plus x_i, the scalars multiply themselves that is $(x_i^T w)^2$. Now the vectors will dot themselves $w^T w$. Well, that is $\|w\|^2$, but we know that the w's that we care about have length 1, so that is just multiplied by one. So, if you do, again, some simplification, these will cancel out. And we are just going to have $\frac{1}{n}\sum_{i=1}^{n} x_i^T x_i - (x_i^T w)^2$.

(Refer Slide Time: 11:03)





Now if you think about this, we are after that w that minimizes this value. So, but if you look at the first term that does not have a w, so, that is a constant that you are adding to each of these terms. So, it will not affect your w, even if you remove that constant. So, equivalently, I can minimize a different function without that constant, which would simply be $\frac{1}{n}\sum_{i=1}^{n} -(x_i{}^T w)^2$.

So, the constant can go away and then the, whichever w minimizes *g(w)*is also the 1 that minimizes *f(w)*. If you are not convinced about this, pause and think about why this constant does not matter in finding w. So, now this is*g(w)*, which has a negative sign. Of course, we want to minimize over all w such that $\|w\|^2$ or $w^T w$ is 1.

So, that is, the goal. But instead of holding on to this negative sign, we can write this as instead, max over W such that, $\|w\|^2$ is 1, without the negative sign, I can write this as

$$\frac{1}{n}\sum_{i=1}^{n}(x_i{}^T w)^2$$

So, we will see why we are doing this simplification.

So, now I can write this objective as$\frac{1}{n}\sum_{i=1}^{n}(x_i{}^T w)(w^T x_i)$. It is the same thing, So, $w^T x_i$ is same as$x_i{}^T w$, I am squaring this. So, I can write this as product of two terms. And why am I doing that?

Well, that is because I can now write this as now remember, w is a vector in R^d, So, which means w transpose can be thought of as like a 1 x d vector. x_i is a d x1 vector x_i^T is a 1xd vector, w is a d x1 vector. So, vectors are all column vectors. By default, the transposes will be row vectors.

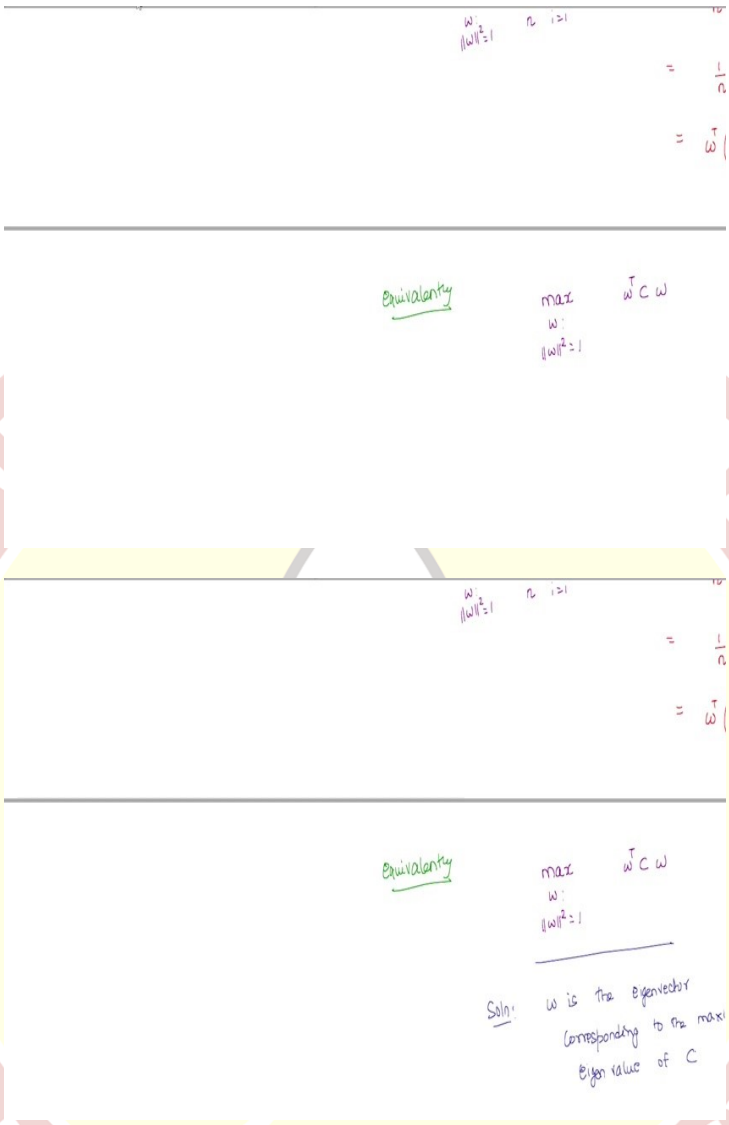
Now I can write this whole thing as$w^T(x_i^T x_i)w^T$, I have just changed the brackets. So, that is still. Because everything is linear, I can do this as well. This is just matrix multiplication. So, it does not matter. I mean, the same trimetric. I mean, I can do whichever way I want. I am not like commuting it.

So, remember that I am just changing the brackets, the order in which the multiplication happens is still the same. Now, note that w does not depend on i. So, I can bring w outside and say that this whole thing is$w^T\left(\frac{1}{n}\sum_{i=1}^{n}(x_i^T x_i)\right)w^T$

. Now, let me note this, that this, what object is this?

So, if you have not, if you are not already seeing it, pause and think about what kind of an object is it? Is it a vector? Is it a scalar? Is the matrix what is it? I will tell you what it is. So, this is a dxd matrix. It is an average of a bunch of dx$d$matrices. 1 dxd matrix per data point. You take all the data points, find their average matrix. So, that is what this is. So, Let us give this matrix and name this matrix. Let us call this C.

(Refer Slide Time: 14:50)



So, essentially, then what we are saying is that equivalently. The problem that we can solve, to achieve our objective is the following, So, we can maximize over $w$such that $\|w\|^2$ is 1, $w^T C w$, where C is$\frac{1}{n}\sum_{i=1}^{n}(x_i{}^T x_i)^2$. What we are saying is that finding that line that passes through the origin which best represents the data, in terms minimum reconstruction error is same as finding that w that maximizes this quantity with respect to a matrix that you can generate from your data.

This is a dxd matrix. And some of you might already recognize what this matrix is. We will talk about this matrix a little bit detail later, but this matrix is nothing but the covariance matrix. So, this matrix is called the covariance matrix of the data set. And basically what we

are trying to find is that we want to find a direction w which maximizes this bilinear form which is $w^T Cw$for the covariance matrix.

Again, for those who have done linear algebra course, already, you would immediately recognize that the solution to this problem is w is the Eigen vector corresponding to the maximum Eigen value of C, the covariance matrix. Basically, your line which best represents the data in terms of error minimization of reconstruction is same as the line which maximizes the $w^T Cw$as C is the covariance matrix and this line is given by the Eigen vector corresponding to the maximum

Eigen value of the covariance matrix. We will talk about this covariance matrix in a little bit more detail a little later. For now, imagine that, well, this is not a hard problem to solve. So, that is the main takeaway at this point. We will understand this covariance matrix and what does it mean to say, where did the covariance come into picture while we are doing error minimization and all that a little later, but for now, assume that well, this problem has a well-known, well understood solution in terms of the Eigen vectors of a certain matrices associated with the dataset.

IIT Madras

ONLINE DEGREE

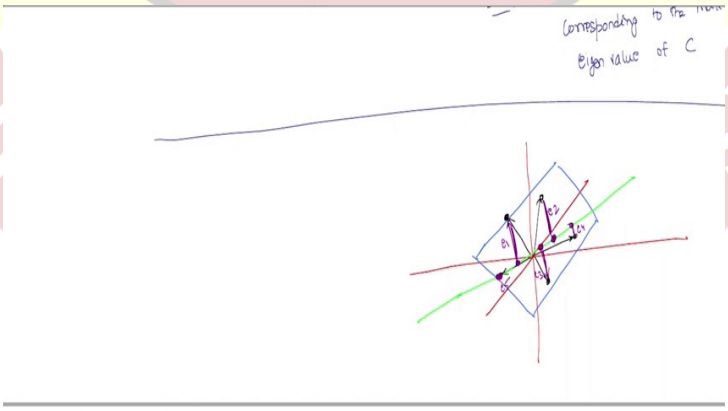
Machine Learning Techniques
Professor Arun Rajkumar
Department of Computer Science and Engineering
Indian Institute of Technology, Madras
Representation Learning: Part 3

Even if you assume that we can solve this problem, the more pertinent question that we should ask ourselves is that are we done? In other words, let us take a step back and think about where we started. So we were given a data set and we asked for that, on which we can find proxies for the data points such that once you find the proxies you can compress the data using just a representative and coefficients along this line.

Now how do we do that? Well, we are saying we can set up an optimization problem and solve it and it turns out also that solution to the optimization problem also turns out to be the eigenvector of a nice matrix called the covariance matrix associated with the data set; that is incidental at this point.

So, but the more important question is let us say we think of this solution to this optimization problem as a black box, we give the data set and then the line comes out, we know what the black box did now, but let us say for the moment we think of it as a black box. Now are we satisfied is the question we should ask, so are we satisfied with this compression that we have in all cases? Well not necessarily; to give an example let us consider the following picture.

(Refer Slide Time: 1:38)



So from 2D let us say we go to 3D, now so now you you are in three dimensions and let us say your data points are bunch of vectors in 3D, so let us say there are a bunch of vectors

here, here, here and so on here. Now imagine that all these vectors were along a plane, so it is a three dimensional space but then not all data points are scattered everywhere around, they all fall along a plane.

Now that is the critical structure that we have in this dataset, the question is we able to, in some sense extract that structure, is our compression trying to extract the structure or not? Well, what is our compression method trying to do? It is trying to find the best line on which you can project the data point such that you lose the least. For this particular data set, well, if we run our algorithm we are going to get a line.

So that line might be something like this; maybe this is the line. So this line will be necessarily on the plane, I mean, you can think about why it has to be on the plane, because if it is not on the plane, if it is outside the plane then it will cost more to project on something outside the plane. So you can avoid that cost. So necessarily the best line is going to be on the plane because all the data points are on the plane.

But where these proxies are going to be, well, these proxies are going to be along this line. So I am going to think of all the data points which were originally on the plane using their proxies along this line. Now the question is, are we satisfied with this? So is this a good compression? Well, it is a compression, if you want to compress this definitely this is a good way to do it, but then does it capture the necessary structure in the data?

In other words what I am saying is the following, our hypothesis is that there is a line that best represents this data, everything else that is the part that we think of as error, well, the part that we leave out while reconstruction is something that we are going to think of as error, but now if the data points lied along a plane, then this part which we are imagining as error may not necessarily be error, that might also contain some information.

So because necessary information, the structure is in a plane not on a line, so the bits that we lose necessarily in this particular case will also contain some information and we might want to extract that information out also, but then we have a nice black box algorithm, which will give you a good line. So we do not want to lose the power of our algorithm that we already have, at the same time we also do not want to just be satisfied with a single line.

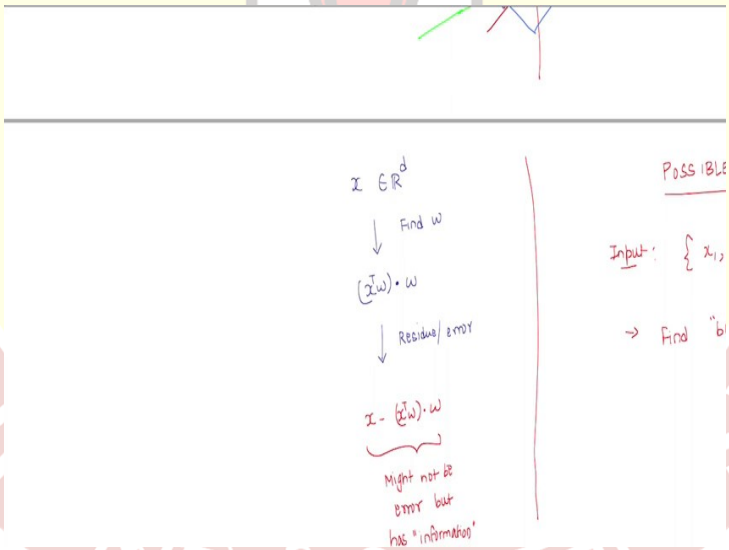
We want to somehow capture the fact that the data might be lying on a two dimensional plane on three dimensional data. So what can we do now? You can do one thing. So if you observe

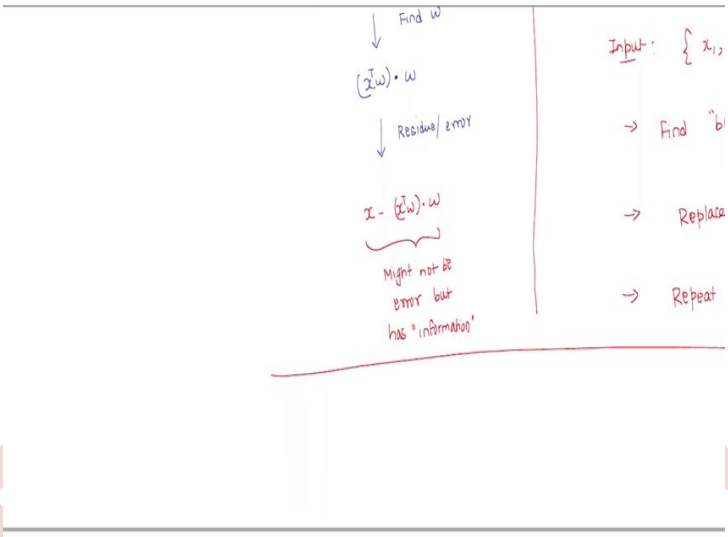
this, so now what we are seeing is that these are the proxies the purple points are the proxies and the purple vectors are the error vectors.

Now if you observe this what we are saying is that the error vectors are not necessarily error, that might also have information, which means that, for example, if I just plotted this error vectors on three dimensional space again, how would these error vectors look like? Now you can see already in this picture that all these error vectors will necessarily align in some direction. So this would be the first error vector corresponding to let us say e_1, e_2, e_3, e_4, e_5.

e_5 is just the origin in this case, because x_5 point was already on the line. This could be e_1, this could be e_3, this could be e_4, maybe this ise_2, and so on. Now as you can see, all these so called error vectors also lie along a line, which means that there is some notion of information in the error vector also and we would want to extract that as well. Now how can we do that?

(Refer Slide Time 06:33)





One procedure that you could follow is the following. Now what we have done so far is we have a data point x, which is a d dimensional data point and what we are seeing is that we will find some w for the data set and once the w is found we are going to represent this data point as its proxy, which is $(x^T w)w$.

This is the representative w and $x^T w$is the coefficient corresponding to this data point. But now the error or the residue or what we might, we have been calling as error that is just the remaining piece. So that is just$x - (x^T w)w$. Now what we are saying is that this might not be error, but might have information, so but has information.

Well, for one if all of this were error they would not necessarily line up along the same direction. So if it is error, the error for each data point has to be kind of in all over the place, so that is the high level idea. So now how can we work with this well we could do the following, so here is a possible algorithm and then we will kind of build this algorithm as we go along. A possible algorithm could be the following.

You start with your original data set which is your input$\{x_1, x_2 \ldots . x_n\}$, where each x_i is in R^d. The first step is whatever we have been doing so far find the best line w by minimizing the error or maximizing the w^T, does not really matter how we do this. For now let us call this w_1 instead of w, we will see why I am calling it w_1. Now, what am I going to do? I am going to replace x_i, the data point x_i with its residue.

So the residue is this vector$x_i - (x_i{}^T w)w$, so now I have a new data set, so now all my the data set, the second level data set is the data set that only contains the residue vectors with

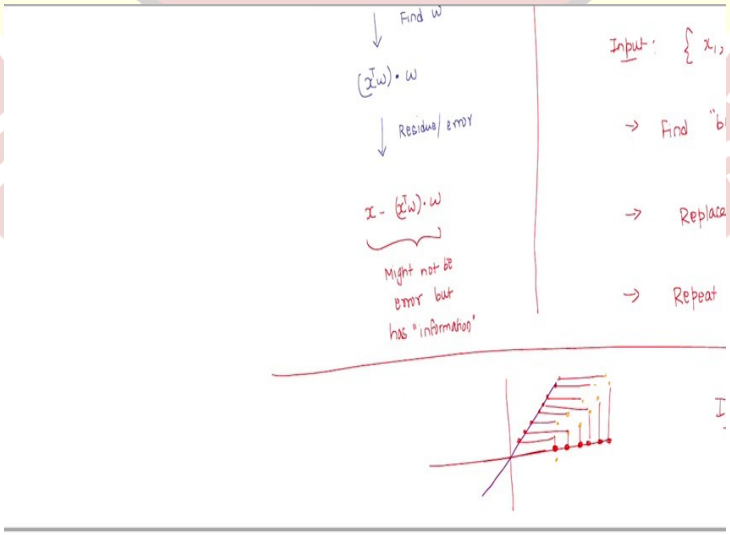
respect to the first line that I have found. Now, what I can do is I can repeat this procedure to obtain w_2, the second level.

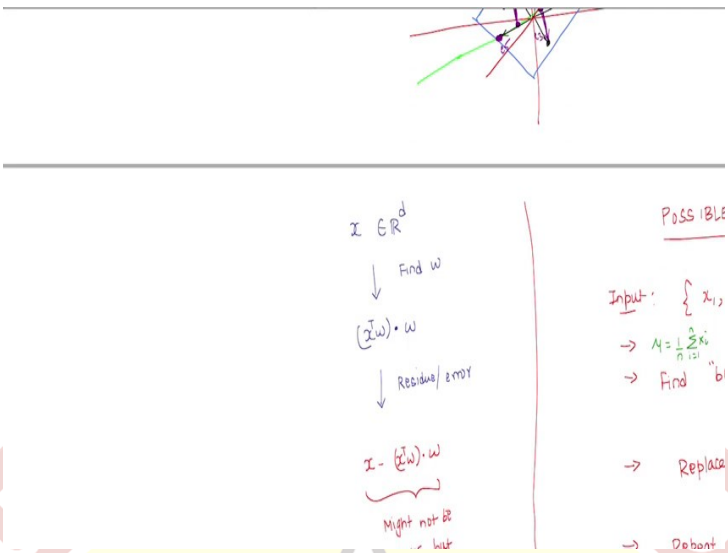
Why should I stop after getting the second level w_2? So the idea was the residue might contain information. Let us say now we had thousand dimensional data so we fit the best line, we found the best line w_1 we looked at the residue, well, let us say we think that the residue also has information, so we compute the residue data set again we find the second line and we could go on and on.

So we can go until, I mean up to the number of dimensions that you have; so you can add more and more data sets create more and more data sets using the residues and we can keep doing this. Now the question is, well, this is a possible algorithm but there are several questions that come up here. So one is, how do we do the reconstruction once we have these different w's that we have found?

What does it mean to say that we have compressed here? So how do we reconstruct? What kind of compression are we achieving? This is an important question. We will answer that, but even before that another question that might arise is a more basic question and we will answer that and then we will answer the more pertinent question about reconstruction and so on.

(Refer Slide Time 10:54)





Another question that might arise is that the data itself may look something like this. So you might have data which is like this even for one dimension, so even for two dimensional data you might have data like this. Now remember, we are only finding lines that pass through the origin, because our notion of compression is representative and a coefficient multiplied by this representative, which means if the coefficient is 0, then the origin should be part of the line that we are considering.

So what might happen is that our data may not necessarily be around the origin, which means though the data is along the direction pointed by the purple line here, but if you really try to find proxies like the way we did, by looking at proxies like this, the errors might be much more than just projecting all the data points along the y-axis.

So if you find proxies along the y-axis, so it might so happen that well the y-axis is, sorry the x-axis, the x-axis in this case might be a better line in terms of the reconstruction than your actual line which you really want to get, which is capturing the direction in which the data is dispersed. So what can we do here? Well, one thing we can do is to recognize the issue first, the issue is that data may not be centered.

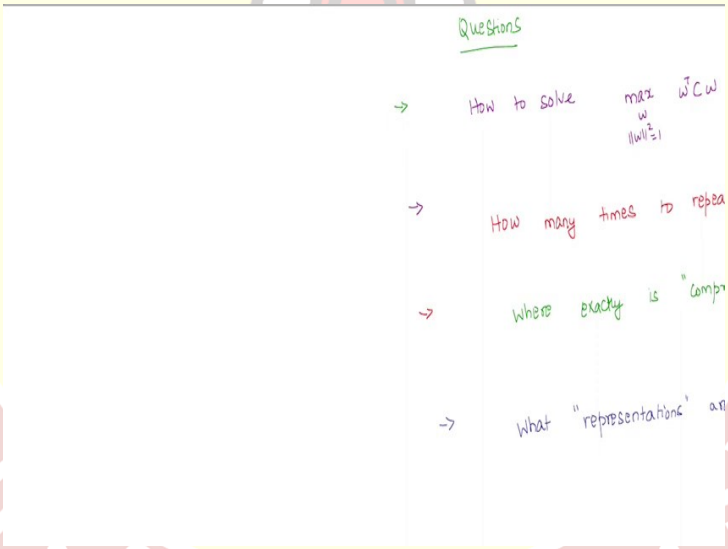
One small fix that you can do is first do a centering of the data set in this possible algorithm that we have, which is first bringing your data such that the center or the origin lies at the center of the data, such that when you search for lines that pass through the origin then you are necessarily finding the line which aligns with our intuition of the direction in which the data itself is dispersed.

So how could you do the centering? Well, the centering is simple easy, so you first find the mean of your data points which is the average of your data set given to you and then you first make your data set centered by subtracting the mean from your data points. So for all x_i, for all I. Now what does that mean?

Well, you are separating the mean from each of the data points, so now if you recompute the mean of this subtracted data set that would be the origin by definition because the mean itself is just the average of your data points, essentially you are moving your data points, translating your data points such that the origin is the mean.

So that that takes care of the centering issue, but still our previous issues are still pertinent. So several issues, so I am going to write down these issues and then we will try to answer each of these in the coming videos.

(Refer Slide Time 14:18)



So the questions that I am going to leave ourselves with right now are the following; of course, the first question is how to solve max of $w^T w$=1 or $\|w\|^2$=1like how I have been using, $w^T Cw$, of course, I did hint at the answer to this that this will be an eigen vector solution, but we have to think about that a little bit more carefully.

The second thing is that, well, in this possible algorithm so how many times should we repeat the procedure, procedure of computing the residues and then finding the line that best fits the residues, repeat the procedure. Perhaps more importantly is the third point, where exactly is the compression happening now in the algorithm.

Earlier it was clear that we were finding one single line, we were finding proxies along this line and we knew that how the compression was playing out. So where exactly, but now, perhaps not that obvious where is the compression really happening. And the fourth point is that well earlier we know that it is a representative and a coefficient and then representative multiplied by this coefficient will give us the proxy.

Now we have multiple lines, perhaps multiple coefficients, so the question is what are the representations that we are really learning using this. So these are some questions that we need to answer to understand our procedure better and we will see that all of this will tie together nicely to whatever you may have learned in your linear algebra classes already in a Machine Learning Foundations' course. If not you this might give you still a different way of thinking about some of the algorithms that you may have already seen.

So if you answer all these four questions carefully then that will lead us to a very, very solid algorithm in data science, which will be our first algorithm for this course, which is a Representation Learning Algorithm for Unsupervised Learning. We will talk about what this algorithm is once we answer all these four questions.

And think about these four questions to see if you already have some ideas about what are the answers for these. So I will leave you with these questions for this point and we will come back and answer all these questions and develop the algorithm that we set out to in the next video. Thank you.