Machine Learning – Stanford C229

# Introduction and Linear Algebra

## In this lecture:

* Introduction and Logistics
* Review of Linear Algebra

## Goals of the course

* By the of the course, you should be an expert in Machine Learning.
  + You should understand the internals of how machine learning algorithms work, not just using them, but understand what is happening inside them.
* Enable you to build machine learning applications.
* Enable you to start doing machine learning research.

### Introduction of the teaching Team

So we're gonna start off today with, um, some introductions of the teaching team. Um, that's me, Anand. Uh, I am a 4th year PhD student in Computer Science. I work with, uh, Professor Andrew Ng on machine learning and applying machine learning, uh, to different problems, uh, mostly in health care. We also have a wonderful team of teaching assistants. You'll meet the rest of them through the quarter and office hours and such. [NOISE] Okay. So the goals of this course, what do we- what do we, uh, what are the goals on this course? So by the end of the course, you will- we want you to be an expert in machine learning, which means you understand the internals of how machine learning algorithms work. Not just using them, but understanding what's happening inside the algorithms. That's the main, um, uh, I would say goal number 1 of this course. Goal number 2 is to enable you to build machine learning applications, which means having a good understanding of what algorithms to use in which scenarios. How do you- how do you, um, decide whether a given problem is a good machine learning problem? Not all problems are good machine learning problems. And once you know that a problem is a good machine learning problem, how- do you treat it as a supervised learning problem, unsupervised learning problem? We'll go through all those things, right? And the third goal is also, uh, enable you to be able to start doing machine learning research, which means making you familiar with a lot of the terms and concepts that you will encounter if you pick up and read a machine learning research paper. So those are the, uh, uh, the top three goals for this course.

### Prerequisites

* Computer science principles
* Be comfortable with Python and NumPy
* Probability – random variables, distribution
* Linear algebra and multivariate calculus

Prerequisites. So, um, machine learning is- is, uh, part mathematical and part computer science in the sense it require- it's- it's really the culmination of both computer science and, um, and- and statistics. So you will need to have basic computer science, uh, uh, principles. Uh, you need to know them. For example, you need to know what is Big O notation, right? You need to, uh, kind of understand what's, you know, uh, what's computation versus what's memory, you know, basic computer science principles. You'll need to be comfortable writing non-trivial, uh, uh, programs with Python and NumPy. It would be great if you are comfortable with recursion. You know, some- some of those, uh, some of the ideas from recursion will be useful when you're translating, um, algorithms to kernel methods. You know, we'll go in- into detail about them later. So, um, uh, you know, we- we will assume that you are- all of you are comfortable writing programs, especially in Python, or at least be willing to put in the time and effort to, uh, pick it up. The other big, um, prerequisite is probability. You should have taken some kind of a probability, um, uh, course in your, you know, uh, academic history so far. So concepts like random variables, distributions should be, you know, um, uh, you should be pretty comfortable with them. You should know what's the difference between a random variable and a distribution, you know, there are, you know, there are distinct things that you should know already what- what's- what's, uh, what they are. You know, things like expectations. We'll be using all these concepts very liberally throughout the- throughout the quarter. The last prerequisite is, uh, linear algebra and multivariate calculus. So you should already be comfortable with things like gradients and Hessians. Um, you shouldn't be seeing them for the first time. You should- you should be comfortable with, uh, these concepts already. Uh, and- and- and you know, things like eigenvalue, eigenvector, um, um, as well. [NOISE] So what about the honor code? So you are strongly encouraged to form study groups. Um, in fact, uh, I would say forming study groups is a key- key step for being successful in this course. Um, especially if you're not so comfortable with the prerequisites, please do form study groups and, you know, work together in study groups. However, you should still write up your homeworks and write up your code independently, right? So it is fine to discuss and, you know, uh, work on solutions together. And once you're done with your study group meeting, put aside all the material that you used during the study group and independently, from scratch, you know, write up your homeworks by yourselves in your own words. Please do not refer to, you know, uh, the material that you used in your study group session while writing up the, um, uh, uh, writing up your ho- uh, your homeworks, right? Um, another important note is that it is in violation of the honor code to refer to homeworks and solutions from previous years. Whether they are the official solutions, uh, released by this course or solutions written by some other students in a GitHub or wherever. So it- it is against the honor code to, um, refer to, um, uh, solutions from the past. And we are very serious about the honor code. And, uh, though we will, you know, uh, we- we will not, uh, actively look for honor code violations, we trust that you will be,

### Course Structure

uh, you'll be following them. So the course structure, there are three homeworks. Uh, each homework will be about, um, two weeks long, and each homework will- will count towards about 20% of your final grade. And there'll be a final exam. It'll be a take-home final exam. Uh, the reason it's a take-home is because you'll need a computer, uh, to- to- um, um, do your final exam. There will be some code, uh, more details on that, you know, uh, towards the end of the quarter. Uh, but it's gonna be a take-home final exam. Very likely a 24-hour take home final exam. Uh, we haven't yet decided what's- what's the exact start of it, uh, when the exam will start, but it's gonna be a final exam during the, uh, finals, uh, final slots. Logistics. So the course website is on, it's- it's up. It is cs229.stanford.edu. On the course website, we have the office hours calendars, the course calendars. There are two calendars up there. One is for all the office hours when- when each TA is holding their office hours and- and where exactly. And, uh, the deadlines are part of the, uh, course calendar. We can have a quick look at the, uh, course website. Oops. Anyways, it- it's not opening over here. Anyway, so the, um, yeah. You can look up the course website. Uh, on the course website there are three big buttons as soon as you open them, you know. One- one button is called syllabus. Um, let me see if I can- there you go. Okay. So you have three big buttons, the syllabus, link to Piazza, and- and, um, the calendars.

### Calendars

So on the calendars, there are two calendars. First one is the office hours. You can- you can click on this and add it into your Google Calendar if you wish to do so. Uh, but the exact office hours of every TA is- is, uh, is out there. And if you scroll further down, you'll see the course calendar. Um, basically we are in this lecture right now. And [NOISE] you'll find details about when each piece set is out, when each piece set is due, um, all the deadlines are in the, um, um, course calendar. So if you want, you could- you could just add this to your, you know, your personal calendar to keep track of the deadlines.

### Syllabus

Um, and then we have the syllabus page. So on the syllabus page, um, we have a collection of topics that, um, you're probably not gonna cover all of these, but we're gonna cover a pretty big subset of these topics. Um, and further down, um, here is gonna be a lecture by lecture of what we are gonna cover in each lecture with the corresponding, uh, slides or, you know, notes corresponding to that lecture. Yeah. And- and the syllabus page, uh, the- this will be updated through the quarter depending on what- what the actual- what the actual progress we make is, okay? And Piazza forum. So all of you should have- um, all of you should have received an invite to Piazza already. If not, click on the Piazza link and enroll right away. Piazza is probably the most important, um, most important platform for the course in terms of logistics, because all announcements will be made on Piazza, the homeworks will be released on Piazza, and, you know, anything that you need to know in terms of, you know, the final exam logistics. All those details will be announced on Piazza. So please sign up on Piazza and- and, you know, um, and monitor it. Gradescope. All the, uh, submissions will be done on Gradescope. Um, if you- if you, um, haven't used Grade- Gradescope before, you know, um, spend a few minutes to, you know, get familiar with it. Um, the homeworks will be uploaded by you into Gradescope directly and we will grade them on Gradescope. Um, again, for Gradescope, also you should have received invites by now. If you have not received your Gradescope invite, you know, create a private Piazza post. So, uh, in terms of any questions that you have through the quarter, your first- um, your first destination will be Piazza. If it is, uh, uh, any- a question related to the course content, about the course logistics, create a question on Piazza. It's ve- it's very likely some others may have already created a similar question. So you can- you can either, um, if you know the answer, help them out or if you, uh, if you don't find the question posted there. If it is a question that is specific to you in the sense, uh, for example, if you're not able to, you know, submit a homework on time and you know that up front and you want an extension, things that are specific to you, or if you have an OAE letter, um, create a private Piazza post, um, you know, things that you don't want the rest of the class to hear about. Um, if there are things that you want to, um, if there are things even beyond that that you wanna, um, um, let's say for example, just let me know, um, and not even let the TS know, just send me an email directly with the word CS229 in the- in the, uh, uh, in the subject. So that's about the logistics. Any questions on the logistics? I'll take that as a no. Okay, moving on. All right.

### What is Machine Learning?

So what is machine learning? So the term machine learning was coined by Arthur Samuel in 1959. Um, and one of the very early programs that- that kind of got people interested in machine learning was a Checkers playing program written by Arthur Samuel. So it was a program, um, probably most of you know what Checkers is. It's- it's a two-person game, there is, um, a checkers board that looks like a chessboard and you have white and black pawns. It has certain rules. So Arthur Samuel wrote a program in, um, few years before 1959 and it became popular in 1959, where the program would learn to play against itself without any- any- any kind of strategy being explicitly programmed into it. Right? It learned to play by- against itself and it improved its, you know, game-playing performance as it experienced more and more games against itself. Right. Eventually, it- it learned to play checkers better than, you know, Arthur Samuel himself. I don't know how good a player he was to begin with, but it got pretty good by just self play. There was no- there was no strategy coded in by Arthur Samuel into the program. Only through experiences, the program learned to, uh, play better. And that was kind of the, uh- the event that kind of sparked interest, widespread interest in machine learning. A more common definition by Tom Mitchell, he's a- a professor at CMU, is machine learning is the study of computer algorithms that improve automatically through experience. Now, experience is, um- is a loose term. What do we mean by experience? Most of the times, by experience, we mean prior data, like examples from the past, where the machine learning algorithm itself is a very generic, um, template algorithm. You don't, um- and you just- you feed the algorithm data from the past, what you call as training data. And the algorithm analyzes the data that's given to it. And you- looking for patterns in the data the- the algorithms improves its performance at- at- at- at whatever task it is- it is, uh, designed to operate on. So the- the key here is data or experience from the past that- that distinguishes machine learning from other fields. Right? Machine learning is also related to the field of artificial intelligence. So AI is a much broader term and we see the terms AI and machine learning being conflated a lot of times, you know, especially in- in the news and media. But strictly, um, if- if you look at the definitions of AI, um, and the definition of machine learning, machine learning is a subset of AI, right? AI is a much broader field. AI is about- is about building programs, where the programs operate at the level of performance of- of say, a human being for example. Again, that's a very loose definition of AI, right? So AI is, uh- is a broader- is a broader definition and it- it deals with algorithms that perform at a cognitive level, similar to a human being. Machine learning is one approach of how to, uh, implement such programs. For example, you could implement an AI program that does not look at any kind of training data. You just use your smarts and write up, you know, a very smart program that- that works really well and behaves as, you know, uh- at the level of a- a- a human in terms of performance, and that's still AI. But that's not machine learning. In machine learning, data is the key. You look at past data, use it as a training set, or you build a simulator from which your program can kind of, um, build experience by interacting with its environment. And that program which is looking at data or which is looking, you know- which is getting experience in some kind of a simulator improves its performance over time. That's machine learning. That's one way to, um, um, make your programs intelligent, right? However, uh, most of the attention, recent attention in artificial intelligence is due to machine learning, right? Artificial intelligence has been around for a while. Machine learning has been around for a while. But some of the recent advances in machine learning has made machine learning very successful. And therefore, the interest in artificial intelligence is- is kind of, uh, reinvigorated again. And even within machine learning, there is a specific, um- a specific type of machine call- uh, learning called deep learning, which has actually seen the most advances recently. So it's, um- it's good to have this clear picture of AI, which is a much broader field, which is- which is, uh, dealing with programs that operate at a- at a similar cognitive level at- as a human being, right? It does not- AI does not tell you how to implement such programs. Within AI, machine learning is a subfield which- which prescribes one way to implement such programs. That's basically look at prior examples and- and- and learn from examples. And even within machine learning, there is a subfield of machine learning called deep learning, uh, which uses something called neural networks, which we will see later in the course. And that- that has- that's this- the field that has made a lot of progress in the last 5 to 10 years.

### Examples of Where Machine Learning Has made A Lot of Progress

Okay. So here are some examples of where machine learning has made a lot of progress.

#### Computer Vision and Image Recognition

So computer vision and image recognition, um, this was, um, uh- this is a field that has probably undergone the most progress in the- in the recent years. There is, uh, a famous, uh, dataset called ImageNet, a computer vision dataset. And there is a kind of model called, uh, the- a convolution neural network, uh, which together, you know, more or less revolutionized computer vision. And this is all recent, you know, since, you know, um, after 2012 and 2015, you know, within the last- last decade. And it has also, um, significantly improved autonomous driving where, you know, machine learning techniques are used to sense where, you know, uh, pedestrians are, where the stop sign is, where the traffic lights are. Um, and it- it- it is- you know, um, it's a key component for- for, uh, autonomous driving. Machine learning has also significantly improved speech recognition. And- and- and things like, you know, uh, it- it has made- and it has made possible things like voice assistance like, you know, Apple Siri or, um, uh, the Google Assistant, etc. Language translation. So Google Translate now uses machine learning or from what we know, from what Google tells us, that it uses, uh, deep learning. Um, and in fact, there was also a paper from Facebook,

#### Unsupervised Translation

I think last year, which does unsupervised translation, which means just give it a corpus of, you know, a whole bunch of, you know, English- um, uh, of English sentences and English documents. And just give it a whole bunch of, um, for example, French documents. And just by looking at them where it was- the algorithm was never told what is a matching English and- and French, uh, statement, but- but just looking at two different corpuses, it learned to, um, uh, translate sentences from one language to another. And similarly with Google Translate, they had this, um, um- they had this paper a few years ago where, um, they built, uh, a single model that could translate from various pairs of languages. And that model automatically learned how to translate from new pairs for which it had never seen a training example before. Tho- tho- those are examples in language translation, you know, a very exciting- exciting field. There's also a lot of progress happening in reinforcement learning,

#### Deep Reinforcement Learning

uh, or deep reinforcement learning. And most of the progress that has been in game playing, like, for example, Deepmind over the last few years. They built, um, reinforcement learning models that could play Atari games at the level of humans. So you- you feed the model, the pixels from the display of an Atari game. And just by looking at the pixels of how the- how the game looks, the, uh- the agent will, uh, um, controls different actions and the agent learned to, um, um, play Atari games at a superhuman level across a whole, you know- across 30, 40 games. And also AlphaGo. Until very recently, uh, Go is another board game. And it was widely considered to be, um, extremely difficult to play because you need a lot of strategy, a lot of planning. And after chess was kind of, um, conquered by- by- by computer programs in- in- in the '80s and, uh, um, around the 1980s, Go was then, you know, thought of as, you know, the next big, uh, uh, board game that's really hard to- to kind of beat. And, you know, uh, AlphaGo, again, you know, uh, was- was, uh, built by Google again, you know, which- which plays, uh, better than like the world's top Go players. Those are, uh, some of the recent, um, uh, progresses and some preview into the course, what we'll be doing, um, uh, through this quarter. So we're gonna look at, um, different supervised learning algorithms, unsupervised deep learning algorithms, some learning theory and some reinforcement learning. So most of the time will be spent on supervised and unsupervised and a- a good amount on learning theory, and maybe a little bit on reinforcement learning theory. So in supervised, uh, learning, we are gonna, um, look at problems and classify them as,

#### Regression Problems

you know, regression problems versus classification problems. What are regression problems? When the output of your machine is- is some kind of a real valued output. Uh, for example, if you're trying to predict the wind speed tomorrow at, you know- at Stanford and you're going to build a machine learning model to predict the wind speed, uh, based on today's, uh, uh, uh, weather condition as the input, the output is going to be a real valued number, some, you know, kilometers per hour or miles per hour. So that's a real valued number.

#### Classification Model

So that's, uh, for example, a regression model. Uh, classification model. Uh, for example, if the output is binary or some kind of class, will it rain tomorrow? Yes or no? Right? That's a classification problem. Given today's conditions. If you wanna predict, you know, uh, whether it's going to rain tomorrow, the answer is yes or no. And, uh, you can- you can, um, classify machine learning algorithms in- in, um- along many different dimensions as generative versus discriminative, probabilistic versus non-probabilistic. So any given machine learning algorithm, you can kind of place it on this hypercube of all these dimensions. Um, and, um, kind of taking a step back, what is supervised learning? Supervised learning is a kind of machine learning, um, uh, technique where the training data that you have come in pairs. Each pair has an input and an output, right? An input is what you feed into the machine learning algorithm, and the output is what the machine learning alg- algorithm should output, right? So- and that's called supervision, because for each example, you're telling the algorithm what the right output is. So that's the su- that's where the supervision comes in. In unsupervised learning, um, algorithms, you're- you give the algorithm some data. There is no explicit supervision and you just ask it to look for patterns or structure- interesting structure in the data. And the output of your algorithm is that interesting structure or, you know, the interesting pattern. And again, uh, we kind of, um, classify unsupervised learning al- algorithms as those that look for clusters versus those that look for subspaces, you know. It's fine if you don't, uh, um, fully understand these terms now,

#### Deep Learning

I'm just giving you a flavor of what's to come, er, you know, in the rest of the quarter. And deep learning is, um, what's also commonly called as representation learning. Where, um, the deep learning can- can, um, can kind of plug into- you can view deep learning in a supervised setting, unsupervised setting, in a reinforcement, ah, setting, etc, where you're trying to learn representations of your data. Uh, whereas, um, in non-deep learning approaches, you are manually feeding in what the right representation is for the supervisor in a supervisor reinforcement learning algorithm. Whereas with deep learning, you're- um, you're letting the algorithm itself learn the representation as well along with the final task. And then we'll, um, cover learning theory, um, super important concepts of, you know, the bias-variance decomposition, and bias-variance trade-off, generalization, and uniform convergence. Why do we expect a model that was trained on this training set to even perform, you know, with any level of accuracy when you- when you start using it in the world, when you put it into production because you trained it on this specific training set.

#### Learning Theory

Now, why do you expect this model to work well in the real world? You know, questions like that, we're gonna answer them, um, in learning theory. And that's probably- the learning theory is probably the- the- uh, the part of the course that- that, um, makes this course interesting, in the sense you learn principles that are common across learning algorithms, you know, uh, common across learning algorithms that have not yet been invented, right? You know, the- the- the foundational principles of why machine learning works. Um, and then we'll spend some time on reinforcement learning as well, uh, though not a lot.

#### Supervised Learning Algorithm Image Classification

So, um, here's an example of a supervised learning, um, algorithm, image classification. So these are examples, a few images of handwritten digits. This comes from a very famous dataset called MNIST, um, where the input is a set of pixels. No, I think this is, um, 24 by 24 pixels or so- some- some- some resolution like that. And, uh, they are black and white images. Uh, in fact, they are grayscale images, um, where you- you have the pixel value at- at- at, um, each location, you feed the set of pixels, um, as input to some learning algorithm. And the output of the learning algorithm is gonna be- is gonna be, um, uh, one of, uh, digits 1 through 9, or 0 through 9, all right? It has 10- it's- it's a- a multi-class classification problem. So the output is, it's- it's a classification but not binary. It's 1 among 10 classes. Um, and you want to learn, a machine learning model given this training data, which- which is able to- um, which is able to predict what the handwritten digit is, for digits it hasn't seen before. Um, in fact, um, you would be doing- you'll be working on this dataset and implementing a neural network to- uh, to build a handwrite- uh, handwriting, ah, um, ah, digit classification in one of your homeworks.

Okay. So unsupervised. Here's an example of, um, an unsupervised, ah, learning.

So in this example, uh, it's what's commonly known as the cocktail party problem. So imagine there are n speakers, in this case, assume there are two speakers who are in a cocktail party, just the two of them. Um, and there are two microphones that are placed in the room at, you know, some- some, um, uh, specific locations. [NOISE] Now, um, the two microphones are recording what the two speakers are speaking, and each of the recording is some mixture of the- the- um, what the two speakers are saying, okay? And the- the task here is now to take these two audio clips, each of them has a different mixture of the two speakers. [NOISE] And the- the machine learning algorithm is expected to output two different- uh, two different waveforms where the source- the- the voices are separated, right? So all that you are feeding into the algorithm is just two clips. You're not saying, you know, um, there's no kind of supervision telling, you know, speaker 1's voice sounds like this or speaker 2's voice sounds like this. There's no supervision whatsoever, you're just feeding it two clips where the clips have mixed audio of two different speakers, and the, um, um, model outputs, um, their separated voices. So let's see if the audio works here. Um, [FOREIGN] [OVERLAPPING] 1, 2, 3, 4, 5, 6, 7, 8, 9, 10. Right? That's, you know, from one microphone there were two speakers. [FOREIGN] [OVERLAPPING] 1, 2, 3, 4, 5, 6, 7, 8, 9, 10. Right? That's from another microphone, probably the- the- the sound of one of the speakers was a little louder and the second one, or- or, um, vice versa. And when you separate them, [FOREIGN] 1, 2, 3, 4, 5, 6, 7, 8, 9, 10. Right? So all what you gave the algorithm was these two audio clips and nothing else, right? Just two wave files and it- you know, it analyzes the, uh, audio wav- waveforms looking for structures, and it's able to separate the voices with no supervision whatsoever. And you're gonna be doing one of these in your homeworks as well. Um, it- it will not be two, but actually, um, um, a set of five audio clips that are mixed together and- and you're given five different, um, audio files. And you will implement the- uh, the ICA algorithm that we're gonna cover in the course and- and separate the audio into five, you know, distinct- distinct clips. [NOISE] And finally, reinforcement learning.

#### Example of Reinforcement Learning

So here's an example of reinforcement learning, um, where the goal of the algorithm is to control what's called- ah, um, what's commonly called the inverted pendulum or the cart-pole. So imagine you have a stick and you're trying to balance a stick on your hand, uh, a vertical stick on your hand, you know. And we are trying to control this- ah, um, this cart-pole agent in order to keep the stick that's placed on it well-balanced, right? So, uh, when you start a- a learning algorithm from scratch, so, uh, this is episode, so each trial, um, is an episode, when- once the stick falls, you start the next episode. And for example, this is episode 0, it is falling, falling, falling. Now it's episode 1, falling again. Episode 2, fell down. Episode 3, fell down. And you- you train this algorithm, you continue trying it for more episodes, and eventually, it will- it'll get better. Um, I stopped the algorithm for- for this- uh, I stopped there at about 130-ish. But if you let it continue, it's- it's gonna learn to- um, learn how to balance the cart-pole for, you know, potentially indefinitely long. There you go. Oops. Oh, no, I think it- it- it falls off the table or something. Anyway, so, um, again, this is gonna be on your homework as well. You're gonna- you're gonna, um, ah, write a reinforcement learning algorithm that's gonna learn in a- in a simulator like this, how to, uh, balance- balance a stick on top of cart-pole. Yes, question? Yeah, just out of curiosity for the previous example you gave, is it required that you have as many clips as you have speakers? So the question is, uh, for- for, uh, those who are watching, uh, the lecture on- online, um, is it necessary to have as many number of audio clips as the number of speakers? Yes, that's exactly right. You need as many number of audio clips as, uh, the number of speakers. Okay, so that's about, uh, it for the introduction.

### Linear Algebra Refresher

So for the rest of today, we will be- yeah, so for the rest of today we'll be covering- um, doing a review of linear algebra, uh, in case you, you know, it's been a while since you took linear algebra to, uh, brush you up and familiar- familiarize you with the concepts that are more important for this course. And in the next- um, next class, we will be doing- covering matrix calculus and- and some probability and statistics. Um, and from, you know, onwards we'll be, uh, getting into actual machine learning, uh, models. Yes, question? You said that working unsupervised learning, you try to find the structures. [inaudible] Yeah. So, ah, the- the, uh, question was the- um, in unsupervised learning you're- you're- jus given a dataset and asked, uh, you know, to- how to find structures in the data, and I also spoke about an example with- with the, uh, language translation where a model was just given unpaired mappings of language and it still learned how to, uh, map from each other. The answer to that is a little technical. Let's take that offline. You know, after the lecture, you know, come up and we- we can talk about that. All right. So I'm gonna be assuming all of you ha- are familiar with some basics of linear algebra in the sense, all of you know what a matrix is, know what a vector is, you know what you can do with vectors and matrices.

#### A vector is:

v **∈ Rd**

y

z

v3

v1

v2

x

**R3**

v =

vT=

Uh, so we'll be reviewing them, but also I'm gonna be, um, assuming some familiarity from, um, all of you. So first of all, some notation. So what's a vector? Right? So, um, for the purposes of this course, you will assume that a given vector, let's call it V, is an element of R^d. Now, what's- let's- let's- let's make sure all of us understand what- what this means, right? This means an element of, you know, from set theory. So V is a vector, it is an element of this set, and this set is a d-dimensional real space, right? So what's a d-dimensional real space? So this is, uh, imagine this is, um, the real line, and this is x, y, z. So this is an example. This is R^3 because there are three dimensions, right? So a vector is an element of a d-dimensional real space, and vectors are- when we say it is an element, it means it is a point that lives in this d-dimensional, um, um, real space. This vector can also be represented as, um, V\_1, V\_2, V\_d. We generally write- um, a vector generally is considered as a column vector. We write it as a column in this course. Um, you can also write a row vector as V^transpose; V\_1, V\_d, right? So what- what- what do these numbers mean? These numbers are basically the coordinates of this point, right? So this is V\_1, this length, and this length is, uh, maybe I should write it here, V\_1, V\_2, V\_3, right?

#### A matrix is:

A **∈ Rm×n**

**A =**

And this point over here is- is V. Okay. Similarly, we- we also have a matrix. So we write matrices as, let's call it a letter A. So generally matrices have- we use capital letters, for vectors we use small letters. Let me say is part of R^m\*n. All right. You can- you can interpret a- a matrix A as, um, as an element living in an m by n dimensional real space. What- what does that actually mean, right? So you can think of that as, [NOISE] so a matrix is- is a grid of numbers, um, real valued numbers for the purpose of this course. Through the course we- we hardly ever deal with complex numbers, so, um, it's gonna be only real numbers. Think of it as a m by n grid of real numbers, and that's a matrix. You can interpret this as n- m- n column vectors of m dimension each, right? Or you can think of it as m row vectors of n dimension each, right?

#### Identity Matrix

**I =**

There is ah, um, a sp- a special matrix called the identity matrix, which is 1s along the diagonal and 0s everywhere else. I just write a big 0 to indicate that, er, all- all the individual cells are 0.

#### Diagonal Matrix

You can also have a diagonal matrix where- a diagonal matrix is one where all the cells except the diagonal are 0s and these could be any value.

#### Symmetric Matrix

A = AT

**=**

Aij = ATji

There's something called a symmetric matrix. [NOISE] A symmetric matrix where A equals A^transpose. What is transpose? Switching rows and columns. Exactly. So a transpose of a matrix, uh, in this case, um, take the first row, I'm I'm- sorry, the first column, and treat it as the row of the transpose, and take the second row- second column, and make it the second row and so on, right? The- the property that, uh, the transpose satisfies is that A\_ij equals A^transpose\_ji. Here, i and j are the, um, row and column, um, indices of- of- of- of- of a given matrix, so when you transpose it, the row number becomes the column number and the column number becomes the row number.

#### Trace of Matrix

Uh, the trace of a matrix is the sum of the- of the diagonal of- of that matrix. So take a square matrix and sum up the diagonals, um, and that- that's the trace. [NOISE] Let's look at some basic operations that you can do with vectors.

#### Inner Product of Vector/Dot Product

x, y **∈ Rd**

or

**xTy**

**=**

So if you have two vectors, you can- you can kind of combine them in two different ways, right? So the first is called the inner product, so first this, so vector-vector operations. Given two vectors, what can you do with them? So the first is- is the inner product. And, um, technically the inner product and dot product are distinct concepts, but in this course they're gonna be the same, so, uh, the dot product. So one way to think of the inner product is, um, if you have two vectors, x,y both in R^d. So in order to perform the inner product, the two vec- the two vectors have to be of the same dimension, for example d, and the inner product is the sum over i equals 1 to d, x\_i y\_i, all right? And this is also written as x^transpose y. So this is the- the- the mathematical representation of the dot-product. You could also think of this as this. You think of a vector written as a row vector, another vector written as a column vector, the two have to have the same dimension, and when you perform the inner product, you get a scalar, right? You feed in two vectors, the output is a- a single number. Right, you- you've lost, um, the angles or- or- or- the orientation of the vectors once you perform an inner product. All right.

#### Outer Product

x **∈ Rd ,** y **∈ Rp**

**xyT** or **yxT**

The other operation you can do with two vectors is the? Anybody? You can do the cross-product. Uh, the cross-product, however, is not very interesting for us. You also have something called the outer product, right? So the cross product and outer product are different things, right? So, uh, in cross-product you, uh- you're given two vectors and you find a new vector that is perpendicular to the plane in which the two- two vectors are, but here we're gonna talk about the outer product. [NOISE] So you have the inner product and you have the outer product. So in the outer product, you can have x in, let's call it R\_d and y in R\_p, right? So for the outer product, the two vectors need not to have the same dimension, right? But inner product they have to have the same dimension. And, um, mathematically we write the outer product as x, y transpose, right? It could also- you- you could also have a different outer product which is y, x transpose. In this case, x transpose y and y transpose x for the inner product was the same. So, you know, you could- you could, uh, switch the order. But for outer products, if you switch the order, you get something which is a little different. Um, the way to think of the outer product, um, at least, uh, pictorially is something like this. Okay. Right? So think of vector 1, vector 2. So we have a column vector and a row vector, right? Um, so in this case, this is d dimensional and this is p dimensional, right? So with the inner product, we took a row vector, a column vector and convert it into a scalar, right? But with an outer product, what we do is you take each- each and every possible pairs of elements from the row vector and the column vector. For example, take the first element from the column vector and say the third element from the row vector, multiply those two elements, right? And that becomes the first row, third column cell of this matrix. So you take two vectors and construct a matrix out of them. All right, by- um, pick the- pick the ith, um, element from this vector, jth element from this vector, multiply them, and that becomes the ij element of- of- of the matrix, right? So this- this, uh, for the outer product, you don't need the vectors to have the same dimension.

#### Rank 1 Matrix

The outer product of two vector produces a rank-1 matrix

Summing two rank-1 matrices will produce a rank-2 matrix

=

2

k

1

≤ min (dpk)

And a matrix that you construct from one row vector and one column vector is also called a rank one matrix, right? Why is it called rank one? Because one way to think of it is it is made of one pair of a row and a column vector, right? So that makes it, um, um, what's, uh, what's also called as a- a rank one matrix. You could also, for example, um, pick two rank one matrices, add them up. So these together will give you, uh, um, one rank one matrix, add it with another rank one matrix. Right? So the row column- the- the column vectors have to be of the same dimension. The row vectors have to be of the same dimension, but the two need not be the same dimension, right? So if, you know, these two have to be d dimensional, these two have to be for example p dimension. Here you get a rank one matrix, you get another rank one matrix. You sum them up element-wise, and you get another matrix of rank. [inaudible]. So you take two rank one matrices, you add them up. [inaudible]. You get a rank two matrix. So you take two rank one matrices and you add them up, you can- you get a rank two matrix. Assuming the vectors are linearly independent. Okay? If the vectors are linearly independent, you take two rank one matrices, add them up, you get a rank two matrix. Now, what happens if, let's say we add, um, k of them, so 1, 2. What's the rank of this matrix? Rank k. So the rank of this matrix is actually going to be the less than or equal to mean of d, p, and k. So the rank of the matrix cannot be bigger than the smaller dimension of the matrix. All right? So by adding the rank one matrices or, you know, um, or linear- adding the linearly independent rank one matrices, you're increasing the rank of your, um, uh, resulting matrix, but you can only go up as high as the smaller of the two dimensions. Yes. Question. Can you define the rank, please? Yes. We will, I- I will define the rank. For now, let's- let's think of a rank as, um, think of this as a sheet of paper. Right, you know, think of this as a matrix and you add more sheets of papers, you're increasing the rank of the matrix, we'll precisely define what the rank of a matrix is in- in a few minutes. Right? And just by looking at a matrix, if you're just given a- a bunch of numbers of- of rows and columns, it's pretty much impossible to tell what the rank of that matrix is. Right? Superficially, you- you cannot just tell by looking at, um, um, looking at a matrix what its rank is, right? It's- it's like an inherent property. There are, you know, simple cases if you are given a diagonal matrix where everything is 0-something diagonal, yes, you can tell the rank, but in general, just by looking at the values for matrix, you cannot tell what the rank of matrix is. Okay, so, um, those are- are vector-vector operations.

#### Matrix Vector Operations

n

m

n

x

A

|  |  |  |
| --- | --- | --- |
| A **∈ Rm×n** | x **∈ Rn** | Ax **∈ Rm** |

Let's look at some matrix-vector operations. Right? So, um, you know matrix vec- so, um, even to our vector- vector operations, the inner product and the outer product. For the inner product, you had to have both the vectors of the same dimension and you would get a scalar. For an outer product, they could be of different dimensions, and you got a rank one matrix, right? Now, let's look at matrix-vector operations. So for a matrix-vector operations, first we're gonna consider operations where you have a matrix, let's call it a and a vector, call it x, and this, let's call it M by N, and this is, uh, n. So A belongs to R\_m by n, and x belongs to R\_n. Right? So this is- um and- and so A\_x is now of dimension. [inaudible] A + 1, mm-hm. So how do you think about, um, this operation? So first you can, um, um, think of, uh, the matrix as a set of rows, right? So each row has n elements. Right? And the vector you're multiplying it is a column vector. And all you're doing is the inner product, right? Take the first row, get the inner product with the matrix, and you get a scalar, right? Take the second row do the inner product, you get another scalar. And you're gonna get as many number of scalars as the number of rows you have in the matrix, right? So- so Ax R\_m. You can write this as m by one but, you know, um, we assume vectors are column vectors. So if I write just, you know, R\_m, you know, just assume it's- it's um, m by 1 if you want to think of it that way. Another interpretation of, um, matrix-vector, um, multiplication is- it's the same matrix, except you're gonna think of them as columns. Okay? m by n. This is n, so this is A and this is x. So this has- I'm just gonna use different symbols. Let's assume, you know, there are three elements, um, and all right this has four. So let's- let's assume this has four elements, and there, um, and this four, A is m by 4 matrix and you're trying to, uh, calculate Ax. The other interpretation here is, um, let me use some colors. So here we did inner products. Right? Here, what we're gonna do is pick the first element of- of x, scale the first column of a. And then pick the second element of x, scale the second column of A by that number. By scaling, I mean, do an element-wise multiplication of- of this number with all the elements here, not just rescale it. And pick the third element of x, scale the third column. Right? And- and then you just, um, um, sum them up along this way. So sum up the scaled versions. All right? So Ax is again, you know, you're summing up, uh, uh, m-dimensional column vectors that are scaled by the corresponding values in the vector. And that's just gonna give you, you know. And whether you do the operation this way or this way, you'll always end up with the same- same right hand side. Right? So that's matrix-vector.

#### Matrix by Matrix

k

n

k

m

Um, and then you have matrix-matrix. [NOISE] So again, for matrix- matrix, there are, kind of, you know, two interpretations of how you, kind of, uh, visualize the product of two matrices. So let's assume you have, uh, two matrices, matrices m by k times k by n. So it is important that the- the number of columns of the first one and the number of rows of the second one are the same, right? And in interpretation number 1, we'll go with the inner product interpretation. So you have m rows of k dimensions each and n columns of k dimensions each, right? The dimensions are matching of, uh, the row and the column. So naturally, we want to do the inner product, right? Now, take every pair of row and column vector. Um, so that- that gives you m times n. So, um, any- any row vector, any column vector perform the inner product. And that becomes the cell value of the corresponding ith row and jth column, right? So pick the, let's say, the second row and the fourth column, do the inner product, and that is the second row, fourth column of- of- of your output matrix. So matrix multiplication, essentially, a whole bunch of inner products that you are doing concurrently in parallel. You can also, um, do the same operation, m by k, k by n. And in this case, think of them as- [NOISE] right? So now we have k column vectors and k row vectors, right? M and n are different, uh, so this is m-dimensional and this is n-dimensional, but we have k of each of them, right? Now, pick them pairwise, not all possible pairs, but pick them pairwise. So first row and first column. I'm sorry, first column and first row. Do the outer product, you get a rank 1 matrix, right? So you get a rank 1 matrix of- so this is column 1 from the first matrix and row 1. Plus, pick the second column and second row, column 1 and, uh, row- column 2 and row 2, right? And add k of them, so column k and row k. So the columns come from the left matrix and the rows come from the right matrix. Um, and you add them up, you get another matrix, right? And the matrix that you calculate this way- all the matrix that you calculate this way are going to be exactly the same. So they're just two different interpretations of the same mathematical operation. [NOISE] Any questions so far? Yeah. [inaudible] Yep, so, uh, the question is, uh, should this be a rank 1 matrix? Uh, so the- the, uh- when you take, ah, ro- a column vector and a row vector and take the outer product, this will be a rank 1 matrix, and this is another separate rank 1 matrix, and this is another rank one matrix. And when you add them up, when you add k of them, you get a potentially, you know, uh, min of, you know, m, n, and k rank matrix. [inaudible]. It will, uh- if- if the rows and columns are linearly independent, it will not be rank 1. You're increasing the rank as- as long as you're taking linearly independent rows and columns. But you cannot- you cannot keep adding, uh, matrices, rank 1 matrices, and increase your rank forever. You have an upper bound which is the smaller of the two dimensions of the row or column. All right, so, um, another question of what the rank was?

#### Why Do We Need To Learn Linear Algebra For Machine Learning?

And before we get to that, um, let's see. Why do we need to learn linear algebra for machine learning? Why is linear algebra even relevant for machine learning? So, uh, some of the, um- some of the situations where we will end up using linear algebra through, you know, um- through the rest of the quarter is first is, um, represent data.

##### Represent Data

So supposing a supervised learning setting, um, you're, let's say, you want to predict, uh, the price of the house and the inputs that you're given are things like, you know, the number of bathrooms, number of bedrooms, you know, the area, etc. Um, you would then represent your training data as x, as, you know. Where each row x's is you- it's also called the design matrix, where each row is a- an example is, a dis- different example. And the columns corresponds to what we also called as features. Uh, for example, uh, this could be house 1, house 2, house 3, house 4, etc. And this could be, say, number of bedrooms area, number of bathrooms, etc. Um, and this is generally called a design matrix, right? So your design matrix x, um, is most conveniently represented as a matrix. Similarly, your supervision values, the actual house prices of those houses, is most conveniently represented as a vector, right? So just to represent your data, it is convenient to use concepts from linear algebra like matrices and vectors, okay? And then we're going to do a whole lot of manipulation with these, like, you know, multiply this, you know, uh, x vector with, you know, uh, another parameter vector called Theta, and so on, and so on. So we are going to be using linear algebra for representing our data and also doing operations on them.

##### Covariance Matrices

S **∈ Rd×d**

Another, um, situations where we will need, um, matrices, uh, is going to be in- in probability to represent what's called covariance matrices. We'll cover probability with you tomorrow, and we will, uh, go over what a covariance matrix is. So for example, covariance matrices, um, they're generally written as, you know- they tend to be symmetric. Um, symmetric, uh, means the matrix and its transpose are the same, right?

##### Calculus

* Gradient – Vector
* Hessian – Matrix (Symmetric)
* Jacobians - Matrix

And we're- we're gonna use, uh, linear algebra for- for calculus, multivariate calculus, right? So, uh, things like, um, gradients, so gradients and, um, um- I'm expecting you know what a gradient is. So, um, um, you um- gradients tend to be vectors, right? So you, you would represent vectors as, as, um, um, column vectors. Hessians, Hessians are matrices. Hessians are basically like, you know, think of them as the second derivative in the multivariate setting. Um, uh, and- and, um Jacobians, Jacobians are derivatives of vector valued output from, you know, a vector value input. Um, all of those can be, um, uh, let's see, gradients, maybe vectors. Hessians matr- matrix, symmetric, Jacobian's, again, a matrix. And Jacobian will, um- will likely not be symmetric, and so forth.

##### Kernel Methods

And we're also gonna see, uh- use linear algebra very literally in- in kernel methods. You're not expected to know what a kernel method is. I'm just listing it here. Uh, um, just so, uh, you know, that, you know, uh, learning linear algebra has uses in all these, um, um, various scenarios, right? And by calculus, I also mean, you know, in general, optimization. You know, given a loss function, uh, we need to minimize it, right? And, and, uh, we're gonna use, uh, gradients and Hessians and, uh- and, um so- so on there, right? So, yeah. So the- uh, so linear algebra is- is is important, and it's very important that you are comfortable manipulating these concepts such as matrices, and vectors, and, you now, multiplying them, taking inverses, etc. Any questions, uh, before we move on to a few more interesting concepts. Uh- [NOISE] No questions? [NOISE] So we saw matrix-vector multiplication over here, right?

#### Geometrical Intepretation

|  |  |
| --- | --- |
| **A ∈ Rm×n**  **x ∈ Rn** | **A(x) ∈ Rm** |

|  |  |  |
| --- | --- | --- |
| z  y  x  Input Space | **A ∈ R3×3**  **B = A-1 R3×3**  FULL RANK | z  y  x  Output Space |

So here's the- let's do a geometrical interpretation of- [NOISE] so linear algebra is one field of mathematics where it's very easy to visualize things which makes it really fun, and- and also kind of easy once you kind of get the core intuitions, right? So we saw the matrix-vector multiplication. Uh, so A, let's- let's call a matrix as A, it's in [NOISE] R m by n. And let's say you have a vector x in Rn, right? Now, when you do a matrix-vector multiplication, you get something called Ax, right? And this is in [NOISE] Rm, right? So one way to- to think of Ax is- is through this, you know, uh, row interpretation or column interpretation, right? But, uh, an even more useful way to think about this is to think of A as some kind of a function, right? Think of A as a function. It's, you know- we- we've been looking at the matrix as a set of numbers, right? So stop thinking of it as a set of numbers, but think of it as a function, a function that takes x as input and outputs, you know, that vector. Now, let's assume you have- we have A in [NOISE] R 3 by 3, which means we have, uh- it takes as input values from a three-dimensional space, and its output is also a three-dimensional space, right? Now we're thinking of A as a function, right? Who- which takes as input- the inputs are vectors, and the outputs are also vectors, right? So x is a vector, h is also a vector. They are in different dimensions, yes? Right? So it takes as input an n-dimensional vector and produces an output which is an m-dimensional vector, right? So let's pick some- so assume this is the A which-which-which, uh, does the multiplication. And [NOISE] so these are- let's call them x-axis, y-axis, z-axis, [NOISE] is 3 by 3. And let's say we- so this is a vector in, you know- this is some x. It lives in a three-dimensional space, and it has, you know, a- a- a- a column representation, right? Multiplied by A, right? And let's say it maps it here, in the output space, right? Now, let's take another vector, let's call it- now this is some other vector, run it through A, it's gonna give you a different output, right? So perhaps it comes here. And similarly, you know, take a- a- a third- third vector, right, through A, it's probably gonna come here, right? So the, uh- what this picture is representing is you have this function A, right? We don't think of it as, you know, numbers anymore. We think of it as a function for which we feed- we- we take some point in the input space as the input, feed it into the function, and it outputs, you know, a- a new vector in- in a diff, you know - possibly a different dimensional space. And in- in this case, it's 3 and 3 to make the visualizations simple, right? Now, if A is full rank, which means its rank is, uh- is 3, in this case, right? There is going to be a unique mapping between every point in the input to every point in the output. There's gonna be a one, you know- a one-by-one mapping, one-to-one mapping between every point in the input to a corresponding point in the output, right? Now, if A is such a matrix that makes such a one-to-one mapping possible, A is a full rank matrix, right? Now, uh, so this is input space, [NOISE] and this is the output space. [NOISE] Right? Now, you could also imagine that there exists some other matrix, call it B, that takes these values as the input, right? Say the red, uh, uh, as the input, run it through B, and it outputs the original point over here, right? Now, if A is a full rank matrix, then such a B will exist, you know, which, which does the reverse mapping, and this B is also called A inverse. Right? So A is the function that takes- that takes vectors as input and maps them to some output. And as long as it is full rank, there will be- there will be another matrix, which is the inverse of it, which takes this as the input and corresponds the- and- and outputs, the original point as- as the output. Right? Now, this is in the case of a full rank matrix.

|  |  |  |
| --- | --- | --- |
| z  y  x  Row Space | **A ∈ R3×3**  Rank-2  **A(x) = A(xR + xN)**  **= A(xR) + A(xn)} = 0**  **= A(xR)**  RANK DEFICIENT | z  y  x  Column Space |

**x = Proj (xi Row Space) + Proj (xj NullSpace)**

Now, what happens if- if, uh, A is not full rank? So this was [NOISE] full rank. Now here, if a matrix is not full rank, it's technically called rank deficient. [NOISE] Right? So again, this is the input space, x-axis, y-axis, z-axis. Similarly, there is an output space x-axis, y-axis, z-axis, right? And we have an A over here. The A is, again, in R 3 by 3. [NOISE] But let's assume this is a rank 2 matrix, right? [NOISE] Right? And this was a full rank matrix. This was a rank 3 matrix, so it could uniquely map every point to every other point, and you could go back through the inverse. You know, now, if it is, uh, rank deficient, what this means is if the rank is 2, it means there exists, uh, a two-dimensional subspace in the input space, you know. Okay. So there exists a- a two-dimensional subspace. What does a subspace mean? So assume this is the ambient space, is three-dimensional. Assume this room to be the ambient three-dimensional space. Any two-dimensional space, for example, this paper that extends indefinitely in all- along all directions, is a subspace that lives in this three-dimensional space, right? Now, if- if the rank of the matrix is- is two, you know, and it is a- a- a- a three by three matrix, then in this three-dimensional ambient space, there exists a two-dimensional subspace that is specific to this matrix, and that subspace must pass through the origin, right? And also, a corresponding subspace in the output space, which is also two-dimensional, which is what makes it, you know, a rank 2 matrix. There is a two-dimensional subspace and a two-dimensional subspace. [NOISE] Again, this also has to go through the- through the origin. And there exists a one-to-one mapping between points on this subspace to points on this subspace here. [NOISE] So a point on this subspace may get mapped here, another point over here may get mapped here. Similarly, you know, a third point here may get mapped here. Now, this- this subspace extends to infinity in all directions, right? In both- in both the input and out- out- output space. And A gives you a unique one-to-one mapping between elements in these subspaces. But A is a three by three matrix, right? It's- it's a function, um, think of it as a function for which you can feed any input. You know, just because a subspace exists for which a one-to-one, uh, uh, mapping exists, doesn't mean, you know, for example, I may take this point that lives outside the subspace, right? This is, uh- you know, I think of this as the ambient three-dimensional space, this is a two-dimensional subspace. What about the point over here where the pen is at? It's not living in the subspace, it's outside the subspace, right? Take- take such a point that, you know, uh, that doesn't live in the subspace. You can still feed this point as an input to this function and it's gonna map it somewhere, right? Where- where- but where will it map it to, right? The way to think about that is any given point in the input, you know, um- in the input space of a matrix can be decomposed, right? You can decompose x [NOISE] into two parts. One part [NOISE] Let's assume this is x, and x lives outside the subspace. You can decompose this x into two parts, where one part lives in the subspace and is in a way nearest to the given x, and another part, which is this residual from that nearest point to the actual point, right? Which means assume the origin is in that corner and this is a two-dimensional subspace that, you know, passes through the origin and we have a point over here, this point can be decomposed into two part- into two parts. This point is represented by a vector that starts at the origin and ends at, you know, this point. You can decompose it into two parts where one part lives strictly in the subspace, and it extends all the way till it is s- nearest to this point, and a second part, which is a point that is perpendicular to the subspace that originates from that nearest point to this point, [NOISE] right? It's like, you know, Pythagoras Theorem. You have two- two components, which are at right angles, and any vector can be decomposed into a component that lies in the subspace and a component that is strictly perpendicular to the subspace, [NOISE] right? And- I'm sorry, what's the question? [inaudible]. Yes, you- you- you can call it a- a- a resolution but a- a more commonly used term is decomposition. So you decompose that vector into two parts, right? And these two parts, when I mean decomposition, I don't mean take the vector, you know, the- and- and split it into, you know, a- a two-dimensional versus a three-dimensional, um, um, you know- What I mean is, you know- you should not be confusing this- [NOISE] this decomposition as splitting it into [NOISE] one part that lives in, uh, uh, the subspace versus another. That's not what we're talking about, right? What we are talking about is, you know, x [NOISE] uh, can be written as- [NOISE] I'm gonna use the word projection over here. [NOISE] So the projection of x onto the subspace [NOISE] plus the- the technical term over here would be the row space [NOISE] plus projection onto the null space. [NOISE] Now, what does that mean? I mean we saw two terms over there, a row space and a null space. Now, row space is actually the subspace which- which gets mapped bijectively to the corresponding, you know, the output subspace, right? This subspace in the input space, right, for which the bijection exist is also called the row space. And it's called the row space because it is made up of precisely those points which can be represented as linear combination of the rows of A, right? Yes. Question. Are all predictions orthogonal? Are all predictions orthogonal? [NOISE] I'll come to that in a moment. Yes. Um, so, um, the row space is made up of all points that- that can be represented as linear combinations of the rows of A, and it is precisely that subspace for which a bijection exists between, [NOISE] you know, the- the output space and- and- and the, er, er, input space. So this is called the row space, and this is called the column space, right? It's also called like, um, um, um, the range of, um, um- it's also called a range or you can, er, think of it as a columns. And this is precisely [NOISE] those- the set of all points which can be represented as linear combinations of the columns of A, right? Now, the- the point over here, which did not exist in the row space, we said, can be represented as the sum of two components. We can decompose it as the sum of two components. One component that lies in the row space and is nearest to x, and another component, which is- you can call it, you know, what's left of the residue, right, which is orthogonal to, uh, uh, your row space. And the question was, is the projection always perpendicular to the space onto which you're projecting it to? Yes, the answer is yes. [NOISE] So the way you wanna think of it is- let's assume this is the subspace. [NOISE] This is a two-dimensional subspace in this ambient three-dimensional space, and you have some point over here, you know, where this, you know, a black pen is, and you wanna project it onto the space- onto the subspace. Now, [NOISE] the- the- the way you do the projection is search all the points that exist in the subspace, calculate the distance from backspace- from that point to this point, and choose the point that- that has the smallest distance, right? [NOISE] That point is the projection of, um, um, the- the- the point having the smallest distance from this point is the projection of this point onto the subspace. And the- the- the line connecting the true point to the projection point is always gonna meet at an angle of 90 degrees. It's- it's always gonna be perpendicular to the subspace itself, [NOISE] right? It's- it's kind of intuitive right? [NOISE] So, um, um, in this [NOISE] subspace, the point that is nearest on this- on this plane to this point is gonna be the one that is like directly below it and so that line is always gonna be, you know, perpendicular. [NOISE] Does- does that answer your question? Okay. [NOISE] Yes. Question. So what is the subspace? Is it possible to [inaudible] origin [inaudible] Subspaces will always pass through the origin. [NOISE] By definition, they pass through the origin because, um, this is always gonna be some linear combination of the row or the columns, and the linear combination can be just 0s, you know, just 0 times Column 1 plus 0 times Column 2. So the origin is always part of both the input [NOISE] subspace and the output subspace. [NOISE] Yes. Question. [NOISE] [inaudible] It is possible [inaudible]? Uh, I don't know [inaudible] Yeah. So the question is what is row space? So row space is precisely the set of all points that can be represented as linear columns of the rows, [NOISE] right? So column space is the set of all points that can be represented as linear combination of the columns of- of A, which means the set of all points that you can obtain by multiplying some vector with that matrix, right? And according to this definition, it is some linear combination of the columns [NOISE] of- of- of A and, you know, um, you take the transpose and you get the row space. Yes. Question. Uh, do you see that these subspaces are, uh, [inaudible] actually [inaudible]? The question is are the subspaces- yeah, yeah [OVERLAPPING] [inaudible] are also [inaudible] A [NOISE] So the question is, are the points of different colors, are they the rows or columns of the matrix A? Is that the question? So, uh, good question. Thank you for asking that question. So the points that I'm choosing over here, right, have nothing to do with. In the sense, they need not be either a row or a column of A, right? The points that lie on this subspace can be represented as some linear combination of the rows of A, right? What it means is one of the- so if A has, has, has three rows, there will be three points in the subspace which corresponds to the three rows of A. It has to, it has to lie there because, you know, um, um, what I mean by that is, let's assume A has three rows. This is A, it has three rows, and this is the subspace obtained by linear combinations of these three rows. Which means one of the combinations could be 1, 0, 0, right? 1 times the first row plus 0 times the second row plus 0 times the third row must lie in the subspace, right? Which means first row exists at some point in the subspace. Does that make sense? Yeah, so, um, yeah, so that's, that's the, um, um, row space and any given x. Now, what happens, you know, back to the question of what happens if you take some x that does not exist in the, in the, uh, row space of x, and you multiply it through- and multiply it uh, by A, what happens? And this is where the, the decomposition, you know, helps us find an answer. So A of x is gonna be A of- I'm gonna abuse notation and write it as x\_R plus x\_N, where it is- this is the projection of x onto the row space, and this is the residue, right? And this the direction in which the residue lives is also called the nullspace of A. Which means any point that lives in the nullspace of A will always get mapped to zero in the output space, that's called the nullspace of A. So any point in the input space can be broken down into two components. A component that lives in the row space and a component that lives in the nullspace, right? And A of x, you know, you can represent it as, you know, the row space component plus the nullspace component. And because A is a linear function, you can write this A of x\_R plus A of x\_N, right? And this is 0 because, you know, you're feeding this, this is, uh, a vector in the nullspace, you multiply it by A, it always goes to 0, right? And so this is just A of x\_R. So the, the operation you wanna think in your mind is given any point in the input space which may or may not lie in the row space. When you multiply it by- multiply it with A, what happens is effectively, you project that space onto the row space and transfer that, you know, and, and, and find the corresponding output of that projection, right? So it, it, it, it essentially means, um, this operation is effectively just pruning out anything that's outside the row space of, of, you know, from, from the input. Find the projection onto the row space and, and, and carry forward that point onto the output space, right? So this concept of projection is super important. You're gonna, um, you're gonna come across this concept- this, this projection again when we, when we, uh, do linear regression, right?

##### Projection

b

v

=

(m×n) (n×n) (n×m)

So what's, what's, uh, technically how do we calculate the projection? Right here, I just represented as some abstract function called projection. And here I used, you know, the, the resulting vector after you do the projection. But how do you actually calculate the protection, right? And that's not too hard either so, so projection. We'll start with the simplest case. The simplest case where you have consider you have a vector. Let's call this V some vector. And you have another vector, lets call it B. There are two different vectors. And now we want to project B onto the direction- onto the subspace spanned by V, right? V is a vector, and it induces a subspace and that subspace is a one-dimensional subspace which is made up of all linear combinations of the vector B, which means take B, multiply it by 2, you get- you take V, you know, 2V is part of that subspace, 1.5V is part of that subspace, minus V is part of that subspace. So this entire line is the subspace spanned by V, where the vector V, And now we want to project the vector B onto the subspace, right? And how do we do that? For that there is, um, for, for a given V, there is something called the projection matrix. So the projection matrix of V is V, V transpose over V transpose V. We decipher this in, in a minute. Now this is a projection matrix, which means you take this matrix and take any other vector. It could be this vector, it could be this vector, right? And if you multiply B by this matrix, the resulting point is gonna be the projection of, projection of, of, of B onto that- onto the subspace spanned by V. Now why is that the case? That's the case because let's look at what's happening here. [NOISE] I'm gonna rewrite this as so V, V transpose B over V transpose V. And this is the same as V over normal V, V, normal V transpose B, right? So V transpose V is the same as the square of the length of V. It's the, it's the norm squared of V. And that- um, so this is the square of the length. I distribute one of the length to this and one of the length to that, right? So when you divide a vector by its length, you are effectively rescaling it into a unit length vector, right? So now I'm going to call this, uh, say V tilde, V tilde transpose B. Where V tilde is a vector along V that has length equal to one. Maybe this point over here is V tilde, right? You can, you can pick any vector along this, along, along, on the subspace. And it's always going to result in the same projection matrix because you are rescaling it by the length. So it doesn't matter which, which specific vector you chose in the subspace. We rescale it by the length comeback to V tilde. And the projection matrix is V tilde, V tilde transpose. Now, this [NOISE] I'm going to this write it as V tilde times V tilde transpose B, right? And this we know is the- when, when you take the inner product between a unit length vector and any other vector, [NOISE] it's gonna give you the, the, the length of the projection of this vector onto a unit length vector. [NOISE] So the- if V transpose is, uh, V tilde is, is, is some unit length vector, you take any other len, uh, uh, vector B, the inner, the inner product between two vectors where one of them is unit length, is going to be the magnitude of the other vector along the direction of, of the unit length vector, right? So this is gonna be the length of the projection of B along the dimension of V. And then this is just some scalar. You're re-scaling V tilde by some scalar again. Okay? So this, this is the projection matrix [NOISE] of some vector V, where the projection matrix projects the input that is fed into the projection matrix onto the subspace spanned by it. Is this clear? Any questions? Now, what happens if instead of one vector V, V give you, V, V have a collection of vectors, a bunch of column vectors. For example, we have, let's say, you know, we have a subspace spanned by the a collection of three row vectors. Now, if these are the vectors onto which, uh, uh, whose subspace onto which we want to, um, project X. We follow something very similar to this. In space, in- in place of V, instead of this being a vector, we will have a matrix where the matrix here is- will be some matrix X, whose columns are vectors which make up the subspace onto which we wanna project it. Right? In this case, it- it was a vector, it- it was subspace with just one vector. But if we want to project B onto a subspace spanned by multiple vectors, right? Then in place of V, we're going to use some matrix X, whose columns are precisely those, uh, you know, uh, set of vectors which make up the subspace. Yes, question? [inaudible] They need to be linearly independent by definition, uh, because those are, you know the, um- um. To make up a subspace. You know, you need a set of linearly independent vectors, right? So, um, take- take, uh, the- s, a- se- some set of linearly independent vectors from that su- subspace, right? And wherever there is V, just plug in X, so we're gonna get X. [NOISE] This looks different from this. And that's because, in V transpose V was a scalar and you could- you can divide something by a scalar, right? But X transpose X is-is going to be a matrix and you cannot divide something by a matrix. It- it- it's not even a meaningful operation. How do you even go about doing it, right? So the- the- the right way to think about this is to rewrite, um, this as V [NOISE]. Right? So I just rewrote this as, um, V- V transpose, V inverse V transpose, which is, uh, which is the same. And it is this form that can generalize into a- a matrix form, right? So the projection matrix for a set of columns is-is this. And we are going to come again, come- come across this concept again when- when we do a linear regression. So, um- um-um, remember that. Yes. [inaudible] I could've put it anywhere. You could have put it anywhere Yeah. [inaudible] So the question is, um, why did we put the X transpose X inverse in the middle rather than to the left or to the right, right? So, um, it's- I mean, the-the answer is- is, um, is- is, I don't know if there's a satisfactory answer, but this is the only thing that makes sense. If you know other, uh, position will match the dimensions, right? So X is m by n, x transpose X, or x transpose x inverse is n by n, and X transpose is n by m. So in a way that's the only place where it kind of fits. In order to kind of multiply things. I know that's not a satisfactory answer, but that's the only operation that kind of makes sense. Right? So this, this, this, um, this concept of what happens when you're [NOISE] this whole idea of-of projecting things and- and finding a point for which, you know, uh, uh, bijections exist and moving over is-is-is going to be a recurring theme. It's- it's- it's something you wanna kind of understand really well, right? Any given, uh, for-for a given matrix A which is not full rank, there exists subspaces in its input and output space. And the-the- the dimension of the subspace is going to be equal to the rank of the matrix. And- and whenever you multiply a vector from the input space, what you're effectively doing is projecting that vector onto the row space that the input subspace for which a bijection exists, [NOISE] and then carry forward that point onto the out- output, um-um dimension. Which also means that now if we have a point that lives outside the subspace in the- in the output space. And we want to find out what point in the input space will give me this as the output. Which is effectively we are asking the question, what is A inverse, right? That does not exist there, for no point in the input space, can we reach a point that lives outside the-the-the column space. Right? This point is unreachable. And-and where, um, in, when we- when we- when we, uh, cover linear regression, we're gonna ask exactly this question. When we have a point that is not unreachable, you know, what are we gonna do about it? How are we going to find an input that, you know, that's- that's basically linear regression, which we'll go on- on- on Friday. But this- this whole idea of- of subspaces, you know, row space, null space decomposition, um, bijection, these are things you want to- you want to kind of digest it well, you know, and absorb them really well. Let's see. We have about eight more minutes, all right. So, uh, we might be able to squeeze in this one. [NOISE] We might run over a little over time today, uh, but I'll try to wrap up as much as possible and we can continue the rest.

##### Eigenvectors

z

y

x

A **∈ R3×3**

All right, so here we saw, um, kind of a visual representation of what A, the matrix A does to the input and output. Uh, now, next we are going to limit ourselves to only square matrices, which means the input and output spaces have the same dimensions, and for the purpose of visualization, we're gonna only consider, let's say, a three-dimensional space. Right? Now, in this diagram we have two different, um, um, um, two different pictures for the input space and the output space. But now, because A is symmetric, I'm sorry, A is- is, um, a square matrix, we're gonna overlay the input and output space onto the same space, right? So here, the input space and output space are- are being overlaid here. Now let's ask the question- let's, you know, A, you know, let's ask the question. We saw what happened, uh, for, you know, pick, you know, choose some points, you know, run it through A, you get an output, uh, uh, output point. Now, what happens if we take the unit sphere around the origin? By unit sphere, I mean, just to the points on the surface of the unit sphere. Think of it as a soccer ball at the center on the origin. And you take every point on the surface, run it through A, you get a corresponding output point for every, you know, input point of- of- of the soccer ball, right? How would the resulting shape look? Right? So that's gonna look as an ellipsoid. It's- it's, you know, almost like an ellipse. So that's gonna be- right? So what- what- what exactly happened here? We- it's- it's a three-dimensional input and output space. We started with the input as- we didn't have one input, but we had a collection of points as inputs. And that collection was precisely those points that live on the surface of a unit sphere. And let's say we- we took this point in the input space, run it through A, we got a corresponding out point, and that point, say, this one. Right? So every point on the input surface maps to some point on the output surface of that shape, right? We could have done this with any input shape, but, you know, sphere is easy to kind of analyze, right? Now, similarly, um, you do it for another point, um, let's say this point, and let's say that maps here, and let's say we pick- what color do we have? Green- green, and let's say we pick this point and that maps here. Okay? Now, we saw what- what- what- what happens when you- when we, uh, take some shape, for example, a sphere and run it through A, instead of thinking of um, um, running a point through A. Now imagine running a full- you know, a full- a full shape through A, which- which essentially means you are running every point in that shape through A and calculating the output point which, you know, which would be some other shape. So now think of A as- as- as taking as a shape as input and outputting a sh- a different shape as the output. If the shape is some unit sphere around, is- is the unit sphere around the origin, the output is going to be an ellipsoid. Right? Now, this ellipsoid, assuming A is also symmetric, right? We'll have, essentially there is two things. So first of all, so this was the input, this was the output, this is one, which means there were two operations being done here. One is- um, there is a change in magnitude, so the magnitude came down a little bit and there is a change in direction, right? A did essentially like, you know, effectively um, um, uh, two things, a change in direction and a change in magnitude. Now, there are going to be some points [NOISE] along, you know, um, uh, some point on the unit sphere for which the only change is gonna be a change in magnitude and no change in direction, [NOISE] right? And those are your eigenvectors. [NOISE] For uh, a full rank three by three matrix, there are gonna be three eigenvectors, right? And this is gonna be one of them, and if it is symmetric, those eigenvectors are gonna be perpendicular. Sorry, I used a wrong color I shouldn't have used. That's it. So this is going to be one eigenvector and there's going to be another eigenvector, which will map this point on the input to a point on the output where the- the only thing that changes is the magnitude and not the direction. Yes. Question. [inaudible] I'm sorry. [inaudible] So, does it necessarily have to have, uh, three eigenvectors? So it will have three eigenvectors. However, some of the eigenvalues may be 0. I'm going to come to- come to it, right? So- so this is going to be, you know, eigenvector number one for which the scaling is the maximum. Right? And the eigenvalue is the- the- the ratio of the length of, uh, the input vector and- and the length of the output vector. So if the output vector is- is- is stretched a lot along the eigen- eigenvector, then you have a large eigenvalue, right? And if- if-, um, if the eigenvector is- is shrinking the point closer to the origin, then the eigenvalue is, you know, less than 1. If it is bigger than, um, um, the origin point is greater than 1. Your- your point may also get reversed in direction, in which case your eigenvector is negative. Right? Now, that's- that's the eigenvector, and, um, um, the eigenvalue is- is the ratio of the output to, uh, um, um, the input magnitude. It may so happen that when you take the- the unit sphere or the soccer ball as the input in field, as the output- the output may not be an ellipsoid, but it may be a flat ellipse, which means you lost your dimension. It got kind of smushed into a- a two-dimensional ellipse instead of having a rigid three-dimensional, um, um, ellipsoid shape.

##### Eigenvalues

Now that means the eigenvalue along the third eigenvector was 0. Right? So along- along- along the third eigenvector, you know, all the lengths kind of got mapped to 0. Doesn't make sense, right? So- so, um, now the- a question was asked earlier on, you know, what is the- what is the meaning of the rank of a matrix? The rank of a matrix is precisely equal to the number of non-zero eigenvalues, right? If the- if the matrix, um, so this is in- in- in a square matrix, um, setting. If a matrix has full rank, then it means a u- a unit sphere will always come out as some kind of an ellipsoid with the same number of dimensions. If you're rank deficit, the ellipsoid that comes out may have lost a few dimensions.