JEREMY KEPNER: All right, welcome.

Thank you so much for coming.

I'm Jeremy Kepner.

I'm a fellow at Lincoln Laboratory.

I lead the Supercomputing Center there, which means I have the privilege of working every day with pretty much everyone at MIT.

I think I have the best job at MIT because I get to help you all pursue your research dreams.

And as a result of that, I get an opportunity to see what a really wide range of folks are doing and observe patterns between what different folks are doing.

So with that, I'll get started.

This is meant to be some initial motivational material, why you should be interested in learning about this mathematics, this mathematics of big data and how it relates to machine learning and other really exciting topics.

It is a math course.

We will be going over a fair amount of math.

But we really work hard to make it very accessible to people.

So we start out with a really elementary mathematical concept here, probably one that hopefully most of you are familiar with.

It's the basic concept of a circle, right?

And I bring that up because many of us know many ways to state this mathematically, right?

It's all the points that are equal distance from a particular point.

There's other ways to describe it.

But this is a basic mathematical concept of a circle that many of us have grown up with.

But, of course, the other thing we know is that, right, this is the big idea.

Although I can write down an equation for circle, which is the equation for a perfect, ideal circle, we know that such things don't actually exist in nature.

There is no true perfect circle in nature.

Even this circle that we've drawn here, it has pixels.

If I zoomed in on it, if I zoomed in on it enough, it wouldn't look like a circle at all.

It would look like a series of blocks.

And so that approximation process, right, where we have a mathematical concept of an ideal circle, right, but we know that there are not really-- they don't really exist nature, but we understand that it is worthwhile to think about these mathematical ideals, manipulate them and then take the results of the manipulation back into the real world.

That's a really productive way to think about things and, really, the basis for a lot of what we do here at MIT.

This concept is essentially the basis of modern or ancient Western thought on mathematics.

If you remember your history courses, this concept of ideal shapes and ideal circles is the foundation of platonic mathematics some 2,500 years ago.

And at the time, though, that they were developing that concept, this idea that there are ideal shapes out there and that thinking about them and manipulating them was a more effective way to reason about the real world, there was a lot of skepticism.

You could imagine 2,500 years ago someone is walking around and saying, I believe there are these things called ideal circles and ideal squares and ideal shapes.

But they don't actually exist in nature.

That would probably not be well-received.

In fact, it was not well-received.

Many of those philosophers who were thinking about this were very negatively received.

And, in fact, if you want to learn about how negative the response was to this, I encourage you to go and read the Allegory of the Cave, which is essentially the story of these philosophers talking about how they're trying to bring the light of this knowledge to the broader world and how they essentially get killed because of it, because people don't want to see it.

So that struggle they experienced 2,500 years ago, it exists today.

You as people at MIT will try and bring mathematical concepts into environments where people are like, I don't see why that's relevant.

And you will experience negative inputs.

But you should rest assured that this is a good bet.

It's worked well for thousands of years.

You know, it's what I base my career on.

People ask me, well, what's the basis of it?

Well, I'm just betting on math here.

It's been a good tool.

So this is why we're beginning to think this way when we talk about big data and machine learning.

So really looking at the fundamentals, what are the ideals that we need in order to effectively reason about the problems that we're facing today in the virtual world, right, and the fact that this mathematical concept described the natural world so well and also described in the virtual world is sometimes called the unreasonable effectiveness of mathematics.

You can look that up.

But people talk about math.

Why does it do such a good job of describing so many things?

And people say, well, they don't really know.

But it seems to be a good bit of luck that it happens that way.

So circles, that gets us a certain way.

But in most of the fields that we work with, and I would say that, in almost any introductory course that you take in college, whatever the discipline is, whether it be chemistry or mechanical engineering or electrical engineering or physics or biology, the basic fundamental theoretical ideas that they will introduce to you will be the concept of a linear model.

So there we have a linear model, right?

And why do we like linear models?

And again, it can be physics.

It can be as simple as F = MA Or, in chemistry, it can be some kind of chemical rate equation.

Or in mechanical engineering it can be basic concepts of friction.

The reason we like these basic linear models is because we can project, right?

I know that if that solid line represents what I believe to-- you know, if I have evidence to support that that is correct, then I feel pretty good about projecting maybe where I don't have data or into a new domain.

So linear models allow us to do this reasoning.

And that's why in the first few weeks of almost any introductory course they begin with these linear models, because they have proven to be so effective.

Now, there are many non-linear phenomena that are tremendously important, OK?

And as a person who deals with large-scale computation, those are a staple of what people do.

But in order to do non-linear calculations or reason about things non-linearly, it usually requires a much more complicated analysis and much more computation, much more data.

And so our ability to extrapolate is very limited, OK?

It's very limited.

So here I am talking about the benefits of thinking mathematically, talking about linearity.

What does this have to do with big data and machine learning?

So we would like to be able to do the same things that we've been able to do in other fields in this new emerging field of big data.

And this often deals with data that doesn't look like the traditional measurements we see in science.

This can be data that has to do with words or images, pictures of people, other types of things that don't feel like the kinds of data that we traditionally deal with in science and engineering.

But we know we want to use linear models.

So how are we going to do that?

How can we take this concept of linearity, which has been so powerful across so many disciplines, and bring them to this field that just feels completely different than the kinds data that we have?

So to begin with, I need to refresh for you what it really means to be linear.

Before, I showed you a line and, hence, the line, linear.

But mathematically, linearity means something much deeper.

And so here's an equation that you may have first seen in elementary school.

We basically have to two times three plus four is equal to two times three plus two times four.

That is called the distributive property.

It basically says multiplication distributes over addition.

And this is the fundamental reason why I would say mathematics works in our world, right?

If this wasn't true very early on in the earliest days of inventing mathematics, it would not have been very useful, right?

To say that I have two of three plus four of something, OK, and then I can change it and do it in this other way, that's really what makes mathematics useful.

And from a deeper perspective, the distributive property is basically what makes math linear.

This is the property that, if this property holds, then we can reason about a system linearly.

Now, you're very familiar with this type of mathematics, but there's other types of mathematics.

So if you'll allow me, hopefully you will let me just replace those multiplication symbols and addition symbols with this funny circle times and circle plus.

And we'll get to why I'm going to do that.

Because it turns out that, while you have done most of your careers with traditional arithmetic multiplication and addition, the kind you would do on your calculator or have done in elementary school, it turns out there's other pairs of operations that also obey this property, this distributive property, and, therefore, allow us to potentially build linear models of very different types of data using this property.

So, as I mentioned, the classic two are circle plus is just equal to regular arithmetic addition, as we show on the first line, and circle times is equal to regular arithmetic multiplication.

So those are the standard ones.

And, by far, this pair, this is the most common pair that we use across the world today.

But there are others.

So, for instance, I can replace the plus operation with max and the multiplication operation with addition, OK?

And the above distributive equation will still hold, right?

That's a little confusing.

I often get confused that multiplications is now addition.

But this pair sometimes referred to as max plus-- you'll sometimes hear about it as max plus algebra-- is actually very important in machine learning and neural networks.

This is actually the back end of the rectified linear unit, is essentially this operation.

If you didn't understand what that meant, that's OK.

We'll get to that later.

It's very important in finance.

There are certain finance operations that rely on this type of mathematics.

There are other pairs, also.

So here's one.

I can replace addition with union and multiplication with intersection, right?

Now, that also obeys that linear property.

This is essentially the pair of operations that, anytime you make a transaction and work with what's called a relational database, that's the mathematical operation pair that's sitting inside it.

It's why those databases work.

It allows us to reason about queries, which are just a series of intersections and unions, and then reorder them in such a way.

In databases, this is called query planning.

And if that property wasn't true, we wouldn't be able to do that.

So this is a deep property of that.

So we can put all different types of pairs in here and reason about them linearly.

And this is why that many, many of the systems we use today work.

And so this class is about really exposing that, that, really, the mathematics that allows us to think linearly about data that we haven't really thought of as maybe obeying some kind of linear model.

This is essentially the critical point of this class.

So it goes beyond that, though.

So hopefully you'll allow me to replace those numbers with letters, right?

So that's basic algebra there.

Just for a refresher, the previous equation, we had A = 2, B = 3, C = 4.

But we're not limited to these variables, or these letters, to being just simple scalar numbers, in this case, real numbers or integers or something like that.

They can be other things, too.

So, for instance, A, B, and C could be spreadsheets.

And that's something we'll go over with extensively in a class, so that I can basically have A, B, and C be whole spreadsheets of data and the linear equation will still hold.

And, in fact, that's probably the key concept in big data, is the necessity to reason about data as whole collections and transforming whole collections.

Going and looking at things one element at a time is essentially the thing that is extremely difficult to do when you have large amounts of data.

A, B, and C can be database tables, right?

Those don't differ too much from spreadsheets.

And as I talked to you in the previous slide, that union/intersection pair naturally lines up and we can reason about whole tables in a database using linear properties.

They can be matrices.

I think, for those of you who have had a linear algebra and matrix mathematics, that would have been the first example, right, when I substituted the A, B, and C and had these linear equations.

Often, in many of the sciences, we think about matrix operations and linearity as being coupled together.

And through the duality between matrices and graphs and networks, we can represent graphs and networks through matrices.

Any time you work with a neural network, you're representing that network as a matrix.

And, of course, all these equations apply there as well and you can reason about those systems linearly.

So that provides a little motivation there.

As we like to say, enough about me, let me tell you about my book.

So this will be the text that will we use in the class.

We are not going to go through the full text, but we have printed out copies of the first seven chapters that we will go through.

And we will hand those out later when you do the class.

So let me now switch gears a little bit and talk about how this relates to, I think, one of the most wonderful breakthroughs that we have seen, or I've seen in my career, and many of my colleagues here at MIT have seen, which is what's been going on in machine learning, right, which is-- it's not hype.

There's a real real there there and it's tremendously exciting.

So let me give you a little history, basic history of this field.

So in a certain sense, before 2010, machine learning looked like this.

And then, after 2015, it kind of looks like this.

So when people talk about the hype in machine learning, or AI, really deep neural networks are the elephant inside the machine learning snake.

It has stormed onto the scene in the last five years and basically allowed us to do things that we had almost taken for granted were impossible.

Just the fact that you're able to talk to computers and they can understand you, that we can have computers that can see at least in a way that approximates the way humans do, these are really almost technological miracles that, for those of us who have been working on this field for fifty years, we had almost literally given up on.

And then all of a sudden it became possible.

So let me give you a little sense of appreciation for this field and its roots.

So machine learning, like any field, is defined as a set of techniques and problems.

When you ask what defines a field, you ask, well, what are the problems that they work on that other fields don't really work on?

And what are the techniques they employ that really are not really being employed by them?

So the core techniques, as I mentioned earlier, are these neural networks.

These are meant to crudely approximate maybe the way humans think about problems.

We have these circles which are neurons.

They have connections to other neurons.

You know, those connections have different weights associated with them.

As information comes in, they get multiplied by those weights.

They get summed together.

And if they pass certain thresholds or criteria, then they send a signal on to another neuron.

And this is, to a certain degree, how we believe the human brain works and is a natural starting point for, how could we make computers do similar things?

The big problems that people have worked on are these classic problems in machine learning, are language, how do we make computers understand human language, vision, how do we make computers see pictures or explain pictures back to us the way we would like, and strategy and games and other types of things like that.

So how do we get them to solve problems?

This is not new.

These core concepts trace back to the earliest days of the field.

In fact, these four figures here, each one is taken from a paper that was presented at the very first machine learning conference in the mid-1950s.

So there was a machine learning conference in the mid-1950s.

It was in Los Angeles.

It had four papers presented.

These were the four papers.

And I will say that three of them were done by folks at MIT Lincoln Laboratory, which is where I work.

And so that was basically the neural networks of language and vision.

And we didn't play games, so that was it.

And you might say, well, why is it?

Why was there so much work going on in Lincoln Laboratory in the mid-1950s that they would want to pioneer in these directions?

At that time, people were first building computers and computers were very special purpose.

So different organizations around the world were building computers to do different things.

Some were doing them to simulate complex fluid dynamics systems, think about designing ships or other types of things like that or airplanes.

Others were doing them to, say, like what Alan Turing was doing, break codes.

And our task was to help people who were watching radar scopes make decisions, right?

How could computers enable humans to watch more sensors and see where they're going?

How could we do that?

So at Lincoln Laboratory, we were building special purpose computers to do this.

And we built the first large computer with reliable, fast memory.

This system had 4,096 bytes of memory, which, at the time, people thought was too much.

What could you possibly do with 4,096 numbers?

The human brain, of course!

Right, that's enough, right?

Most of us can remember five, six, seven digits, right?

So a computer that can remember 4,096 numbers should be able to do things like language and vision and strategy.

So why not?

So they went out and they started working on these problems, OK?

But Lincoln Laboratory, being an applied research laboratory, we are required to get answers to our sponsors in a few years' time frame.

If problems are going to take longer than that, then they really are the purview of the basic research community, universities.

And it became apparent pretty early on that this problem was going to be more difficult.

It was not going to be solved right away.

So we did what we often do, is we partnered.

We found some bright young people at MIT, people just like yourselves.

In this case, we found a young professor named Marvin Minsky.

And we said, why don't you go and get some of your friends together and create a meeting where you can lay out what the fundamental challenges are of this field?

And then we will figure out how to get that funded so that you can go and do that research.

And that was the famous Dartmouth AI conference which kicked off the field.

And the person leading this group, Oliver Selfridge at Lincoln Laboratory, basically arranged for that conference to happen and then subsequently arranged for what would became the MIT AI Lab that was founded by Professor Minsky.

And likewise, Professor Selfridge also realized that we would need more computing power.

So he left Lincoln Laboratory and formed what was called Project MAC, which became the Laboratory for Computer Science.

And then those two entities later merged 30 years later to become CSAIL.

So that was the initial thing.

Now, it was pretty clear that, when this problem was handed off to the basic research community, there was a feeling that these problems would be solved in about a decade.

So we were really thinking by the mid-1960s is when these problems would be really solved.

So it's like giving someone an assignment, right?

You all are given assignments by professors and they give you a week to do it.

But it took a little longer.

In this case, it took five weeks or, in this case, five decades to solve this problem.

But we have.

We have now really, using those techniques, made tremendous progress on those problems.

But we don't know why it works.

So we made this tremendous progress but we don't really understand why this works.

So let me show you a little bit what we have learned, and this course will explore the deeper mathematics to help us gain insight.

We still don't know why it works.

At least we can lay the foundations and maybe you can figure it out.

So here I am, fifty years later, a person from Lincoln Laboratory saying, "All right.

Question one has been answered.

Here's question two."

Ha.

Why does this work and hopefully you can begin, be the generation figured it out.

Hopefully it'll take less than fifty years.

Historically this type once we know how it works, it usually takes about twenty years to figure out why.

So I mean impasses but maybe maybe you know some people are smarter and they'll figure it out faster.

So this is what a neural network looks like.

On the left you have your input, in this case, a vector, y zero.

It's just these dots that are called features.

What is a feature?

Anything can be a feature.

That is the power of neural networks, is they don't require you to *a priori* state what the inputs can be.

They can be anything.

People have said, well, you know, neural networks, machine learning, it's just curve fitting.

Yeah, but it's curve fitting without domain knowledge.

Because domain knowledge is so costly and expensive to create that having a general system that can do this is really what's so powerful.

So the inputs: we have a input feature.

It could be a vector, which we call y sub zero.

And that can just be an image, right, the canonical thing being an image of a cat, right?

And that can just be the pixels, values just rolled out into a vector, and they will be the inputs.

And then we have a series of layers.

These are called hidden layers.

The circles are often referred to as neurons, OK?

And each line connecting each dot has a value associated with it, a weight.

And the strength of the connection between any two neurons is given by that weight.

And then, ultimately, the output, in this case, the output classification, the series of blue dots there, are the different possible categories.

So if I put in a cat picture, one of those dots would be cat, maybe one would be dog, maybe one would be apple or orange, whatever I desired.

And the whole idea is that, if I put in a picture of a cat and I set all these values correctly, then the dot corresponding to cat will end up with the highest score, right?

And then I mentioned earlier that each one of these neurons collects inputs.

And if it's above a certain threshold, it then chooses to pass on information to the next.

And that's where these b values, which are vectors, are just the thresholds associated with each one of those.

It's a vector, one value associated with each one of those that does those.

This entire system can be represented relatively simply with one equation, which is that yi plus one, which is the next vector in the layer, OK, can be computed by the previous vector, yi matrix multiplied by the weight, W. So whenever you see transformations from one set of neurons to the next layer, you should think, oh, I have a matrix that represents all those weights and I'm going to multiply it by the vector to get the next one.

Then we apply these thresholds, all right?

So we add these, the bi's, and then we have a function that we pass it through.

Typically, this h function has been given the name rectified linear unit.

It's much simpler than that.

It's just, if the value is greater than what comes out of this matrix multiplied, if the value is greater than zero, don't touch it.

Just let it pass through.

If it's less than zero, make it zero, right?

You know, it's a pretty complicated name for a very simple function.

That's actually critical.

If you didn't have that h function, this nonlinear function there, then we could roll up all of these together and we would just have one big matrix equation, right?

So that's really considered a pretty important part of it.

So that's pretty much what's going on.

When you want to know what the big deal is of neural networks, that's all that's going on.

It's just that equation.

The challenge is we don't know what the W's and the b's are.

And we don't know how many layers there should be.

And we don't know how many neurons there should be in each layer.

And although the features can be arbitrary, picking the right ones do matter.

And picking the right categories do matter.

So when people talk about, I do machine learning or I'm off working on-- they're basically playing with all of these parameters to try and find the ones that will work best for their problem.

And there's a lot of trial and error.

And you'll hear about there's now systems that try and use machine learning to do that process automatically.

You know, how do you make machines that learn how to do machine learning?

The basic approach is a trial and error approach.

I take a whole bunch of pictures of cats that I now have cats in them, OK, and other things, right?

And I randomly set all those weights and thresholds.

And I put in the vector and I see what the system-- I guess what I think the number of layers and neurons and all that should be and I run it through the system and I get an estimate or a calculation for what I think these final values should be and I compare it with the truth.

That is, I just basically subtract it.

And then I use those corrections to very carefully adjust the weights.

Basically, with the last weights first, I do what's called back propagate these little changes to try and make a better guess on what these weights should be.

So if you hear the term back propagation, that's that process of taking those differences and using them to adjust these weights by about 0.01% at a time.

And then we just do this over and over again until eventually we get a set of weights that we think does the problem well enough for our purpose.

So that's called back propagation, all right?

Once we have the set of weights and we have a new picture that we want to know what it is, we drop it in there and it tells us it's a cat or a dog or whatever.

That forward step is called inference.

These are two words you'll hear frequently in machine learning, back propagation and inference.

And that's all there is to it.

There's really nothing else to that.

If you can understand this equation, you'll be way ahead of most people in machine learning, you know?

You know, there's lots of people who understand about all the software and the packages and the data.

All of them are just doing that.

And I'd say this is one of the most powerful ways to be ahead in your field, is to actually understand the mathematical principles.

Because then the software and what it's doing is much clearer.

And other people who don't understand these mathematical principles, they're really guessing.

They're like, oh, well, I do this and I throw this module in.

They don't really know that all it's doing is making adjustments to these various equations, how many different layers there are and stuff like that.

Now, why is this important?

You're like, well, what does it matter?

As I said before, we have this system.

It works but we don't know why.

Well, why is it important to know why?

Well, there's two reasons.

One is that, if we want to be able to apply this incredible innovation to other domains-- so many of you probably want to do that.

Many of you want to say, how can I apply machine learning to something else other than language or vision or some of these other standard problems?

I kind of need some theory to know.

Like, OK, if I have a problem that's like this one over here and I changed it in this way, there's a good chance it'll work.

There's some basis for why I'm going to try something, right?

Right now there's a lot of trial and error.

It's like, well, it's an idea.

But if you can have some math that says, you know, I think that will probably work, that really is a great way to guide your reasoning and guide your efforts.

Another reason is that-- so here's a picture of a very cute poodle, right?

And the machine learning system correctly identifies it as poodle.

One thing we realized is that the way you and I see that picture is actually very, very different than the way the neural network sees that picture, all right?

And, in fact, I can make changes to that picture that are imperceptible to you or me but will completely change how the neural network-- that is, given our neural network, I can basically make it think anything, right?

And so, for instance, this is a famous paper.

And they got the system to think that that was an ostrich, right?

And you can basically show this for anything, right?

So what's called robust AI, or robust machine learning, machine learning that can't be tricked, is going to become more and more important.

And again, having a deeper understanding of the theory is very, very critical of that.

So how are we going to do this?

What's the main concept that we are going to go through in this class?

This has mostly been motivational.

But how are we going to understand the data at a deeper level?

You know, what's the big idea?

And the big idea is captured now in this, I apologize for this eye chart slide, which is what we call declarative mathematically rigorous data.

So we have this mathematical concept called an associative array.

And it's corresponding algebra that basically encompasses the data you would put in databases, the data that you would put in graphs, the data that would put in matrices and it makes it all a linear system.

And the key operations are outlined there at the bottom.

If you recall, we have our basic little addition and multiplication.

And then what's going to be very important, probably the real workhorse for this-- and i didn't show it before-- is called essentially array multiplication or matrix multiplication.

And that's the far one on the right there, which we often abbreviate just with no symbol, just A B. But if we really want to explicitly call out that its matrix multiplication as a combination of both multiplication and addition, we put in what we call the punch-drunk emoji, which is a plus dot times.

You're probably all young enough that you don't even remember emojis when they had to type them out with just little characters and we didn't have icons, right?

So that meant you went to the bar and lost to the fight, right?

But, anyway, that's really going to be the workhorse of what we're doing here.