MICHALE FEE: OK, let's go ahead and get started. All right, so today, we're going to continue talking about feed-forward neural networks, and we're going to keep working on some interesting aspects of linear algebra-- matrix transformations. We're going to introduce a new idea from linear algebra, the idea of basis sets. We're going to describe some interesting and important properties of basis sets, such as linear independence. And then we're going to end with just a very simple formulation of how to change between different basis sets.

So let me explain a little bit more, motivate a little bit more why we're doing these things. So as people, as animals, looking out at the world, we are looking at high-dimensional data. We have hundreds of millions of photoreceptors in our retina. Those data get compressed down into about a million nerve fibers that go through our optic nerve up to our brain. So it's a very high-dimensional data set.

And then our brain unpacks that data and tries to make sense of it. And it does that by passing that data through layers of neural circuits that make transformations. And we've talked about how in going from one layer of neurons to another layer of neurons, there's a feed-forward projection that essentially does what looks like a matrix multiplication, OK? So that's one of the reasons why we're trying to understand what matrix multiplications do.

Now, we talked about some of the matrix transformations that you can see when you do a matrix multiplication. And one of those was a rotation. Matrix multiplications can implement rotations. And rotations are very important for visualizing high-dimensional data. So this is from a website at Google research, where they've implemented different viewers for high-dimensional data, ways of taking high-dimensional data and reducing the dimensionality and then visualizing what that data looks like.

And one of the most important ways that you visualize high-dimensional data is by rotating it and looking at it from different angles. And what you're doing when you do that is you take this high-dimensional data, you rotate it, and you project it into a plane, which is what you're seeing on the screen. And you can see that you get a lot out of looking at different projections and different rotations of data sets.

Also, when you're zooming in on the data, that's another matrix transformation. You can stretch and compress and do all sorts of different things to data. Now, one of the cool things is that when we study the brain to try to figure out how it does this really cool process of rotating data through its transformations that are produced by neural networks, we record from lots of neurons. There's technology now where you can image from thousands, or even tens of thousands, of neurons simultaneously. And again, it's this really high-dimensional data set that we're looking at to try to figure out how the brain works.

And so in order to analyze those data, we try to build programs or machines that act like the brain in order to understand the data that we collect from the brain. It's really cool. So it's kind of fun. As neuroscientists, we're trying to build a brain to analyze the data that we collect from the brain.

All right, so the cool thing is that the math that we're looking at right now and the kinds of neural networks that we're looking at right now are exactly the kinds of math and neural networks that you use to explain the brain and to look at data in very powerful ways, all right? So that's what we're trying to do. So let's start by coming back to our two-layer feed-forward network and looking in a little bit more detail about what it does.

OK, so I introduced the idea, this two-layer feed-forward network. We have an input layer that has a vector of firing rates, a firing rate that describes each of those input neurons, a vector of firing rates. That, again, is a list of numbers that describes the firing rate of each neuron in the output layer. And the connections between these two layers are a bunch of synapses, synaptic weights, that we can use to calculate to transform the firing rates at the input layer into the firing rates at the output layer.

So let's look in a little bit more detail now at what that collection of weights looks like. So we describe it as a matrix. That's called the weight matrix. The matrix has in it a number for the weight from each of the input neurons to each of the output neurons. The rows are a vector of weights onto each of the output neurons. And we'll see in a couple of slides that the columns are the set of weights from each input neuron to all the output neurons.

A row of this weight matrix is a vector of weights onto one of the output neurons. All right, so we can compute the firing rates of the neurons in our output layer for the case of linear neurons in the output layer simply as a matrix product of this weight vector times the vector of input firing rates. And that matrix multiplication gives us a vector that describes the firing rates of the output layer.

So let me just go through what that looks like. If we define a column vector of firing rates of each of the output neurons, we can write that as the weight matrix times the column vector of the firing rates of the input layer. We can calculate the firing rate of the first neuron in the output layer as the dot product of that row of the weight matrix with that vector of firing rates, OK? And that gives us the firing rate. v1 is then W of a equals 1 dot u.

That is one particular way of thinking about how you're calculating the firing rates in the output layer. And it's called the dot product interpretation of matrix multiplication, all right? Now, there's a different sort of complementary way of thinking about what happens when you do this matrix product that's also important to understand, because it's a different way of thinking about what's going on.

We can also think about the columns of this weight matrix. And we can think about the weight matrix as a collection of column vectors that we put together into matrix form. So in this particular network here, we can write down this weight matrix, all right? And you can see that this first input neuron connects to output neuron one, so there's a one there. The first input neuron connects to output neuron two, so there's a one there. The first input neuron does not connect to output neuron three, so there's a zero there, OK?

All right. So the columns of the weight matrix represent the pattern of projections from one of the input neurons to all of the output neurons. All right, so let's just take a look at what would happen if only one of our input neurons was active and all the others were silent. So this neuron is active. What would the output vector look like? What would the pattern of firing rates look like for the output neurons in this case?

Anybody? It's straightforward. It's not a trick question. [INAUDIBLE]?

AUDIENCE: So--

MICHALE FEE: If this neuron is firing and these weights are all one or zero.

AUDIENCE: The one neuron, a--

MICHALE FEE: Yes? This--

AUDIENCE: Yeah, [INAUDIBLE].

MICHALE FEE: --would fire, this would fire, and that would not fire, right? Good. So you can write that out as a matrix multiplication. So the firing rate vector, in this case, would be the dot product of this with this, this with this, and that with that. And what you would see is that the output firing rate vector would look like this first column of the weight matrix. So the output vector would look like 1, 1, 0 if only the first neuron were active.

So you can think of the output firing rate vector as being a contribution from neuron one-- and that contribution from neuron one is simply the first column of the weight matrix-- plus a contribution from neuron two, which is given by the second column of the weight matrix, and a contribution from input neuron three, which is given by the third column of the weight matrix, OK? So you can think of the output firing rate vector as being a linear combination of a contribution from the first neuron, a contribution from the second neuron, and a contribution from the third neuron. Does that make sense?

It's a different way of thinking about it. In the dot product interpretation, we're asking, what is the-- we're summing up all of the weights onto neuron one from those synapses. We're summing up all the weights onto neuron two from those synapses and summing up all the weights onto neuron three from those synapses. So we're doing it one output neuron at a time.

In this other interpretation of this matrix multiplication, we're doing something different. We're asking, what is the contribution to the output from one of the input neurons? What is the contribution to the output from another input neuron? And what is the contribution to the output from yet another input neuron? Does that makes sense? OK.

All right, so we have a linear combination of contributions from each of those input neurons. And that's called the outer product interpretation. I'm not going to explain right now why it's called that, but that's how that's referred to. So the output pattern is a linear combination of contributions.

OK, so let's take a look at the effect of some very simple feed-forward networks, OK? So let's just look at a few examples. So if we have a feed forward-- this is sort of the simplest feed-forward network. Each neuron in the input layer connects to one neuron in the output layer with a weight of one. So what is the weight matrix of this network?

AUDIENCE: Identity.

MICHALE FEE: It's the identity matrix. And so the firing rate of the output layer will be exactly the same as the firing rates in the input layer, OK? So there's the weight matrix, which is just the identity matrix, the firing rate. And the output layer is just the identity matrix times the firing rate of the input layer. And so that's equal to the input firing rate, OK?

All right, let's take a slightly more complex network, and let's make each one of those weights independent. They're not all just equal to one, but they're scaled by some constant-- lambda 1, lambda 2, and lambda 3. The weight matrix looks like this. It's a diagonal matrix, where each of those weights is on the diagonal. And in that case, you can see that the output firing rate is just this diagonal matrix times the input firing rate. And you can see that the output firing rate is just the input firing rate where each component of the input firing rate is scaled by some constant. Pretty straightforward.

Let's take a look at a case where the weight matrix now corresponds to a rotation matrix, OK? So we're going to let the weight matrix look like this rotation matrix that we talked about on Tuesday, where are the diagonal elements are cosine of sum rotation angle, and the off-diagonal elements are plus and minus sine of the rotation angle.

So you can see that this weight matrix corresponds to this network, where the projection from input neuron one to output neuron one is cosine phi. Input neuron two to output neuron two is cosine phi. And then these cross-connections are a plus and minus sine phi. OK, so what does that do?

So we can see that the output firing rate vector is just a product of this rotation matrix times the input firing rate vector. And you can write down each component like that. All right, so what does that do? So let's take a particular rotation angle. We're going to take a rotation angle of pi over 4, which is 45 degrees. That's what the weight matrix looks like. And we can do that multiplication to find that the output firing rate vector looks like-- one of the neurons has a firing rate that looks like the sum of the two input firing rates, and the other output neuron has a firing rate that looks like the difference between the two input firing rates.

And if you look at what this looks like in the space of firing rates of the input layer and the output layer, we can see what happens, OK? So what we'll often do when we look at the behavior of neural networks is we'll make a plot of the firing rates of the different neurons in the network. And what we'll often do for simple feed-forward networks, and we'll also do this for recurrent networks, is we'll plot the input firing rates as in the plane of u1 and u2.

And then we can plot the output firing rates in the same plane. So, for example, if we have an input state that looks like u1 equals u2, it will be some point on this diagonal line. We can then plot the output firing rate on this plane, v1 versus v2. And what will the output firing rate look like? What will the firing rate of v1 look like in this case?

AUDIENCE: [INAUDIBLE]

MICHALE FEE: Yeah let's say this is one and one. So what will the firing rate of this neuron look like? [INAUDIBLE]?

AUDIENCE: [INAUDIBLE]

MICHALE FEE: What's that?

AUDIENCE: [INAUDIBLE]

MICHALE FEE: So the firing rate of v1 is just this quantity right here, right? So it's u1 plus u2, right? So it's like 1 plus 1 over root 2. So it will be big. What will the firing rate of neuron v2 look like? It'll be u2 minus u1, which is?

AUDIENCE: Zero.

MICHALE FEE: Zero. So it will be over here, right? So it will be that input rotated by 45 degrees. And input down here-- so the firing rate of the one will be the sum of those two. Those two inputs are both negative. So v1 for this input will be big and negative. And v2 will be the difference of u1 and u2, which for anything on this line is?

AUDIENCE: Zero.

MICHALE FEE: Zero. OK. And so that input will be rotated over to here. So you can think of it this way-- any input in this space of u1 and u2, in the output will be just rotate by, in this case, it's minus 45 degrees. So that's clockwise, are the minus rotations. So you can just predict the output firing rates simply by taking the input firing rates in this plane and rotating them by minus 45 degrees.

All right, any questions about that? It's very simple. So this little neural network implements rotations of this input space. That's pretty cool. Why would you want a network to do rotations? Well, this solves exactly the problem that we were working on last time when we were talking about our perceptron, where we were trying to classify stimuli that could not be separated in one dimension, but rather, can be separated in two dimensions.

So if we have different categories-- dogs and non-dogs-- that can be viewed along different dimensions-- how furry they are-- but can't be separated-- the two categories can't be separated from each other on the basis of just one dimension of observation. So in this case, what we want to do is take this base of inputs and rotate it into a new what we'll call a new basis set so that now we can take the firing rates of these output neurons and use those to separate these different categories from each other. Does that make sense?

OK, so let me show you a few more examples of that. So this is one way to think about what we do when we do color vision, OK? So you know that we have different cones in our retina that are sensitive to different wavelengths. Most colors are combinations of those wavelengths. So if we look at the activity of, let's say, a cone that's sensitive to wavelength one and the activity in a cone that's sensitive to wavelength two, we might see-- and then we look around the world. We'll see a bunch of different objects or a bunch of different stimuli that activate those two different cones in different ratios.

And you might imagine that this axis corresponds to, let's say, how much red there is in a stimulus. This axis corresponds to how much green there is in a stimulus. But let's say that you're in an environment where there's some cloud of contribution of red and green. So what would this direction correspond to in this cloud? This direction corresponds to more red and more green. What would that correspond to?

AUDIENCE: Brown.

MICHALE FEE: So what I'm trying to get at here is that the sum of those two is sort of the brightness of the object, right? Something that has little red and little green will look the same color as something that has more red and more green, right? But what's different about those two stimuli is that the one's brighter than the other. The second one is brighter than the first one.

But this dimension corresponds to what? Differences in the ratio of those two colors, right? Sort of changes in the different [AUDIO OUT] wavelengths, and that corresponds to color. So if we can take this base of stimuli and rotate it such that one axis corresponds to the sum of the two colors and the other axis corresponds to the difference of the two colors, then this axis will tell you how bright it is, and this axis will tell you what the hue is, what the color is. Does that makes sense?

So there's a simple case of where taking a rotation of a inputs base, of a set of sensors, will give you different information than you would get if you just had one of those stimuli. If you were to just look at the activity of the cone that's giving you a red signal, if one object has more activity in that cone, you don't know whether that other object is just brighter or if it's actually more red, that looked red. Does that makes sense?

So doing a rotation gives us signals in single neurons that carries useful information. It can disambiguate different kinds of information. All right, so we can use that simple rotation matrix to perform that kind of separation. So brightness and color. Here's another example.

I didn't get to talk about this in this class, but there are-- so barn owls, they can very exquisitely localize objects by sound. So they hunt, essentially, at night in the dark.

They can hear a mouse scurrying around in the grass. They just listen to that sound, and they can tell exactly where it is, and then they dive down and catch the mouse.

So how did they do that? Well, they used timing differences to tell which way the sound is coming from side to side, and they use intensity differences to tell which way the sound is coming from up and down. Now, how do you use intensity differences? Well, one of their ears, their right ear pointed slightly upwards. And their left ear is pointed slightly downwards.

So when they hear a sound that's slightly louder in the right ear and slightly softer in the left ear, they know that it's coming from up above, right? And if it's the other way around, if it's slightly louder in the left ear and softer in the right ear, they know it's coming from below horizontal. And it's extremely precise system, OK?

So here's an example. So if they're sitting there listening to the intensity, the amplitude of the sound in the left ear and the amplitude of the sound in the right ear, some sounds will be up here with high amplitude in both ears. Some sounds will be over here, with more amplitude in the right ear and less amplitude in the left ear. What does this dimension correspond to? That dimension corresponds to?

AUDIENCE: Proximity.

MICHALE FEE: Proximity or, overall, the loudness of the sound, right? And what does this dimension correspond to?

AUDIENCE: Direction.

MICHALE FEE: The difference in intensity corresponds to the elevation of the sound relative to the horizontal. All right? So, in fact, what happens in the owl's brain is that these two signals undergo a rotation to produce activity in some neurons that's sensitive to the overall loudness and activity in other neurons that's sensitive to the difference between the intensity of the two sounds. It's a measure of the elevation of the sounds.

All right, so this kind of rotation matrix is very useful for projecting stimuli into the right dimension so that they give useful signals. All right, so let's come back to our matrix transformations and look in a little bit more detail about what kinds of transformations you can do with matrices.

So we talked about how matrices can do stretch, compression, rotation. And we're going to talk about a new kind of transformation that they can do. So you remember we talked about how a matrix multiplication implements a transformation from one set of vectors into another set of vectors? And the inverse of that matrix transforms back to the original set of vectors, OK? So you can make a transformation, and then you can undo that transformation by multiplying by the inverse of the matrix.

OK, so we talked about different kinds of transformations that you can do. So if you take the identity matrix and you make a small perturbation to both of the diagonal elements, the same perturbation to both diagonal elements, you're basically taking a set of vectors and you're stretching them uniformly in all directions. If you make a perturbation to just one of the components of the identity matrix, you can take the data and stretch it in one direction or stretch it in the other direction. If you add something to the first component and subtract something from the second component, you can stretch in one direction and compress in another direction.

We talked about reflections and inversions through the origin. These are all transformations that are produced by diagonal matrices. And the inverse of those diagonal matrices is just one over the diagonal elements. OK, we also talked about rotations that you can do with this rotation matrix. And then the inverse of the rotation matrix is, basically, you compute the inverse of a rotation matrix simply by computing the rotation matrix with a minus sign for this, using the negative of the rotation angle.

And we also talked about how a rotation matrix-- for a rotation matrix, the inverse is also equal to the transpose. And the reason is that rotation matrices have this antisymmetry, where the off-diagonal elements have the opposite sign. One of the things we haven't talked about is-- so we talked about how this kind of matrix can produce a stretch along one dimension or a stretch along the other dimension of the vectors. But one really important kind of transformation that we need to understand is how you can produce stretches in an arbitrary direction, OK? So not just along the x-axis or along the y-axis, but along any arbitrary direction.

And the reason we need to know how that works is because that formulation of how you write down a matrix to stretch data in any arbitrary direction is the basis of a lot of really important data analysis methods, including principal component analysis and other methods. So I'm going to walk you through how to think about making stretches in data in arbitrary dimensions. OK, so here's what we're going to walk through.

Let's say we have a set of vectors. I just picked-- I don't know, what is that-- 20 or so random vectors. So I just called a random number generator 20 times and just picked 20 random vectors. And we're going to figure out how to write down a matrix that will transform that set of vectors into another set of vectors that stretched along some arbitrary axis. Does that make sense?

So how do we do that? And remember, we know how to do two things. We know how to stretch a set of vectors along the x-axis. We know how to stretch vectors along the y-axis, and we know how to rotate a set of vectors. So we're just going to combine those two ingredients to produce this stretch in an arbitrary direction. So now I've given you the recipe-- or I've given you the ingredients. The recipe's pretty obvious, right?

We're going to take this set of initial vectors. Good. Lina?

AUDIENCE: You [INAUDIBLE]. That's it.

MICHALE FEE: Bingo. That's it. OK, so we're going to take-- all right, so we're going to rotate this thing 45 degrees. We take this original set of vectors. We're going to-- OK, so first of all, the first thing we do when we want to take a set of points and stretch it along an arbitrary direction, we pick that angle that we want to stretch it on-- in this case, 45 degrees. And we write down a rotation matrix corresponding to that rotation, corresponding to that angle. So that's the first thing we do.

So we've chosen 45 degrees as the angle we want to stretch on. So now we write down a rotation matrix for a 45-degree rotation. Then what we're going to do is we're going to take that set of points and we're going to rotate it by minus 45 degrees.

So how do we do that? How do we take any one of those vectors x and rotate it by-- so this that rotation matrix is for plus 45. How do we rotate that vector by minus 45?

AUDIENCE: [INAUDIBLE] multiply it by the [INAUDIBLE].

MICHALE FEE: Good. Say it.

AUDIENCE: Multiply by the inverse of that.

MICHALE FEE: Yeah, and what's the inverse of a--

AUDIENCE: Transpose.

MICHALE FEE: Transpose. So we don't have to go to Matlab and use the inverse matrix in inversion. We can just do the transpose. OK, so we take that vector and we multiply it by transpose. So that does a minus 45-degree rotation of all of those points. And then what do we do? Lina, you said it. Stretch it. Stretch it along?

AUDIENCE: The x-axis?

MICHALE FEE: The x-axis, good. What does that matrix look like that does that? Just give me-- yup?

AUDIENCE: 5, 0, 0, 1.

MICHALE FEE: Awesome. That's it. So we're going to stretch using a stretch matrix. So I use phi for a rotation matrix, and I use lambda for a stretch matrix, a stretch matrix along x or y. Lambda is a diagonal matrix, which always just stretches or compresses along the x or y direction. And then what do we do?

AUDIENCE: [INAUDIBLE]

MICHALE FEE: Good. By multiplying by? By this. Excellent. That's all. So how do we write this down? So, remember, here, we're sort of marching through the recipe from left to right. When you write down matrices, you go the other way. So when you do matrix multiplication, you take your vector x and you multiply it on the left side by phi transpose. And then you take that and you multiply that on the left side by lambda. And then you take that.

That now gives you these. And now to get the final answer here, you multiply again on the left side by phi. That's it. That's how you produce an arbitrary stretch-- a stretch or a compression of a data in an arbitrary direction, all right? You take the data, the vector. You multiply it by a rotation matrix transpose, multiply it by a stretch matrix, a diagonal matrix, and you multiply it by a rotation matrix. Rotate, stretch, unrotate.

So let's actually do this for 45 degrees. So there's our rotation matrix-- 1, minus 1, 1, 1. The transpose is 1, 1, minus 1, 1. And here's our stretch matrix. In this case, it was stretched by a factor of two. So we multiply x by phi transpose, multiply by lambda, and then multiply by phi. So we can now write that down.

If you just do those three matrix multiplications-- those two matrix multiplications, sorry, yes? One, two. Two matrix multiplications. You get a single matrix that when you multiply it by x implements this stretch. Any questions about that? You should ask me now if you don't understand, because I want you to be able to do this for an arbitrary-- so I'm going to give you some angle, and I'll tell you, construct a matrix that stretches data along a 30-degree axis by a factor of five. You should be able to write down that matrix.

All right, so this is what you're going to do, and that's what that matrix will look like, something like that. Now, we can stretch these data along a 45-degree axis by some factor. It's a factor of two here. How do we go back? How do we undo that stretch? So how do you take the inverse of a product of a bunch of matrices like this?

So the answer is very simple. If we want to take the inverse of a product of three matrices, what we do is we just-- it's, again, a product of three matrices. It's a product of the inverse of those three matrices, but you have to reverse the order. So if you want to find the inverse of matrix A times B times C, it's C inverse times B inverse times A inverse. And you can prove that that's the right term as follows.

So ABC inverse times ABC should be the identity matrix, right? So let's replace this by this result here. So C inverse B inverse A inverse times ABC would be the identity matrix. And you can see that right here, A inverse times A is i. So you can get rid of that. B inverse times B is i. C inverse times C is i. So we just proved that that is the correct way of taking the inverse of a product of matrices, all right?

So the inverse of this kind of matrix that stretches data along an arbitrary direction looks like this. It's phi transpose inverse lambda inverse phi inverse. So let's figure out what each one of those things is. So what is phi transpose inverse, where phi is a rotation matrix?

AUDIENCE: Just phi.

MICHALE FEE: Phi, good. And what is phi inverse?

AUDIENCE: [INAUDIBLE]

MICHALE FEE: [INAUDIBLE]. Good. And lambda inverse we'll get to in a second. So the inverse of this arbitrary rotated stretch matrix is just another rotated stretch matrix, right? Where the lambda now has-- lambda inverse is just given by the inverse of each of those diagonal elements. So it's super easy to find the inverse of one of these matrices that computes this stretch in an arbitrary direction.

You just keep the same phi. It's just phi times some diagonal matrix times phi transpose, but the diagonals are inverted. Does that makes sense?

All right, so let's write it out. We're going to undo this 45-degree stretch that we just did. We're going to do it by rotating, stretching by 1/2 instead of stretching by two. So you can see that compresses now along the x-axis. And then we rotate back, and we're back to our original data. Any questions about that?

It's really easy, as long as you just think through what you're doing as you go through those steps, all right? Any questions about that? OK. Wow. All right. So you can actually just write those down and compute the single matrix that implements this compression along that 45-degree axis, OK? All right.

So let me just show you one other example. And I'll show you something interesting that happens if you construct a matrix that instead of stretching along a 45-degree axis does compression along a 45-degree axis. So here's our original data. Let's take that data and rotate it by plus 45 degrees. Multiplied by lambda, that compresses along the x-axis and then rotates by minus 45 degrees.

So here's an example where we can take data and compress it along an axis of minus 45 degrees, all right? So you can write this down. So we're going to say we're going to compress along a minus 45 degree axis. We write down phi of minus 45.

Notice that when you do this compression or stretching, there are different ways you can do it, right? You can take the data. You can rotate it this way and then squish along this axis. Or you could rotate it this way and squish along this axis, right? So there are choices for how you do it. But in the end, you're going to end up with the same matrix that does all of those equivalent transformations.

OK, so here we are. We're going to write this out. So we're writing down a matrix that produces this compression along a minus 45-degree axis. So there's 5 minus 45. There's lambda, a compression along the x-axis. So here, it's 0.2001. And here's the phi transpose. So you write all that out, and you get 0.6, 0.4, 0.4, 0.4.

Let me show you one more. What happens if we accidentally take this data, we rotate it, and then we squish the data to zero? Yes?

AUDIENCE: [INAUDIBLE]

MICHALE FEE: It doesn't. You can do either one. Let me go back. Let me just go back to the very first one. So here, we rotated clockwise and then stretched along the x-axis and then unrotated. We could have taken these data, rotated counterclockwise, stretched along the y-axis, and then rotated back, right? Does that make sense?

You'll still get the same answer. You'll still get the same answer for this matrix here. OK, now watch this. What happens if we take these data, we rotate them, and then we compress data all the way to zero? So by compressing the data to a line, we're multiplying it by zero. We put a zero in this element of the stretch matrix, all right? And what happens? The data get compressed right to zero, OK?

And then we can rotate back. So we've taken these data. We can write down a matrix that takes those data and squishes them to zero along some arbitrary direction. Now, can we take those data and go back to the original data? Can we write down a transformation that takes those and goes back to the original data? Why not?

AUDIENCE: Lambda doesn't [INAUDIBLE].

MICHALE FEE: Say it again.

AUDIENCE: Lambda doesn't [INAUDIBLE].

MICHALE FEE: Good. What's another way to think about that?

AUDIENCE: We've lost [INAUDIBLE].

MICHALE FEE: You've lost that information. So in order to go back from here to the original data, you have to have information somewhere here that tells you how far out to stretch it again when you try to go back. But in this case, we've compressed everything to a line, and so there's no information how to go back to the original data.

And how do you know if you've done this? Well, you can take a look at this matrix that you created. So let's say somebody gave you this matrix. How would you tell whether you could back to the original data? Any ideas? Abiba?

AUDIENCE: [INAUDIBLE]

MICHALE FEE: Good. You look at the determinant. So if you calculate the determinant of this matrix, the determinant is zero. And as soon as you see a zero determinant, you know right away that you can't go back. After you've made this transformation, you can't go back to the original data. And we're going to get into a little more detail about why that is and what that means.

And the reason here is that the determinant of lambda is zero. The determinant of a product matrices like this is the product of the determinants. And in this case, the determinant of the lambda matrix is zero, and so the determinant of the product is zero, OK?

All right, so now let's talk about basis sets. All right, so we can think of vectors in abstract directions. So if I hold my arm out here and tell you this is a vector-- there's the origin. The vectors pointing in that direction. You don't need a coordinate system to know which way I'm pointing. I don't need to tell you my arm is pointing 80 centimeters in that direction and 40 centimeters in that direction and 10 centimeters in that direction, right?

You don't need a coordinate system to know which way I'm pointing, right? But if I want to quantify that vector so that-- if you want to quantify that vector so that you can maybe tell somebody else precisely which direction I'm pointing, you need to write down those numbers, OK? So you can think of vectors in abstract directions, but if you want to actually quantify it or write it down, you need to choose a coordinate system.

And so to do this, you choose a set of vectors, special vectors, called a basis set. And now we just say, here's a vector. How much is it pointing in that direction, that direction, and that direction? And that's called a basis set. So we can write down our vector now as a set of three numbers that simply tell us how far that vector is overlapped with three other vectors that form the basis set.

So the standard way of doing this is to describe a vector as a component in the x direction, which is a vector 1, 1, 0, sort of in the standard notation; a component in the y direction, which is 0, 1, 0; and a component in the z direction, 0, 0, 1. So we can write those vectors as standard basis vectors. The numbers x, y, and z here are called the coordinates of the vector. And the vectors e1, e2, and e3 are called the basis vectors. And this is how you would write that down for a three-dimensional vector, OK?

Again, the little hat here denotes that those are unit vectors that have a length one. All right, so in order to describe an arbitrary vector in a space of n real numbers, Rn, the basis vectors each need to have n numbers. And in order to describe an arbitrary vector in that space, you need to have n basis vectors. You need to have-- in n dimensions, you need to have n basis vectors, and each one knows basis vectors has to have n numbers in them.

So these vectors here-- 1, 0, 0; 0, 1, 0; and 0, 0, 1-- are called the standard basis. And each one of these values has one element that's one and the rest are zero. That's the standard basis. The standard basis has the property that any one of those vectors dotted into itself is one. That's because they're unit vectors. They have length one.

So i dot ei is the length squared of the i-th vector. And if the length is one, then the length squared is one. Each vector is orthogonal to all the other vectors. That means that each e1 dot e2 is zero, and e1 dot e3 is zero, and e2 dot e3 is zero. You can write down as e sub i dot e sub j equals zero for i not equal to j.

You can write all of those properties down in one equation-- e sub i dot e sub j equals delta i j. Delta i j is what's called the Kronecker delta function. The Kronecker delta function is a one if i equals j and a zero if i is not equal to j, OK? So it's a very compact way of writing down this property that each vector is a unit vector and each vector is orthogonal to all the other vectors.

And the set with that property is called an off an orthonormal basis set. All right, now, the standard basis is not the only basis-- sorry. I'm trying to do x, y, and z here. So if you have x, y, and z, that's not the only orthonormal basis set. Any basis set that is a rotation of those three vectors is also an orthonormal basis.

Let's write down two other orthogonal unit vectors. We can write down our vector v in this other basis set as follows. We just take our vector v. We can plot the basis vectors in this other basis. And we can simply project v onto those other basis vectors. So we can project v onto f1, and we can project v onto f2.

So we can write v as a sum of a vector in the direction of f1 and a vector in the direction of f2. You can write down this vector v in this different basis set as a vector with two components. This is two dimensional. This is R2. You can write it down as a two-component vector-- v dot f1 and v dot f2. So that's a simple intuition for what [AUDIO OUT] in two dimensions. We're going to develop the formalism for doing this in arbitrary dimensions, OK? And it's very simple.

All right, these components here are called the vector coordinates of this vector basis f. All right, now, basis sets, or basis vectors, don't have to be orthogonal to each other, and they don't have to be normal. They don't have to be unit vector. You can write down an arbitrary vector as a sum of components that aren't orthogonal to each other.

So you can write down this vector v as a sum of a component here in the f1 direction and a component in the f2 direction, even if f1 and f2 are not orthogonal to each other and even if they're not unit vectors. So, again, v is expressed as a linear combination of a vector in the f1 direction and a vector in the f2 direction. OK, so let's take a vector and decompose it into an arbitrary basis set f1 and f2.

So v equals c1 f1 plus c2 f2. The coefficients here are called the coordinates of the vector in this basis. And the vector v sub f-- these numbers, c1 and c2, when combined into this vector, is called the coordinate vector of v in the basis f1 and f2, OK? Does that makes sense? Just some terminology.

OK, so let's define this basis, f1 and f2. We just pick two vectors, an arbitrary two vectors. And I'll explain later that not all choices of vectors work, but most of them do. So here are two vectors that we can choose as a basis-- so 1, 3, which is sort of like this, and minus 2, 1 is kind of like that.

And we're going to write down this vector v in this new basis. So we have a vector v that's 3, 5 in the standard basis, and we're going to rewrite it in this new basis, all right? So we're going to find the vector coordinates of v in the new basis. So we're going to do this as follows.

We're going to write v as a linear combination of these two basis vectors. So c1 times f1-- 1, 3-- plus c2 times f2-- minus 2, 1-- is equal to 3, 5. That make sense? So what is that? That is just a system of equations, right? And what we're trying to do is solve for c1 and c2. That's it.

So we already did this problem in the last lecture. So we have this system of equations. We can write this down in the following matrix notation. F times vf-- vf is just c1 and c2-- equals v. So there's F-- 1, 3; minus 2, 1. Those are our two basis vectors. Times c1 c2-- the vector c1, c2-- is equal to 3, 5. And we solve for vf. In other words, we solve for c1 and c2 simply by multiplying v by the inverse of this matrix F.

So the coordinate vector in this new base is said is just the old vector times f inverse. And what is f inverse? F inverse is just a matrix that has the basis vectors as the columns of the matrix. So the coordinates of this vector in his new basis set are given by f inverse times v. We can find the inverse of f. So if that's our f, we can calculate the inverse of that.

Remember, you flip the diagonal elements. You multiply the off-diagonals by minus 1, and you divide by the determinant. So f inverse is this times v is that, and v sub f is just 13/7 over minus 4/7. So that's just a different way of writing v. So there's v in the standard basis. There's v in this new basis, all right? And all you do to go from the standard basis to any arbitrary new basis is multiply the vector by f inverse.

And when you're actually doing this in Matlab, this is really simple. You just write down a matrix F that has the basis sets in the columns. You just use the matrix inverse function, and then you multiply that by the data matrix, by the data vector. All right, so I'm just going to summarize again. In order to find the coordinate vector for v in this new basis, you construct a matrix F, whose columns are just the elements of the basis vectors.

So if you have two basis vectors, it's a two-- remember, each of those basis vectors. In two dimensions, there are two basis vectors. Each has two numbers, so this is a 2 by 2 matrix. In n dimensions, you have n basis vectors. Each of the basis vectors has n numbers. And so this matrix F is an n by n matrix, all right?

You know that you can write down v as this basis times v sub f. You solve for v sub f by multiplying both sides by f inverse, all right? That performs whats called change of basis. Now, that only works if f has an inverse. So if you're going to choose a new basis to write down your vector, you have to be careful to pick one that has an inverse, all right? And I want to show you what it looks like when you pick a basis that doesn't have an inverse and what that means.

All right, and that gets to the idea of linear independence. All right, so, remember I said that if in n dimensions, in Rn, in order to have a basis in Rn, you have certain requirements? Not any vectors will work. So let's take a look at these vectors. Will those work to describe an-- will that basis set work to describe an arbitrary vector in three dimensions? No? Why not?

AUDIENCE: [INAUDIBLE] vectors, so if you're [INAUDIBLE].

MICHALE FEE: Right. So the problem is in which coordinate, which axis?

AUDIENCE: Z-axis.

MICHALE FEE: The z-axis. You can see that you have zeros in all three of those vectors, OK? You can't describe any vector with this basis that has a non-zero component in the z direction. And the reason is that any linear combination of these three vectors will always lie in the xy plane. So you can't describe any vector here that has a non-zero z component, all right?

So what we say is that this set of vectors doesn't span all of R3. It only spans the xy plane, which is what we call a subspace of R3, OK? OK, so let's take a look at these three vectors. The other thing to notice is that you can write any one of these vectors as a linear combination of the other two.

So you can write f3 as a sum of f1 and f2. The sum of those two vectors is equal to that one. You can write f2 as f3 minus f1. So any of these vectors can be written as a linear combination of the others. And so that set of vectors is called linearly dependent. And any set of linearly dependent vectors cannot form a basis.

And how do you know if a set of vectors that you choose for your basis is linearly dependent? Well, again, you just find the determinant of that matrix. And if it's zero, those vectors are linearly dependent. So what that corresponds to is you're taking your data and when you transform it into a new basis, if the determinant of that matrix F is zero, then what you're doing is you're taking those data and transforming them to a space where they're being collapsed.

Let's say if you're in three dimensions, those data are being collapsed onto a plane or onto a line, OK? And that means you can't undo that transformation, all right? And the way to tell whether you've got that problem is looking at the determinant.

All right, let me show you one other cool thing about the determinant. There's a very simple geometrical interpretation of what the determinant is, OK? All right, sorry. So if f maps your data onto a subspace, then the mapping is not reversible.

OK, so what does the determinant correspond to? Let's say in two dimensions, if I have two orthogonal unit vectors, you can think of those vectors as kind of forming a square in this space. Or in three dimensions, if I have three orthogonal vectors, you can think of those vectors as defining a cube, OK? And if there unit vectors, then they define a cube of volume one.

Here, you have the square of area one. So let's think about this unit volume. If I transform those two vectors or those three vectors in 3D space by a matrix A, those vectors get rotated and transformed. They point in different directions, and they define-- it's no longer a cube, but they define some sort of rhombus, OK?

You can ask, what is the volume of that rhombus? The volume of that rhombus is just the determinant of that matrix A. So now what happens if I have a cube in three-dimensional space and I multiply it by a matrix that transforms it into a rhombus that has zero volume? So let's say I have those three vectors. It transforms it into, let's say, a square.

The volume of that square in three dimensional space is zero. So what that means is I'm transforming my vectors into a space that has zero volume in the original dimensions, OK? So I'm transforming things from 3D into a 2D plane. And what that means is I've lost information, and I can't go back.

OK, notice that a rotation matrix, if I take this cube and I rotate it, has exactly the same volume as it did before I rotated it. And so you can always tell when you have a rotation matrix, because the determinant of a rotation matrix is one. So if you take a matrix A and you find the determinant and you find that the determinant is one, you know that you have a pure rotation matrix.

What does it mean if the determinant is minus one? What it means is you have a rotation, but that one of the axes is inverted, is flipped. There's a mirror in there. So you can tell if you have a pure rotation or if you have a rotation and one of the axes is flipped. Because in the pure rotation, the determinant is one. And in an impure rotation, you have a rotation and a mirror flip.

All right, and I just want to make a couple more comments about change of basis, OK? All right, so let's choose a set of basis vectors for our new basis. Let's write those into a matrix F. It's going to be our matrix of basis vectors. If the determinant is not equal to zero, then these vectors, that set of vectors, are linearly independent. That means you cannot write one of those vectors as a linear combination of-- any one of those vectors as a linear combination of the others.

Those vectors form a complete basis in that n dimensional space. The matrix F implements a change of basis, and you can go from the standard basis to F by multiplying your vector by F inverse to get the coordinate vector and your new basis. And you can go back from that rotated or transformed basis back to the coordinate basis by multiplying by F, OK? Multiply by F inverse transforms to the new basis. Multiplying by F transforms back.

If that set of vectors is an orthonormal basis, then-- OK, so let's take this matrix F that has columns that are the new basis vectors. And let's say that those form an orthonormal basis. In that case, we can write down-- so, in any case, we can write down the transpose of this matrix, F transpose. And now the rows of that matrix are the basis vectors.

Notice that if we multiply F transpose times F, we have basis vectors in rows here and columns here. So what is F transpose F for the case where these are unit vectors that are orthogonal to each other? What is that product?

AUDIENCE: [INAUDIBLE]

MICHALE FEE: It's what?

AUDIENCE: [INAUDIBLE]

MICHALE FEE: Good. Because F1 dot F1 is one. F1 dot F2 is zero. F2 dot F1 is zero, and F2 dot F2 is 0. So that's equal to the identity matrix, right? So F transpose equals F inverse. If the inverse of a matrix is just its transpose, then that matrix is a rotation matrix. So F is just the rotation matrix.

All right, now let's see what happens. So that means the inverse of F is just this F transpose. Let's do this coordinate-- let's [AUDIO OUT] change of basis for this case. So you can see that v sub f, the coordinate vector in the new basis, is F transpose v. Here's F transpose-- the basis vectors are in the rows-- times v. This is just v dot F1, v dot F2, right?

So this shows how for a orthonormal basis, the transpose, which is the inverse of F-- taking the transpose of F times v is just taking the dot product of v with each of the basis vectors, OK? So that ties it back to what we were showing before about how to do this change of basis, OK? Just tying up those two ways of thinking about it.

So, again, what we've been developing when we talk about change of basis are ways of rotating vectors, rotating sets of data, into different dimensions, into different basis sets so that we can look at data from different directions. That's all we're doing. And you can see that when you look at data from different directions, you can get-- some views of data, you have a lot of things overlapping, and you can't see them. But when you rotate those data, now, all of a sudden, you can see things clearly that used to be-- things get separated in some views, whereas in other views, things are kind of mixed up and covering each other, OK?

And that's exactly what neural networks are doing when they're analyzing sensory stimuli. They're doing that kind of rotations and untangling the data to see what's there in that high-dimensional data, OK? All right, that's it.