MICHALE FEE: OK, let's go ahead and get started. So today we're turning to a new topic called that basically focused on principal components analysis, which is a very cool way of analyzing high-dimensional data. Along the way, we're going to learn a little bit more linear algebra. So today, I'm going to talk to you about eigenvectors and eigenvalues which are one of the most fundamental concepts in linear algebra. And it's extremely important and widely applicable to a lot of different things.

So eigenvalues and eigenvectors are important for everything from understanding energy levels and quantum mechanics to understanding the vibrational modes of a musical instrument, to analyzing the dynamics of differential equations of the sort that you find that describe neural circuits in the brain, and also for analyzing data and doing dimensionality reduction. So understanding eigenvectors and eigenvalues are very important for doing things like principal components analysis.

So along the way, we're going to talk a little bit more about variance. We're going to extend the notion of variance that we're all familiar with in one dimension, like the width of a Gaussian or the width of a distribution of data to the case of multivariate Gaussian distributions or multivariate-- which means, it's basically the same thing as high-dimensional data.

We're going to talk about how to compute a covariance matrix from data which describes how the different dimensions of the data are correlated with each other, what the variance in different dimensions is, and how those different dimensions are correlated with each other. And finally, we'll go through actually how to implement principal components analysis, which is useful for a huge number of things.

I'll come back to many of the different applications of principal components analysis at the end. But I just want to mention that it's very commonly used in understanding high-dimensional data and neural circuits. So it's a very important way of describing how the state of the brain evolves as a function of time. So nowadays, you can record from hundreds or even thousands or tens of thousands of neurons simultaneously. And if you just look at all that data, it just looks like a complete mess.

But somehow, underneath of all of that, the circuitry in the brain is going through discrete trajectories in some low-dimensional space within that high-dimensional mess of data. So our brains have something like 100 billion neurons in them-- about the same as the number of stars in our galaxy-- and yet, somehow all of those different neurons communicate with each other in a way that constrains the state of the brain to evolve along the low-dimensional trajectories that are our thoughts and perceptions.

And so it's important to be able to visualize those trajectories in order to understand how that machine is working. OK, and then one more comment about principal components analysis, it's not actually the best way often of doing this kind of dimensionality reduction. But the basic idea of how principal components analysis works is so fundamental to all of the other techniques. It's sort of the base on which all of those other techniques are built conceptually. So that's why we're going to spend a lot of time talking about this.

OK, so let's start with eigenvectors and eigenvalues. So remember, we've been talking about the idea that matrix multiplication performs a transformation. So we can have a vector x that we multiply it by matrix A. It transforms that set of vectors x into some other set of vectors y. And we can go from y back to x by multiplying by A inverse-- if the determinant of that matrix A is not equal to zero.

So we've talked about a number of different kinds of matrix transformations by introducing perturbations on the identity matrix. So if we have diagonal matrices, where one of the elements is slightly larger than 1, the other diagonal element is equal to 1, you get a stretch of this set of input vectors along the x-axis. Now, that process of stretching vectors along a particular direction has built into it the idea that there are special directions in this matrix transformation. So what do I mean by that?

So most of these vectors here, each one of these red dots is one of those x's, one of those initial vectors-- if you look at the transformation from x to y going-- so that's the x that we put into this matrix transformation. When we multiply by y, we see that that vector has been stretched along the x direction. So for most of these vectors, that stretch involves a change in the direction of the vector. Going from x to y means that the vector has been rotated.

So you can see that the green vector is at a different angle than the red vector. So there's been a rotation, as well as a stretch. So you can see that's true for that vector, that vector, and so on. So you can see, though, that there are other directions that are not rotated. So here's another. I just drew that same picture over again. But now, let's look at this particular vector, this particular red vector. You can see that when that red vector is stretched by this matrix, it's not rotated. It's simply scaled. Same for this vector right here. That vector is not rotated. It's just scaled, in this case, by 1.

But let's take a look at this other transformation. So this transformation produces a stretch in the y direction and a compression in the x direction. So I'm just showing you a subset of those vectors now. You can see that, again, this vector is rotated by that transformation. This vector is rotated by that transformation. But other vectors are not rotated. So again, this vector is compressed. It's simply scaled, but it's not rotated. And this vector is stretched. It's scaled but not rotated. Does that make sense?

OK, so these transformations here are given by a diagonal matrices where the off-diagonal elements are zero. And the diagonal elements are just some constant. So for all diagonal matrices, these special directions, the directions on which vectors are simply scaled but not rotated by that matrix by that transformation, it's the vectors along the axes that are scaled and not rotated-- along the x-axis or the y-axis.

And you can see that by taking this matrix A, this general diagonal matrix, multiplying it by a vector along the x-axis, and you can see that that is just a constant, lambda 1, times that vector. So we take this times this, plus this times this, is equal to lambda 1. This times this plus this times this is equal to zero. So you can see that A times that vector in the x direction is simply a scaled version of the vector in the x direction. And the scaling factor is simply the constant that's on the diagonal.

So we can write this in matrix notation as this lambda, this stretch vector, this diagonal matrix, times a unit vector in the x direction. That's the standard basis vector, the first standard basis vector. So that's a unit vector in the x direction is equal to lambda 1 times a vector in the x direction. And if we do that same multiplication for a vector in the y direction, we see that we get a constant times that vector in the y direction. So we have another equation.

So this particular matrix, this diagonal matrix, has two vectors that are in special directions in the sense that they aren't rotated. They're just stretched. So diagonal matrices have the property that they map any vector parallel to the standard basis into another vector along the standard basis.

So that now is a general n-dimensional diagonal matrix with these lambdas, which are just scalar constants along the diagonal. And there are n equations that look like this that say that this matrix times a vector in the direction of a standard basis vector is equal to a constant times that vector in the standard basis direction. Any questions about that? Everything else just flows from this very easily. So if you have any questions about that, just ask.

OK, that equation is called the eigenvalue equation. And it describes a property of this matrix lambda. So any vector v that's mapped by a matrix A onto a parallel vector lambda v is called an eigenvector of this matrix. So we're going to generalize now from diagonal matrices that look like this to an arbitrary matrix A. So the statement is that any vector, that when you multiply it by a matrix A that gets transformed into a vector parallel to v, it's called an eigenvector of A.

And the one vector that this is true for that isn't called an eigenvector is the zero vector because you can see that a zero vector here times any matrix is equal to zero. OK, so we exclude the zero vector. We don't call the zero vector an eigenvector. So typically a matrix, an n-dimensional matrix, has n eigenvectors and n eigenvalues. Oh, and I forgot to say that the scale factor lambda is called the eigenvalue associated with that vector v.

So now, let's take a look at a matrix that's a little more complicated than our diagonal matrix. Let's take one of these rotated stretch matrices. So remember, in the last class, we built a matrix like this that produces a stretch of a factor of 2 along a 45-degree axis. And we built that matrix by multiplying it together by basically taking this set of vectors, rotating them, stretching them, and then rotating them back. So we did that by three separate transformations that we applied successively. And we did that by multiplying phi transpose lambda and then phi.

So let's see what the special directions are for this matrix transformation. So you can see that most of these vectors that we've multiplied by this matrix get rotated. And you can see that even vectors along the standard basis directions get rotated. So what are the special directions for this matrix? Well, they're going to be these vectors right here. So this vector along this 45-degree line gets transformed. It's not rotated. It gets stretched by a factor of 1. And this vector here gets stretched.

OK, so you can see that this matrix has eigenvectors that are along this 45-degree axis and that 45-degree axis. So in general, let's calculate what are the eigenvectors and eigenvalues for a general rotated transformation matrix. So let's do that. Let's take this matrix A and multiply it by a vector x. And we're going to ask what vectors x satisfy the properties that, when they're multiplied by A, are equal to a constant times x. So we're going to ask what are the eigenvectors of this matrix A that we've constructed in this form?

So what we're going to do is we're going to replace A with this product of matrices, of three matrices. We're going to multiply this equation on both sides by phi transpose on the left side, by phi transpose. OK, so phi transpose times this, is equal to A sabai, x subai, times 5 transpose on the left. What happens here? Remember phi is a rotation matrix. What is phi transpose phi? Anybody remember?

Good. Because for rotation matrix, the inverse, the transpose of a rotation matrix, is its inverse. And so phi transpose phi is just equal to the identity matrix. So that goes away. And we're left with lambda phi transpose x equals A phi transpose x. So remember that we just wrote down that if we have a diagonal matrix lambda, that the eigenvectors are the standard basis vectors. So what does that mean?

If we look at this equation here, and we look at this equation here, it seems like phi transpose x is an eigenvector of this equation as long as phi transpose x is equal to one of the standard basis vectors. Does that make sense? So we know this solution is satisfied by phi transpose x is equal to one of the standard basis vectors. Does that make sense? So if we replace phi transpose x with one of the standard basis vectors, then that solves this equation.

So what that means is that the solution to this eigenvalue equation is that the eigenvalues A are simply the diagonal elements of this lambda here. And the eigenvectors are just x, where x is equal to phi times the standard basis vectors. We just solve for x by multiplying both sides by phi transpose inverse. What's phi transpose inverse? phi. So we multiply both sides by phi. This becomes the identity matrix. And we have x equals phi times this set of standard basis vectors.

Any questions about that? That probably went by pretty fast. But does everyone believe this? We went through that. We went through both examples of how this equation is true for the case where lambda is a diagonal matrix and the e's are the standard basis vectors. And if we solve for the eigenvectors of this equation where A has this form of phi lambda phi transpose, you can see that the eigenvectors are given by this matrix times a standard basis vector. So any standard basis vector times phi will give you an eigenvector of this equation here.

Let's push on. And the eigenvalues are just these diagonal elements of this lambda. What are these? So now, we're going to figure out what these things are, and how to just see what they are. These eigenvectors here are given by phi times a standard basis vector. So phi is a rotation matrix, right? So phi times a standard basis vector is just what? It's just a standard basis vector rotated.

So let's just solve for these two x's. We're going to take phi, which was this 45-degree rotation matrix, and we're going to multiply it by the standard basis vector in the x direction. So what is that? Just multiply this out. You'll see that this is just a vector along a 45-degree line. So this eigenvector, this first eigenvector here, is just a vector on the 45-degree line, 1 over root 2. It's a unit vector. That's why it's got the 1 over root 2 in it.

The second eigenvector is just phi times e2. So it's a rotated version of the y standard basis vector, which is 1 over root 2 minus 1, 1. That's this vector. So our two eigenvectors we derived for this matrix that produces this stretch along a 45-degree line, the two eigenvectors are the vector, 45-degree vector in this quadrant, and the 45-degree vector in that quadrant. Notice it's just a rotated basis set. So notice that the eigenvectors are just the columns of our rotated matrix.

So let me recap. If you have a matrix that you've constructed like this, as a matrix that produces a stretch in a rotated frame, the eigenvalues are just the diagonal elements of the lambda matrix that you put in there to build that thing, to build that matrix. And the eigenvectors are just the columns of the rotation matrix. OK, so let me summarize. A symmetric matrix can always be written like this, where phi is a rotation matrix. And lambda is a diagonal matrix that tells you how much the different axes are stretched.

The eigenvectors of this matrix A are the columns of phi. They are the basis vectors, the new basis vectors, in this rotated basis set. So remember, we can [AUDIO OUT] this rotation matrix as a set of basis vectors, as the columns. And that set of basis vectors are the eigenvectors of any matrix that you construct like this. And the eigenvalues are just the diagonal elements of the lambda that you put in there. All right, any questions about that? For the most part, we're going to be working with matrices that are symmetric, that can be built like this.

So eigenvectors are not unique. So if x eigenvector of A, then any scaled version of x is also an eigenvector. Remember, an eigenvector is a vector that when you multiply it by a matrix just gets stretched and not rotated. What that means is that any vector in that direction will also be stretched and not rotated. So eigenvectors are not unique. Any scaled version of an eigenvector is also an eigenvector. When we write down eigenvectors of a matrix, we usually write down unit vectors to avoid this ambiguity.

So we usually write eigenvectors as unit vectors. For matrices of n dimensions, there are typically n different unit eigenvectors-- n different vectors in different directions that have the special properties that they're just stretched and not rotated. So for our two-dimensional matrices that produce stretch in one direction, the special directions are-- sorry, so here is a two-dimensional, two-by-two matrix that produces a stretch in this direction. There are two eigenvectors, two unit eigenvectors, one in this direction and one in that direction.

And notice, that because the eigenvectors are the columns of this rotation matrix, the eigenvectors form a complete orthonormal basis set. And that is true. That statement is true only for symmetric matrices that are constructed like this. So now, let's calculate what the eigenvalues are for a general two-dimensional matrix A. So here's our matrix A. That's an eigenvector. Any vector x that satisfies that equation is called an eigenvector. And that's the eigenvalue associated with that eigenvector.

We can rewrite this equation as A times x equals lambda i times x-- just like A equals b, then equals 1 times b. We can subtract that from both sides, and we get A minus lambda i times x equals zero. So that is a different way of writing an eigenvalue equation. Now, what we're to do is we're going to solve for lambdas that satisfy this equation. And we only want solutions where x is not equal to zero.

So this is just a matrix. A minus lambda i is just a matrix. So how do we know whether this matrix has solutions where x is not equal to zero? Any ideas? [INAUDIBLE]

AUDIENCE: [INAUDIBLE]

MICHALE FEE: Is, so what do we need the determinant to do?

AUDIENCE: [INAUDIBLE]

MICHALE FEE: Has to be zero. If the determinant of this matrix is not equal to zero, then the only solution to this equation is x equals zero. OK, so we solve this equation. We ask what values of lambda give us a zero determinant in this matrix. So let's write down an arbitrary A, an arbitrary two-dimensional matrix A, 2D, 2 by 2. We can write A minus lambda i like this. Remember, lambda i is just lambdas on the diagonals.

The determinant of A minus lambda i is just the product of the diagonal elements minus the product of the off-diagonal elements. And we set that equal to zero. And we solve for lambda. And that just looks like a polynomial. OK, so the solutions to that polynomial solve what's called the characteristic equation of this matrix A. And those are the eigenvalues of this arbitrary matrix A, this 2D, two-by-two matrix.

So there is characteristic equation. There is the characteristic polynomial. We can solve for lambda just by using the quadratic formula. And those are the eigenvalues of A. Notice, first of all, there are two of them given by the two roots of this quadratic equation. And notice that they can be real or complex. They can be complex. They are complex in general. And they can be real, or imaginary, or have real and imaginary components.

And that just depends on this quantity right here. If what's inside this square root is negative, then eigenvalues will be complex. If what's inside the square root is positive, then the eigenvector will be real. So let's find the eigenvalues for a symmetric matrix. a, d on the diagonals and b on the off-diagonals. So let's see what happens. Let's plug these into this equation. The 4bc becomes 4b squared.

And you can see that this thing has to be greater than zero because a minus d squared has [INAUDIBLE] has to be positive. And b squared has to be positive. And so that quantity has to be greater than zero. And so what we find is that the eigenvalues of a symmetric matrix are always real. So let's just take this particular-- just an example-- and let's plug those into this equation. And what we find is that the eigenvalues are 1 plus or minus root 2 over 2. So two real eigenvalues.

So let's consider a special case of a symmetric matrix. Let's consider a matrix where the diagonal elements are equal, and the off-diagonal elements are equal. So we can update this equation for the case where the diagonal elements are equal. So a equals d. And what you find is that the eigenvalues are just a plus b and a minus b-- so a plus b and a minus b. And the eigenvectors can be found just by plugging these eigenvalues into the eigenvalue equation and solving for the eigenvectors.

So I'll just go through that real quick-- a times x. So we found two eigenvalues, so there are going to be two eigenvectors. We can just plug that first eigenvalue into here, call it lambda plus. And now, we can solve for the eigenvector associated with that eigenvalue. Just plug that in, solve for x. What you find is that the x associated with that eigenvalue is 1, 1-- if you just go through the algebra. So that's the eigenvector associated with that eigenvalue. And that is the eigenvector associated with that eigenvalue.

So I'll just give you a hint. In most of the problems that I'll give you to deal with on an exam or many of the ones in the problem sets, I think, in the problem set will have a form like this and [INAUDIBLE] eigenvectors along a 45-degree axis. So if you see a matrix like that, you don't have to plug it into MATLAB to extract the eigenvalues. You just know that the eigenvectors are on the 45-degree axis.

So the process of writing a matrix as phi lambda phi transpose is called eigen-decomposition of this matrix A. So if you have a matrix that you can write down like this, that you can write in that form, it's called eigen-decomposition. And the lambdas, the diagonal elements of this lambda matrix, are real. And they're the eigenvalues. The columns of phi are the eigenvalues, and they form an orthogonal basis set.

And this, if you take this equation and you multiply it on both sides by phi, you can write down that equation in a slightly different form-- A times phi equals phi lambda. This is a matrix way, a matrix equivalent, to the set of equations that we wrote down earlier. So remember, we wrote down this eigenvalue equation that describes that when you multiply this matrix A times an eigenvector equals lambda times the eigenvector, this is equivalent to writing down this matrix equation.

So you'll often see this equation to describe the form of the eigenvalue equation rather than this form. Why? Because it's more compact. Any questions about that? We've just piled up all of these different f vectors into the columns of this rotation matrix phi. So if you see an equation like that, you'll know that you're just looking at an eigenvalue equation just like this.

Now in general, when you want to do eigen-decomposition, when you have a symmetric matrix that you want to write down in this form. It's really simple. You don't have to go through all of this stuff with the characteristic equation, and solve for the eigenvalues, and then plug them in here, and solve for the eigenvectors. You can do that if you really want to. But most people don't because in two dimensions, you can do it. But in higher dimensions, it's very hard or impossible.

So what you typically do is just use the eig function in MATLAB. If you just use this function eig on a matrix, it will return the eigenvectors and eigenvalues. So here, I'm just constructing a matrix A-- 1.5, 0.5, 0.5, and 1.5, like that. And if you just use the eig function, it returns the eigenvectors as the columns of the matrix and the eigenvalues as the diagonals of this matrix. So you have to pass it. Arguments F and V equals eig of A. And it returns eigenvectors and eigenvalues. Any questions about that?

So let's push on toward doing principal components analysis. So this is just the machinery that you use. Oh, and I think I had one more panel here just to show you that if you take F and V, you can reconstruct A. So A is just F, V, F transpose. F is just phi in the previous equation. And V is the lambda. Sorry, they didn't have phi and lambda, and they're not options. For variable names, I used F and V. And you can see that F, V, F transpose is just equal to A. Any questions about that? No?

All right, so let's turn to how do you use eigenvectors and eigenvalues to describe data. So I'm going to briefly review the notion of variance, what that means in higher dimensions, and how you use a covariance matrix to describe data in high dimensions.

So let's say that we have a bunch of observations of a variable x-- so this is now just a scaler. So, we have m different observations, x superscript j is the j-th observation of that data. And you can see that if you make a bunch of measurements of most things in the world, you'll find a distribution of those measurements. Often, they will be distributed in a bump.

You can write down the mean of that distribution just as the average value overall distributions by summing together all those distributions and dividing by the number of observations. You can also write down the variance of that distribution by subtracting the mean from all of those observations, squaring that difference from the mean, summing up over all observations, and dividing by m.

So let's say that we now have m different observations of two variables, pressure and temperature. We have a distribution of those quantities. We can describe that observation of x1 and x2 as a vector. And we have m different observations of that vector. You can write down the mean and variance of x1 and x2. So for x1, we can write down the mean as mu1. We can write down the variance of x1. We can write down the mean and variance of x2, of the x2 observation.

And sometimes, that will give you a pretty good description of this two-dimensional observation. But sometimes, it won't. So in many cases, those variables, x1 and x2, are not correlated with each other. They're independent variables. In many cases, though, x1 and x2 are dependent on each other. The observations of x1 and x2 are correlated with each other, so that if x1 is big, x2 also tends to be big.

In these two cases, x1 can have the same variance. x2 can have the same variance. But there's clearly something different here. So we need something more than just describing the variance of x1 and x2 to describe these data. And that thing is the covariance. It just says how do x1 and x2 covary? If x1 is big, does x2 also tend to be big? In this case, the covariance is zero. In this case, the covariance is positive.

So we're taking if a fluctuation of x1 above the mean is associated with a fluctuation of x2 above the mean, then these points will produce a positive contribution to the covariance. And these points here will also produce a positive contribution to the covariance. And the covariance here will be some number greater than zero. That's closely related to the correlation, just the Pearson correlation coefficient, which is the covariance divided by the geometric mean of the individual variances.

I'm assuming most of you have seen this many times, but just to get us up to speed. So if you have data, a bunch of observations, you can very easily fit those data to a Gaussian. And you do that simply by measuring the mean and variance of your data. And that turns out to be the best fit to a Gaussian. So if you have a bunch of observations in one dimension, you measure the mean and variance of that set of data. That turns out to be a best fit in the least squared sense to a Gaussian probability distribution defined by a mean and a variance.

So this is easy in one dimension. What we're interested in doing is understanding data in higher dimensions. So how do we describe data in higher dimensions? How do we describe a Gaussian in higher dimensions? So that's what we're going to turn to now. And the reason we're going to do this is not because every time we have data, we're really trying to fit a Gaussian into it. It's just that it's a powerful way of thinking about data, of describing data in terms of variances in different directions.

And so we often think about what we're doing when we are looking at high-dimensional data is understanding its distribution in different dimensions as kind of a Gaussian cloud that optimally best fits the data that we're looking at. And mostly because it just gives us an intuitive about how to best represent or think about data in high dimensions.

So we're going to get insights into how to think about high-dimensional data. We're going to develop that description using the vector and matrix notation that we've been developing all along because vectors and matrices provide a natural way of manipulating data sets, of doing transformations of basis, rotations, so on. It's very compact. And those manipulations are really trivial in MATLAB or Python.

So let's build up a Gaussian distribution in two dimensions. So we have, again, our

Gaussian random variables, x1 and x2. We have a Gaussian distribution, where the probability distribution is proportional to e to the minus 1/2 of x1 squared. We have probability distribution for x2-- again, probability of x2. We can write down the probability of x1 and x2, the joint probability distribution, assuming these are independent. We can write that as the product of p-- the product of the two probability distributions p of x1 and p of x2.

And we have some Gaussian cloud, some Gaussian distribution in two dimensions that we can write down like this. That's simply the product. So the product of these two distributions is e to the minus 1/2 x1 squared times e to the minus 1/2 x2 squared. And then, there's a constant in front that just normalizes, so that the total area under that curve is just 1.

We can write this as e to the minus 1/2 x1 squared plus x2 squared. And that's e to the minus 1/2 times some distance from the origin. So it falls off exponentially in a way that depends only on the distance from the origin or from the mean of the distribution. In this case, we set the mean to be zero.

Now, we can write that distance squared using vector notation. It's just the square magnitude of that vector x. So if we have a vector x sitting out here somewhere, we can measure the distance from the center of the Gaussian as the square magnitude of x, which is just x dot x, or x transpose x. So we're going to use this notation to find the distance of a vector from the center of the Gaussian distribution. So you're going to see a lot of x [INAUDIBLE] axis.

So this distribution that we just built is called an isotropic multivariate Gaussian distribution. And that distance d is called the Mahalanobis distance, which I'm going to say as little as possible. So that distribution now describes how these points-- the probability of finding these different points drawn from that distribution as a function of their position in this space. So you're going to draw a lot of points here in the middle and fewer points as you go away at larger distances.

So this particular distribution that I made here has one more word in it. It's an isotopic multivariate Gaussian distribution of unit variance. And what we're going to do now is we're going to build up all possible Gaussian distributions from this distribution by simply doing matrix transformations. So we're going to start by taking that unit variance Gaussian distribution and build an isotopic Gaussian distribution that has an arbitrary variance-- that means an arbitrary width.

We're then going to build a Gaussian distribution that can be stretched arbitrarily along these two axes, y1 and y2. And we're going to do that by using a transformation with a diagonal matrix. And then, what we're going to do is build an arbitrary Gaussian distribution that can be stretched and worked in any direction by using a transformation matrix called a covariance matrix, which just tells you how that distribution is stretched in different directions. So we can stretch it in any direction we want. Yes.

AUDIENCE: Why is [INAUDIBLE]?

MICHALE FEE: OK, the distance squared is the square of magnitude. And the square of magnitude is x dot x, the dot product. But remember, we can write down the dot product in matrix notation as x transpose x. So if we have row vector times a column vector, you get the dot product. Yes, Lina.

AUDIENCE: What does isotropic mean?

MICHALE FEE: OK, isotropic just means the same in all directions. Sorry, I should have defined that.

AUDIENCE: [INAUDIBLE] when you stretched it, it's not isotropic?

MICHALE FEE: Yes, these are non-isotropic distributions because they're different. They have different variances in different directions. So you can see that this has a large variance in the y1 direction and a small variance in the y2 direction. So it's non-isotropic. Yes, [INAUDIBLE].

AUDIENCE: Why do you [INAUDIBLE]?

MICHALE FEE: Right here. OK, think about this. Variance, you put into this Gaussian distribution as the distance squared over the variance squared. It's distance squared over a variance, which is sigma squared. Here it's distance squared over a variance. Here it's distance squared over a variance. Does that makes sense? It's just that in order to describe these complex stretching and rotation of this Gaussian distribution in high-dimensional space, we need a matrix to do that.

And that covariance matrix describes the variances in the different direction and essentially the rotation. Remember, this distribution here is just a distribution that's stretched and rotated. Well, we learned how to build exactly such a transformation by taking the product of phi transpose lambda phi. So we're going to use this to build these arbitrary Gaussian distributions.

OK, so I'll just go through this quickly. If we have an isotopic unit variance Gaussian distribution as a function of this vector x, we can build a Gaussian distribution of arbitrary variance by writing down a y that's simply sigma times x. We're going to transform x into y, so that we can write down a distribution that has an arbitrary variance. Here this is variance 1. Here this is sigma squared.

So let's make just a change of variables y equals sigma x. So now, what's the probability distribution as a function of y? Well, there's probability distribution as a function of x. We're simply going to substitute y equals sigma x with x equals sigma inverse y. We're going to substitute this into here. The Mahalanobis distance is just x transpose x, which is just sigma inverse y transpose sigma inverse y.

And when you do that, you find that the distance squared is just y transpose sigma to the minus 2y. So there is our Gaussian distribution for this distribution. There's the expression for this Gaussian distribution with a variance sigma. We can rewrite this in different ways.

Now, let's build a Gaussian distribution that stretched arbitrarily in different directions, x and y. We're going to do the same trick. We're simply going to make a transformation y equals matrix, diagonal matrix, s times x and substitute this into our expression for a Gaussian. So x equals s inverse y. The Mahalanobis distance is given by x transpose x, which we can just get down here. Let's do that with this substitution. And we get an s squared here, s inverse squared, which we're just going to write as lambda inverse.

And you can see that you have these variances along the diagonal. So if that's lambda inverse, then lambda is just a matrix of variances along the diagonal. So sigma 1 squared is the variance in this direction. Sigma 2 squared is the variance in this direction.

I'm just showing you how you make a transformation to this vector x into another vector y to build up a representation of this effective distance from the center of distribution for different kinds of Gaussian distributions. And now finally, let's build up an expression for a Gaussian distribution with arbitrary variance and covariance.

So we're going to make a transformation of x into a new vector y using this rotated stretch matrix. We're going to substitute this in, calculate the Mahalanobis distance-- is now x transpose x. Substitute this and solve for the Mahalanobis distance. And what you find is that distance squared is just y transpose phi lambda inverse phi transpose times y. And we just write that as y transpose sigma inverse y. So that is now an expression for an arbitrary Gaussian distribution in high-dimensional space. And that distribution is defined by this matrix of variances and covariances.

Again, I'm just writing down the definition of sigma inverse here. We can take the inverse of that, and we see that our covariance-- this is called a covariance matrix-- it describes the variance and correlations of those different dimensions as a matrix. That's just this rotated stretch matrix that we been working with. And that's just the same as this covariance matrix that we described for distribution.

I feel like all that didn't come out quite as clearly as I'd hoped. But let me just summarize for you. So we started with an isotopic Gaussian of unit variance. And we multiplied that vector, we transformed that vector x, by multiplying it by sigma so that we could write down a Gaussian distribution of arbitrary variance. We transformed that vector x with a diagonal covariance matrix to get arbitrary stretches along the axes.

And then, we made another kind of transformation with an arbitrary stretch and rotation matrix so that we can now write down a Gaussian distribution that has arbitrary stretch and rotation of its variances in different directions. So this is the punch line right here-- that you can write down the Gaussian distribution with arbitrary variances in this form. And that sigma right there is just the covariance matrix that describes how wide that distribution is in the different directions and how correlated those different directions are.

I think this just summarizes what I've already said. So now, let's compute the covariance matrix from data. So now, I've shown you how to represent Gaussians in high dimensions that have these arbitrary variances. Now, let's say that I actually have some data. How do I fit one of these Gaussians to it? And it turns out that it's really simple. It's just a matter of calculating this covariance matrix. So let's do that.

So here is some high-dimensional data. Remember that to fit a Gaussian to a bunch of data, all we need to do is to find the mean and variants in one dimension. For higher dimensions, we just need to find the mean and the covariance matrix. So that's simple. So here's our set of observations. Now, instead of being scalars, they're vectors. First thing we do is subtract the mean. So we calculate the mean by summing all of those observations, dividing those numbers, divide by m. So there we find the mean.

We compute a new data set with the mean subtracted. So from every one of these observations, we subtract the mean. And we're going to call that z. So there is our mean subtracted here. I've subtracted the mean. So those are the x's. Subtract the mean. Those are now our z's, our mean-subtracted data. Does that makes sense? Now, we're going to calculate this covariance matrix.

Well, all we do is we find the variance in each direction and the covariances. So it's going to be a matrix in low-dimensional data. It's a two-by-two matrix. So we're going to find the variance in the z1 direction. It's just z1 times z1, summed over all the observations, divided by m. Th variance in the z2 direction is just the sum of z2, j, z2, j divided by m.

The covariance is just the cross terms, z1 one times z2 and z2 times z1. Of course, those are equal to each other. So in a covariance matrix, it's symmetric. So how do we calculate this? It turns out that in MATLAB, this is super-duper easy. So if this is our vector, that's our vector, one of our observations, we can compute the inner product z transpose z. So the inner product is just z transpose z, which is z1, z2, z1, z2. That's the square magnitude of z. There's another kind of product called the outer product. Remember that.

So the outer product looks like this. This is a 1 by 2. That's a rho vector times a column vector is equal to a scalar. 1 by 2 times 2 by 1 equals by 1 by 1-- two rows, one column-- times 1 by 2, gives you a 2 by 2 matrix that looks like this. z1 times z1, z1, z2, z1, z2, z2, z2. Why? It's z1 times z1 equals that. z1 times z2, z2 z1, one z2 z2. So that outer product already gives us the components to compute the correlation matrix.

So what we do is we just take this vector, z the j-th observation of this vector z, and multiply it by the j-th observation of this vector z transpose. And that gives us this matrix. And we sum over all this. And you see that is exactly the covariance matrix.

So if we have m observations of vector z, we put them in matrix form. So we have a big, long data matrix. Like this. There are m observations of this two-dimensional vector z. The data dimension, the data vector has, mentioned 2. Their are m observations. So m is the number of samples. So this is an n-by-m matrix. So if you want to compute the covariance matrix, you just in MATLAB, literally do z. This big matrix z times that matrix transpose. And that automatically finds the covariance matrix for you in one line of MATLAB.

There's a little trick to subtract the mean easily. So remember, your original observations are x. You compute the mean across the rows. Thus, you're going you're going to sum across columns to give you a mean for each row. That gives you a mean of that first component of your vector, mean of the second component. That's really easy in the lab. mu is the mean of x summing cross the second component. That gives you a mean vector and then you use repmat to fill that mean out in all of the columns and [INAUDIBLE] subtract this mean from x to get this data matrix z.

So now, let's apply those tools to actually do some principal components analysis. So principal components analysis is really amazing. If you look at single nucleotide polymorphisms and populations of people, there are like hundreds of genes that you can look at. You can look at different variations of a gene across hundreds of genes. But it's this enormous data set.

And you can find out which directions in that space of genes give you information about the genome of people. And for example, if you look at a number of genes across people with different backgrounds, you can see that they're actually clusters corresponding to people with different backgrounds.

You can do single-cell profiling. So you can do the same thing in different cells with a tissue. So you look at RNA transcriptional profiling. You see what are the genes that are being expressed in individual cells. You can do principal components analysis of those different genes and find clusters for different cell types within a tissue. This is now being applied very commonly in brain tissue now to extract different cell types. You can use images and find out which components of an image actually give you information about different faces.

So you can find a bunch of different faces, find the covariance matrix of those images, take that, do eigendecomposition on that covariance matrix. And extract what are called eigenfaces. These are dimensions on which the images carry information about face identity. You can use principal components analysis to decompose spike waveforms into different spikes. This is a very common way of doing spike sorting. So when you stick an electrode in the brain, you'd record from different cells at the end of the electrode.

Each one of those has a different way of form and you can use this method to extract the different waveforms people have even recently used this now to understand the low-dimensional trajectories of movements. So if you take a movie--

SPEAKER: After tracking, a reconstruction of the global trajectory can be made from the stepper motor movements, while the local shape changes of the worm can be seen in detail.

MICHALE FEE: OK, so here you see a c elegans, a worm moving along. This is an image, so it's a very high-dimensional. There are 1,000 pixels in this image. And you can decompose that image into a trajectory in a low-dimensional space. And it's been used to describe the movements in a low-dimensional space and relate to a representation of the neural activity in low dimensions as well. OK, so it's a very powerful technique.

So let me just first demonstrate PCA on just some simple 2D data. So here's a cloud of points given by a Gaussian distribution. So those are a bunch of vectors x. We can transform those vectors x using phi s phi transpose to produce a Gaussian, a cloud of points with a Gaussian distribution, rotated at 45 degrees, and stretched by 1.7-ish along one axis and compressed by that amount along another axis.

So we can build this rotation matrix, this stretch matrix, and build a transformation matrix-- r, s, r transpose. Multiply that by x. And that gives us this data set here. OK, so we're going to take that data set and do principal components analysis on it. And what that's going to do is it's going to find the dimensions in this data set that have the highest variance. It's basically going to extract the variance in the different dimensions. So we take that set of points. We just compute the covariance matrix by taking z, z transpose, times 1 over m.

That computes that covariance matrix. And then, we're going to use the eig function in MATLAB to extract the eigenvectors and eigenvalues of the covariance matrix OK, so q-- we're going to call q is the variable name for the covariance matrix it's zz transpose over m. Call eig of q. That returns the rotation matrix. And that rotation matrix, the columns of which are the eigenvectors, it returns the matrix of eigenvalues, the diagonal elements are the eigenvalues.

Sometimes, you need to do a flip-left-right because I sometimes return the lowest eigenvalues first. But I generally want to plot put the largest eigenvalue first. So there's the largest one, there's the smallest one. And now, what we do, is we simply rotate. We [AUDIO OUT] basis. We can rotate this data set using the rotation matrix that the principal components analysis found.

OK, so we compute the covariance matrix. Find the eigenvectors and eigenvalues of the covariance matrix right there. And then, we just rotate the data set into that new basis of eigenvectors and eigenvalues. It's useful for clustering. So if we have two clusters, we can take the clusters, compute the covariance matrix. Find the eigenvectors and eigenvalues of that covariance matrix. And then, rotate the data set into a basis set in which the dimensions in the data on which variances largest are along the standard basis vectors.

Let's look at a problem in the time domain. So here we have a couple of time-dependent signals. So this is some amplitude as a function of time. These are signals that I constructed. They're some wiggly function that I added noise to. What we do is we take each one of those times series, and we stack them up in a bunch of columns. So our vector is now a set of 100 time samples. So there is a vector of 100 different time points. Does that make sense? And we have 200 observations of those 100-dimensional vectors.

So we have a data vector x that has columns. That are hundreds dimensional. And we have 200 of those observations. So it's 100-by-200 matrix. 100-by-200 matrix.

We do the means subtraction we subtract the mean using that trick that I showed you. Compute the covariance matrix. So there we compute the mean. We subtract the mean using repmat. Subtract the mean from the data to get z. Compute the covariance matrix Q. That's what the covariance matrix looks like for those data.

And now, we plug it into eig to extract the eigenvectors and eigenvalues. OK, so extract F and V. If we look at the eigenvalues, you can see that there are hundreds eigenvalues because those data have 100 dimensions. So there are hundreds eigenvalues. You could see that two of those eigenvalues are big, and the rest are small. This is on a log scale. What that says is that almost all of the variance in these data exist in just two dimensions. It's 100-dimensional space. But the data are living in two dimensions. And all the rest is noise. Does that makes sense?

So what you'll typically do is take some data, compute the covariance matrix, find the eigenvalues, and look at the spectrum of eigenvalues. And you'll very often see that there is a lot of variance in a small subset of eigenvalues. Then, it tells you that the data are really living in a lower-dimensional subspace than the full dimensionality of the data. So that's where your signal is. And all the rest of that is noise.

You can plot the cumulative of this. And you can say that the first two components explain over 60% of the total variance in the data. So since there are two large eigenvalues, let's look at the eigenvectors associated with those. And we can find those. Those are just the first two columns of this matrix F that the eig function returned to us. And that's what those two eigenvectors look like.

That's what the original data looked like. The eigenvectors, the columns of the F matrix, are an orthogonal basis set. A new basis set. So those are the first two eigenvectors. And you can see that the signal lives in this low-dimensional space of these two eigenvectors. All of the other eigenvectors are just noise. So we can do is we can project the data into this new basis set. So let's do that. We simply do a change of basis.

The f is a rotation matrix. We can project our data z into this new basis set and see what it looks like. Turns out, that's what it looks like. There are two clusters in those data corresponding to the two different wave forms that you could see in the data.

Right there, you can see that there are kind of two wave forms in the data. If you projected data into this low-dimensional space, you can see that there are two clusters there.

If you projected data into other projections, you don't see it. It's only in this particular projection that you have these two very distinct clusters corresponding to the two different wave forms in the data. Now, almost all of the variance is in the space of the first two principal components. So what you can actually do is, you can project the data into these first two principal components, set all of the other principal components to zero, and then rotate back to the original basis set.

That is that you're setting as much of the noise to zero as you can. You're getting rid of most of the noise. And then, when you rotate back to the original basis set, you've gotten rid of most of the noise. And that's called principal components filtering. So here's before filtering and here's after filtering. OK, so youve found the low-dimensional space, in which all the data sits, in which the signal sits, everything outside of that space is noise. So you rotate the data into a new basis set. You can filter out all the other dimensions that just have noise. You filter back.

And you just keep the signal. And that's it. So that's sort of a brief intro to principal component analysis. But there are a lot of things you can use it for. It's a lot of fun. And it's a great intro to all the other amazing dimensionality reduction techniques that there are. I apologize for going over.