

FEATURE TRANSFORMATION

x_1, x_2, x_3, x_4

x_1, x_2, x_3, x_4

$2x_1 + x_2 + x_3$

THE PROBLEM OF PRE-PROCESSING A SET OF FEATURES TO CREATE A NEW (SMALLER? MORE COMPACT?) FEATURE SET, WHILE RETAINING AS MUCH (RELEVANT? USEFUL?) INFORMATION AS POSSIBLE:

$x \sim F^N \rightarrow F^M$

$M < N$ (usually)

$P^T x$ (usually)

XOR

a, b

a, b, c

linear combinations of original features

1. Hi again Michael. >> Hey, how's it going? >> It's going pretty well. We are ready to do our last lesson of this mini course on unsupervised learning and randomized optimization. >> Cool, what's it about? >> It is about feature transformation. As opposed to feature selection. >> Cool. So they can change from vehicles into robots, I assume. >> yeah, typically. Usually it's from robots into cars, but that's a technical detail. >> Okay. >> Okay. So I'm going to define feature transformation for you, or at least I'm going to do my best to. It turns out that it's a slightly more slippery definition than the one that we used for feature selection but it has a lot of things in common. Okay? >> Yep. >> And you tell me if this makes sense. Okay, so feature transformation as opposed to feature selection which we talked about last time is the problem of doing some kind of pre-processing on a set of features. In order to create a new set of features. Now typically, we expect that new set of the future to be smaller or in some way more compact. Okay? But while we create this new set of features, we want to explicitly retain as much information as possible, and when I say retain as much information as possible, I probably mean, I think we'll, we'll see as we continue this conversation. Information that is relevant and hopefully useful. Now, the first thing you might ask is, what's the difference between this and feature selection. Is that a question you might ask, Michael? >> I was thinking about that, though I think you kind of told me at the beginning of the feature selection lecture. >> well, I told you that I was going to to tell you. I'm not sure I actually told you. >> Hm. >> So one thing you might say Michael, is that if I looked at this definition, this seems to actually be consistent with feature selection as well. I'd take a set of features, a feature selection. Then I'd create a new feature set which happens to be smaller. And my goal was to retain as much information as possible and we talked about the difference between relevant features and useful features, but really this sort of describes feature selection as well. You see that? >> Yeah, definitely. >> Now [the difference \(between feature selection and feature transformation\)](#) is, in fact probably the right way to think about is this that [feature selection is in fact a subset of feature transformation](#). Where the pre-processing you're doing is literally taking a subset of the features. Here, feature transformation can be more powerful, and can be an arbitrary pre-processing, not just something that goes from a set to to a subset. But what we're going to do is we're going to restrict our notion of future transformation to what's called linear future transformation. So, let

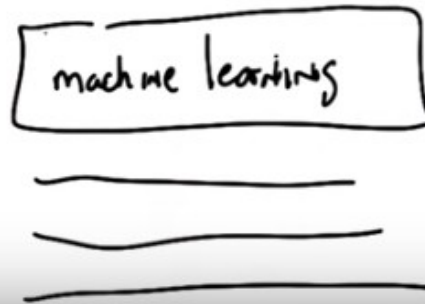
me define that for you explicitly. So as before with the feature subset, we said that we were taking a bunch of features or a bunch of instances that were in some individual feature space and transforming it into another feature space of size M . Yup. >> And in the, the feature selection problem, m was meant to be less than N , hopefully much less than N . And that's typically the case of feature transformation, though as we'll discuss towards the very end, it doesn't have to be. But when I say usually we expect M to be less than N , usually means almost always. >> [LAUGH] Okay. And that's because of the cause of Dimensionality problem. But the difference between what we were doing with feature selection and feature transformation, because this is exactly what I wrote before, is that this transformation operator is usually something like this, in linear transformation operator. So the goal here is to find some matrix P , such that I can project my examples, my points, in my instance space, into this new subspace, typically smaller than the original subspace. And I'll get some set of new features which are combinations in a particular linear combinations of the old features. Does that make sense? I think so. So then that P matrix would want to be N by M . >> Yes. >> So the transpose would be M by N and that multiplies by the N dimensional feature space in X and projects it out to an M dimensional feature space. >> Right. >> Okay. Yeah. Pretty good. So if we wanted to write that out. We could say that feature selection was about taking features like X_1, X_2, X_3 and X_4 and finding a subset like X_1 and X_2 . And that would be feature selection. But feature transformation would be taking something like taking X_1, X_2, X_3 , and X_4 , and translating into something like $2X_1 + X_2$. Which creates a single new feature ($2X_1 + X_2$). Which is a linear combination of the subset of the. Does that make sense? >> Yeah, but it could actually make other feature combinations as well, right. It's just projected down to one. >> Right, it could be projected down to two (features), or it could be projected down to three or could even project it out to a different four. >> Okay. >> And in principal you could imagine that you could even project up into other dimensions which is something that we've done before. Conceptually, anyway. >> When we talked about kernels (SVM 裡面)? >> Yeah, although those were typically non-linear transformations implicit in the notion was doing a non-linear feature transformation but we even did it before we even learned about kernels. Very second thing I think we did. Perceptrons. How do perceptrons go into a higher dimensional space. >> Well, when we talked about XOR. >> Oh, right. >> What we effectively did was we showed that we could project the original. Two dimensional space into what looks like a three dimensional space where the third dimension was a combination of the first two. And then you could actually do it with a linear separator. >> But that wasn't a linear transformation, that c was a nonlinear transformation. >> Right. Because we were talking about Boolean verbs. >> Yeah. >> But, in the end of the day it was still a kind of feature transformation and in this case a feature transformation where we went to a higher number of features. And today, what we're going to be talking about is linear transformations as opposed to non-linear transformations. And we're going to be focusing specifically on the case where we're trying to reduce the number of dimensions. So the implicit assumption here, right, the kind of domain knowledge that we're bringing to here or the belief that we're bringing here is that. We don't need all N features. We need a much smaller set of features that'll still capture all the original information, and therefore, help us to overcome the curse of dimensionality. It's just that that smaller subset might require bringing in information across the various features. Okay? >> Yeah. >> Alright.

FEATURE TRANSFORMATION

W442

INFORMATION RETRIEVAL

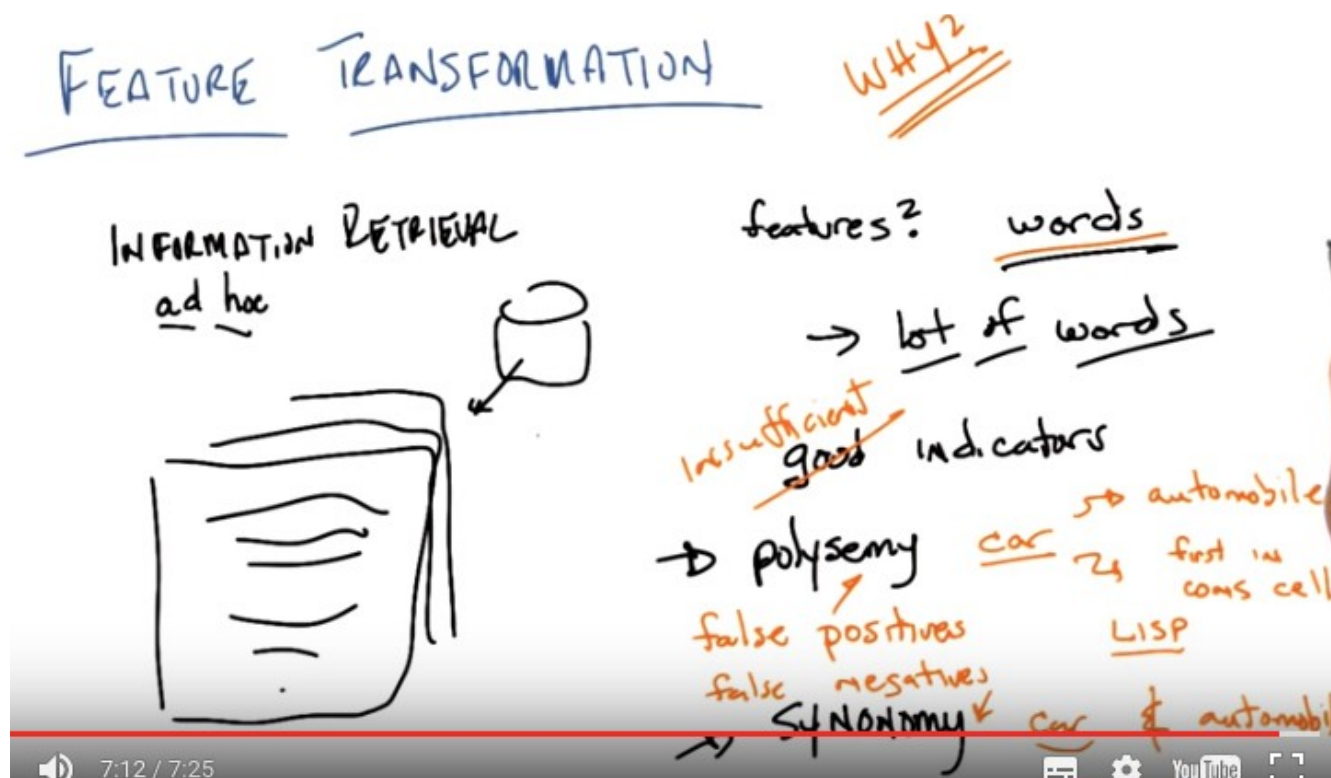
ad hoc
google



Lucky?

2. Okay, Michael, so you might ask me why we would do feature transformation, and I think the answer is obvious even to the most casual observer. We actually just went through a couple of examples in the previous slide, like X or the kernel methods, all these things we've done before where effectively we were doing feature transformation. In fact, you could argue that things like neural networks. Already doing feature transformation. So we've been doing feature transformation all along, and so, the real question here is, *is there some specific reason why we should do this feature transformation explicitly and separate from some specific learning algorithm that is doing its own feature transformation such as a perceptron or a neural network, or support vector machines.* So I'm just going to very quickly go through *a simple example* that I hope illustrates *why you might want to do this*, but it's still kind of a large relevant example okay. You with me? >> I'm ready. >> Okay, so here's the problem, I'm going to describe, and I'm not going to do it in a lot of detail. I, I really just want to give you an intuition. *That problem is information retrieval. So in particular, I'm talking about what's called, ad hoc retrieval.* Do you know what the ad hoc retrieval problem is? >> I assume that if you start with a key, and you add hoc to it, you get hockey. >> No, that would be a kind of feature transformation. We would call that the pun transformation. *The ad hoc retrieval problem, maybe I can help you out by calling it by another nickname, it's the Google problem [LAUGH] which is, you have a whole bunch of documents in some database somewhere. This is me drawing a database and you want to retrieve exactly the subset of documents that are relevant to some query. [SOUND] So this is me drawing a data base. So I might type in a word like machine-learning, into some text box somewhere, click on some button, and then get back a list of relevant documents* nicely sorted against the white background. >> [LAUGH] *And this is the ad hoc query problem*, the ad hoc retrieval problem. I give you some features, which in this case are words, and you return objects, in this case documents, that are somehow relative to the, relevant to those features. Okay? So we do this all the time. >> What makes it ad hoc as, as opposed to what? There's various other kinds of information retrieval problems, this one's just called that ad hoc one. *What's ad hoc about it is you don't already know what the queries are.* You aren't doing something like clustering the documents. It's just that you're going to be given some unknown query, which is in some feature space, and you're just going to have to retrieve the things that

are relevant so. Its just always been called the ad hoc problem as far as I know. I guess because you don't know what these are going to be beforehand. >> Yeah, and like as opposed to maybe some kind of standing query, like I want any information that talks about the OMS program, you know, please alert me. >> Right, exactly, and that's actually got a name too, it's not the standing query problem, but it's a kind of triage problem, I can't remember what it's called right now. But yeah, so there's different versions of this. This one (ad hoc) is in some ways the hardest, because you can't do a lot of very clever things up front, in order to figure out the best way to store your documents, because you don't know what this ad hoc query is going to be. Okay? >> Yup. >> All right. So. Why is this problem hard, other than by saying I don't know what this is going to be? So let's think about it in terms of the features that you have.



3. Okay Michael, so what are our features? When we have a bunch of documents let's say full of words? >> I'm not sure, it could be things like the number of words in the document? >> That could be, but what sort of, and that's actually perfectly reasonable thing but what's the simplest set of features that you might start with from which you might then compute more interesting features? >> Well there's a super, duper big set of features which are the words. >> Right, so in fact that is exactly what we have, is we have words. Maybe we have punctuation and, you know. Words are sort of the obvious thing to do. I typed in machine learning, why don't I just return every single document that has machine followed by learning. >> Maybe you should. >> Maybe I should and maybe that's a perfectly reasonable thing to do. In the case of machine learning, should I return documents that contain the word Machine but don't contain the word learning? >> I wouldn't score them very highly. >> I might not score them as highly, but I might still return them as being at least more relevant than documents that contain neither Machine nor Learning. >> Right, that are just about turtles, say. >> Right exactly, although it's turtles all the way down Michael. >> I see. >> OK. So if our features are words, which are the sort of the most obvious things to get at, we can compute other things like the number of words or whatever. But basically our features are going to be words or counts of words or something like that. As it's sort of a. Reasonable first step. And, in fact, the very early retrieval systems, which, you know,

predate both of us, actually Michael, used simple things like words. Now, there's a lot of way, details to this you could imagine. Like maybe you'd want to transform all plurals into their singular version, get rid of words like the. And there's a bunch of complicated stuff you might want to do. But, it's not particularly relevant for this discussion. So, just assume whatever you want to assume about the kind of words that you have. Okay? >> Okay. >> Okay. So, [what's the problem with using words?](#) Can you think of any problems with using words? >> [Well, there's a lot of them.](#) >> Right. That's actually the first thing. There's a lot of words. [Which means there are a lot of features. Which means the curse of dimensionality is going to hurt us.](#) >> I would think that they'd be pretty good indicators of meaning, except I guess there's kind of [two complimentary problems. One is that some words mean more than one thing.](#) >> Mm. I like that. So we have good indicators. Words are good indicators of meaning because, you know. The words, but, you could be in the case where you have words mean multiple things. You said there were two problems, what's [the second one.](#) >> Sort of the opposite, which is, [you can say the same thing using completely different words.](#) >> yes. So that's that many words mean the same thing. In particular, words, [the fact that words can have multiple meanings. It's called polysemy. The fact that many words can mean the same thing is called synonymy.](#) So, [can you think of a word that might have multiple meanings? That indicates the problem of polysemy?](#) >> Well, so, you know, learning, in the machine learning example, learning. Sometimes refers to, this statistical process that we've been talking about. But it also refers to the thing that, people do. And in some, in some scenarios, it actually means to teach. Like, I'm going to learn you something. >> That's true, but that's just too on point. I'm going to pick something else. That I think you will appreciate at the end. Let's think of a word like car. So [car has multiple meanings,](#) Michael. >> Hmm. >> Can you think of them? >> I'm mostly stuck on one at the moment. Unless you're thinking of. >> Which one is that? >> I'm thinking of [the vehicle that you drive around.](#) >> Yes. >> But I guess [one could also be referring to a list deconstructor in LISP.](#) >> Yes, exactly. It's the first in what are those things called? >> CON, CON cells? >> CON. Thank you. Right, so car is either an automobile or [it's the first element in a cons cell,](#) which all of you people who've heard of lisp knows is an awesome reference. [LAUGH] For those of you who've never used lisp, which is clearly the greatest language ever written in all of history, or that ever will be written. Because it is a superset and subsumes all other languages, including natural languages. Imagine this word, instead, is apple. And so apple can refer to a fruit, or it can refer to a computer company, or to a music company for that matter. >> Hm. >> But, I prefer car. So, car can mean automobile, or it can mean the first element in a cons cell. If you're using Lisp. Now sticking to this car example [synonymy is a similar problem in that car and automobile often refer to the same thing.](#) So you see this. So polysemy is a problem multiple things and synonymy is a problem meaning the same thing. And a particular word like car in this case. Can cause both polysemy problems and synonymy problems. >> Yeah, I can see that. So in the example you gave before about machine learning, you would, if you'd just returned documents that had machine and learning right after each other, then you'd miss all the stuff on, for example, data mining. Right, in fact, that's a very good example so, there's a huge split in the community between those who care about data mining and those who care about machine learning. And often the people in one camp don't believe the people in the other camp are doing what they're doing. But for the person who isn't religiously committed to one of those camps or the others, when you talk about machine learning you probably also care about data mining. When you talk about data mining you probably also care about what people inside the field might call it instead of machine learning. And so you would be missing out on a whole swath of papers or interesting discussions if you happened to put in the word machine learning. But similarly if you put in the word like data mining you might end up getting all kinds of documents that are about the data or about the data than you get when you literally mine for local. Who knows? And it would cause you huge problems for getting the exactly relevant set of documents that you want. So we can actually talk about this in terms of, the kind of errors that you get in say supervised learning, [what kind of errors do polysemy give you,](#) Michael? >> False, well there's false positive and false negative and this one of the

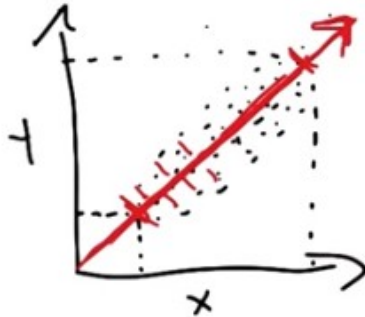
things where we, its going to return things that aren't relevant, its going to say they're positive when they're not. So you've given me the answer, false positive. Okay. >> Looks like false positive. By contrast synonymy gives you what? >> True positives. >> No, >> No wait, I negated the wrong word, false negatives. >> Exactly. >> In other words it's going to, wait is that right? Does the, so false negative means, oh it's going to tell you something's not there, when actually it really is. >> Right, so >> Typing in car will not just give me the automobile articles about my Tesla, which is a black P85. With black leather, [LAUGH] fully loaded and is like the greatest car on the planet, but it will also give me really awesome, but in this case not what I'm looking for, documents about LISP. Meanwhile, when I type in car, I will actually get all these things on automobiles. But I won't get articles that talk about automobiles without actually using the word car. So you can imagine that these sorts of problems come up all the time. You've got a set of features, in this case words, which have this problem that although they're good indicators. They are insufficient, that is we have a set of features that will generate false positives and false negatives on their own and more to the point, doing feature selection will not solve this problem. I can throw away a bunch of irrelevant words and even useless words, but I am still going this problem of generating false positives. For our polysemy and false negatives, or synonymy. And this goes beyond simply, information retrieval and text retrieval into any generic problem where you have possibly a large set of features that have this problem with false negatives and false positives.

(本段圖和上段圖一樣)

4. So, what we're going to get out of feature transformation is the ability to combine these features, these words, somehow, into some new space where hopefully we will do a better job of eliminating our false positives and false negatives by combining words together. So, we're going to leave this example for now, but just to give you an intuition about how a proper kind of feature transformation might help you just let me point out that if I type in a word like car, you would expect, since I know what car kind of means. Let's say automobiles in this case, I should pick up documents or score documents higher if they don't contain the word car, but they do contain the word automobile. Or perhaps they contain a word like motor or race track, or Tesla, right, that somehow words like Tesla, and automobile, and motorway, and anything else you can think of that has to do with cars probably are highly correlated or highly related to cars in some way. And so, it would make sense to combine words like automobile and car together into new features to pick up documents that are somehow related together that way. Does that make sense? So that's the intuition I want you to think of for the three algorithms that we're going to go over next. Okay? >> Yeah, I could definitely see how synonymy is going to be a win, when we index the documents, if we include, well, if we map them down to a lower dimensional feature space where one feature's used for anything sort of car related. I'm not sure how that's going to help with polysemy, but I definitely see how this feature transformation idea could be a win with synonymy. The way, so that's right, and I think it's, it's much easier to see how it works with synonymy than with polysemy. The way it will, it could in principle help you with polysemy is that it will combine a bunch of features that together eliminate the ambiguity of any particular word. So as you type in more words together it'll start to pick those thing up while also eliminating synonymy. So we'll see. The way it's going to work in practice actually is that if you think of it. Now we are talking about unsupervised learning. But if you think of this as a supervised learning problem, then you can imagine how you can ask yourself how to combine sets of these features, these words together such that they can still give you correct labels. And if you can solve that problem you will end up solving both polysemy and synonymy. Or at least minimizing their impact. >> Cool. >> OK. So lets go over this specific example to far more abstract examples and talk about three specific algorithms. .

Principal Components Analysis

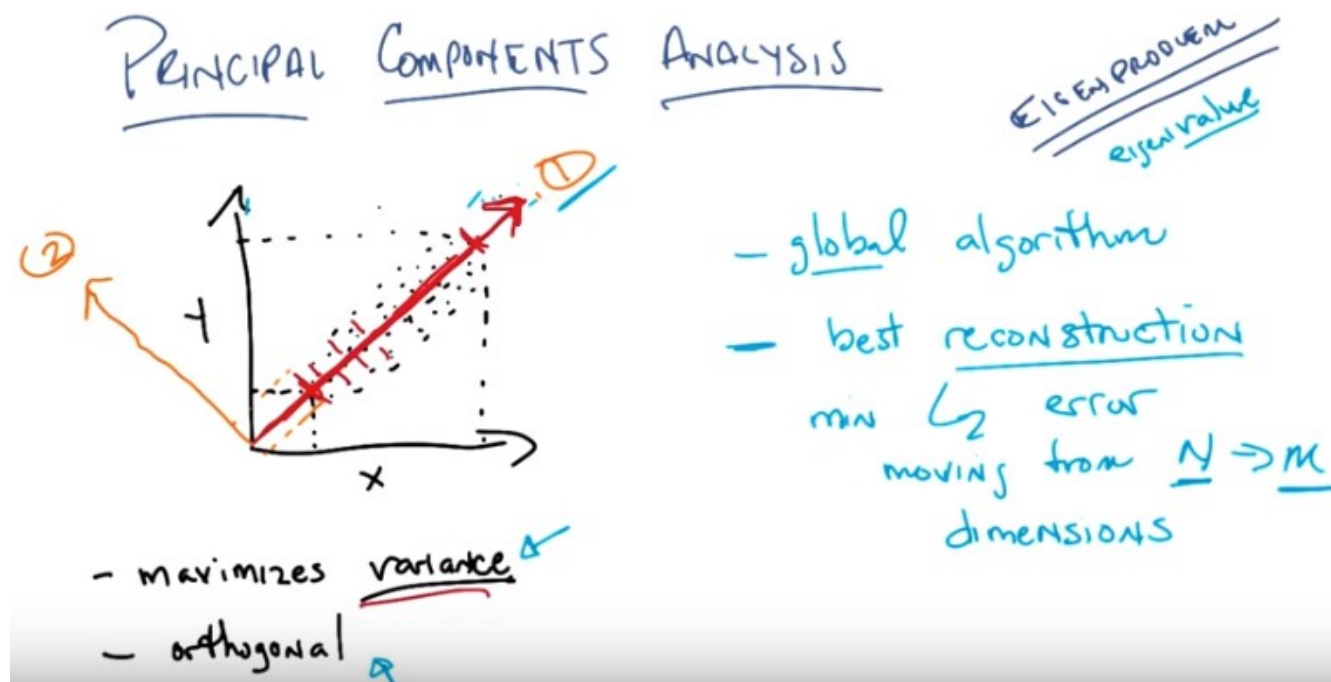
Eigen Problem



- maximizes variance
- orthogonal

5. Okay, so the first linear transformation algorithm we're going to discuss, is something called Principal Components Analysis. Okay Michael? >> Sure. >> Now just to be clear here, the amount of time it would take me to derive Principal Components Analysis and work it all out in its gory detail would take forever and so I'm not going to do that, I'm going to leave that to reading. But I want people to understand what Principal Components Analysis actually does and what its properties are, okay? >> Yeah. >> Okay, so, Principal Components Analysis has a long history. It's a particular example of something called an eigenproblem. >> Mm. >> Okay. It's a particular example something called an eigenproblem which you either already know what that means or you don't. And if you don't then it means you haven't read the material that we've given you so I'm going to ask you to do that. But, whether you have or have not let me just kind of give you an idea of what Principal Components Analysis actually does. And I think the easiest way to do that is with a very simple two dimensional example. So here's my very simple two dimensional example. All right, so you see this picture Michael? >> Yep. >> So this is a bunch of dots sampled from some distribution that happens to lie on a two dimensional plane like this, okay? >> Yep. >> Now, what, so this is in fact two dimension. So we have two features here, we'll just call them x and y . We could have called them one and two, it doesn't really matter, this is just the x, y plane. And let me tell you what Principal Components Analysis actually does. What Principal Components Analysis does, is it finds directions of maximal variance. Okay, does that make sense? >> Variance of what? >> The variance of the data (我的理解: 兩點之間的 variance 即為它們在二維空間中的距離. 我們要找的, 就是一個方向(或向量), 將所有點投影在此向量上, 這些投影點在此方向上的 span 出來的最大距離, 即為 variance). So if I had to pick a single direction here (沒指到哪裡), such that if I projected it onto that dimension, onto that direction, onto that vector. And then, I computed the variance. Like, literally the variance of the points that were projected on there. Which direction will be maximal? >> I would think it would be the one, that is sort of diagonal. It kind of, blobs along that particular direction. >> Right, that's exactly right. And, and to see that, imagine that we projected all of these points onto just the x dimension. That's the same thing as just taking that particular feature. Well, if we projected all of them down, we would end up with all of our points living in this space (x 軸上虛線間的區域). And when we compute the variance, the variance is going to be something that captures the distribution between here and here (x 軸上虛線間的區域). Does that make

sense? >> Yep. >> Similarly, if we projected it onto the y axis. Which is the equivalent of. >> Those are examples of feature selection. >> Yes, of fea, it's exactly, it's a, it's a feature selection. It's equivalent of just looking at the second feature here, y. I'm going to end up comput having a variance that spans this space between here and here (y 軸上虛線間的區域). By contrast, if we do what you want to do, Michael and we pick a direction that is about 45 degrees if I drew this right. We would end up projecting points between here and here (紅線上兩個粗刻度間的區域). Now, it's not as easy to see in this particular case but the variance of points as they get projected onto this line will have a much higher variance than on either of these two dimensions. And in particular it turns out that for data like this which I've drawn as an oval that you know sort of has an axis at it's 45 degrees, this direction or axis is in fact the one that maximizes variance. So, Principal Components Analysis, if it had to find a single dimension would pick this dimension. Because it is the one that maximizes variance. Okay, you got it? >> Sure. >> Okay. Now. What's the second thing what's the second component that PCA or Principal Components Analysis would find? Do you know? >> I don't know what you mean by second. This is now a direction. That ha, that has high variance. >> Yes. >> No. >> The first one. >> Yes. >> because it seems like you know either the x or the y is pretty high or something that looks just like that red line but it's just a little bit tilted from it would also be very, very high. >> Right, that's exactly right so, in fact what Principal Components Analysis does is it has one other constraint that I haven't told you about. And that constraint is, it finds directions that are mutually orthogonal. So in fact, since there are only two dimensions here, the very next thing that Principal Components Analysis would do, is it would find a direction that is orthogonal or we think about it in two dimensions, as perpendicular to the first component that it found. >> I see. So there really only, really only is one choice at that point. >> That's right. Or. You know there is two choices because you, doesn't matter which direction you pick in principal.

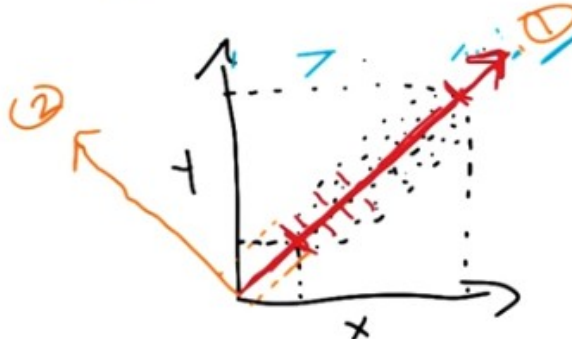


6. So this is called the first component, or the principle component, and this is called the second, or second principle component of this space. Okay, does that make sense? >> Yep. >> Now, here's what's interesting about principle components analysis. You might ask me exactly how you do this. There are several, several mechanisms for doing it. For those of you who have dealt with linear algebra before,

something like this singular value decomposition might be familiar to you. It's one way of finding the principal component. But principal components analysis basically has a lot of really neat properties, so let me just describe some of those properties to you. The first property is, well, the two that I've written here. It finds directions that maximize variants and it finds directions that are mutually orthogonal. Mutually orthogonal means it's a global algorithm (是 algorithm, 不是 maximum). And by global here I mean that all the directions, all the new features that they find have a big global constraint, namely that they must be mutually orthogonal. It also turns out that you can prove. Which I will try to give you a little bit of evidence for. But I'm going to prove formally, that the PCA actually gives you the best reconstruction. Now what do I mean by best reconstruction? What I mean is, if you think of each of these directions that it found, in this case. You found this one first and found this one second. The first thing you, I, I hope you see is that if I return these two dimensions, I have actually lost no information. That is, this is just a linear rotation of the original dimensions. So if I were to give you back these two different features, you could reconstruct all of your original data. You see that? >> Wait. When you say features you mean, what we're going to give it is, for each of those little black data points, we're going to say how far along the red axis is it and how far along the orange axis is it. >> Right. >> So it really is just a, a, a kind of a relabeling of the. of the dimensions. Right, so just like when I think about these points in x and y space, the original feature space, whenever I give you value here for this feature I'm just describing how far along a black dot is on this dimension or this projection. Now the second dimension or the second feature just tells me how far along a dot is. On this particular dimension or axis, and similarly about projecting on the read and on the orange, I'm telling how far along a point is along this axis and along that axis. So if I were to return, if I were to take X and Y and transform them into this new one and two, I would have given you different values than I did from X or Y, but they're actually just the same point. And so I've thrown away no information. >> So that's a pretty good reconstruction. >> That's a pretty good reconstruction. But what principle components analysis does for you. Is if I take just on of these dimensions, in particular, the first one, the principle component, I am guaranteed that if I project only in to this space and then try to reproject into the original space. I will minimize what's called L2 error. So do you understand that? >> I'm not sure let me check. So, you're saying if we instead of, all right we take the little black dots, they now have a red dimension and orange dimension one two, and now if we reconstruct using only the first dimension, I guess it just puts the black dots on the red line. >> Yep. And so there is no, they have no existence in that second dimension. >> Correct. >> And now you're saying of all the different ways that I could do that to kind of project it to a, to a linear sequence. This is the one that's going to have the smallest. L2 error which if im not mistaken is the same kind of reconstruction error we talked about in all the other times we talked about reconstruction. So it's like squared error. That's exactly right, its squared error. This particular notion is called a [UNKNOWN] norm but that's really just talking about distance. What this means is that if I project onto this single axis here, and then I compare it to where it was in the original space, the distance, the sum of all the distances (準確地說, 應該是 sum of all distance error squared, 其中的 distance error 即 '紅線上的 distance' 減去 '原本在二維空間中的 distance') between those points will actually be the minimal that I could get for any other projection. >> Cool. >> And you can, you can sort of prove this (其實很好理解, 因為這些點基本上都是按紅線的方向分佈的, 故沿紅線方向的距離誤差最小). It kind of makes sense if you just think about the fact that points. Always start out in some orthogonal space. And, I'm basically finding in scaling and a rotation such that I don't lose any information. And I maximize variance along the way. By maximizing variance, it turns out I'm maximizing or maintaining distances as best I can in any given dimension. And, so, that gives me the best reconstruction that I can imagine. Now, you might ask yourself, Is there anything else nice about principal components analysis given this reconstruction error? And there's another property of PCA that is very, very useful. And it boils down to the fact that it's an eigenproblem. What happens when you do principal components analysis is you get all of these axes back, and in fact, if you start out with N dimensions, you get back N dimensions again, and the job here for a feature transformation as you

might recall, is you want to pick a subset M of them hopefully much smaller than N . Well, it turns out that associated with each one of these new dimensions that we get. Is its eigenvalue (我的理解: eigenvalue 可以理解為 distance 之和, 即所有點在該方向 span 出的區域長度). That eigenvalue is guaranteed to be non-negative, it has a lot of other neat properties. But what matters to us here is that the eigenvalues monotonically non-increase, that is, they tend to get smaller as you move from the principal to the second principal, to the third, to the fourth, to the fifth, to the sixth, and so on to the n th dimension. And so, you can throw away the ones with the least eigenvalue. And that's a way of saying that you're throw away these projections, or the directions, or the features, with the least amount of variance.

PRINCIPAL COMPONENTS ANALYSIS filtering EIGENPROBLEM eigenvalue



- global algorithm
- best reconstruction
- min L_2 error moving from $N \rightarrow M$ dimensions
- \emptyset eigenvalue \equiv ignorable
- well studied \rightarrow fast
- classification

- maximizes variance
- orthogonal

4:48 / 5:01

7. Okay, wait. So let me see if I can echo some of that back. So, it's almost as if what we're doing here is we're doing a transformation into a new space where feature **selection** can work. >> Exactly, and in fact, here's something kind of interesting for you. It turns out that if the eigenvalue of some particular dimension is equal to zero, then it means it provides no information whatsoever in the original space. So, if I have a direction, if I have a dimension that has an eigen of zero, I can throw it away and it will not affect my reconstruction. >> I'm trying to remember if that makes it irrelevant or useless. >> Well, it certainly makes it irrelevant (因為 irrelevance 是關於 information 的, usefulness 是關於 error 的). If there's no variance, that's the same thing as saying it has zero entropy, because it never changes. So it's irrelevant. Now whether it's useful or not, well, probably not, but it might be useful for something like, our simple, perceptron example that we used last time. Gotcha. >> Okay. So does this all make sense? >> So one question I have is in the example that we just kind of worked through there was a blob of data and when we drew the red line through the maximum variance direction it went through the xy origin. >> Um-huh. >> Does it have to? Is it necessarily the case that it's going to? Or you know is the algorithm restricted to have to put things through the origin? >> Well so my answer to you is that, that's actually a very complicated question. And in principle it, so to speak it, it doesn't really matter. But in practice what people do and their doing something like PCA is they actually subtract the mean of the data, or the central of the data from all the data points. So it ends up being centered around an origin, for the origin. But what this means is that you can then interpret what we're doing is finding

maximum variants. As capturing correlation. >> Okay. That's helpful. >> And otherwise, if you don't do that, what you end up with is effectively a principle component that at least intuitively kind of captures the notion of where the origin should be, and it's not terribly helpful for what it is we're trying to do. So my answer is, no it doesn't, and yes it does. >> [LAUGH] >> Okay? >> Sure. >> Okay. So, that's basically PCA. It's got all these neat properties. Let's sort of summarize them again. It's, well, actually. Let me add a couple beyond these. It is a global algorithm. It does give you best reconstruction error. Which seems like a very nice thing to, thing to have. And, it tells you, which of the new features that you get out are actually important with respect to this notion of reconstruction by simply looking at their corresponding eigenvalues. They also have a couple of other practical properties. In particular, it's very well studied. And what that means in this case is that there are very fast algorithms that do a very good job, even in weird cases. Even with large data sets, at least if they're appropriately sparse, of being able to compute these things. People have been working on this problem again, since before we were born Michael, and they've gotten really good at finding principle components even for what would be very difficult spaces. But this does lead me to a question though which is another question, which is okay, so you've got an algorithm that probably gives you the best reconstruction. But what does it have to do with classification? This is kind of like the question we had before about relevance versus usefulness. So we find a bunch of projections which are relevant in the sense that they allow you to do reconstruction. But it's not clear that if you threw away some of these projections, the ones with low eigenvalue, that even though you'd be able to reconstruct your. Original data is not clear that would help you do classification later, can you see how that would work out? >> Sure, I mean like, it just could be that that's not where the information is about what the labels ought to be. >> Right, so imagine for example that one of your original dimensions is in fact directly related to the label and all the rest are just, say Gaussian noise. But the, the variance of that particular direction is extremely small. >> It might end up throwing it away. >> It'll almost certainly end up throwing it away. Which means you'll end up with a bunch of random data that doesn't actually later help you do classification. And that's because this looks a lot like, if I can do the analogy, a filter method (沒理解). >> I was going to say that, yeah. >> Oh. So this is kind of like filtering. And in fact it's going to turn out that the other two items we're looking at too are like filtering although their particular criterion might be more relevant. We'll see. Okay? Alright, so do you understand Principal Components Analysis? Again, I'm not asking you to understand exactly how you would run a principal components analysis algorithm. That stuff's in, in all the text, but just to sort of understand exactly what is is trying to do, what it's trying to accomplish. >> Yeah, I think so. So, it's, it's taking the data, it's finding a different set of axis, that are just like regular axis in that they're mutually orthogonal. But it lines up the variance of the data with those axis, so that we can drop the least significant ones, and that gives us a way to do feature selection. But the whole thing is a feature transformation algorithm, in the sense that it first moved the data around, to be able to do that. >> That's exactly right. So, you do transformation into a new space, where you know how to do filtering. >> Got it. >> Excellent.

INDEPENDENT COMPONENTS ANALYSIS

- PCA ~ correlation, maximizing variance \Rightarrow reconstruction

- ICA ~ independence

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \end{bmatrix} \rightarrow \begin{bmatrix} y_1 \\ y_2 \\ \vdots \end{bmatrix} \quad \begin{array}{l} y_i \perp y_j \\ \cdot I(y_i, y_j) = 0 \\ \cdot I(\underline{y}, \underline{x}) = \uparrow \end{array}$$

8. Okay Michael, so the second algorithm that we're going to look at is just like principle components analysis, except it's called Independent Components Analysis. Okay, and the major difference is really the difference between the first word principle and independent. So the main idea here is that PCA is about finding correlation. And the way it does that is by maximizing variance. And what that gives you, is the ability to do reconstruction. What independent components analysis is doing, or often called ICA by those in the know, is it's trying to maximize independence. Very simply put, it tries to find a linear transformation of your feature space, into a new feature space, such that each of the individual new features are mutually independent and I mean that in a statistical sense. So, you are converting your $X_1, X_2, X_I \dots$ And there's some new features space let's call it I don't know let's call it a $Y, Y_1, Y_2, Y_I \dots$ Such that each one of the new features are statistically independent of one another, that is to say their mutual information is equal to zero ($I(y_i, y_j) = 0$). Does that make sense? >> And this is going to be a linear transformation? >> It's going to be a linear transformation. >> To make them statistically independent. >> So, I find some linear transformation here, which is going to take my original feature space, which I'm representing with these X 's, and transform it into a new feature space, such that, if I were to treat each of these new features as random variables and compute their mutual information, I would get for all pairs a mutual information of zero. That's part one and the second thing that it's trying to do is that it's trying to make certain that the mutual information between all of the features, y and the original feature space x is as high as possible. So in other words, we want to be able to reconstruct the data. We want to be able to predict an X from a Y and a Y from a X . While at the same time making sure that each of the new dimensions is in fact mutually independent. In a statistical sense. >> I think I'm going to need an example.

INDEPENDENT COMPONENTS ANALYSIS

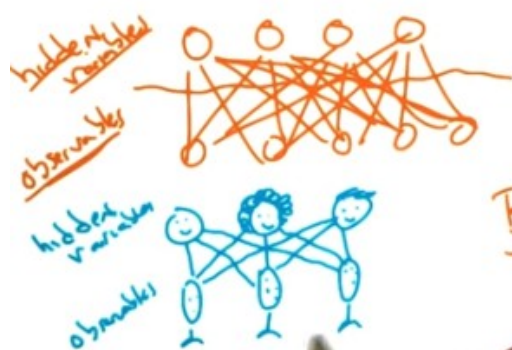
- PCA ~ correlation, maximizing variance \Rightarrow reconstruction

- ICA ~ independence

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \end{bmatrix} \rightarrow \begin{bmatrix} y_1 \\ y_2 \\ \vdots \end{bmatrix} \quad y_i \perp y_j$$

$$\cdot I(y_i; y_j) = 0$$

$$\cdot I(\underline{y}; \underline{x}) = \uparrow$$



Blind Source Separation
(COCKTAIL PARTY)

9. So it is important to get an example I, I think that the, the best way to kind of see an example here is to draw a little picture for you that tells you what the sort of underlying assumption behind ICA is. So here's the kind of, or at least the way I see what the fundamental assumption of ICA is. And that is this. You're, you're assuming that there's a bunch of things in the world. And these are going to be hidden variables. And they've got some properties that, that are associated with them. They're random variables. They are mutually independent of one another. That is if I know the value of one it doesn't tell me anything about the value of the other. But we don't know what they are. And what we get to see are observables (不是 observers). These observables (都是 hidden variable 的 linear combination) are given rise to by the value of the hidden variables and they somehow combine in a sort of linear fashion and our job, the job of any learner in this case, any unsupervised learner in this case is given your observables try to find the hidden variables. Under the assumption that these hidden variables are in fact independent of one another. So, what's a concrete example of that? I'm going to give you a concrete example of that then I'm going to give you a demo that we found on the web. Okay? >> Okay. >> So, one problem where we know this is true is something called the Blind Source Separation Problem. So what's the Blind Source Separation problem? It also has another name which is the Cocktail Party Problem do you remember the Cocktail Party Problem? >> I think so, yeah. >> So, describe it to me. >> So, not that I go to a lot of cocktail parties, but if you're in a large group of people like in a cafeteria or something like that. >> Mhm. >> And you're trying to listen to a conversation. It can be very challenging because theres all these different conversations happening simultaneously. >> Mm-hm. >> And we need to be able to pull out that one source that, that we're, that we're caring about, so that we can listen to it while kind of separating it out from what all the other noise sources are. >> So what you just described to me is that somehow you have a bunch of people, so there's Charles, there's Michael and there's Pushgar. And there all talking at once. And let's imagine in place of ears we have microphones. And these microphones are actually recording everything that well everything that they hear. So this means that this first microphone is going to hear me. The second microphone is also going to hear me and so does the third microphone... However, they're going to each hear me at a

slightly different volume at a slightly different delay. Okay? Because they're placed sort of randomly around the room. You with me? >> I think so, yeah. >> Okay, surely, Michael, who just keeps talking and talking and talking mostly about puns, is also going to be picked up by each of the three microphones, and Pushgar, mainly complaining that we aren't doing enough quizzes, is also going to be heard by the three microphones. In this case if we look at our other examples, the people are the hidden variables. That is, they're the things that are causing events to happen in the world. But what we have are observable's are the microphones, and what it is that they're recording. Now, if we're in a small enough room, it turns out that physics tell us that, even though our voices are all nonlinear and sound doesn't travel in a very linear fashion the way that we would like, it turns out that in a small enough room with all of the reflections and everything, each of these microphones are well modeled as receiving unknown linear combination of all of the sources. So this means that each one of these microphones actually contains a, a different linear combination of each of our voices. Hey, you with me? >> Yeah, it's interesting. >> Right. And so, really, if I were to give you these three recordings and I wanted to figure out what say, Michael Lipman was saying, all that information is here, but I can't extract Michael Lipman from microphone one, or microphone two, or microphone three. But given all of these microphones, I can in fact recover, just Michael alone or just Charles alone or just Pushcar alone, because ICA exactly tries to find this notion of mutual independence (三個人說的話是互相獨立的) without losing any information from the output. And this model here, of three people talking, mostly independently, and being recorded by linear combination by different microphones is exactly the model that ICA was designed to represent. So, just to sort of drive this home, let me give you an actual example that we found on the web that they do exactly this problem. Okay? >> Sure.

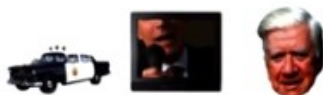
COCKTAIL PARTY PROBLEM

Imagine you're at a cocktail party. For you it is no problem to follow the discussion of your neighbours, even if there are lots of other sound sources in the room: other discussions in English and in other languages, different music, etc... You might even hear a siren from the passing-by police car.

It is not known exactly how humans are able to separate the different sound sources. Independent component analysis is able to do it, if there are at least as many microphones or 'ears' in the room as there are different simultaneous sound sources. In this demo, you can select which sounds are present in your cocktail party. ICA will separate them without knowing anything about the different sound sources or the positions of the microphones.

ORIGINAL SOUND SOURCES

By clicking the icons you can listen to the original sound sources.



SAMPLES AT THE COCKTAIL PARTY

Listen to the mixtures by clicking the microphones.



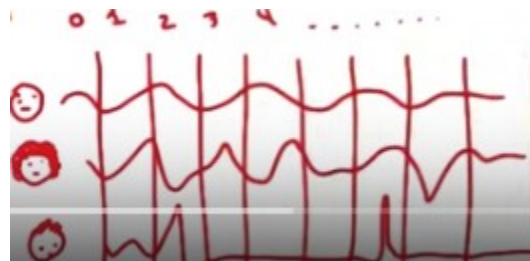
FOUND SOUND SOURCES

Below are the sound sources separated by ICA. Note that they might be in different order than the original ones.



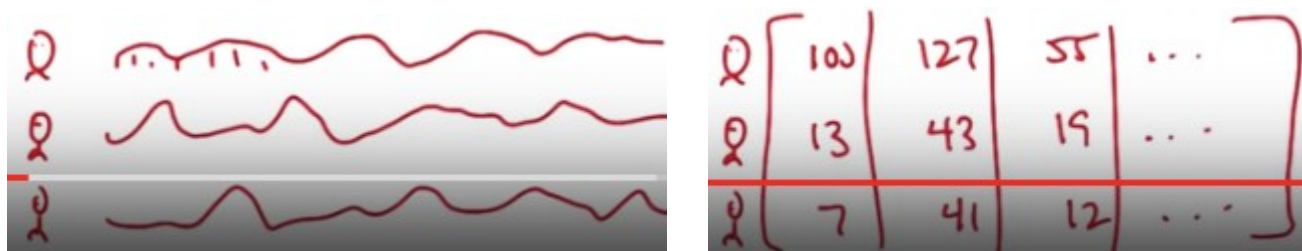
10. Okay, Michael, so I pulled up this webpage. We will provide a link to it, on our own webpages. And you'll notice there's a really nice copyright here, and this is all free to use. So I'm not, I'm not doing anything wrong here. They're going to use Independent Components Analysis, as a way of recovering, original sounds. So, here you see that I've clicked onto a police car. Somebody talking in a commercial and, let's say, this dude here. And, what it's going to do is it's going to, generate sounds from each of these three sources that are all independently generated. And, it's going to mix them. So if I click on these icons here, of these microphones. See, these look just like what I drew before. You'll be able to

hear each of their sounds combined together. So I'm going to put my microphone very close to it so you can hear it. [SOUND] [FOREIGN] And here is another one of the microphones. [SOUND] [FOREIGN] Okay, so, Michael, could you hear that? >> Yes. >> Okay. So, what I did is, I played two of the, the, the mix sounds together. And, you know, to the untrained ear, they sound very much alike. But I think the, the most important thing is, all three of these sounds are over on top of one another, and as human beings we actually do a pretty good job on being able to focus on one of them, because I don't know, we're designed to do that. But machines have historically had a terrible time of doing this. And independent components analysis was the first sort of generic algorithm that was able to solve this. So, we're going to use ICA now to separate the sound sources. And as you see, we get these little gramophone things. And I'm going to play, each one of these and see if they did a good job of separating those into the original sources. [FOREIGN] The guardians of the electronic stock market NASDAQ who have been burned by past ethics questions are. [SOUND] Okay, so wasn't that kind of cool Michael? I mean we, [we took these completely mixed up sounds and we were able to recover the original, by using independent components analysis.](#) >> That's fascinating. I don't still understand what it's doing, but it's pretty neat that it could do that, because it sounded pretty mixed up to me. >> Well it was, it was in fact each one of these three sources, that we eventually heard were in fact arbitrarily, linearly, combined into each of these separate microphones. And there's really sort of no way in which you'd be able to recover the original sounds, because, there's no reason for you to be able to do that. But people do it all the time. And now with something like independent components analysis, you can also do it. And the reason it works with independent components analysis. And that fundamental assumption that it's making, is that whatever is generating in these cases these sounds, the sources, are statistically independent of one another. Which happens to be true in that case. And they're combined together in a way that is a linear combination. Now there's a lot a details here. And again when you read the materials you see exactly how independent components analysis works. But, intuitively all it's doing is taking this particular model, and by using mutual information directly to find things that are independent of one another, while not losing any information, from the observables, it's able to reconstruct these sounds. And it's able to do this incredibly quickly, and incredibly well. Under a large number of conditions.



11. Okay Michael. So let's try to be a little bit more detailed about, kind of, the mechanics of how you, how you would make this work. The algorithms for finding Independent Components Analysis, are in all the reading. But, I do think it's pretty easy to kind of get lost. If you don't, sort of, turn these matrices into actual numbers. So, let me just, sort of, give you [a quick example of how we would do this specific thing here.](#) And see if that helps, okay? >> Mm-hm. >> Okay. So, [how would you go about turning this into](#) a problem that something like, you know, [an actual algorithm that uses numbers could do.](#) Well, it turns out it's fairly straightforward. And let's just take this, this particular example here, with people talking into a microphone. Basically you create a matrix. Which are samples of all of the sounds. So, in our original space, we have, you know, this handsome person here. We have this other, let's say, similarly handsome but in a different way person here, and we have this other person, with my poor attempt to draw hair over here, and they're talking. Well, what we end up doing is we basically take the sound wave and we sample it. And what does it mean to sample it? Well, you know,

you represent sound in a computer as a sequence of numbers, just like you represent pictures as a sequence of numbers, and in fact, you represent words as a sequence of numbers. And so we basically turn these sound waves into a matrix full of values.

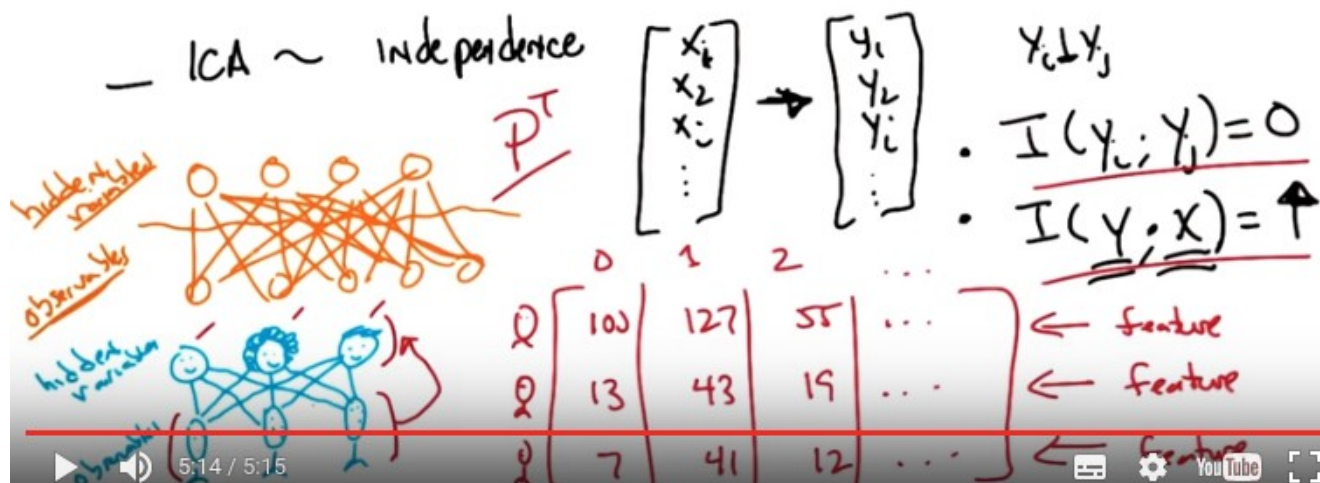


So, as I described before, Michael, each one of these microphones. Is actually getting a linear combination of speech from each of these three individuals. So if we think of microphone one, it is seeing some kind of sound wave, which is again a linear combination from each of these people generating a sound wave. The same is true for microphone two and similarly. For microphone three. Now, of course, we're talking about recording these and we're thinking about computer programs, which means we take this continuous sound wave and we sample it at a very fast rate. And what we end up with is a sequence of numbers that represent the particular pressure that we're getting in these sound waves.

INDEPENDENT COMPONENTS ANALYSIS

- PCA ~ correlation, maximizing variance \Rightarrow reconstruction

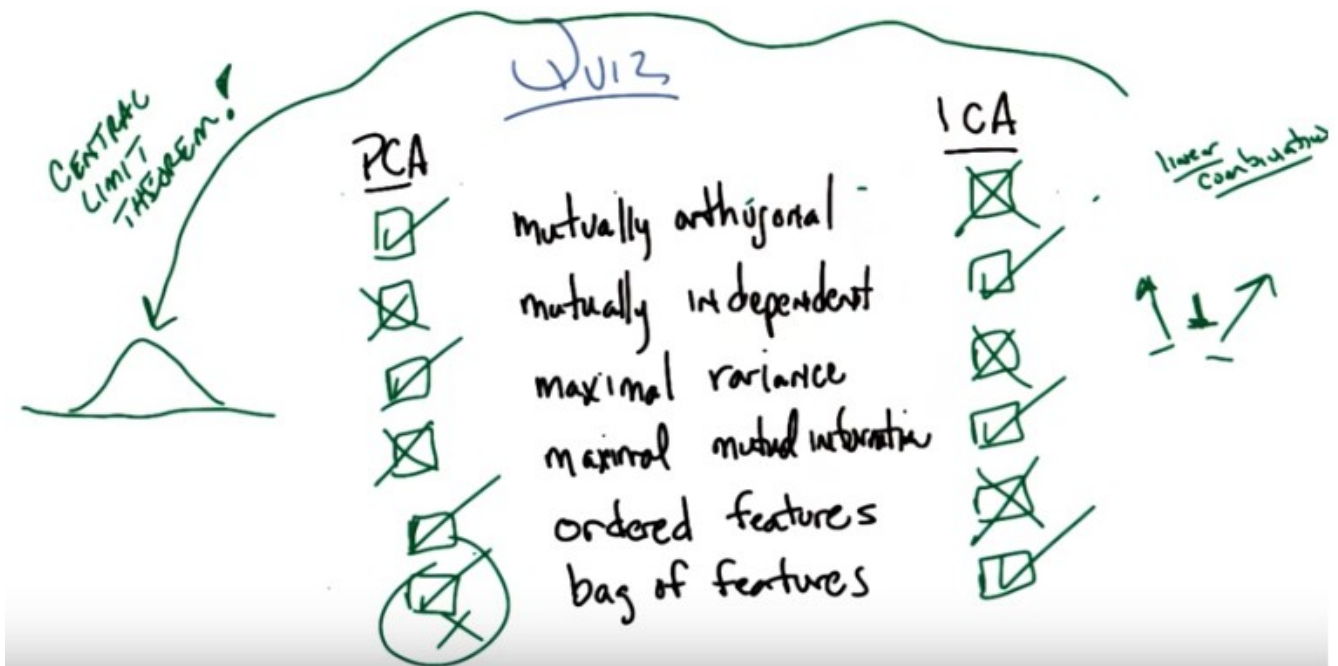
- ICA ~ independence



So, these sound waves get converted into a sequence of numbers and I'm just going to make up some numbers for this. And now what we have is a matrix where each row represents a feature in our original feature space. And by original here, I mean, the feature space that we see each of our microphones. And every column represents a sample. Say at time 0, time 1, time 2, and so on. Does that make sense? >> Yes. And so, since each of these is a linear combination of these voices we get here, our goal is to find a projection like we did before and the original definition of the problem, such that if we projected this on to here, we would end up with a new feature space that corresponds individually to each of

these people. And you could do this with anything, it doesn't have to be sounds, but if we think about the Adhoc query problem that we went through earlier, each of these would be words, and say their presence of words would be your features, and each of these columns would be documents, say. And the goal would be to find a projection given a set of documents that allowed us to recover some underlying structure that was useful for classification, or for information retrieval. Does that make sense? >> Yeah. And can you say, what does it mean for that sequence to have mutual information? >> so as you recall. Because I know that you listen to Push Cars. A discussion about mutual information and entropy and so forth. All mutual information is a measure of how much one variable tells you about another variable. And, the assumption here is that each of these people talking has no mutual information. That is, they're talking independently of one another, or the sounds that are coming out of their mouths, don't allow us to predict the sounds that are going to come out of another person's mouth. So it may mean that these people are talking to one another as we often do, Charles might be talking to Michael, who's talking to push cars, who's talking back to Michael, who's talking. To Charles, but the exact sound waves that we see are actually independent of one another. So if that turns out to be true, what independent component analysis is trying to do is trying to recover features, in this case. It turns out the corresponding individual speakers, such that their sound waves, or the values that you see, are statistically independent of one another. Okay? And that's what this does. But, since we can always come up with arbitrary projections that are statistically independent, it's important that whatever our new transformation gives us, it actually has some relationship with the original values that we saw. So somehow we want to be able to learn from this, into this in a way that we don't lose any information. In much the same way that PCA was trying to minimize the loss of information, and so we not only want each of the individual new features, in this case people to be statistically independent, we want the amount of data that we get from these people. Compared to the amount of data that we got originally from the microphones, to actually strongly predict one another. And if the mutual information between the microphones and our candidates for the people's voices have high mutual information, then it means that we haven't lost anything. And so those two things together, those two constraints, forces into a situation where, if this model is true, we (can) construct the original voices.

12. >> Okay Michael, so I'm going to see if you understood that fire hose of words that I threw at you by giving you a quick quiz. >> Thanks. >> Yeah, you're welcome. So, what I have down in the middle here is a bunch of phrases that might describe PCA, might describe ICA, might describe both of them or might describe neither. Okay? And what I want you to do is check off under PCA each one of these that applies. And check off under ICA each one of these that applies. Okay. Does that make sense? >> Sure. >> Okay. Go.



第三個是 maximal variance, 不是 maximal relevance

13. Okay Michael you got answers for me? >> Think so. >> Okay good. Alright so lets look at the first one. Mutually orthogonal. Does that apply to PCA, ICA, both, or neither? >> So, it was one of the defining properties in PCA, so I would say PCA. >> Okay that's fair enough. That is actually correct. What about ICA? >> I don't, I don't know. It's not one of the defining features. It wasn't, it wasn't even thinking about orthogonality. >> That's right. And in fact ICA by finding independent. Projections are almost all but guaranteed to find ones that are not mutually arthugoial, it doesn't care at all. So in fact **this is what makes PCA a global algorithm since it has this global constraint of mutual arthuguality.** Where ICA really in its definition that cares about that, so this should be unchecked. >> Okay. >> I'm going to put an X there to represent unchecked. Okay, got it? >> Yeah. >> Okay. What about mutually independent? >> So, that was how ICA was trying to construct its, I don't want to say features, yeah I guess the, the transformed features. >> That's right. >> It was trying to create them to be mutually independent, so I would check ICA in that case. And, PCA didn't use that language at all, so I would just not do that. >> Okay. That's fair. I will point out though, that it turns out that PCA is trying to do something that sometimes will create things that are mutually independent. But we'll see that when we answer the next question. But you're right, PCA does not care about mutually independents. Okay, what about the third phrase, maximal variance? >> Alright. Again, I feel like this was one of the defining features of PCA. So it was trying to choose dimensions that maximize variance. >> That's right, and what about ICA? >> In terms of variance? Again, the, the, there, it wasn't really discussed in that context. >> Right. ICA is specifically interested in notions of independence, statistical independence. Not inpu, not in issues of variance. So in fact ICA does not care about maximizing variants. Now that we have gotten this far let me point something out, it turns out that because of this arthugonal deconstraint this lack of independent constraint, but this gold and maximizing variance, there are cases under which PCA happens to find independent projections. What's really going on here with these three constraints or, or lack of at least one in this case, is that **PCA is tending to find things that are uncorrelated. By maximizing variance along arthogonal dimensions is finding uncorrelated dimensions.** And that makes some sens given what it's trying to do. But that is not the same thing as finding things that are statistically independent. Again there is this

particular case, there is a specific case where that does work out (即 PCA 找出的 features 是 independent 的), and that's when all of your data is in fact Gaussian. >> Oh, interesting. >> And why Gaussian? Because it turns out that the distribution that maximizes variance is in fact the normal distribution. That's my interpretation of a normal distribution. So maximizing variance means that what PCA is doing is it's trying to find a bunch of orthogonal Gaussians. More or less. Does that make sense? >> Yeah but, you were saying that it, and it aligns with ICA in that case? >> In that case yes, because it turns out that uncorrelated ends up being independent under very specific distributions. But that's a coincidence it's not a normal fact. >> Ha, ha. >> But by the way this, this is probably worth pointing out here something that at least I think is kind of interesting here. Which is since ICA is only trying to find things that are independent of one another. Here's our little symbol for independence. As opposed to things that are uncorrelated, things that are Gaussian. It turns out that whatever it is PCA is doing, it is not working under the same underlying model as ICA. Lets think about what ICA is trying to do. ICA is trying to find a bunch of these prjections all of which are statically independent. Right? >> Mm-hm. >> What happens if I take a whole bunch of statically independent variables and I add them together? In other words I create linear combination of them. What am I going to get? A bunch of sums of things that are independent? Right and what does that tend towards in the limit? >> I want to say that the law of large numbers tells us that it turns normal. >> That's exactly right. If I take a bunch of independent variables and I sum them together, that is, I create a linear combination, I in fact end up with a Gaussian. And that is the central limit theorem. So one argument you might make is that if you believe in a world where there are independent causes giving rise to observables, which is what ICA believes, then whatever you do, you should not be maximizing variance, because you're guaranting the summing together otherwise independant variables (意思是此時不能用 PCA, 否則得到的是這些 feature 的 sum, 而不是這些 independent features. 第 14 段的紅字是一個很好的例子). Oh, I see. I see, I see. So by trying to find things that are maximal variants it's trying to mix together through the central unit theorem all these things that are independent, so it's, it's, it's, it's, specifically not teasing apart the independent things, it's trying to smush together the independent things. >> Right. Under certain assumptions about the distributions of these individual variables. So another assumption that I see us making is not just that these variables are independent, but that they are highly non-normally distributed. And if that's the case, then ending up with things that look like Gaussians, has got to be exactly the wrong thing to do. If that assumption holds to be true. Okay. What about maximum mutual information? >> My understanding of what you describe for ICA said that this is what it's trying to do. It's trying to find a new feature space where the features are maximally. The mutual information between them is large as possible. So I would check the ICA in that case and not the PCA. >> Let me be Let me clarify soething you said. How can it be trying to maximize mutual information while also trying to make things that are mutually independent. I think you would describe them both in the same language. So what is it trying to make mutual independent? Different features. The, the new, the new transform features. >> Right. So each of the new transform features is independent with all the other new transform features. So what is it trying to maximize mutual information between? >> Oh, the, I see. The the information that content of the original features and the new features. >> Right. So this is about joint mutual information between all the original features together and All of the transform features. There it's trying to maximize mutual information. But inside the new transform features it's trying to make them pair-wise mutually independent. >> Alright. I think I said that wrong because I understood it wrong. So thank you for clarifying. >> You're welcome. But now you understand it right. >> Maybe. >> Okay, let's go with that because you got the checkmark right. What about PCA? >> Just X that. I don't understand what that would mean. >> Right. So if I were to put something here that it could get a check mark for, what I put down is maximal reconstruction. >> right. >> Right. And notice that maximal reconstruction of your original data is not the same thing as maximizing mutual information. Thought of course in the limit they work out to be the same. >> Interesting, okay. >> But the one project that maximizes variance is not necessarily the same as the first projection you find for

maximizing mutual information. So these things really are doing two completely different things. Okay, last two what about ordered features? >> So, in PCA, it was actually assigning, you know, taking the maximum variant to dimension first and then the next, >> Mm-hm >> You know, after that's been subtracted out, whatever has the largest remaining variance and so forth. So that the features end up coming out in, in a very specific order. And it has the property that you could drop the, the last group of features if you want to still have as high a reconstruction area as possible, given the number of features that you keep. So I would check PCA for that. >> Okay, good. What about ICA? >> You didn't say anything about the ordering or how you'd actually find features. It seemed like in the blind source separation example you gave It just, came out with the three, so I'm going to say not ordered. >> That's right and in fact, if you think about the blind source separation example, how in the world would you order people anyway. I mean, other than in the obvious way. It just doesn't really mean anything. I say it doesn't have a notion of causes being more important than other causes merely that they're independent. So, it doesn't really worry about ordered features. It turns out in practice That you can actually try to order the features by using something called kurtosis. Which is the fourth central moment of a distribution. But that's really just something that's useful in some specific cases, almost by coincidence. ICA, itself, does not particularly care, about ordering. At least not classical ICA. Okay, what about the last one, bag of features? >> So, I would view what you just said about ICA as implying that what ICA produces is a bag of features. There's no particular ordering to them. >> That's right. >> It's just a collection of things that make up the whole. I guess, you know, PCA, after you've thrown away whichever features you don't want. The features that remains are just features. They could be treated as a bag, I guess. So I don't know if I would check that or not. I. For symmetry I guess I would say not. >> Okay, but I'm going to say yes because in ordered set of features is still a bag of features. But we would accept either. Either a check or an x, both are sort of fine. So then, what do we really learn from this, Michael? I think what we've learned is these things (PCA & ICA) have fundamentally different assumptions and are really trying to do completely different things. >> Okay. >> The only thing they have in common is that they're still trying to capture the original data somehow. >> Alright. I understand that, but I also learned the opposite, which is that they are really closely related and are trying to do very similar things. >> Yeah. But their underlying models are different. So maybe that, that's actually a good point, Michael. So maybe a better way of saying it is, their sort of fundamental assumptions are different. Even though they're trying to do the same thing, which is capture the original data in some new transform space that is somehow better. But if you think about it that way, there are two different optimization functions, two different fitness functions, two different cost functions. So even though they're trying to do the same thing. Reconstructing. Keeping the data around. Their basic assumptions about the way that data is constructed is very different. >> Okay. >> Okay.

PCA vs ICA cont'd

→ BSS ✓ICA ✗PCA

→ directional ✓ICA ✗PCA

→ FACES



→ NATURAL SCENES

→ ICA

← edges

→ DOCUMENTS → TOPICS

BSS: blind source separation problem

14. >> Okay, so, let me give you just a few more examples, Michael, of a, how PCA and ICA differ. And I'm going to do this mainly by talking about certain things that ICA does. And this is really just for your edification, but I think it really helps, to think about, those sort of underlying models and how they differ, by thinking about how they react differently to different kinds of data. So, just, [here's a couple of examples](#). The one we covered right away, was the blind source separation problem. And of course, what, what we recall is that ICA was, in some sense, designed to solve the blind source separation problem and in fact ICA does an excellent job at solving the blind source separation problem. Meanwhile, PCA does a terrible job at solving the blind source separation problem and that's just because it's assuming this kind of Gaussian distribution, it just does not recover what's going on. But here's something that ICA does differently from PCA. Is because it's got this kind of blind source root in it, it's actually directional. And what I mean by that is, you recall I drew this sort of matrix before, where we had features this way (每行). And we had samples of those features like sound, time samples of sound this way (每列). It turns out that for PCA, it doesn't matter whether I give you this matrix or I give you the transpose of this matrix. It ends up finding the same answer. >> Hm. >> And that should make sense, right? Because it's just basically finding a new rotation of the data, and these are effectively just rotations of each other. For the purposes of, of, if you just kind of think about it geometrically in space. ICA on the other hand, gives you completely different answers, if you give it this (上面的矩形) versus giving it this (下面的矩形). So ICA is highly directional. And PCA much less so. But ICA, because it's making this kind of fundamental assumption, does end up generating some really cool results. I'm going to give you another example of one. That's like a blind source separation. In fact, I'm going to give you two. Okay? So, imagine you had a bunch of inputs of faces. Okay? So, here's my input faces. I give you bunches and bunches and bunches of faces. What do you think PCA

would do? What do you think the first principle component of PCA would be over pictures of thousands and thousands and thousands of faces. >> Over all darkness of the image. >> Actually that is exactly right. The first thing that PCA tends to do with images, we are actual, we are talking pictures, not just sketches here, is it finds the direction of maximum variance and that tends to be brightness or luminous. Which kind of makes sense because that's typically the kind of thing that kids vary the most. So in fact, the first thing people often do when they're trying to use PC on faces is they normalize all of that away. Because the first principal component isn't terribly helpful. It's just kind of giving you the average light. What do you think the second thing it would find it would be? >> Hair versus not hair? >> No. Interestingly enough. What it ends up finding is sort of the average face. Which is kind of like the question you asked me earlier about what happens if you go through the origin. >> Mm. >> So, there's actually names for this. It's called the Eigen Faces because, you know, as I noted before, it's an Eigen problem. And it basically finds kind of the average face. And that's kind of weird. But it works for reconstruction. I don't know what the average face here would look like. Probably something like this (問號右邊那個). Yeah, so you can see how useful that would be. Not even clear that's a face. But it works for reconstruction. Do you know what ICA ends up finding? >> Noses? >> Yes, it finds noses. It finds eye selectors. It finds mouth selectors. It finds hair selectors. And, I think, intuitively the way I would think about this is because PCA is doing this global orthogonality thing, it's going to be forced to find global features. ICA I didn't say was global, and that's because it's basically local, it doesn't care about orthogonality. So it tends to find parts of. If you feed it the metrics the right way, it tends to find parts of. And so it ends up finding these little selectors. So this has a really nice outcome in cases like natural images or natural scenes. So what do you think happens Michael, if I take natural scenes, you know, pictures of the forest and grass and walking around. Just things that I would see if I were just walking out in the world and taking bunches of pictures. And I feed it into ICA. Well the answer for PCA, by the way, is the same as before, is you get brightness, you get sort of the average image, things that really make sense if your goal is to do reconstruction. But ICA actually gives you something different. What do you think the independent components, the underlying causes, so to speak, of natural scenes are? >> I'm still thinking about the face parts. But, by analogy, it seems like it would be the, you know, things in the world like, trees, and rocks, and ground. >> Not quite, and that's I think in part because, there are too many of those things that kind of overlap in too many different ways. It actually finds something more fundamental. >> Edges. >> That's exactly right. ICA finds edges. Now for me, that is incredibly satisfying. It says that the independent components of the world are edges. Now there are two things that come out of this. One is just the satisfying feeling you get by realizing that in it there's a algorithm that, on it's own, recovers something fundamental like edges. That's very nice. But the second thing that's nice about it is, once I use ICA to do my feature transformation and discover that what it's learning are edge detectors, well then I can just write edge detectors. I can just write algorithms that are very fast, very efficient, that can go through images of natural scenes and pull out the edges. It's unclear how to do that with, you know, different principled projections, but if edges are the fundamental building blocks of natural scenes then there are people who know how to write edge projectors very quickly. By the way, you get a similar result, I won't talk about it, but you get a similar result in our information retrieval problem where you have documents. And you can read this in the material we gave you. But what ICA ends up giving you for documents are topics. And they're very easily interpreted that way. Collections of words that select for specific topics that are themselves made up of collections of words that select for topics and collections of uncorrelated words that get rid of distracter documents. So the independent components of scenes are edges and the independent components of documents are topics. And that feels very nice, but in both of these cases, both with edges and with topics, it turns out that the form of these topics are something that you can compute very, very quickly. And sort of independently of the underlying ICA algorithm. So why does that matter, Michael? Well, remember what I said originally about unsupervised learning and what it was good for, was understanding your data. Doing kind of data analysis from a human being's point of view.

So what something like ICA, and other algorithms we might imagine, do, is they allow you to do analysis over your data, to discover fundamental features of them, like edges, or topics or noses and eyes. And then once you understand that's what's making up your data, you can simply write code that will select for it, or figure out what the important parts of your data are correspondingly. And so here, we haven't just done this feature transformation problem for the sake of doing classification, feature transformation actually helps us to understand the fundamental underlying causes, or at least structure of our data.

ALTERNATIVES

- RCA = Random Components Analysis
 → generates random directions!
 $N \Rightarrow M$ $M \ll N$ $P^T X$ works?!
 $m > n!$ → correlations

15. >> Okay Micheal. So we've talked about PCA and ICA. And they both work remarkably well in the specific domains that they're designed for. And they've been applied for decades on a wide variety of problems for doing this sort of feature transformation. But I'm going to just very briefly describe two other alternatives, and sort of give you a notion of the space, okay? >> Sure. >> Okay, the first one is kind of irritating, but I feel obligated to share it with you. And it's called, well it's got many different names, but I'm going to call it RCA just because I like the symmetry. And RCA stands for random components analysis. So what do you think random components analysis does? This is also called random projection. >> I'm going to guess, instead of finding dimensions with, say, high variance, it's just going to pick any direction. >> That's exactly right. What RCA does is it generates random directions. >> Then I guess it projects the data out into those directions. >> That's exactly right. It's like saying it picks a random P to a random matrix to project your data onto. This matrix is, in some sense, just any random linear combination. And you want to know something? It works. It works remarkably well. >> At what though? In terms of like reconstruction? >> At reconstruction. Well not particularly well at reconstruction. But you know what it works really well, it works really well if the next thing you're going to do is some kind of classification. >> Hm, >> Now why is it you think that it actually works? Can you imagine why just picking a bunch of random directions and projecting onto those random directions might work? >> Well, it does mix things together differently. I don't know why the original data wouldn't work. Then this would work better. Unless the original data is somehow is purposely made to not work. >> Well remember what we're doing here, right. We're starting with N dimensions. And we're projecting down to M dimensions, where M is significantly lower than N. So, I started with a bunch of dimensions. Now remember, the real problem here is not that I can't gather the data from the N dimensions. It's that there's a whole bunch of them, curse of dimensionality. >> Yes. >> So I need to have a lot of data. So if I don't have a lot of data, at least certainly not an exponential amount of data, so to speak, and I project a lower dimension, why would random projections still give me something that helps with classification? >> So it's, it's as if it's

maintaining some of the information from these other dimensions, even though there's fewer of them now. They're all kind of mixed together, but they signal might still be there. >> That's exactly right. And because you project into a lower dimension, you end up dealing with a curse of dimensionality problem, which is sort of the whole point of this, or one of the whole points of this in the first place. And really, a way I think of summarizing what you're saying is, that [this manages to still pick up some correlations. So if I take random linear combinations of all of my features, then there's still information from all of my features there.](#) So in practice, at least in my experience, Michael, [it turns out that \$M\$, the number of lower dimensions you project into, for a randomized components analysis, or randomized projections, tends to be bigger than the \$M\$ that you would get by doing something like PCA. So you don't end up projecting down to sort of the lowest possible dimensional space. But you still project down to a lower dimensional space that happens to capture your correlations, or at least capture some of the correlations, which often ends up working very well for a learner or a classifier down the road. You can actually see how, in this case, you might even project into another set of dimensions \$M\$, where those number dimensions are actually bigger than the number of dimensions you started out with. This, in some sense, is almost exactly what we did with perceptrons in solving XOR. Basically, you're projecting into higher dimensional spaces by doing this. Does that all make sense? >> Yeah, I think so. I mean it makes sense to me that it would be not as efficient in some sense, as PCA, because it sort of reminds me of, you know, if I want to, if I want to paint my wall, I can very carefully paint all the little pieces of it. Or, I could just splatter stuff at it. It generally takes more when you splatter because you're not being as systematic, but it does, ultimately, cover your wall. >> \[LAUGH\], Yeah. I think that's an interesting analogy, and I'm going to go with an apt one. Okay, so, this sort of thing works. What advantages does it actually have over PCA and ICA? Can you imagine one? There's one in particular which I think sort of jumps out at you. >> RCA? Well, I don't know. Is that, is that a good quiz question maybe? >> Sure, let's make it a quick quiz.](#)

ALTERNATIVES

- RCA = Random Components Analysis

→ generates random directions!

$N \rightarrow M$ $M \ll n$ $P^T X$ works?!
 $m > n!$ → correlations

QUIZ
 BIG ADVANTAGE
 OF RCA?
FAST

16. Okay, so here's the quiz question then. [What is the big advantage of RCA](#) and there is a single word to describe it. So your job Michael is to look into my brain and imagine what word it is I'm thinking of and I gave you a hint by saying it jumps out at you, which means if you don't come up with it you're going to feel really bad. [LAUGH] Okay, so you ready? >> Ready. >> Okay. Go!

17. >> Okay Michael, what did you come up with? >> I have a bunch. Well, so, the big advantage of ICR, RCA, it's cheap. >> It is cheap. >> It's simple. >> It is simple. >> It's easy. >> It is easy, in a sort of comical or complexity sense. >> Sort of practically free. Those are all words like, free, easy.

Those are, those are the words that came to mind. You're saying that that's, none of those is your word. >> No, none of those was the word that I was coming up with. >> How about, works? >> No. >> It did start with the letter f, though. >> Famous. >> No. >> Foul mouthed. >> That's two words. >> Friendly. >> No. >> Alright, I need more letters. I'm still going back to famous. Fabulous. Fast. >> That's right, it's fast. But I would give you, definitely give you cheap or easy. >> Cheap is like fast. >> So is easy. >> [LAUGH] Alright. >> [LAUGH] At least your, that's what the slang term meant where I grew up. Anyway, okay, so anything that captures fast, cheap, easy, whatever would do here. But what's irritating about this and actually things like k means and k-NN and a whole bunch of other algorithms, is that they're so freaking simple, and yet they manage to work. And that can be kind of annoying sometimes because, you know, as machine learning people, we want to come up with these complex things that work, but as a very brilliant man once said to me, you must earn your complexity. So, RCA, randomized components analysis, or really randomized projections, has this very simple way of thinking about the world. I'll just randomly throw things together, and it turns out to pick up correlations on the way and worked pretty well. And its big advantage, other than working sometimes, is that it's very, very fast. PCA can take hours in supercomputers, and ICA can take hours upon hours in even more supercomputers. But generating a bunch of random numbers, computers are really good at that, and it happens to go really, really fast.

ALTERNATIVES

- RCA = Random Components Analysis

↳ generates random directions!

$N \rightarrow M$ $M \ll n$ $P^T X$ works?!

$m > n!$ ↳ correlations

- LDA = LINEAR DISCRIMINANT ANALYSIS

↳ finds a projection that discriminates based on the label

18. >> So, the second alternative I just wanted to mention very briefly is something called LDA, which is short for Linear Discriminant Analysis. So, any guess on what linear discriminant analysis might do, Michael? >> It'll make, well, linear. So it's got a linear in it. Discriminant reminds me of, like, classification lines. >> Yes. So, what linear discriminant analysis does is, it finds a projection that discriminates based on the label. This actually will feel, this feels a lot like supervised learning, right? You take advantage of the label, and you come up with a transformation, such that you will, in fact, project things into different clumps or clusters, or otherwise separate them based upon their label. So, you can imagine using a lot of algorithms, as we've used in the past. In some sense, what they're doing is, linear discriminant analysis. They're finding lines or finding linear separators between different clumps of points. And if you think about the binary case, for example, you could think of SVM as doing something like this, where what you've done is, you transform your points not into a specific

label plus or minus, one or zero, yes or no, but you instead project it onto the line such that it would clump things accordingly. And you use the value of that projection as a way of re-representing the data. Does that make sense? >> Yeah, and this, this approach seems a little different from all the other ones, in that it's actually paying attention to how the resulting components are going to get used. Right. >> That's exactly right. >> It actually is going to try to help you, specifically, with the classification. >> Alright, that's exactly right. All three of the examples that we gave before, feel almost like filter methods, right? In that they have some criterion they're trying to maximize. Even though randomized projections is, who knows what it's trying to maximize besides randomness. But they don't care about the ultimate learner, or the ultimate label that's associated with them. Whereas LDA does care explicitly about the labels and wants to find ways to discriminate, finds sort of projections or features that make it easier for you to do that discrimination. Now it also does not care about what learner's going to happen next, but it does care about the label. So it does end up doing something slightly different, and works pretty well in a world where you have very simple things that can be linearly discriminated. Okay? >> Yeah. >> So that's it for the alternatives. I actually think that's pretty much it for, feature transformation. So why don't we wrap up. [CROSSTALK] >> Just a quick, just a quick question, though. [CROSSTALK] So, LDA, I've heard of LDA in this context before meaning something else, like, Latent Dirichlet Allocation? Yeah, but that happened long after Linear Discriminant Analysis and we haven't moved past the 90s. >> Well, I'm just saying that it does seem. And I think it is actually an unsupervised approach. >> Oh latent, oh no, absolutely. But it is well beyond the scope of the discussion that we're going to have here. >> All right. As you wish. >> But it is in fact. Every time you see a D you can think Dirichlet. >> Which is the computer science pronunciation. Other people pronounce it differently. >> Yes, because the correct way of pronouncing it is sort of beyond my. It has another syllable in there. Is it German or something? >> Yeah. >> Who is Dirichlet. Is it Dirichlet? [LAUGH] That sounded German. [LAUGH] Okay, so let's wrap up Michael. >> Alright. >> Okay.

WHAT HAVE WE LEARNED?

- A is for analysis! PCA, ICA, LDA, RCA
- RELATIONS BETWEEN DIFFERENT TRANSFORMATION ALGS.
- analysis of the data \rightarrow structure
- PROBABILITY VS LINEAR ALGEBRA

19. Okay Michael, so we've gone on a journey of discovery. [LAUGH] Through unsupervised learning, and so my question to you is, what have we learned? >> I think we learned a little bit about ourselves. >> And a little bit about America. So what A's have we learned today, Michael? >> So there was PCA. >> Mm-hmm. >> ICA. >> Mm-hmm. >> LDA. Mm-hmm. RCA and USA. >> [INAUDIBLE] USA, we're number one! Whoo! >> [LAUGH] >> Okay, we're just going to erase that little bit. [LAUGH] >> [LAUGH] >> Okay, yeah, okay, so we learned about a lot of A's today. >> Uh-huh. >>

Which is the same grade that all our students are going to get, I am sure. That would be great. >> Or that's if they're truly independent. If they aren't independent, then the central limit theorem says, there will be a normal distribution across grades. >> Mm-mmm. >> Mm-mmm. >> Ring that bell curve. >> Aw, yeah, baby. Okay. So we learned about PCA, ICA, LDA and RCA. What else did we learn about? >> Well, I think that was it. But we talked about specifically. we, we talked in detail about the relationships between some of these. >> Mm-hm. >> In particular, these are all examples of feature transformation. >> That's right. Okay, so we found out about the relationships between different transformation analysis. Oh, here's something we learned. We learned that the A doesn't just stand for analysis. In the algorithms, but it actually does stand for the analysis of the data. >> Because that's unsupervised learning. >> That's right. And that in particular I gave some examples where [ICA tells you what the underlying structure of the data is. You can use it to find structure. So that, for example, the independent components of natural scenes are edges.](#) So it's interesting, because I feel like the other time that you emphasized structure was when you were talking about, mimic which was a piece of work that you did when you were a graduate student. >> Yep >> One would almost want to guess that maybe you worked on ICA when you were a graduate student. >> I actually did. My very first paper as a young graduate student was on mimic and my very last paper as a young graduate student. My actual dissertation, was on independent components analysis. >> I had that sense from the number of strong points you felt the need to make [LAUGH] about ICA. >> Well listen man, really, structure runs my life. As you know, everything about my life is well-structured. >> Yeah, sure. >> [LAUGH] Okay, did we learn anything else? >> So, yeah, so I mean I feel like we spent a lot of time talking about so P, [ICA is a more probabilistic kind of modeling, method and PCA is a more I want to say linear algebraic, modeling model.](#) >> That's a really good point Michael. So, we didn't say it explicitly this way, but actually, even in our own work it often comes up that sometimes, you want to think about, information theory. You want to think about probability. And sometimes, you really just want to think about linear algebra. [And you could see PCA as being really about linear algebra. And sometimes only coincidentally being about probability. Where as ICA is all about probability and information theory, and only coincidentally ever about linear algebra.](#) >> Yeah, that's helpful. That does seem to be a fundamental split in a lot of work that happens in machine learning. >> Yeah and it, and it makes some sense. I mean, we, we know what the right answer is in probability, but we know what the right answer is in linear algebra. And I guess it's, it's often the case Michael, would you agree that. That [the linear algebra approach is often easier to think about, or easier to do in practice and sometimes,](#) it can be interpreted as if it's probability. And that typically breaks down on the edge cases, but you know, you can kind of work around it for sort of common cases. Yeah, that, that [the linear algebra algorithms are often cheaper to implement, cheaper to execute less prone to local minima issues. There's sort of a well defined answer that they're finding. But it's often not quite the answer that you want, and the probability methods give you the answer that you want, but can be very hard to find. Right. And in fact you can see that in ICA and PCA, in that PCA is very well understood. There are lots of fast algorithms for it. And you know that the principle components always exist. Interestingly, we didn't talk about this but by contrast, ICA with its more information, theoretic and probabilistic roots, has a very specific model. And it isn't always the case that that model fits, and so in fact, sometimes you can't find independent components. Because they don't actually exist, except in the most trivial sense. So it's both more expensive, because of the way you end up searching the space. And it doesn't always produce an answer. But when it does produce an answer, it tends to produce very satisfying ones.](#) >> Well, I think that's a good place to stop, in the sense that I wanted to know just one more interesting fact about ICA. And now that I've got that, I feel fully satisfied. Well, there's another fact I can tell you which is that it's the only one of these algorithms that start with a vowel. >> And now I'm more than satisfied. >> It is always my goal to leave you more than satisfied. >> [LAUGH]. Alright, then! >> OK. Well I think we are done with this entire sub lesson mini course thingy. >> About unsupervised learning. Well that, well that's great! >> About unsupervised learning. >> What does that, what does that leave us to do?

Doesn't lead us to do anything, at least not with this particular, mini-course. I think actually the description we had here, what we've learned for this particular, lesson actually applies, even going backwards to some of our other lessons. And what we're going to get to do next is, decision problems and reinforcement learning. Ooo, exciting. >> It is exciting. >> But I think first people probably have some homeworky stuff to do, projects to do in the context of this minicourse. >> Yes, and probably an exam of some sort. >> [LAUGH] >> [LAUGH] >> Good luck to everyone on that. >> Yes, we're absolutely sure you'll do fine. Be sure to go over these lessons and be sure to read all of the material, because there's a lot of detail in the material that wouldn't make a lot of sense for us to cover in this format. But do give it a read, come back, look at the stuff that we've talked about, it should help you understand the intuition behind what's really happening there. Excellent. Well this is great, thanks, thanks Charles, I learned a lot, >> I did too. Bye Michael I will hear from you soon. Alright, awesome. >> Bye.