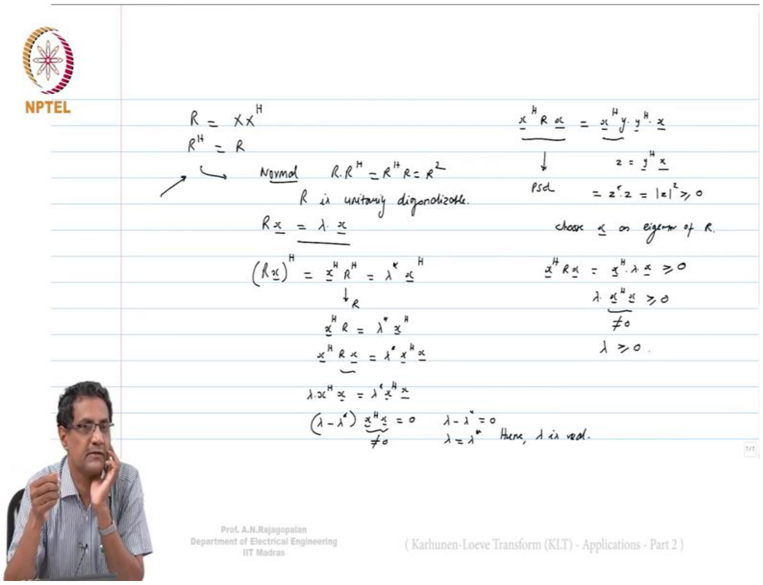
Image Signal Processing
Professor. A.N.Rajagopalan
Department of Electrical Engineering
Indian Institute of Technology, Madras
Lecture No. 54
Karhunen-Loeve Transform (KLT) – Applications – Part 2

So, after the class there will some people who asked about why is for a complex matrix right apparently I just assumed that some of these things are kind of generally know, so I will just show those things within kind of 5 minutes and then we will kind a move ahead, some of these things I sometimes assume, you know and so it is like saying that you know especially the doubt is like if you have a complex matrix, how does it work out?

(Refer Slide Time: 00:47)



$R = XX^H$

$R^H = R$

$\hookrightarrow$ Normal $R.R^H = R^H R = R^2$

R is unitarily diagonalizable.

$\underline{R\underline{x} = \lambda . \underline{x}}$

$(R\underline{x})^H = \underline{x}^H R^H = \lambda^* \underline{x}^H$

$\downarrow R$

$\underline{x}^H R = \lambda^* \underline{x}^H$

$\underline{x}^H \underbrace{R \underline{x}} = \lambda^* \underline{x}^H \underline{x}$

$\lambda . x^H \underline{x} = \lambda^* \underline{x}^H \underline{x}$

$(\lambda - \lambda^*) \underbrace{\underline{x}^H \underline{x}}_{\neq 0} = 0 \quad \lambda - \lambda^* = 0$

$\lambda = \lambda^*$ Hence, λ is real.

$\underbrace{\underline{x}^H R \underline{x}} = \underbrace{\underline{x}^H \underline{y}} . \underline{y}^H . \underline{x}$

$\downarrow$ psd

$z = \underline{y}^H \underline{x}$

$= z^* . z = |z|^2 \geq 0$

choose $\underline{x}$ as eigenvector of R.

$\underline{x}^H R \underline{x} = \underline{x}^H . \lambda . \underline{x} \geq 0$

$\lambda . \underbrace{\underline{x}^H \underline{x}}_{\neq 0} \geq 0$

$\lambda \geq 0$.

Prof. A.N.Rajagopalan
Department of Electrical Engineering
IIT Madras

(Karhunen-Loeve Transform (KLT) - Applications - Part 2)

So, it is like this so just exactly similar to the way that it works out when you have a real matrix, so R when you have a covariance R you can write it as some xx Hermitian, if it is a complex matrix, which is the form of R, whenever you say it is a covariance it comes as xx Hermitian, therefore clearly R Hermitian is equal to R, because R Hermitian will again give you xx Hermitian, which means that because of this reason it is of course normal, because A Hermitian is equal to A Hermitian A, that will get satisfied, because RR Hermitian is equal to R Hermitian R is equal to R square.

So, so in that sense we know for a fact that even if you have the complex type, it does not matter I mean the this one that spectral theorem for say this one normal matrices and all will hold. So, which means that R is unitarily diagonalizable, this follows so from this it follows that R is unitarily diagonalizable, then coming to the eigenvalues being real, if you look at let us say here if you take if you take an eigenvector x it again it can be complex, so you know that this should be some lambda times x, lambda can also be complex.

Let us say, suppose we assume that lambda can be complex, then what you can do is, you know if you take Rx the whole Hermitian, that will be x Hermitian R Hermitian which on the right hand side will become lambda star, because lambda is a scalar and then x Hermitian. And then what you can do is you can now multiply now, you know that R Hermitian is R because we showed it there.

So, which then means that x Hermitian R is equal to lambda star x Hermitian, now the simple trick that you play is you know multiply the so it multiply the multiply on the by x, so this gives you lambda star x Hermitian x know but Rx is lambda x, that comes from here, so you get x Hermitian, so lambda I can pull it out it is a scalar so x equal to lambda star x Hermitian x or this means lambda minus lambda star into x Hermitian x, it is equal to 0.

But then because because swe started by saying that x is an eigenvector, x Hermitian x, it cannot be 0, this it is not 0 because it is an eigenvector, which then forces lambda minus lambda star to be equal to 0, or which means lambda is equal to lambda star, so hence lambda is real. And then the other thing was then another thing is if you in general, if you if you take some x Hermitian and multiply need not be an eigenvector suppose we take Rx, if x is a covariance then this can this you can write as some yy Hermitian into x.

Now, since this is a scalar call call z as y hermitian, z will be a scalar now, call that as y hermitian x, now x hermitian y will then be see there so x hermitian y will then be z star, so this will be like z star z which is equal to magnitude z square, which will be a number which you solve this kind of larger than or greater then or equal to 0, so which is why we say that this is a PSD.

So, all that you must have done for real, actually just goes through in a straightforward manner, then now you right based on this condition choose x to be an eigenvector, choose x is x is a eigenvector of R eigenvector of R, then we know x Hermitian R is R axis this turns out to be x Hermitian, Rx is lambda x and since we know that in general for any x Hermitian Rx is a number greater than or equal to 0, so then this means that this number or to be have a greater than or equal to 0, or lambda times x Hermitian x should be a number greater than or equal to 0.

But since this is this cannot be 0 this norm of x it is an eigenvector, therefore lambda is then kind of say greater than or equal to 0. So, which is a reason why we had assumed all of this as (())(05:26) when we did we said eigenvalues will be real they will be greater than or equal to 0, R is PSD unit diagonlise were all this all this comes from the simple background. And then of course somebody was asking why do we choose based upon the eigenvalues? Why do we put significance in terms of in the when you pick the significant eigenvector, why is it that eigenvalues are chosen for that?

The answer is that after all right when you do a diagonalisation, so when you have this eigenvector, so this eigenvectors are actually orienting towards maximum variations. So, for example in that to say ellipsoidal kind of thing that had shown if you are if you had a distribution like that, indicate of a 2D space whether it means that the maximum variance is probably along One direction and then orthogonal to that there is another spread.

So, these eigenvectors orient because of the fact that they actually they actually represent variations, it is a variance, when you kind of look at look at lambda it represents the variance of the covariance matrix. So, therefore it is kind of telling that the variance is maximum in this direction followed by another whatever the second maximum the second direction and maybe third and so on. So, you have this order and the whole idea behind doing this eigenvalue, eigenvectors are of a decomposition is to be able to capture those say directions along which the variances are kind of maximum.

And then the idea is that if you know attach significance if you want to compute, what is the significance of the variants that comes to lambda, the orientation comes via the eigenvector, the action significance if you know how much is spread comes from your eigenvalues. And therefore you choose the eigenvalues to kind of make up your mind as to how many eigenvectors choose, which ones to choose and so on.

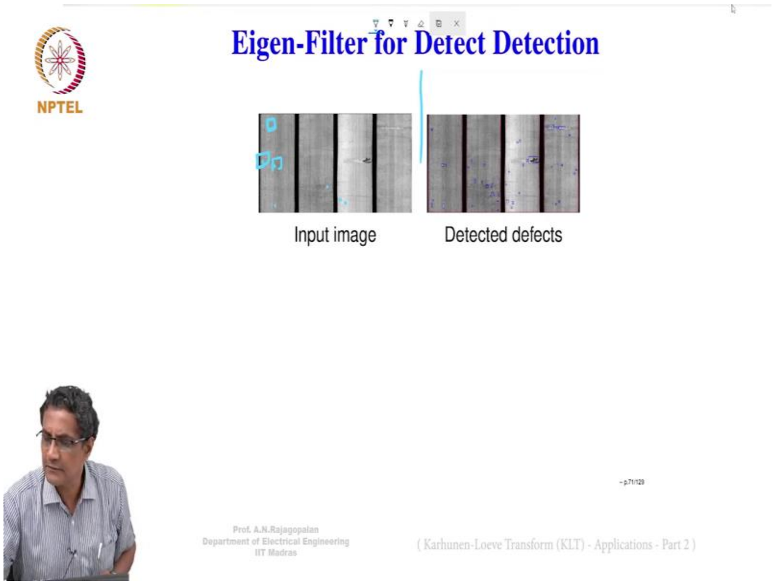
Then there was one more doubt as to how do you put the how do you fix how many eigenvectors do you want? It is very hard to tell, there is no sort of a single rule that says that universally that these many eigenvectors are enough, or these many significant eigenvectors are enough, you have to for each application it can change. For some applications you may need like I told you yesterday when we had to do this kind of face location, kind of a detection, we had to use fewer eigenvectors because the problem was one of you know trying to separate something like a background patch from a face patch.

The idea was not to reconstruct the face that well it was just to use minimum vectors on both sides for the background as well as for the face be be able to tell that you know, tell that patch a part whether it belongs to background or foreground. So, that in fact you do not want to use to use you know too many eigenvectors because to start using too many of them, then you are say reconstruction capability actually goes up in which case you will be you will find it hard to separate a background from the foreground.

So, so you have to use the notion of sparsity use fewer eigenvectors, but if you are looking at a recognition problem, sub delighted you wanted to identify somebody then maybe you know, maybe it makes them more sense to make sense to use more number of eigenvectors there because you want to reconstruct properly and then you want to know whether you got you got you got the person's face correct or not.

So, there you may need more, so all of this is totally application dependent there is no there are no general rules is to know when to stop or where to start. So, people have different ways of kind of say defining it, some people might say that look at this look at the highest eigenvalue then go down till about a fraction of that you know somebody may say 5 percent 10 percent 20 percent all are that is completely in your hands depending on the application.

(Refer Slide Time: 08:48)



Eigen-Filter for Detect Detection

- p.71/129

Prof. A.N.Rajagopalan
Department of Electrical Engineering
IIT Madras

(Karhunen-Loeve Transform (KLT) - Applications - Part 2)

But you know what if you see this image on the left, this is an input image and then there is something like an output image on the right, ignore the one on the right first. Now, if you see this image, what do you think this image is? Of course, it is hard to guess, it is actually the image of the rim of a wheel, this was a problem that somebody somebody approached us long ago 10 years ago, so so so they said that when they manufacture a rims of wheel you know, so what happens is sometimes those rims may have faults in them, that could be small kind of can say defects.

So, for example, I do not know whether you can make out but there is a defect there is defect there, there is a defect here, there is a defect here and so on, there is a there is a defect I think somewhere here, so you have the small small kinds of things which are very hard to I mean that you have to strain your eyes to see those see those kind of a kind of say defects. So, they said can we kind of build something that can identify those small little small little see defects.

And at the same time you should be able to ignore other things, you know, which are kind of more or less more or less more or less kind of what you say and also something like this right this pattern that you see there, it just looks like a slightly darkish pattern on that image, you cannot classify that as a defect, because that some pattern that is running on the wheel but some of these other things for example, this might be kind of classified as a this one defect and things like that.

Now, what they said was, so in this case what we did was, we tried everything, like let us say anyone would do, we tried edge detection, whatever, we tried typically that is what you would do some kind of sort of the thresholding, whether you know, that will kind of tell these apart but what happens is if you set the set the threshold to low then even things that are that are not actually defect will show up and so many of them will show up, it is again the same problem like in faces, you know, there is one face you do not want to show that are 4 faces, similarly there is 1 defect or if there are 3 defects you cannot claim that there are 7 of them.

Again, then the whole, you know rate what you call the old significance of your algorithm gets lost. And at the same time if you if you set a threshold that is too tight then what happens is you might actually miss a defect, which is which is which is very expensive mean you cannot afford to miss a defect. So, what do you think we did?

On the right there is sort of a result which looks pretty so what you do is, you know so you kind of look at this image and it is because they have already pointed to do you pointed out to you which are the defective parts, so what you do is you take a small patch, around this like for example, you might take a patch like that which could be whatever they the size and all you have to fix up, these are all sort of ad hoc.

But let us say 16 cross 16 or something and then and then when you get to move across you will get several such 16 cross 16 patches. Now, what you do is you go through this and then you take the second image again look at all those reasons that are clean, go through all those patches and now what you have is something like that like like sort of you know a data matrix that you have wherein wherein you have all the all the sub images or the patches that that sort of see represent represent actually clean data.

Now, you can now because of the fact that you can have those overlapping patches and because they are small in size you can get you can get a 100 I mean easily around let us say 5, 10,000 effect more, depending upon the size of the image you can even get get more numbers, then what you do is you compute a covariance matrix out of that, then after you compute the covariance then you again look at the significant eigenvectors. And then what you now do is you come back and here what now you go back revisit every patch right in a kind of a moving window fashion and then for every patch you try see how well you can get your u hat.

Because you are going to not reconstruct it exactly, you are going to throw away a lot of eigenvectors, take only the significant ones and then when you compute this u minus u hat norm, the moment the patch lands on lands on something which has a defect, you will automatically see that this error error will kind of go up, very similar to what I said is yesterday with respect to locating a face versus a non-face, so when you land in (())(12:41) window lands in those only thing is what would happen is, when you move around you will get not just one window you will get multiple windows.

Because that patch could be that small little defect, it could be anywhere within the patch, so when you are moving there you do not know where it is. So, when you move you will get kind of multiple multiple boxes around it and that is typically removed by what is called a non-maximus separation that is something out of this, but that is what you do in order to be able to be able to you know merge all of them into a single box, I mean you do not want to flag, 7 boxes when there is actually 1. And that is how you do and I know and incidentally what I wanted to point out was this is called an Eigen filter by the way.

And Eigen filter is not very good of well-known at all, if you look at books there is hardly anybody that talks about Eigen filters, but then the fact is actually I Eigen filters are very good, in fact, there was another problem that we solved again using Eigen filters after we after we succeeded with this there was another semi-conductor wafer chip where they were they wanted to see whether the whether the wafer had leaked and that leak, if you look at that image it will be very hard to make out with your with your know ordinary eyes, it is very hard to see where that you know if you want to see the contour of the leak is very hard, it just kind of see merges with the with whatever with that is the substrate.

And it is very hard to tell, it is a very soft edge and I know anything that you do will sort of fly on will the flag so many contours on that image and again what we what we did was we learnt the background or whatever is supposedly clean and then and then again right used this kind of a window based approach computed the reconstruction error, that flag. So, what I want to say is tomorrow any time you encounter situations of this kind where there are where there are see defects are very subtle, Eigen filters are very good I mean a that just that very well-known I mean it kind of surprising that people do not seem to use them that much.

So, the so now why I wanted to point this out is because so the PCA on the one hand you can use it as a benchmark then second as I said you know you can use it for dimensionality reduction, third as I said if you want to if you want to see I mean do some kind of you know, some kind of a graphical this one a visualization that is 3 then fourth what it is been used for one of the most talked about things for face recognition. Then 5 is this Eigen filter and there are so many others also.

But at least I mean this many, so in that sense that sense PCA you would hopefully have understood the significance of actually doing a PCA, not simply from a unitary transform point of view, but also from the point of view and in fact if you look at pattern recognition learning on all that, they know which came later is all fundamentally based on this, covariance has so much structure that right anything when you say data mining when you say learning from data all of that if you can really trace back it is will all starts here, there is so much so much hidden there is so much pattern hidden at cross data that if you can if you can only extract those patterns and you can start start doing wonderful things.