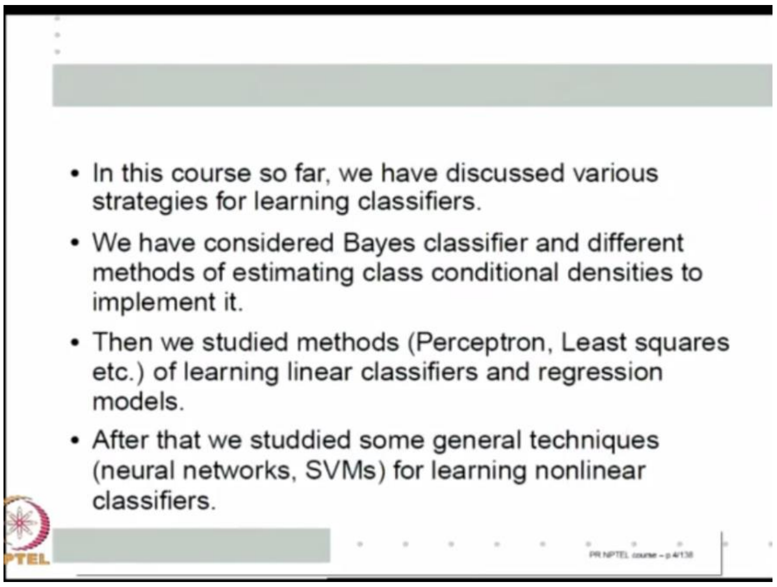
# Pattern Recognition

**Prof. P. S. Sastry**
**Department of Electronics and Communication Engineering**
**Indian Institute of Science, Bangalore**

## Lecture - 38
## Feature Selection and Dimensionality Reduction; Principal Component Analysis

Hello, and welcome to this next lecture in pattern recognition. With the last lecture, we have discussed, we have completed discussion on kernel based methods. So, with that we kind of completed as the part of the course, which discusses about learning classifiers. So, we can, we can take a global look now and ask what, what all have we done so far. So, in this course so far, we have discussed various strategies for learning classifiers. We started with pattern classification said the classifier design is, what admits lot of mathematical structures and we discussed different strategies for learning classifiers.

(Refer Slide Time: 01:02)

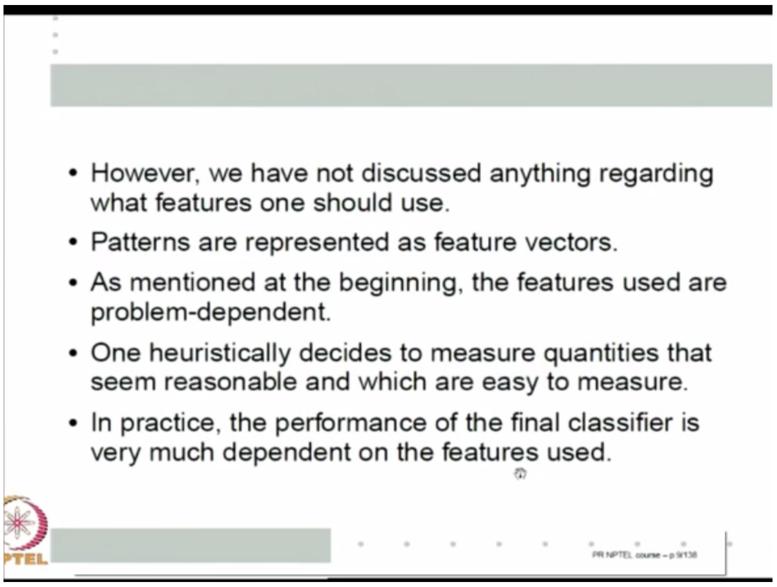


- In this course so far, we have discussed various strategies for learning classifiers.
- We have considered Bayes classifier and different methods of estimating class conditional densities to implement it.
- Then we studied methods (Perceptron, Least squares etc.) of learning linear classifiers and regression models.
- After that we studdied some general techniques (neural networks, SVMs) for learning nonlinear classifiers.

PR NPTEL course – p.4/138

So, specifically we have considered the Bayes classifier, the optimal Bayes classifier; the different methods for implementing it by learning class conditional densities. Then we looked at various linear classification regression techniques, percepetron least squares LMS specially linear discriminant so many of these techniques we looked at. We looked at various pros and cons of them. Then we went into statistical learning theory to show what kind of bounds, we can put on them. And then we looked at strategies for learning nonlinear classifier. We just concentrated so far on how on some general techniques for learning classifiers given both linear, nonlinear classifiers and some theory of statistical learning which also deals with us.

(Refer Slide Time: 01:54)



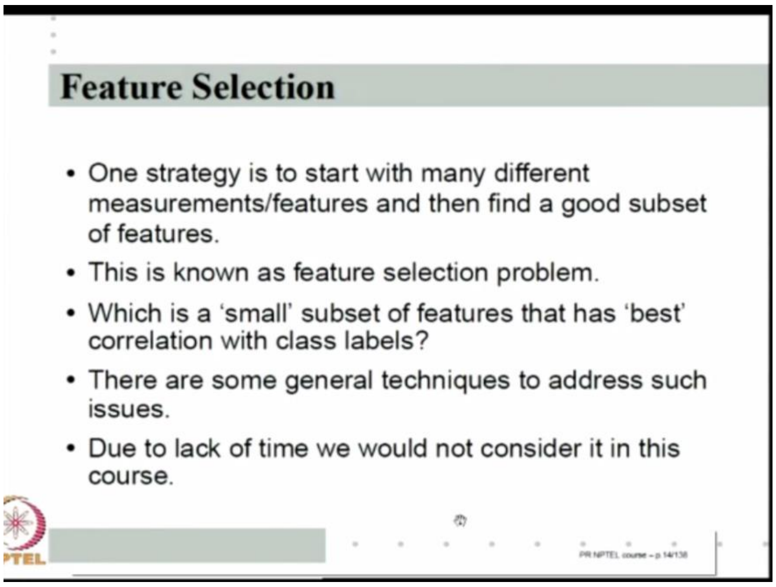
- However, we have not discussed anything regarding what features one should use.
- Patterns are represented as feature vectors.
- As mentioned at the beginning, the features used are problem-dependent.
- One heuristically decides to measure quantities that seem reasonable and which are easy to measure.
- In practice, the performance of the final classifier is very much dependent on the features used.

PR NPTEL course – p.9/138

So, one question that we never addressed really is what features does one use right in all our discussions so far. Patterns are represented as feature vectors, patterns are abstract for us and patterns are always some d dimensional feature vectors. And we have no never in this course said anything about, what should we what kind of features does one use. Very beginning of the course, we seen many examples of specific pattern recognition task, and there we discussed the kind of features people use in those examples. And the, the overall import is that features are very much problem dependent which is still true. So, what features to use very much depends on the problem and the kind of pattern is said, if I am doing a SVM regression task. I would have a one dimensional signal and I may do some frequency content out of it as my features. If I am doing a phase recognition task image as the feature, and you know the, the kind image as the pattern and the kind of features, I measure would be totally different.

So, in general features are very much problem dependant. Essentially one heuristically decides on what kind of quantity is or, relevant for the classification decision. And also, what kind of quantities are easy to measure, easy to calculate and that is how one comes up with a decision on what features to use. Of course, in practice the final classifier would very much be dependent on the features use so, even the so called Bayes optimal classifier is optimal in the sense given that particular feature vector. And the corresponding underlying class conditional densities, no other classification will can do better, that is all it means. It does not mean that, you cannot and that problem, we cannot do better. Obviously, if I change the features I should, I may be able to do better. So, while we cannot give general theory of what features to measure there it is still possible to talk, about if I can measure so many features out of that can I double a features which are useful.

(Refer Slide Time: 04:07)



## Feature Selection

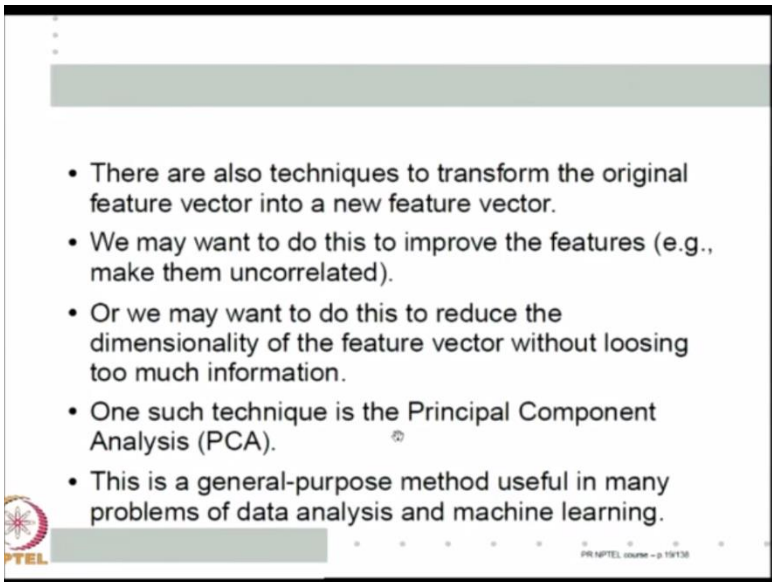
- One strategy is to start with many different measurements/features and then find a good subset of features.
- This is known as feature selection problem.
- Which is a 'small' subset of features that has 'best' correlation with class labels?
- There are some general techniques to address such issues.
- Due to lack of time we would not consider it in this course.

PR NPTEL course – p.14/138

So, one strategy, what is called feature selection is that we start with many different measurements. We measured, whatever we think are possibly relevant. And then out of that given our training samples try and (( )) a good subset of features. This is known as feature selection problem, and the; and the problem we want to address here is essentially, which subset of features are the best correlation with the class labels. So, we are asking, given my training data, what is the best subset of features can I use? So, looking at the training data, I am asking which features have better correlation with the class label, which features have less correlations class labels, which features you know seem to give me the best discriminative information.

So, there are some information ways of formulating this question, saying which subset of features gives me the best information regarding the class label, and we can use such notions to decide, which subset of features to select. So, there are some general techniques, if you, you if you can measure this 500 different features, I might be able to provide you, with some 50 good subset of features. These are computational expensive, but there often useful, but in this course we will not be considering feature selection it is, it. These techniques are computationally rather expensive, and it is difficult to implement a completely theoretically justified method. So, very often they turn out to be heuristic, so with that and also because it, takes us a little outside the purview of what we want to discuss here, we will not be considering these. However, I would like all of you to know that there is lot of work, on how one can select a subset of features.

(Refer Slide Time: 06:17)



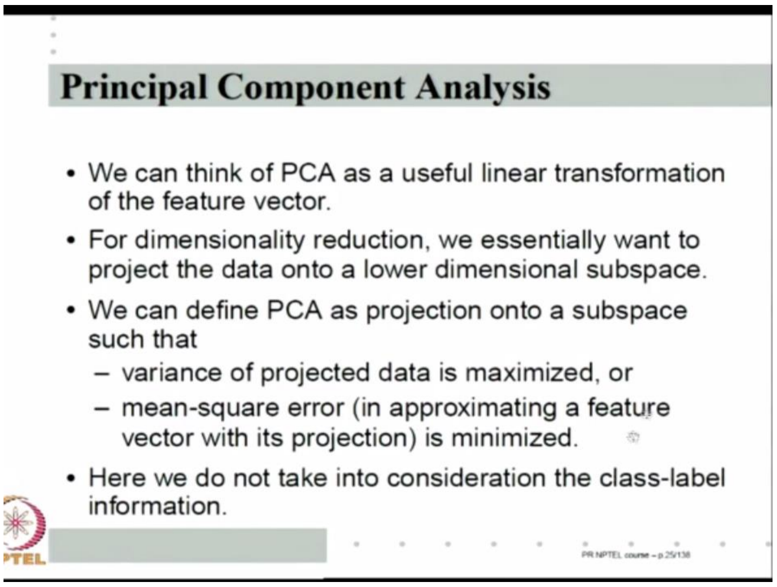
- There are also techniques to transform the original feature vector into a new feature vector.
- We may want to do this to improve the features (e.g., make them uncorrelated).
- Or we may want to do this to reduce the dimensionality of the feature vector without loosing too much information.
- One such technique is the Principal Component Analysis (PCA).
- This is a general-purpose method useful in many problems of data analysis and machine learning.

PR NPTEL course – p.19/138

As you said, there is also another way of looking at getting better features. Given that, I have measured these features, can I manufacture a new features out of these features? For example, may be certain linear combinations of these features may give me better features for discrimination. So, there are also techniques to transform, the original features vector into new feature vector. We may want to do it for many different, we may want to do this to improve the features for example, you may want to make them uncorrelated, or you may want to make them better at classification. Or we may want to do dimensional trade action I am ask, a can I instead of having this hundred dimensional feature vector. Can I have some fifty dimensional feature vector of course, just dropping a 50 of them may not be the best solution. But if I transform them first and then drop a 50 it might be better.

So, I may want to transform them to get a better dimensionality reduction. One such very popular technique is called principal component analysis or PCA for short. And this is a very, very important technique in many, many different problems of data analysis. And machine learning apart from pattern recognition. So, whenever you have training data and you want to understand the, the geometry of training data to ask, how many dimensions in the vectors are needed to capture most of the training data. So, principal component analysis is a, is a good technique, it is in general used as a very general purpose dimensionality reduction technique, give some data. So, we will consider that in detail in this class.

(Refer Slide Time: 07:58)



## Principal Component Analysis

- We can think of PCA as a useful linear transformation of the feature vector.
- For dimensionality reduction, we essentially want to project the data onto a lower dimensional subspace.
- We can define PCA as projection onto a subspace such that
  - variance of projected data is maximized, or
  - mean-square error (in approximating a feature vector with its projection) is minimized.
- Here we do not take into consideration the class-label information.

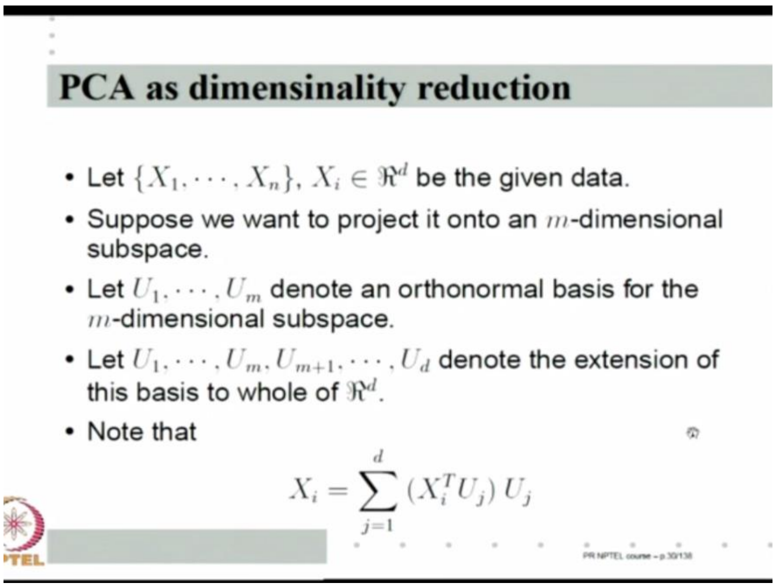
PR NPTEL course – p.25/138

So, we can think of PCA as a useful linear transformation of the feature vector. So, given the feature vector, you want to find a different linear transform, transform to transform the feature vector into some other space, or linearly transform into some other space and that is useful for classification. In the dimensional direction sense, we essentially, want to project the data on to some lower dimensional subspace. The linear transform would be projection actually, so if I may have a D dimensional feature vector. I may want to project it onto on a; onto on a, m dimensional subspace of the D dimensional space. So, we can in general, we can define PCA in 2 different ways. A PCA is a projection of data onto a subspace, such that one variance of the projected data is maximized that is even though, I am projecting onto a m dimensional subspace. Most of the variance in the data is captured by this, by the projection onto the m dimensional subspace.

So, give a fixed M, which particular, m dimensional subspace, we will capture as much as the variance of the original data as possible to is, if I want to approximate the original feature vector with the projection, I want to minimize the mean square error. So, I want to find that m dimensional subspace such that approximating a feature vector with this projection, the in the sense of mean square error is best as it turns out solution for both is same. So, that is the PCA, we will see how to derive how what this particular transform would be. But whether I want to think of it as best approximation in the m dimensional subspace or a fixed m on both are for a fixed m for a fixed m capture most of the variation in the original data. The final solution turns out with the same transform.

So that is what PCA is that is what, we are going to consider now. And I just to want to want you to understand that here we were not considering the class label information at all we are just looking at data, all the data and asking if I want to project it to a smaller dimensional subspace which subspace will, will give me the best approximation. We are not bothered about you know based on the class label information, which is the best approximation. So, here this is nothing to do with the class label, we are just give large set of data vectors, what is a best lower dimensional representation of the data vectors?

(Refer Slide Time: 10:42)



## PCA as dimensinality reduction

- Let $\{X_1, \cdots, X_n\}$, $X_i \in \Re^d$ be the given data.
- Suppose we want to project it onto an $m$-dimensional subspace.
- Let $U_1, \cdots, U_m$ denote an orthonormal basis for the $m$-dimensional subspace.
- Let $U_1, \cdots, U_m, U_{m+1}, \cdots, U_d$ denote the extension of this basis to whole of $\Re^d$.
- Note that

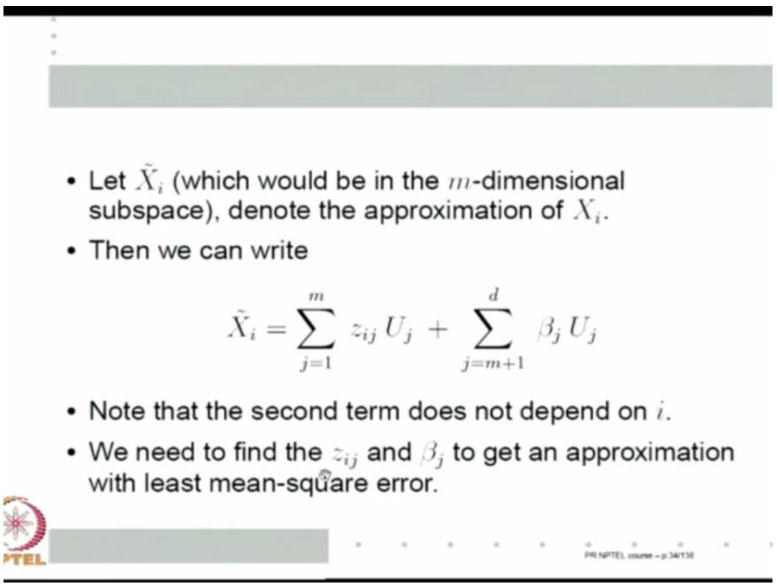
$$X_i = \sum_{j=1}^{d} (X_i^T U_j)\, U_j$$

PR NPTEL course – p.30/138

So, let us start define the problem, let us say we first found to look at it from the point of view of dimensionality reduction. Let X 1 to X n be the given data, as I said now, class labels are not important. So, we will only look at X 1 to X n. Let us say all of them are in are d dimensional, we want to project it on a m dimensional subspace. So, let us say U 1 to U m denotes the orthonormal basis for the m dimensional subspace. So, ultimately want to determine the vectors U 1 to U m, because they determine the subspace. So, we will, we will the subspace is of course, given by its basis and we will choose a orthonormal basis. So, let us say U 1 to U m is an orthonormal basis for the m dimensional subspace. And let us say, because that will be subspace of R d. If I add U m plus 1, U d it should become an orthonormal basis for R d.

So, U m plus 1 to U d define them to be those orthogonal vectors, which when added to U 1 to U m become a orthonormal basis for R d. So, U 1 to U m is the orthonormal basis for the m dimensional subspace I am looking for U 1 to U m, U m plus 1, all the way up to U d is the extension of this basis to the whole of R d. So, because this is orthonormal basis any X i can of course, be represented as a linear combination of the vectors used U 1 to U d. And the coefficient of the linear combination will be simply, X i transpose U j, because orthonormal basis. So, X i can be written as sum over j is equal to 1 to d, X i transpose U j, U j. So, it is written as a linear combination of these basis vectors U j and the (( )) linear combinations X i transpose U j. These are the standard, standard representation any orthonormal basis.

(Refer Slide Time: 12:32)



- Let $\tilde{X}_i$ (which would be in the $m$-dimensional subspace), denote the approximation of $X_i$.
- Then we can write

$$\tilde{X}_i = \sum_{j=1}^{m} z_{ij}\, U_j \;+\; \sum_{j=m+1}^{d} \beta_j\, U_j$$

- Note that the second term does not depend on $i$.
- We need to find the $z_{ij}$ and $\beta_j$ to get an approximation with least mean-square error.

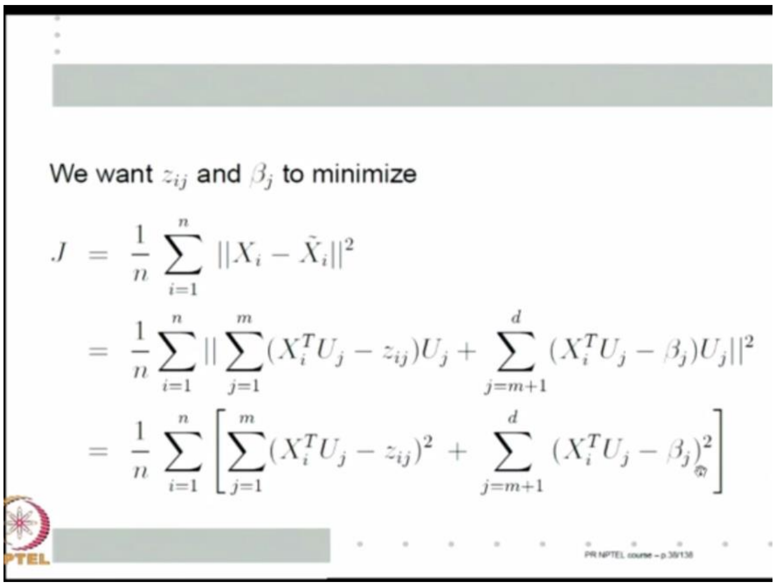
PR NPTEL course – p.34/138

Now, let us say X i tilde is the approximation of X i in the m dimensional subspace, because the dimensional subspace which is spanned by U 1 to U m, X i tilde can be written as Z i j, U j j is equal to 1 to m. So, Z i j are the coefficients in the U j coordinate system for representing tilde. In addition, we will have a constant, we will put second term which is a constant. So, that is, there is any, any mean to be taken care of there is always better to take (( )) constant to get a low mean square approximation an X i. So, because this is a constant and whatever is in the m dimensional subspace can easily be accounted for by j i j. The part of the constant which is not in the m dimensional subspace is what, we have to worry about.

So, that we will write it as j is equal to m plus 1 to d beta j U j. Now, we can see that this expression is independent of i. This term does not depend on the index i, so this is a constant for all X i, it simply some constant vector, which is in the orthogonal complement of this m dimensional subspace. The only reason for putting it like this is essentially, just represent it in terms of these U j is, there might be a constant which when added might lower my mean square error. Of course, if I do not use the constant; the constant will turn out to be 0 that is when, I optimize, but having a constant we will help me to find a better approximation.

So, what is the task now? We have to find Z i j, b z to get the approximation with the least mean square error, of course, you have to get also U j. So, we will solve the problem in 2 steps. Given any particular, U j that means given any subspace. what are the best Z i j's in b j 's in, terms of U j's. Then my, I will write the mean square error and find the best Z i j's and b j's. So, ultimately my mean square error will now, only depends on U j. Now, I can minimize the mean square error by choosing the right U j. So, we will first want to find the best Z i j's and beta j's, if you get a good approximation.

(Refer Slide Time: 14:48)



We want $z_{ij}$ and $\beta_j$ to minimize

$$J = \frac{1}{n}\sum_{i=1}^{n} ||X_i - \tilde{X}_i||^2$$

$$= \frac{1}{n}\sum_{i=1}^{n} || \sum_{j=1}^{m}(X_i^T U_j - z_{ij})U_j + \sum_{j=m+1}^{d}(X_i^T U_j - \beta_j)U_j||^2$$

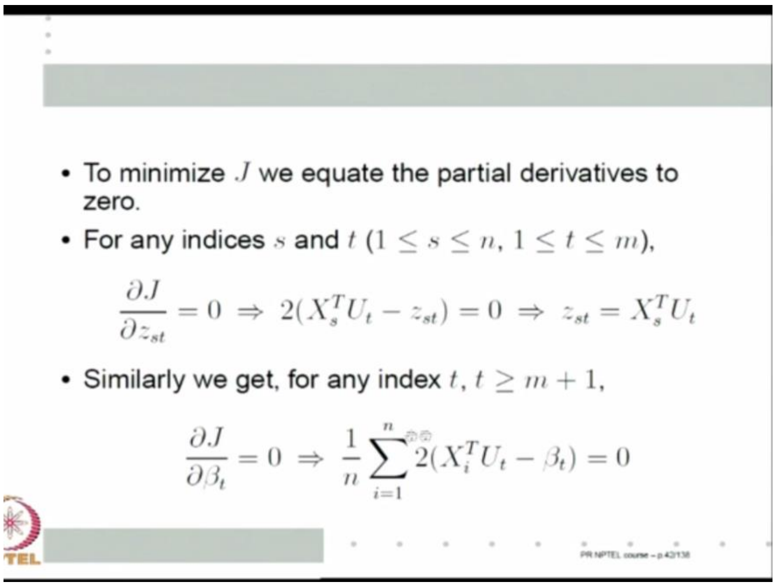
$$= \frac{1}{n}\sum_{i=1}^{n} \left[\sum_{j=1}^{m}(X_i^T U_j - z_{ij})^2 + \sum_{j=m+1}^{d}(X_i^T U_j - \beta_j)^2\right]$$

PR NPTEL course – p.38/138

What is my approximation error? I want X i to be close to X i tilde; X i tilde is my low dimensional approximation. So, my error is X i minus X i tilde norm square summed over, i is equal to 1 to n divided by 1 by n. This is the mean square error, this is the error; this is the square of the error add and divide 1 by n to get the mean. So, mean square error is what, I want to subtract. Now, we know that X i X i is given by this X i transpose U j U j where, X i tilde is given by up to U m, it is some Z i j U j and from U m plus 1. It is beta j so, for want X i minus X i tilde up to m, X i as X i transpose U j are the coefficient and X i tilde Z i is at coefficient. So, this is X i transpose U j minus Z i j into U j j is equal to 1 to m. From, m plus 1 to d X i, still has X i transpose U j, but X i tilde has beta j.

So, I just left everything is still the norm square X i minus X i tilde vector is represented in terms of U j's like this. Now, because U is, are an orthonormal basis if I have sum a linear combination of U is and I want the norm, norm square is simply equal to the square of the individual coefficients. So, this is 1 by n summation is equal to 1 to n that comes from this j is thing. And here I have to square all the coefficients. So, j is equal to 1 to m X i transpose U j minus Z i whole square plus j is equal to m plus 1 to d X i transpose U j minus beta j whole square so, this is the mean square error. So, I want to minimize this to find Z i j and beta j for given U j is.

(Refer Slide Time: 16:58)



- To minimize $J$ we equate the partial derivatives to zero.
- For any indices $s$ and $t$ $(1 \leq s \leq n,\ 1 \leq t \leq m)$,

$$\frac{\partial J}{\partial z_{st}} = 0 \ \Rightarrow\ 2(X_s^T U_t - z_{st}) = 0 \ \Rightarrow\ z_{st} = X_s^T U_t$$

- Similarly we get, for any index $t$, $t \geq m+1$,

$$\frac{\partial J}{\partial \beta_t} = 0 \ \Rightarrow\ \frac{1}{n}\sum_{i=1}^{n} 2(X_i^T U_t - \beta_t) = 0$$

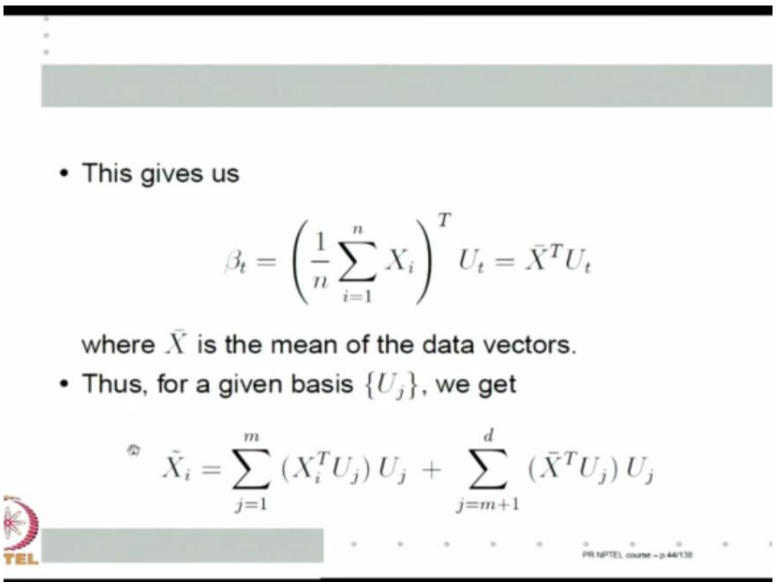
PR NPTEL course – p.42/138

So, how do I find them we, we first have to take partial derivatives of j with and equate it to 0 so, I have essentially, find Z i j and beta j. So, first let us look at derivative with respect to Z i j. So, if I take any indices as n T and i want derivative with respect to delta Z as T with respect to X T. If I want partial derivative, there is only 1 term in this entire expression that contain Z s T that happens, when i takes a value Z and j takes the value T and that term is nothing. But X i transpose U j minus Z i j is are the X i transpose U t minus Z s T whole square. So, if I differentiate that I will get this 2 into X s transpose U t minus Z s T equate it to 0.

So, what does that give me that gives me Z s T is X s transpose U t so, Z i j is X i transpose U j. So, essentially to find the coefficients of representation X tilde with respect to the first U 1 to U m, you might use it as the, the same thing as the original X s that is X i transpose U j. What about beta? If I want to differentiate with respect to particular beta t that happens when the j takes value t so, i fully remains. So, the derivative with respect to beta t will be 1 by n summation, i is equal to n derivative of this with respect to beta t. That happens only, when j is equal to T and this term does not have any betas.

So, only this will come that will be 2 X i transpose U j minus X i transpose U t minus beta t so for any t, t greater than m plus 1 only t for t greater than m, m plus 1. We have beta t is so del j by del beta d is equal to 0. I implies 1 by n, i is equal to 1 to n, 2 times X i transpose U t minus beta t is equal to 0. If I expand this, I get 1 by n, 2 will anyway go. So, I will get 1 by n summation, beta t that will be n beta d by n, will be beta t. This second that will come on the one is, are the equality that will be equal to what one by n summation i is equal to 1 to n X i whole transpose U t.

(Refer Slide Time: 19:26)



- This gives us

$$\beta_t = \left(\frac{1}{n}\sum_{i=1}^{n} X_i\right)^T U_t = \bar{X}^T U_t$$

where $\bar{X}$ is the mean of the data vectors.

- Thus, for a given basis $\{U_j\}$, we get

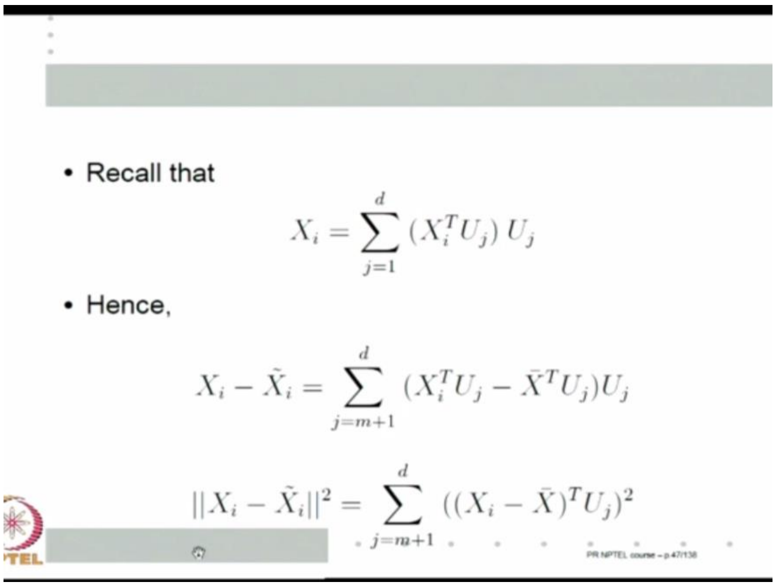
$$\tilde{X}_i = \sum_{j=1}^{m} (X_i^T U_j)\, U_j + \sum_{j=m+1}^{d} (\bar{X}^T U_j)\, U_j$$

PR NPTEL course – p.44/138

So, this gives me beta t equal to 1 by n summation, i is equal to n, X i transpose U t. This is nothing but mean of X i so, let us represent it by X bar. So, beta t is X bar t into t so, my best approximation in the m dimensional subspace given a m dimensional subspace. That is U 1 to U m and U m plus 1 to U d is for the first time components I use X i transpose U j as my coefficients and for all the remaining ones, the constant I have to put is X bar transpose U j essentially, if X bar is equal to 0. I do not need it, X bar is not 0, i need this constant.

So, this is my best m dimensional approximation. By m dimensional approximation, we mean the X i tilde vectors leave in a m dimensional subspace of d dimensional space. Of course, this is a linear combination of the vectors U 1 to U d each of the vectors U 1 to U d are (( )) R d. So, the right and side of this is a d dimensional vector, there is no; there is no question about that. So, this expression does not show that X i tilde is an m dimensional vector, but the expression is such that we will see shortly, how it happens that essentially, X i tilde leaves in an m dimensional subspace plus a constant. This constant is independent of phi. So, accept for a constant X i leaves in a m dimensional subspace so, if know this constant separately, let us say if X bar is equal to 0, which I, i can always take then these m numbers, X i transpose U j will completely specify, X i tilde. That is; that is; that is what is meant by it is a; it is a sits in a m dimensional subspace. So, this is X i tilde, tilde; this is the best representation, we get for a given j is.

(Refer Slide Time: 21:35)



- Recall that

$$X_i = \sum_{j=1}^{d} (X_i^T U_j)\, U_j$$

- Hence,

$$X_i - \tilde{X}_i = \sum_{j=m+1}^{d} (X_i^T U_j - \bar{X}^T U_j) U_j$$

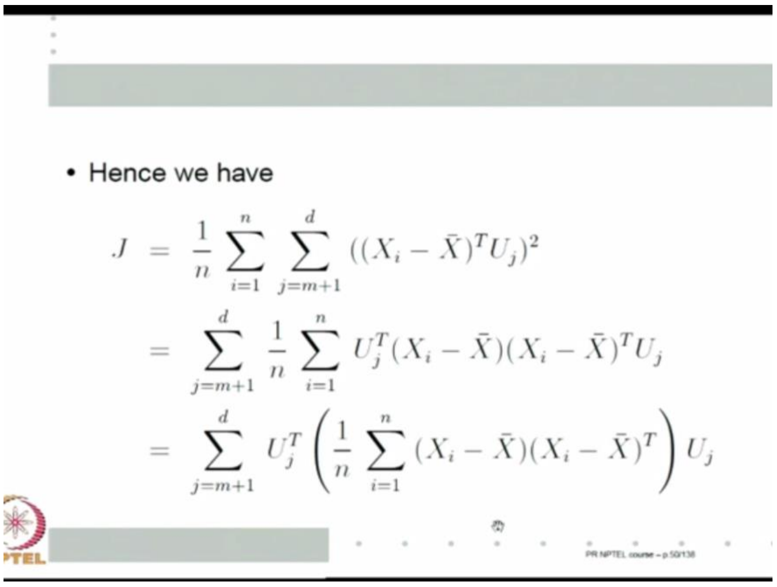
$$||X_i - \tilde{X}_i||^2 = \sum_{j=m+1}^{d} ((X_i - \bar{X})^T U_j)^2$$

PR NPTEL course – p.47/138

And we know that X i is this so, if I want X i minus X i tilde the first m components in the first m U j’s it is same, X i tilde here is j is equal to 1 to m X i transpose U j U j and here, also i same thing. So, for j is equal to 1 to m the U j’s coefficient if I take X i minus X i tilde i will be 0, only this will survive. So, I can now write X i minus X i tilde at j is equal to 1 plus m 1 to d X I is coefficients for using U j. Now, is X i transpose U j whereas, X tilde coefficient is X bar transpose U j only, if I am j is equal to m plus 1 to d.

So, this is my error so the, the norm square of the error will be what because, once again because, U j’s are an orthogonal basis, U j’s are orthogonal vectors if I take norm of this that will be simply equal to squares of this, because U j’s are orthonormal vectors. So, that will be X i transpose U j minus X bar transpose U j whole square, which I can always write as X i minus X bar whole transpose U j whole square. So, my for a given i my square of the error X i minus X i tilde norm square is some j is equal to m plus 1 to d X i minus X bar whole transpose U j whole square. So, my actual j, my actual mean square error is some of this over i is equal to 1 to n divided by 1 plus n.

(Refer Slide Time: 23:18)



- Hence we have

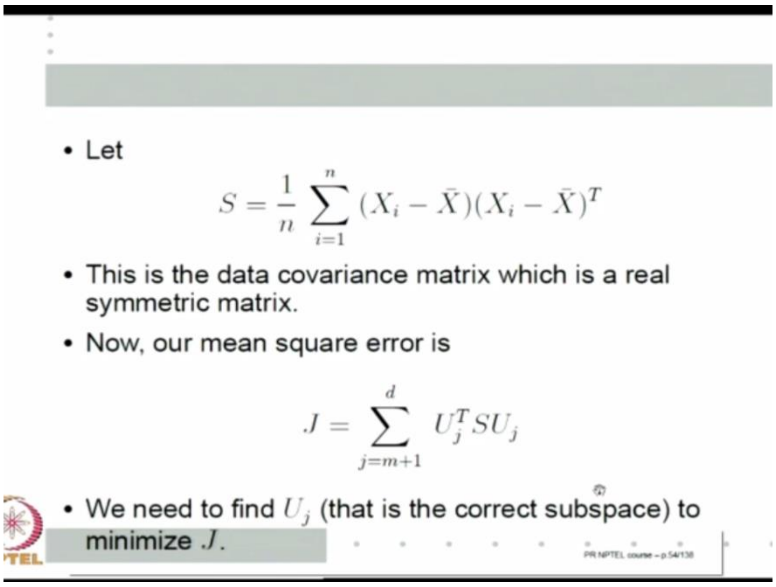
$$J = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=m+1}^{d}((X_i - \bar{X})^T U_j)^2$$
$$= \sum_{j=m+1}^{d}\frac{1}{n}\sum_{i=1}^{n} U_j^T (X_i - \bar{X})(X_i - \bar{X})^T U_j$$
$$= \sum_{j=m+1}^{d} U_j^T \left(\frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X})(X_i - \bar{X})^T\right) U_j$$

PR NPTEL course – p.50/138

So, my j is 1 by n summation, i is equal to 1 to n X i minus X tilde whole square, which is this. So, I can write that as j is equal to m plus 1 to d X i minus X tilde transpose U j whole square. Now, X i minus X bar transpose U j is a scalar. So, I can, I can write it as X i minus X bar transpose U j or U j transpose X i minus X bar. So, I can write this square as, I first interchange the 2 summation both are finite summations j is equal to m plus 1 to d 1 by n, i is equal to 1 to n, i write the square as 2 times U j transpose X i minus X bar into X i minus X bar transpose U j. If I write it like that now, I have a very interesting structure here.

Now, I can, I can work it out that these, because these U j's do not depend on I, the U j transpose can come out of the ith summation and, this U j can also be outside the i summation. So, I can write it as summation, j is equal to m plus 1 to d U j transpose. 1 by n summation, i is equal to 1 to n, X i minus X square; X i minus X square transpose U j. So, what I have here is a quadratic form. U j transpose something, U j and this something is a matter X. This X i minus X bar is a d by 1 vector and X i minus X bar transpose is a 1 by d vector. This is a d by d matrix is known as an outer product, I, I hope many of you have come across this earlier, this called an outer product of 2 vectors. So, I am summing over, i is equal to 1 to n so, this whole thing is a d by d matrix. So, j can be written as a quadratic form, so let us give a name to this matrix, what is this matrix? X i minus X bar, X bar is the mean of X i, so X i minus X bar X i minus X bar transpose summed over, i, i is equal to 1 to n is nothing.

(Refer Slide Time: 25:21)



- Let

$$S = \frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X})(X_i - \bar{X})^T$$

- This is the data covariance matrix which is a real symmetric matrix.
- Now, our mean square error is

$$J = \sum_{j=m+1}^{d} U_j^T S U_j$$

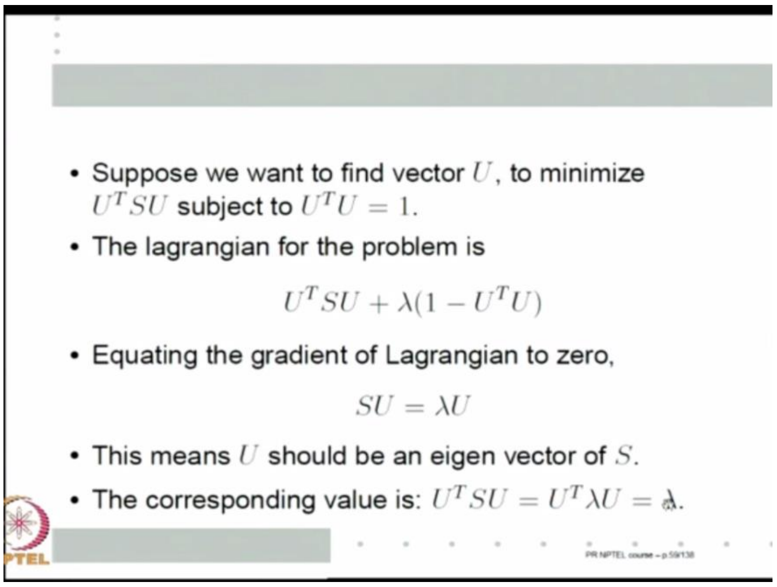
- We need to find $U_j$ (that is the correct subspace) to minimize $J$.

PR NPTEL course – p.54/138

But the covariance matrix, for random vector X, the covariance matrix is defined as expected value of X minus mu X into X minus mu X transpose. Now, if I do not know how to evaluate such an notation, I can evaluate it as a sample means, and this is nothing but the sample mean estimator that expectation. So, this is the data covariance matrix am I with come across the data covariance matrix earlier when we did the Fischer linear discriminant. So, S is nothing but the data covariance matrix and j now can be written as U j transpose SU j, J is equal to m plus 1 to d .

So, now my mean square error given, given any subspace defined by the U 1 to U m, U m plus 1 to U d. The mean square error in approximating a vector by its projection onto this m dimensional subspace is given by j is equal to m plus 1 to d U j transpose SU j where S is the data covariance matrix. So, my job is to find U j's to minimize this, so we need to find U j's finding U j is, is same as finding the correct subspace to minimize j. Let us so, what we need to minimize is; essentially, a quadratic form of a matrix with respect to vector of course, here I need to minimize sum of such quadratic forms with respect to vectors, which are constrained to be orthonormal, constrained to have norm 1 and orthogonal to each other.

(Refer Slide Time: 27:13)



- Suppose we want to find vector $U$, to minimize $U^TSU$ subject to $U^TU = 1$.
- The lagrangian for the problem is

$$U^TSU + \lambda(1 - U^TU)$$

- Equating the gradient of Lagrangian to zero,

$$SU = \lambda U$$

- This means $U$ should be an eigen vector of $S$.
- The corresponding value is: $U^TSU = U^T\lambda U = \lambda$.

PR NPTEL course – p.59/138

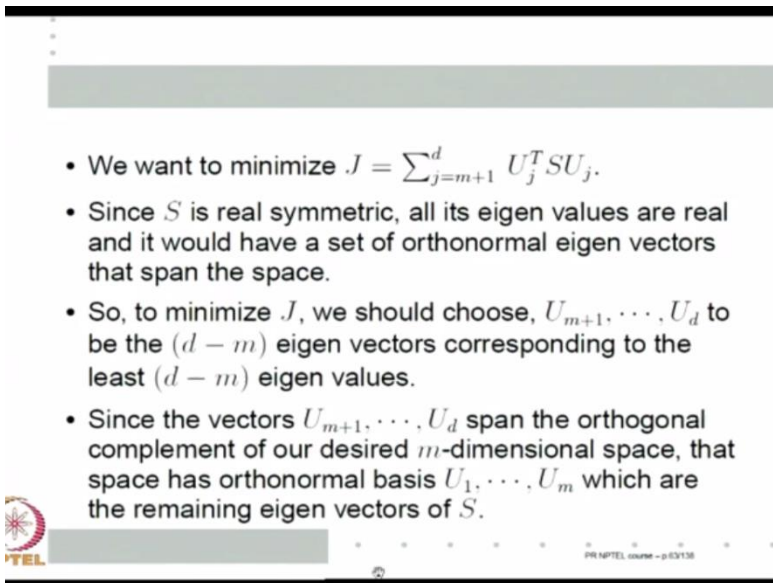
So, just look at how 1 minimizes a quadratic form suppose I want to find a vector U to minimize U, U transpose S U subject to the constraint the norm of U is equal to 1. This is constraint optimization problem, that we already seen when we stated with SVM. So, you want to maximize or minimize this objective function subject to this constraint. So, we have to write a Lagrangian, so the Lagrangian will be U transpose SU plus lambda times 1 minus U transpose U. So, this is the Lagrangian for this constraint optimization problem and to find the optimum, we need to equate the gradient as the Lagrangian to 0. The, what I am trying to find is U, so I am optimizing with respect to the vector U. So, the gradient with respect to U that gives me SU 2 will go away anyway from both side. And this side I will get minus lambda U, so equating gradient to 0 we get SU is equal to lambda U, what is this equation? U is a vector S is the given matrix.

So, S is equal to lambda U is nothing but the Eigen vector equation for the matrix. So, this shows that any U that minimizes this has to be a Eigen vector of the matrix S. That means U should be an Eigen vector first. So, if I want to minimize U transpose SU subject to U transpose U is equal to U 1 then U has to be an Eigen vector, which Eigen vector. We can actually plug this Eigen vector property into our objective function. So, if I take U to be any Eigen vector, what is the value of my objective function? U transpose S. SU, SU is nothing but lambda U, U transpose lambda U, transpose U is 1 so that is lambda.

So, if I want to minimize U transpose SU subject to the constraint that U transpose SU is equal to 1. Then U has to be an Eigen vector take as pointed to some Eigen value lambda, and then if I take any Eigen vector corresponding, Eigen value lambda. The actual value of U transpose SU is nothing but lambda, how can this value be lambda? Is lambda i real number, i complex number. This can be lambda because, lambda is real number. Why is lambda a real number? As I said, this is the data covariance matrix as you already, know the data covariance matrix is real.

Because is all it is easy enough to see it is; it is; it is, if all X i is are real, it is real and is also symmetric that is also quite easy to see. Because of the outer product form the matrix that you get is symmetric. So, S is a real symmetric matrix and hence all its Eigen values are real. So, if I want to minis U transpose SU subject to U transpose is equal to 1, then the minimizing U has to be a Eigen vector and if I take it to be Eigen vector with Eigen value lambda, then the value of U transpose SU is equal to lambda, which mean, we actually minimize it I, I need to take the Eigen vector which has, which corresponds to the least Eigen value.

(Refer Slide Time: 30:51)



- We want to minimize $J = \sum_{j=m+1}^{d} U_j^T S U_j$.
- Since $S$ is real symmetric, all its eigen values are real and it would have a set of orthonormal eigen vectors that span the space.
- So, to minimize $J$, we should choose, $U_{m+1}, \cdots, U_d$ to be the $(d-m)$ eigen vectors corresponding to the least $(d-m)$ eigen values.
- Since the vectors $U_{m+1}, \cdots, U_d$ span the orthogonal complement of our desired $m$-dimensional space, that space has orthonormal basis $U_1, \cdots, U_m$ which are the remaining eigen vectors of $S$.

PR NPTEL course – p.63/138

Now, coming back to our problem, we want to minimize j is equal to m plus 1 to d U j transpose S U j. Where our constraints are not only that U j's are orthogonal U j's are unit norm, that is U j transpose, U j's equal to 1 they are also orthogonal. U j transpose U i equal to 0 if phi is not equal to j. With those constraints, I want to minimize this, because we can once again write the Lagrangian and do dirty algebra. But we do not need to do all that because, I know for any one of these terms, if I want to minimize the U has to be a Eigen vector.

So, ultimately U is all, U is have to be Eigen vectors. Now, I can there, there are a S is a d by d matrix, there are d possible Eigen vectors corresponding to d possible Eigen values. And if I choose U j to be any one Eigen vector, I know U j transpose SU j will be the corresponding Eigen value. So, this particular j would be some of Eigen values which Eigen values the, the Eigen vectors that you picked up the Eigen values correspond to Eigen vectors you picked up. Now, I want this to be as small as possible, so what is that mean? Now, we know because S is symmetric all Eigen values are real and also it will always have a set of Eigen vectors, which can be orthonormal. So, S will have orthonormal set of Eigen vectors, these Eigen values are real and the Eigen vectors of S will span the space. So, Eigen vectors of S can be an orthonormal basis for R d. I know the so I can take the Eigen vectors of S to be orthonormal base for R d, then they satisfy the constraints of U j.

Now, we already know that any Eigen vector will satisfy the gradient equal to 0 equation. So, essentially we have to ask which Eigen vector would you choose to minimize this. Any Eigen vector I pick here, will be the, the U j transpose U j's the Eigen value and obviously, I cannot repeat any Eigen vector, because I want U j's to be orthogonal like they have to span the space. So, that tells me that the best I can do is to make my so, I have to pick U I want to minimize this sum over j is equal to m plus 1 is equal to d, that means you have to pick U m plus; U m plus 1 up to U d.

So, I need to pick U m plus 1, U m plus 2 up to U d and each one of them has to be a distinct Eigen vector S. And, whichever Eigen vector I pick up that term in this sum transpose of the corresponding Eigen value. So, which means U m plus 1, U m plus 1 to up to U d should be the d minus 1 Eigen vectors corresponding to the least d minus 1 Eigen values. You take the Eigen values of S, take the least d minus m Eigen values and take the corresponding Eigen vectors, that is the least, I can make this to be.

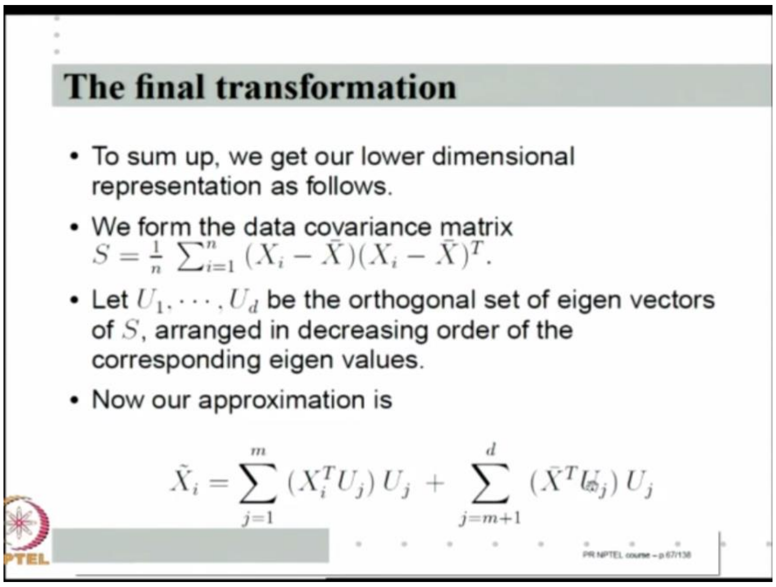
So, what we now see shown is that if, I want to minimize this, my choice of U m plus 1, U m plus 2 up to U d are the d minus m Eigen vectors correspond to the least d minus m. Eigen values. Of course, minimizing this I will be fixing only U m plus 1 m up to U d.

Why is that so, because my j just turns out to be that. When we did that my, my, my j turns out to be that, because my X i minus X tilde up to first term; up to first m terms in the summation X i and X tilde are same. And hence, my X i minus X tilde sums over only j is equal to 1 plus 1 to d of something into U j.

So, ultimately my J turns out to be this. So, because my J turns out to be this and I minimize that, I essentially pick U m plus to U d, but that does it mean that I have; I have a choice about remaining. I do not. Because, the, the U m plus 1 to U d completely specify the orthogonal complement to the m dimensional space that I want to want to be in. So, my optimal m dimensional space space is one whose orthogonal complement is U m plus one to U d which means the an orthonormal basis for that space U 1 to U m.

So, ultimately you have to shown is that if I want X tilde the lambda (()) projection onto the m dimensional subspace to be close to the origin X i in a in the sense of minimizing mean square error. Then the m dimensional subspace is has a orthonormal space given by U 1 to U m where, U 1 to U m are the Eigen vectors of the data covariance matrix here corresponding to the top m Eigen values.

(Refer Slide Time: 36:25)



## The final transformation

- To sum up, we get our lower dimensional representation as follows.
- We form the data covariance matrix $S = \frac{1}{n} \sum_{i=1}^{n} (X_i - \bar{X})(X_i - \bar{X})^T$.
- Let $U_1, \cdots, U_d$ be the orthogonal set of eigen vectors of $S$, arranged in decreasing order of the corresponding eigen values.
- Now our approximation is

$$\tilde{X}_i = \sum_{j=1}^{m} (X_i^T U_j)\, U_j + \sum_{j=m+1}^{d} (\bar{X}^T U_j)\, U_j$$

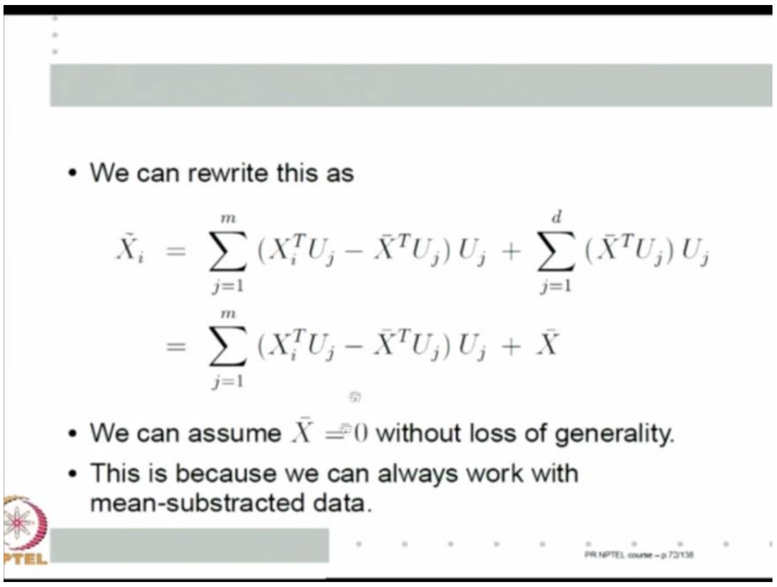
PR NPTEL course – p.67/138

So, let us sum this up so, given some data X 1 to X n all living in the d dimensional real Euclidean space, we do a lower dimensional representation as follows. We first form the data covariance matrix S is 1 by n, i is equal to 1 to n X m as m bar X i minus X bar transpose, this is a d by d matrix. Then you find all its Eigen vectors. Let us say U 1 to U d be the orthogonal set of Eigen vectors because, S is real symmetric. We it, will have a complete, complete set of Eigen vectors meaning its Eigen vectors span the space.

Since, the Eigen vectors span the space we actually make the Eigen vectors be orthogonal an orthogonal set of vectors and then a of course, I can normalize that so I can have a orthonormal set of Eigen vectors of S. Let us say U 1 to U d are that such Eigen vectors which are arranged in a decreasing order of corresponding n value that means U 1 corresponds to the highest Eigen value U 2 corresponds to the next highest Eigen value and so on. And I can talk about highest and next Eigen value, because the Eigen values are real because S is real symmetric.

Once, I have this, this complete Eigen spectrum arranged like this, my low dimensional approximation now X hat tilde is for j is equal to 1 to m X i transpose U j, U j where this U 1 to U m are the Eigen vectors corresponding to top m. Eigen values of S and then there is a constant j is equal to m plus 1 to d X bar transpose U j. This is my lower dimensional approximation I can slightly rewrite it, what I can do is, I can for from each of these terms I can subtract X transpose U j and then add X transpose U j that means, I will be adding j is equal to 1 to m X transpose U j, U j to this term.

(Refer Slide Time: 38:21)



- We can rewrite this as

$$\begin{aligned}\tilde{X}_i &= \sum_{j=1}^{m} (X_i^T U_j - \bar{X}^T U_j)\, U_j + \sum_{j=1}^{d} (\bar{X}^T U_j)\, U_j \\ &= \sum_{j=1}^{m} (X_i^T U_j - \bar{X}^T U_j)\, U_j + \bar{X}\end{aligned}$$

- We can assume $\bar{X} = 0$ without loss of generality.
- This is because we can always work with mean-substracted data.

PR NPTEL course – p.72/138

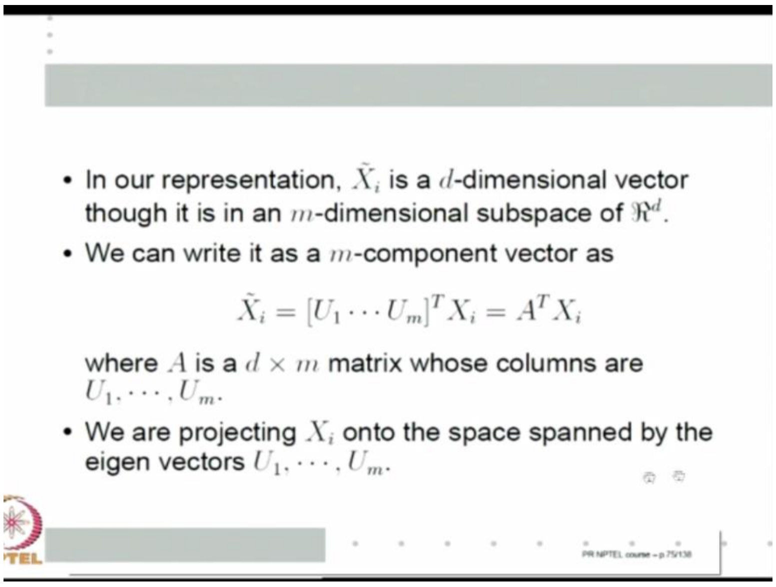
So, I can write my X tilde as j is equal to 1 to m X i transpose U j minus X bar transpose U j, what I did is from each of these terms I subtract X bar transpose U j. Now, I got j is equal to and plus this, other term j is equal to 1 to d X bar transpose U j, U j now I can to get all my X bar transpose U j terms together. Once I subtract them I have to add it, I already have m plus 1 to d and I am adding j is equal to 1 to m. So, I am adding j is equal to 1 to d.

Now, j is equal to 1 to d X bar transpose U j, U j’s nothing but the representation of X bar in the coordinate system of U j. So, this entire sum is nothing but X bar so I can write it as X bar, X a transpose U j minus X bar transpose U j, U j plus X bar. This is my low dimensional representation. Writing like this is nice because, if I assume X bar to be 0 then it is simply X i tilde is X i transpose U j U j first m components. Essentially, a first m components in the coordinate system of the Eigen vectors.

So, instead of what my original coordinate system I now represent my axis in the coordinate axis of the Eigen vectors and then take the first m components. The, the extra term comes only to take care of the bias that means the, the non zero mean of the data. And this never causes any problem, because you can always work with mean subtracted if you give me X 1, X 2, X n as data. I can actually redefine X 1 prime, X 2 prime, X n prime where, X i prime is X i minus X bar.

So, now these prime variables have zero mean, I can always work with zero mean data and the, the covariance matrix of the original data is same as the mean subtracted data. So, my U j’s will be the same, so whichever loss has generally I can assume X power to be 0 from now, on we will write these expressions with X bar is equal to 0. So, that they are much simpler as I already told you this does not look like an m dimensional vector is of course, it lives in m dimensional subspace because, this is just a constant. So, this is X, this is expanded in terms only U 1 to U m so to represent X i tilde you need to give me only m numbers. But it is still written in a vector notation as a, d dimensional vector because each of the U j’s are d dimensional.

(Refer Slide Time: 40:54)



- In our representation, $\tilde{X}_i$ is a $d$-dimensional vector though it is in an $m$-dimensional subspace of $\Re^d$.
- We can write it as a $m$-component vector as

$$\tilde{X}_i = [U_1 \cdots U_m]^T X_i = A^T X_i$$

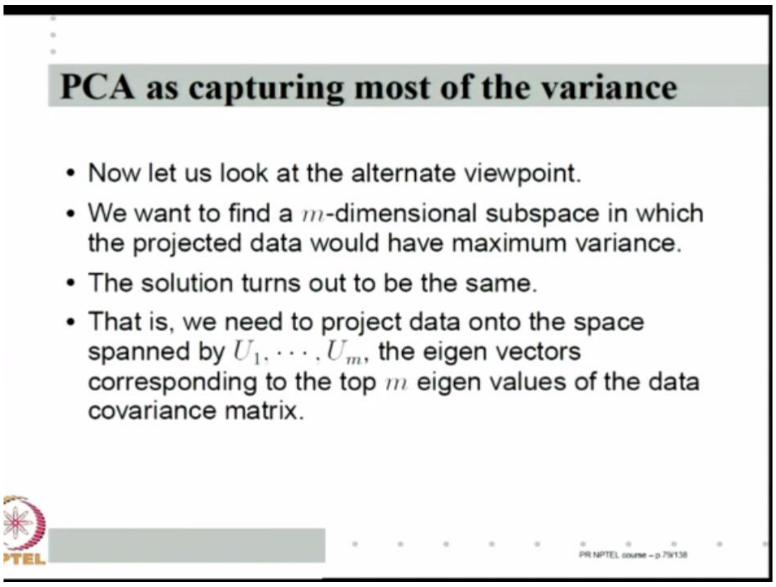
where $A$ is a $d \times m$ matrix whose columns are $U_1, \cdots, U_m$.
- We are projecting $X_i$ onto the space spanned by the eigen vectors $U_1, \cdots, U_m$.

PR NPTEL course – p.75/138

We can of course, rewrite it as a d vector so, as I said in our representation X i tilde is a d dimensional vector. Because each of the U j's are d dimensional vectors, but we can; obviously, write as an m dimensional vector because, it is; it is an m dimensional subspace of R d. How do I write as m dimensional vector? So X i tilde can be written as U 1 transpose X i U 2 transpose X i so on. It becomes a transpose X i written as a vector now of d components m components. Essentially, what it means is that once we take X x bar to be 0. Essentially I can write as m component vector where the first component is X i transpose U 1. Second component is X i transpose U 2 and so on.

So that is what this representation is so I can actually write X i tilde as a transpose X i where A is a d by m matrix, this matrix whose columns are the vectors U 1 to U m. So you make a, you make a matrix whose columns are the Eigen vectors, you take as many as you want. And then write X i tilde as a transpose X i where X i is the original data then that is the lower dimensional representation of X. This is in this sense that it is a linear transform so essentially projecting X i onto the space spanned by the Eigen vectors we want to use.

(Refer Slide Time: 42:28)

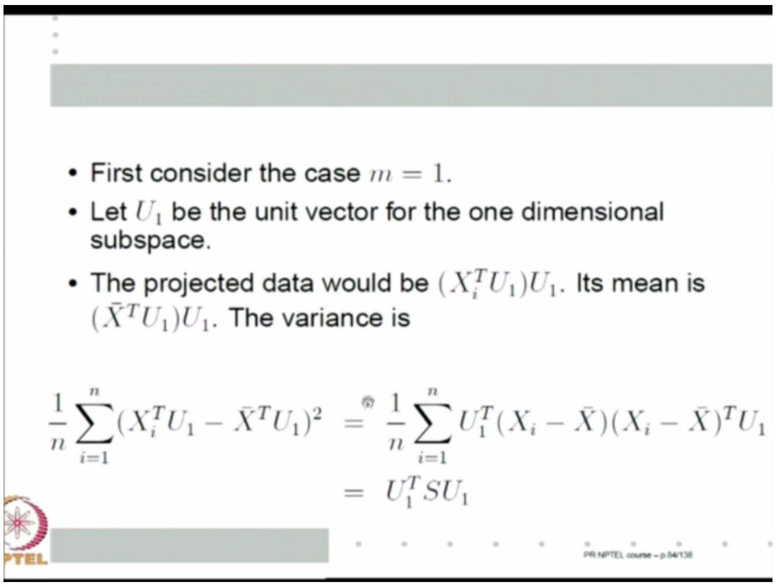


## PCA as capturing most of the variance

- Now let us look at the alternate viewpoint.
- We want to find a $m$-dimensional subspace in which the projected data would have maximum variance.
- The solution turns out to be the same.
- That is, we need to project data onto the space spanned by $U_1, \cdots, U_m$, the eigen vectors corresponding to the top $m$ eigen values of the data covariance matrix.

PR NPTEL course – p.79/138

(Refer Slide Time: 42:57)



- First consider the case $m = 1$.
- Let $U_1$ be the unit vector for the one dimensional subspace.
- The projected data would be $(X_i^T U_1)U_1$. Its mean is $(\bar{X}^T U_1)U_1$. The variance is

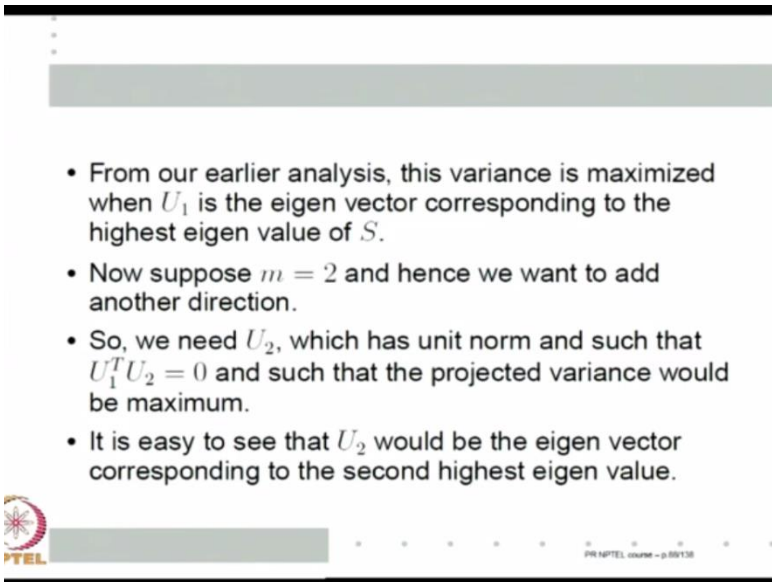
$$\frac{1}{n}\sum_{i=1}^{n}(X_i^T U_1 - \bar{X}^T U_1)^2 = \frac{1}{n}\sum_{i=1}^{n} U_1^T (X_i - \bar{X})(X_i - \bar{X})^T U_1$$
$$= U_1^T S U_1$$

PR NPTEL course – p.84/138

Now, this is a as far as the U point of finding the best mean square approximation is concerned. So, let us look at the other view point of maximizing variance, let we want to find an m dimensional subspace in which the projected data would have the maximum variance. As you already said, this solution will turns out to be same once again, you have to project data onto the space spanned by U 1 to U m, the orthonormal Eigen vectors correspond with the top m Eigen values of the data covariance matrix. We will see, we will; we will not even give as much detail for this proof as the earlier one to get our (( )) on let us consider one dimensional case.

Let us say U 1 is unit vector for the one dimensional subspace that we want to find; we want to find U 1. So, the projected data will be X i transpose U 1 along U 1 so, X i transpose U 1, U 1, this is the vector notation. So, its mean will be X bar transpose U 1 U 1, what will be its variance? X i transpose U 1 minus X bar transpose U 1 whole square summed over I is equal to 1 to n, 1 by n. This will be 1 by n sum I is equal to 1 to n, U 1 transpose X i minus X bar, X i minus X bar transpose U 1. We will all you know, what is this because 1 by n I is equal to 1 to n, X i minus X bar into X i minus X bar transpose is S, this is used U 1 transpose X U 1.

(Refer Slide Time: 43:53)

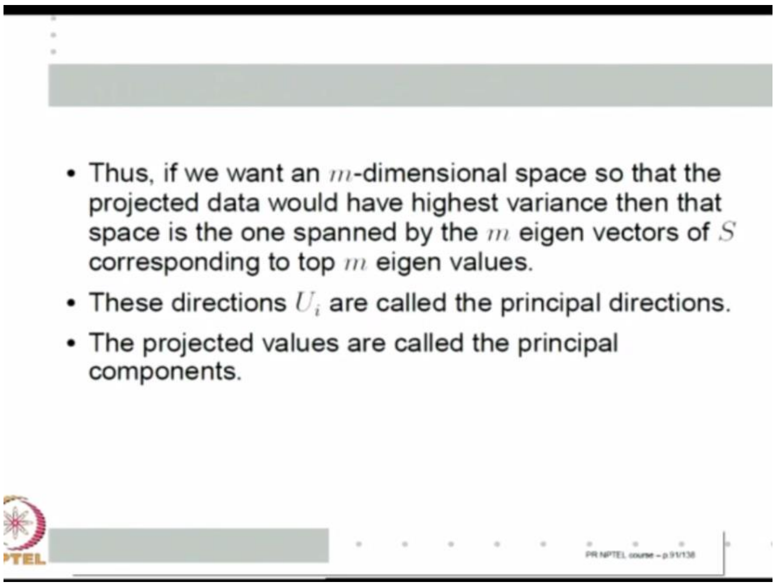


- From our earlier analysis, this variance is maximized when $U_1$ is the eigen vector corresponding to the highest eigen value of $S$.
- Now suppose $m = 2$ and hence we want to add another direction.
- So, we need $U_2$, which has unit norm and such that $U_1^T U_2 = 0$ and such that the projected variance would be maximum.
- It is easy to see that $U_2$ would be the eigen vector corresponding to the second highest eigen value.

And we already know that, this is maximized if U 1 is the Eigen vector corresponding, the highest Eigen value. See maximum minima, maximizing and minimizing is the same. Essentially, if I equate the gradient to 0, I get solution which might be either maxima or minima that tells me, that it has to be an Eigen vector and for any particular Eigen vector, I take U 1 to be, this U 1 transpose U 1 will be corresponding Eigen value. So, if you want to maximize it, U 1 should be the Eigen vector corresponding to the maximum Eigen value. If you want to minimize it, it should be Eigen vector corresponding to the least Eigen value.

So, he will want to maximize it so, my first U vector U 1 should be the Eigen vector of at covariance matrix corresponding to the maximum value. Now, suppose m is equal to 2 and we want to add one more direction, what is that we need;,we need a U 2 which has a unit now, which is that U 1 transpose U 2 is equal to 0 and the projected variance should be maximum once again you will get the same thing. And U 2 should become the Eigen vector corresponding to the second largest Eigen value and so on.

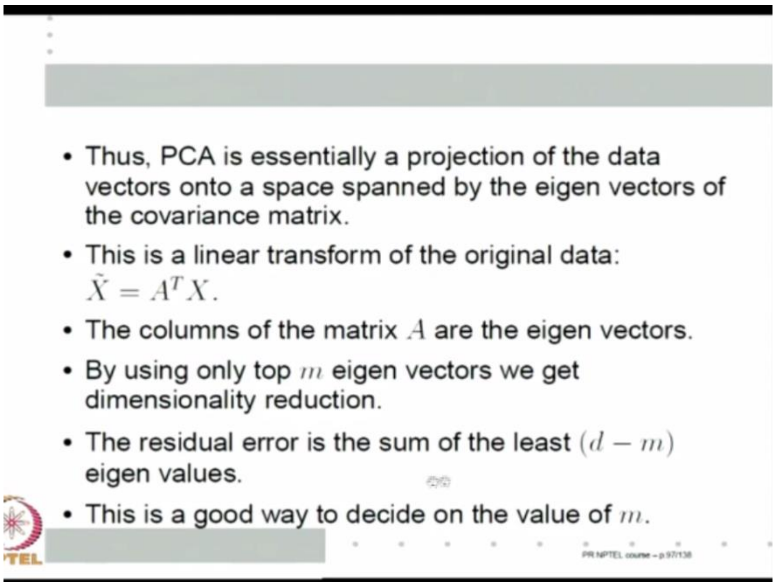
(Refer Slide Time: 45:11)



- Thus, if we want an $m$-dimensional space so that the projected data would have highest variance then that space is the one spanned by the $m$ eigen vectors of $S$ corresponding to top $m$ eigen values.
- These directions $U_i$ are called the principal directions.
- The projected values are called the principal components.

PR NPTEL course – p.91/138

(Refer Slide Time: 45:37)



- Thus, PCA is essentially a projection of the data vectors onto a space spanned by the eigen vectors of the covariance matrix.
- This is a linear transform of the original data: $\tilde{X} = A^T X$.
- The columns of the matrix $A$ are the eigen vectors.
- By using only top $m$ eigen vectors we get dimensionality reduction.
- The residual error is the sum of the least $(d - m)$ eigen values.
- This is a good way to decide on the value of $m$.

PR NPTEL course – p.97/138

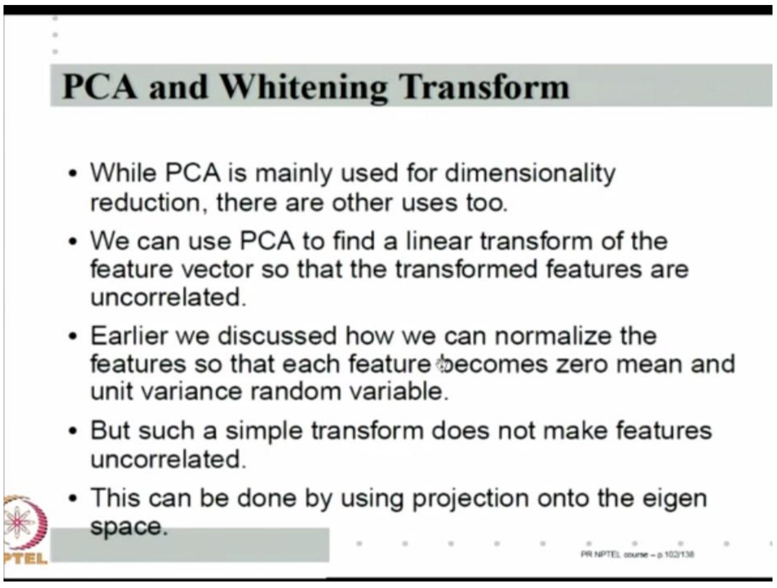
So, that is how we can say if you want an m dimensional subspace that the projected data would have the highest variance. Then that subspace is the one spanned by the m Eigen vectors of S corresponding to the top m Eigen values. These directions U i are called the principal directions and the projected values U i transpose X i and so on. They are called the principle components that is why the, this, this n type way of finding is called principal component analysis.

So, to sum up the PCA is essentially a projection of the data vectors onto a space spanned by the Eigen vectors for the covariance matrix. So, this, this is a linear transform of the original data X tilde j tilde transpose X, where columns of a are the Eigen vectors. Of course, they can choose all of them, even then I am transforming the data into a better data because, but essentially, to take by, by using only top m Eigen vectors.

So that I make a to be d by m matrix so, X tilde will be m dimensional subspace. So, if I use only top m Eigen vectors, we get dimensionality reduction. Now, what will be the; what will be the residual error? The residual error as you already seen j is equal to m plus 1 to d U j transpose U S U j which means nothing but the sum of the remaining Eigen values. So, the residual be the sum of the least d minus m Eigen values so, you find all the Eigen values look from the bottom most.

So, keep adding them that how much error you can tolerate that many Eigen values you can reject that tells me, what m I should use. This has often one way of deciding what m I look at these Eigen values and based on these decide, how many Eigen values, I can reject. If I have got a few high Eigen values and rest of the Eigen values are all small. Then along those, Eigen directions I do not have much variation, so that component of data is not important. I will only look at the top m Eigen values, that is how PCA works.

(Refer Slide Time: 47:18)



## PCA and Whitening Transform

- While PCA is mainly used for dimensionality reduction, there are other uses too.
- We can use PCA to find a linear transform of the feature vector so that the transformed features are uncorrelated.
- Earlier we discussed how we can normalize the features so that each feature becomes zero mean and unit variance random variable.
- But such a simple transform does not make features uncorrelated.
- This can be done by using projection onto the eigen space.

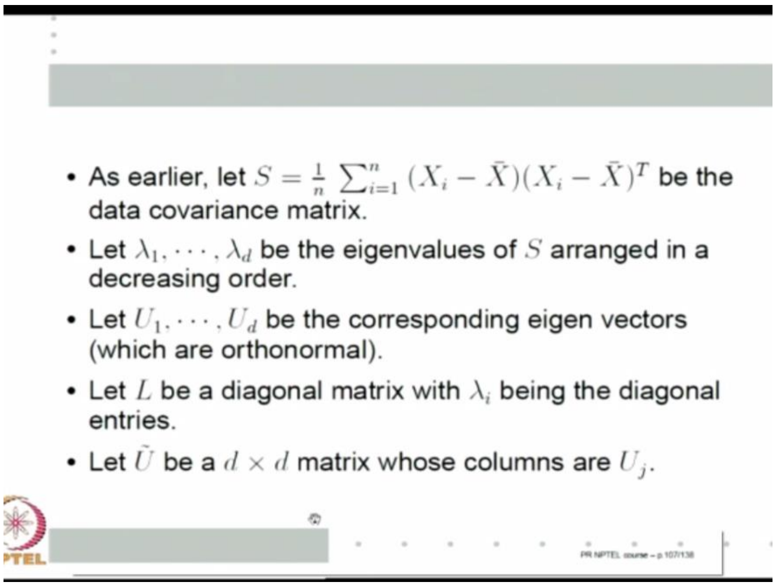
PR NPTEL course – p.103/138

We will quickly look at a couple of generic variations of PCA. First is (( )) whitening transform. Of course, so far we looked at PCA mainly for dimensional reduction that is its main use most of the time, it is used as a dimensional reduction technique. But there are other uses, we will just illustrate 1. We can use PCA to find a linear transform of the feature vector so, the transformed feature vectors are uncorrelated, when we considered our linear least squares, as well as our neural networks and so on.

We mention that it is always nice to normalize individual feature components. That is we want each feature component of zero mean and variance. So, that can be done by taking each feature component, find its mean from the data. I mean it is sample mean from the data its sample variance from the data and then do a simple linear transformation so that the mean and variance of each feature component can be standardized.

So, we discussed, how we can normalize, so at each feature component at zero mean and unit variance. But did I mean that the features themselves are uncorrelated because, it still leaving the features using PCA. We can actually transform the feature vector, so that the components can become uncorrelated which could be quite useful in pattern recognition.

(Refer Slide Time: 48:33)



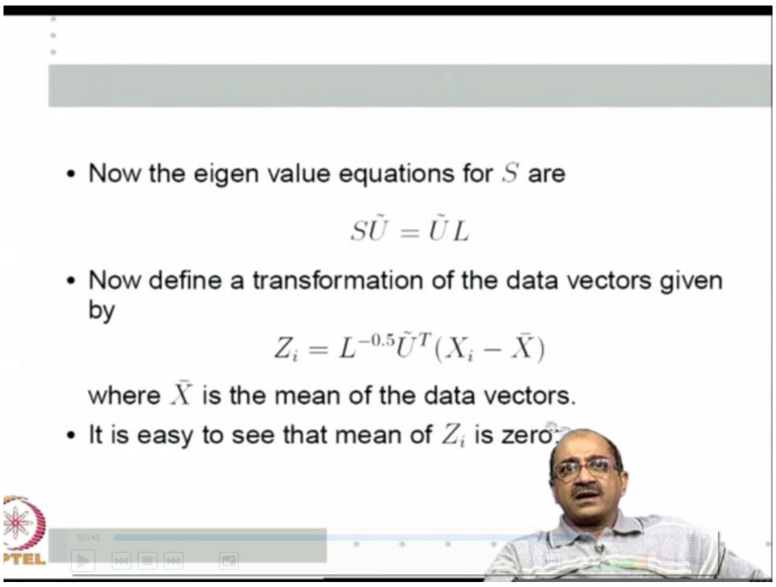
- As earlier, let $S = \frac{1}{n} \sum_{i=1}^{n} (X_i - \bar{X})(X_i - \bar{X})^T$ be the data covariance matrix.
- Let $\lambda_1, \cdots, \lambda_d$ be the eigenvalues of $S$ arranged in a decreasing order.
- Let $U_1, \cdots, U_d$ be the corresponding eigen vectors (which are orthonormal).
- Let $L$ be a diagonal matrix with $\lambda_i$ being the diagonal entries.
- Let $\tilde{U}$ be a $d \times d$ matrix whose columns are $U_j$.

PR NPTEL course – p.107/138

So, as earlier let S be the data covariance matrix and lambda 1 to lambda d be the Eigen values of S arranged in the decreasing order. U 1 to U d be the corresponding Eigen vectors, which are orthonormal remember that, because S is real symmetric and has a complete set of Eigen vectors. We can have a orthonromal, an orthonromal setup Eigen vectors of S and they are arranged in the order of lambda 1 to lambda n, that is decreasing order of the values of course.

Then let of course, this we, we will not particularly use the fact that they are decreasing values in the whitening transform. Let L be a diagonal matrix with lambda I being the diagonal entries and let U tilde be the matrix, whose columns are U j. So, U tilde be U tilde is the matrix of Eigen vectors and because these Eigen vectors are orthogonal U tilde is a orthogonal or even tilde matrix. So, U tilde is a d by d matrix because, they are d Eigen vectors each of d dimensions.

(Refer Slide Time: 49:39)



- Now the eigen value equations for $S$ are

$$S\tilde{U} = \tilde{U}L$$

- Now define a transformation of the data vectors given by

$$Z_i = L^{-0.5}\tilde{U}^T(X_i - \bar{X})$$

where $\bar{X}$ is the mean of the data vectors.

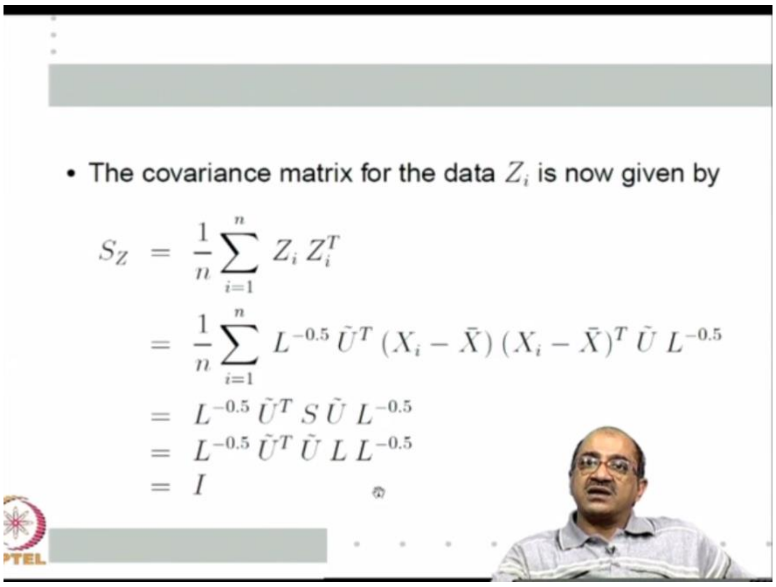
- It is easy to see that mean of $Z_i$ is zero.

So, the Eigen value, Eigen vector equation for S is S U tilde is U tilde L. So I have written all the Eigen value, Eigen vector equations together this, because L is diagonal with lambda, this is nothing but the Eigen value equation for S. Now, let me define a transformation of the vector, so I am transforms X i is into Z i is; Z i is given by l to the power minus half where L is the diagonal matrix like the Eigen values on the diagonal L to the power minus half U tilde transpose X i minus X tilde. Where X bar X i minus X bar, where X bar is the mean of the data vectors.

Now, it is very easy to see that Z i has zero mean now the, the transformation makes Z i, zero mean simply, because X i minus X bar as zero mean. So, Z i will add all the Z i is, this is a constant then that will be same as summation X i minus X i bar that will be 0.

Now, so zero mean, we achieve, we are asking, what is the covariance matrix of Z i? We want Z i covariance matrix to be identity matrix. So, that not only each Z i has unit variance Z i Z j are uncorrelated. Let us calculate the covariance matrix of Z i.

(Refer Slide Time: 51:04)

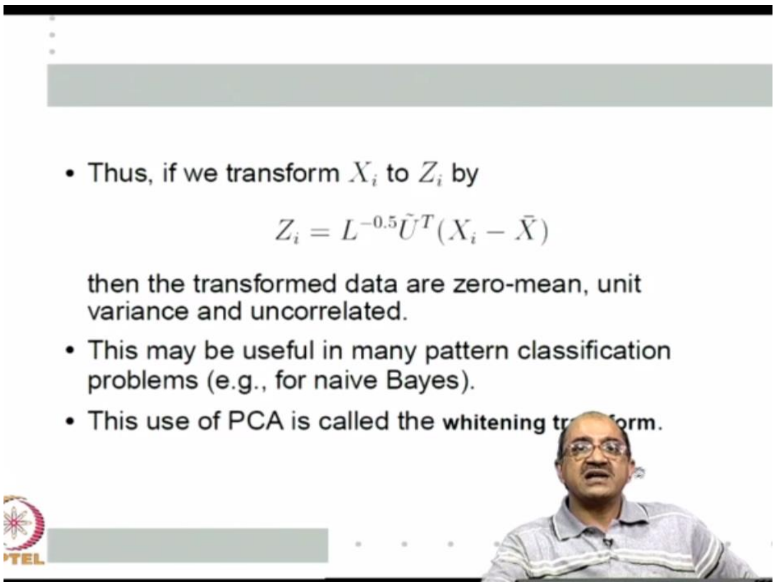


- The covariance matrix for the data $Z_i$ is now given by

$$\begin{aligned} S_Z &= \frac{1}{n}\sum_{i=1}^{n} Z_i Z_i^T \\ &= \frac{1}{n}\sum_{i=1}^{n} L^{-0.5}\,\tilde{U}^T\,(X_i-\bar{X})\,(X_i-\bar{X})^T\,\tilde{U}\,L^{-0.5} \\ &= L^{-0.5}\,\tilde{U}^T\,S\,\tilde{U}\,L^{-0.5} \\ &= L^{-0.5}\,\tilde{U}^T\,\tilde{U}\,L\,L^{-0.5} \\ &= I \end{aligned}$$

Why is the covariance matrix of Z i Z i Z i transpose because Z i is zero mean. So, let us substitute for Z i Z i is nothing but l to the power minus half U tilde transpose X i minus X bar, z i transpose is X i minus X bar transpose U tilde L to the power minus half. Now, I have this S matrix sitting in the middle 1 by n, i is equal to 1 to n, X i minus X bar X i minus X bar transpose, put that in. So I got L to the power minus half U tilde transpose S U tilde L to the power minus half. Now, because U tilde is the matrix of Eigen vectors S U is U i i S U tilde is U i. So, this is L to the power minus half U tilde transpose U i, L to the power minus half U is a orthogonal matrix these columns are orthogonal. So, U transpose U tilde transpose U tilde is identity so this goes away. And now I get L to the power minus half by L to the power minus half. This is diagonal matrix so, this is a identity. So, the covariance matrix of the data Z i now is the identity matrix.

(Refer Slide Time: 52:11)



- Thus, if we transform $X_i$ to $Z_i$ by

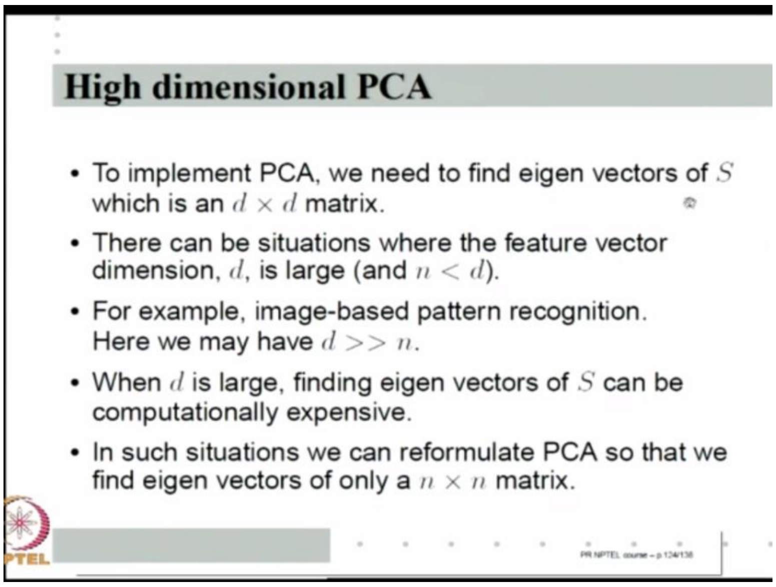
$$Z_i = L^{-0.5}\tilde{U}^T(X_i - \bar{X})$$

then the transformed data are zero-mean, unit variance and uncorrelated.
- This may be useful in many pattern classification problems (e.g., for naive Bayes).
- This use of PCA is called the **whitening tr**[illegible]**orm**.

What does this mean? If you transform X i to Z i by using Z i is equal to L to the power minus half U tilde transpose X i minus X bar. Where X bar is the mean of the original data and U tilde is the matrix, whose columns are the Eigen vectors of the data covariance matrix. And L is a diagonal matrix, whose diagonals are the Eigen values of the data covariance matrix.

Then this transformation, transform vectors into zero mean unit variance uncorrelated components. This can often be useful many for example, in naive Bayes, we assumed different features are un independent. This transformation can at least give uncorrelated a PCA used like this to make my feature vector components uncorrelated is called a whitening transform. So, this is another use of PCA apart from dimensional direction, I can use PCA as a whitening transform.

(Refer Slide Time: 53:09)



## High dimensional PCA

- To implement PCA, we need to find eigen vectors of $S$ which is an $d \times d$ matrix.
- There can be situations where the feature vector dimension, $d$, is large (and $n < d$).
- For example, image-based pattern recognition. Here we may have $d >> n$.
- When $d$ is large, finding eigen vectors of $S$ can be computationally expensive.
- In such situations we can reformulate PCA so that we find eigen vectors of only a $n \times n$ matrix.

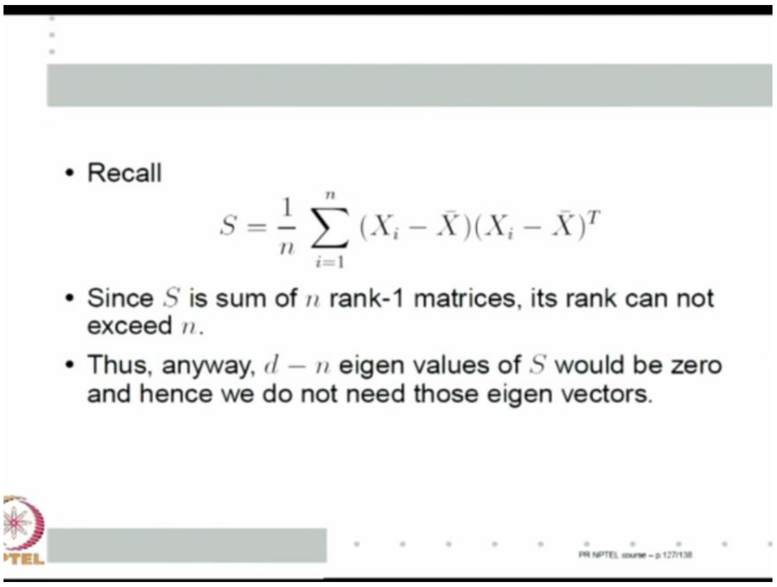
PR NPTEL course – p.124/138

There is one other interesting take on PCA that is worth knowing. To implement the PCA, we need to find Eigen values and Eigen vectors of S. The data covariance matrix is a d by d matrix, there can be situations when the feature vector dimension is large. And so large that as a matter of fact the number of examples, I have is less than d what is such situations image based pattern, when we think of let us say we I am doing face recognition.

So, I take each face image as a, as a pattern, so if I simple think of it as a vector suppose I have a 32 by 32 measure very small, highly scaled on image is still has 1000 components. If I have 64 by 64, it has a million components. That even 32 by 32, have 1000 component face image. If you want face recognition, how many different images can, can I get for one particular person on may be 20 may be 10. I cannot get 10,000 images.

So, very often n will be much less than d, because I often work with more than 32 by 32 images and I do not have so many face images of a single person and even, if I have all the face put together would not become too large. So, I may have d much larger than n and I need to find the Eigen space of S. Now, when d is large the finding the Eigen vectors of S can be computationally expensive and also may be useless. Basically, because it computationally expensive, we do not want to find all the Eigen values. So, they are asking because, n is much less than d. Is there a way I can find the Eigen vectors of some other n by n vectors instead of this d by d matrix, we show that it can be done.

(Refer Slide Time: 55:03)



- Recall

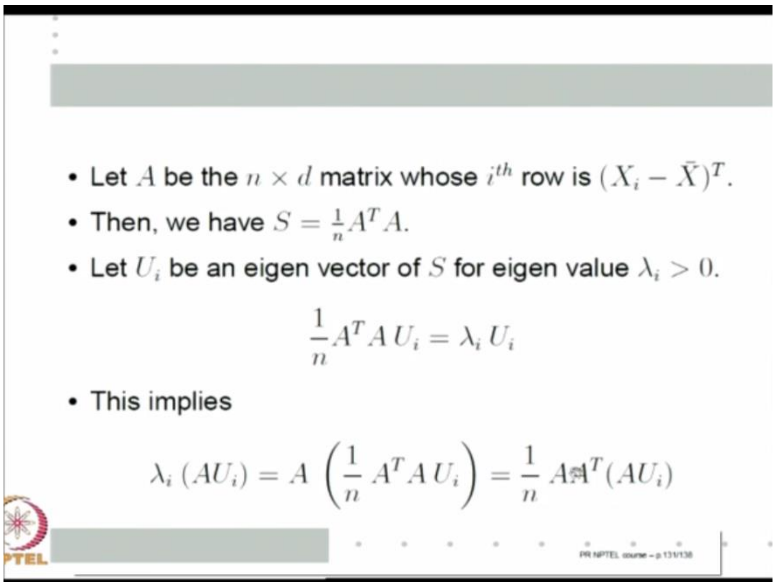
$$S = \frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X})(X_i - \bar{X})^T$$

- Since $S$ is sum of $n$ rank-1 matrices, its rank can not exceed $n$.
- Thus, anyway, $d - n$ eigen values of $S$ would be zero and hence we do not need those eigen vectors.

PR NPTEL course – p.127/138

The main reason why, we choose an n by n matrix is, S is given by this each of X I, this is some vector so, X i minus X bar X i minus X bar transpose, this is an outer product. So, if I take a vector a, and i form the matrix a, a transpose; obviously, it is a rank 1 matrix because, every column is a multiple of another column. So, because this is a rank 1 matrix, I am adding n rank 1 matrices. The rank of S can be at most n which means by any way know d minus n Eigen values will be 0. Because, the rank will be n and hence I do not need the Eigen vectors correspond to I would not use them, because there are zero Eigen values anyway drop those dimensions. So, I should be able to work just n Eigen vectors, if I want to work with n Eigen vectors I should be able to find n, Eigen vectors having n by n matrix.

(Refer Slide Time: 55:57)



- Let $A$ be the $n \times d$ matrix whose $i^{th}$ row is $(X_i - \bar{X})^T$.
- Then, we have $S = \frac{1}{n} A^T A$.
- Let $U_i$ be an eigen vector of $S$ for eigen value $\lambda_i > 0$.

$$\frac{1}{n} A^T A\, U_i = \lambda_i\, U_i$$

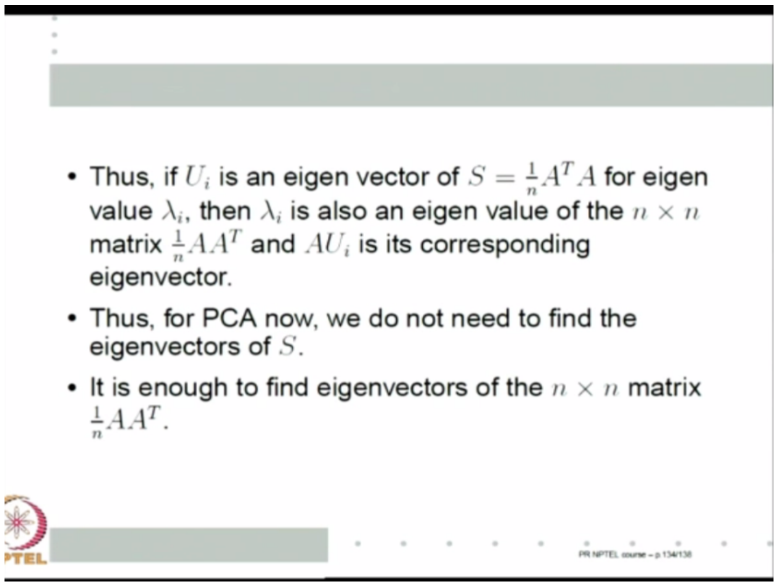
- This implies

$$\lambda_i\, (AU_i) = A \left( \frac{1}{n} A^T A\, U_i \right) = \frac{1}{n} AA^T (AU_i)$$

PR NPTEL course – p.131/138

So, what n by n matrix should I use so let A be the n by n d matrix whose ith row is X i minus X bar transpose. Then by definition S is 1 by n a transpose A X i minus X bar, X i minus X bar transpose summed over i. Let U i be an Eigen vector of S with an Eigen value lambda for a Eigen value greater than 0. So, that means 1 by n A transpose A into U i is lambda U i, if I multiply both size with a that is A lambda, i times A U i will be a times 1 by n A transpose A U i the because matrix multiplication associated. We can write as 1 by n this is scalar bring it out, I have A, A transpose and A U can be put that means I have 1 by n A, A transpose times A U is lambda times A U. So, A U i is a Eigen vector corresponding to the Eigen value lambda i of the matrix one by n A, A transpose A is an n by d matrix. So, A, A transpose is a n by n matrix, S is of course, A transpose A instead of that, if I do 1 by n, A A transpose the A U i is are Eigen values.

(Refer Slide Time: 57:09)

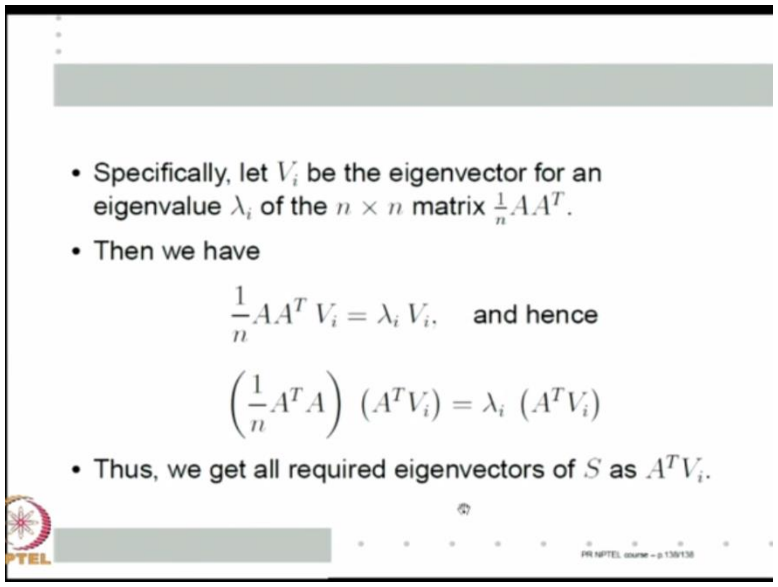


- Thus, if $U_i$ is an eigen vector of $S = \frac{1}{n}A^T A$ for eigen value $\lambda_i$, then $\lambda_i$ is also an eigen value of the $n \times n$ matrix $\frac{1}{n}AA^T$ and $AU_i$ is its corresponding eigenvector.
- Thus, for PCA now, we do not need to find the eigenvectors of $S$.
- It is enough to find eigenvectors of the $n \times n$ matrix $\frac{1}{n}AA^T$.

PR NPTEL course – p.134/138

So, U i is an Eigen vector of S 1 by a transpose a for an Eigen value lambda i. Then lambda i is also happens to be Eigen value of the n by n matrix 1 by n A, A transpose and A U i is the corresponding Eigen vector. Which means for taking PCA now, I do not need to find the Eigen vectors of S, I can work with the Eigen vectors of the n by n matrix. Specifically, what can I do?

(Refer Slide Time: 57:30)



- Specifically, let $V_i$ be the eigenvector for an eigenvalue $\lambda_i$ of the $n \times n$ matrix $\frac{1}{n}AA^T$.
- Then we have

$$\frac{1}{n}AA^T\, V_i = \lambda_i\, V_i, \quad \text{and hence}$$

$$\left(\frac{1}{n}A^T A\right)\ (A^T V_i) = \lambda_i\ (A^T V_i)$$

- Thus, we get all required eigenvectors of $S$ as $A^T V_i$.

PR NPTEL course – p.135/138

Let V i be the Eigen vector of corresponding Eigen value lambda i of the n by n matrix 1, 1 by n A, A transpose. I hope you remember, what A is, A is the n by d matrix whose i through X i minus X power transpose. So, S is actually 1 by n A transpose A instead of that, we are looking at 1 by N A, A transpose 1 by n A transpose A is d by d matrix, 1 by 1 A A transpose A is a n by n matrix. If V i is Eigen vector, I know 1 by n A A transpose V i is lambda i V i. And hence, 1 by n A transpose A into A transpose V i. So, if I multiply both sides of the equation is a transpose I can put a transpose A here, and A transpose V i here, because of associatively so, 1 by n A transpose A into a transpose V i is lambda i times a transpose A.

So, I find all the Eigen vectors V i of 1 by n A transpose, this is an n by n matrix. And then I can get the Eigen vectors of 1 by n A transpose A, which is my data covariance matrix as simply A transpose V i. That is we get all the required Eigen vectors of S. So, this is how one handles high dimensional PCA. So, we stop the PCA here, so we would not come to feature vector selection anymore next class. We look at a few other special topics of pattern recognition.

Thank you.