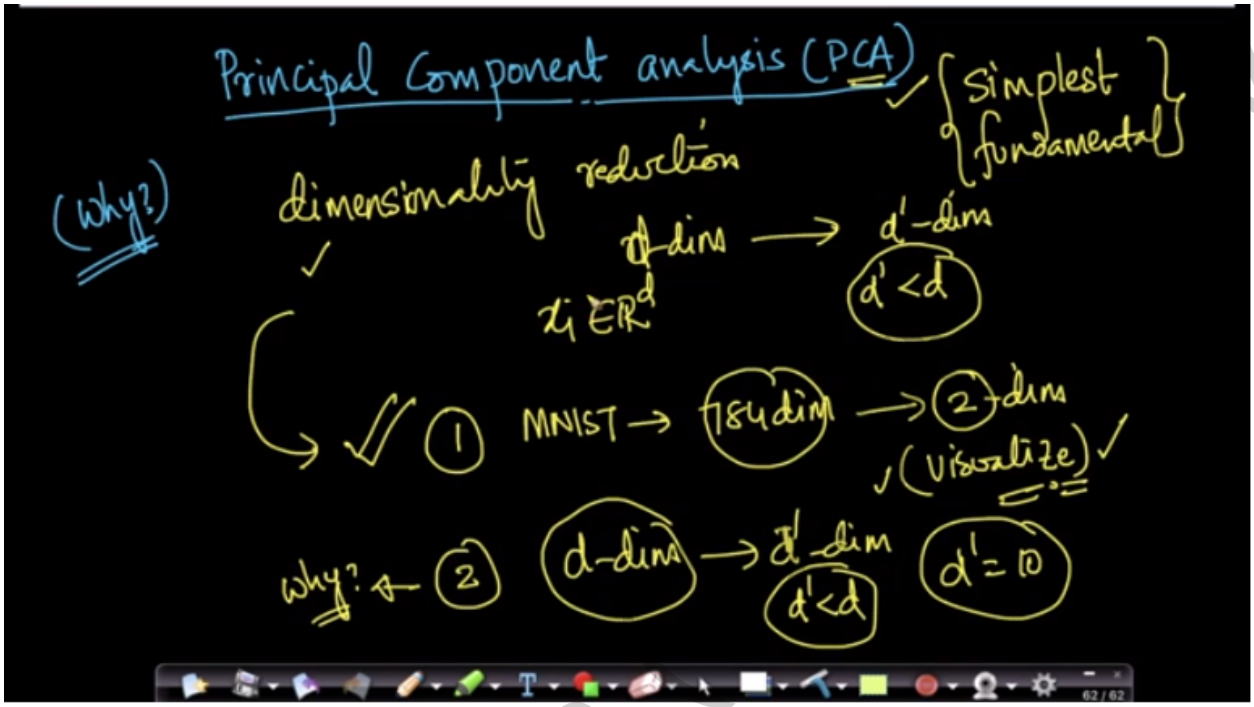
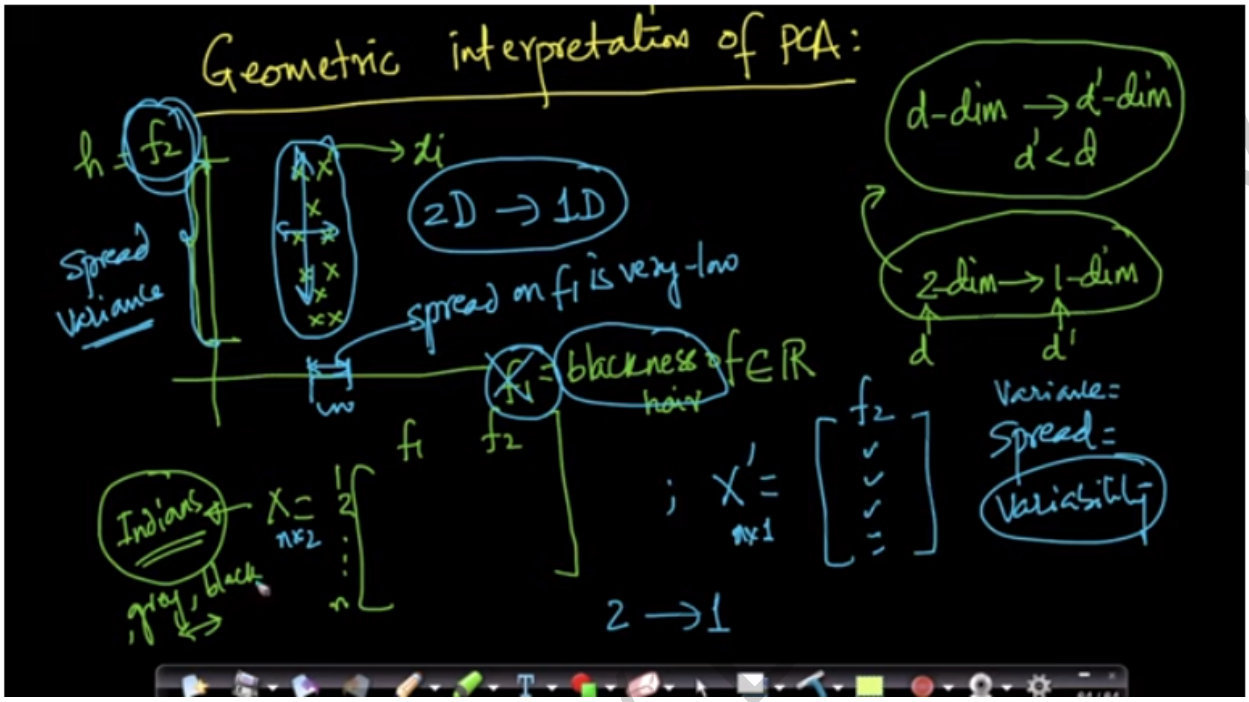
26.1 Why learn PCA?



Timestamp: 3:25

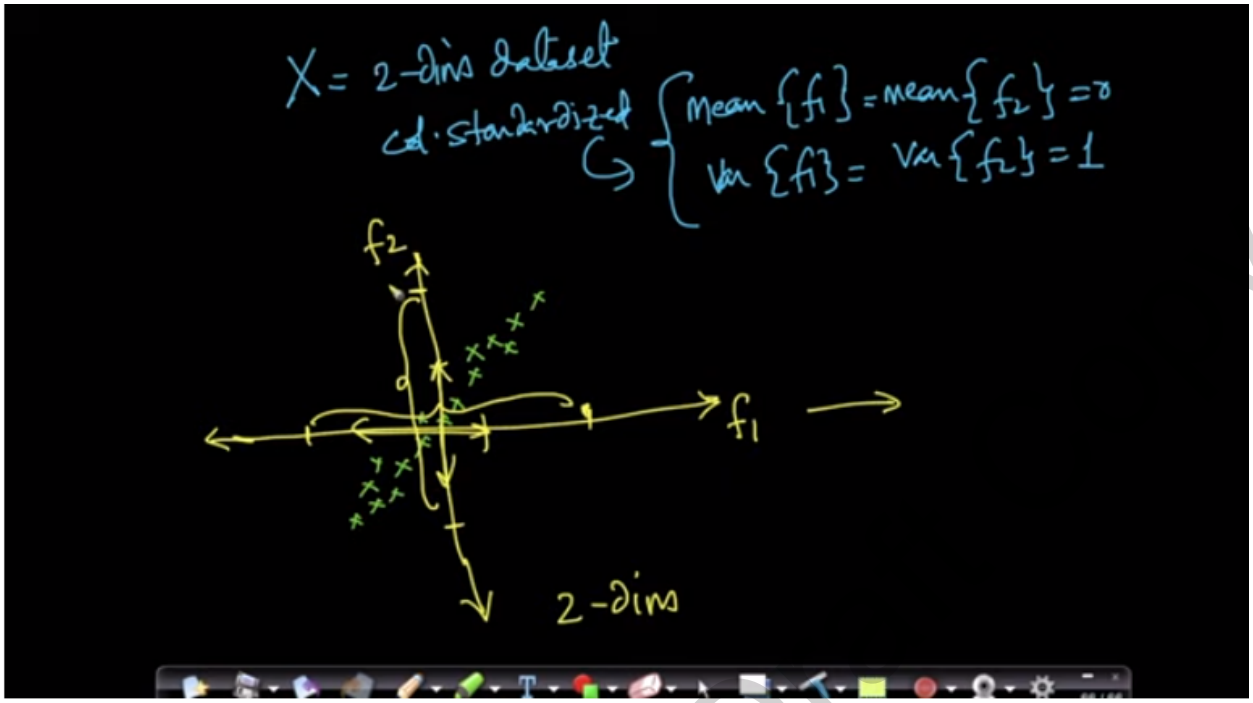
PCA is used for dimensionality reduction to convert data from d dimensions to d' dimensions where d'<d. It has applications to both visualize the data by converting it into 2-D or 3-D. Apart from visualization reducing the dimension of data using PCA has other applications which will be discussed in further chapters.

26.2 Geometric intuition of PCA



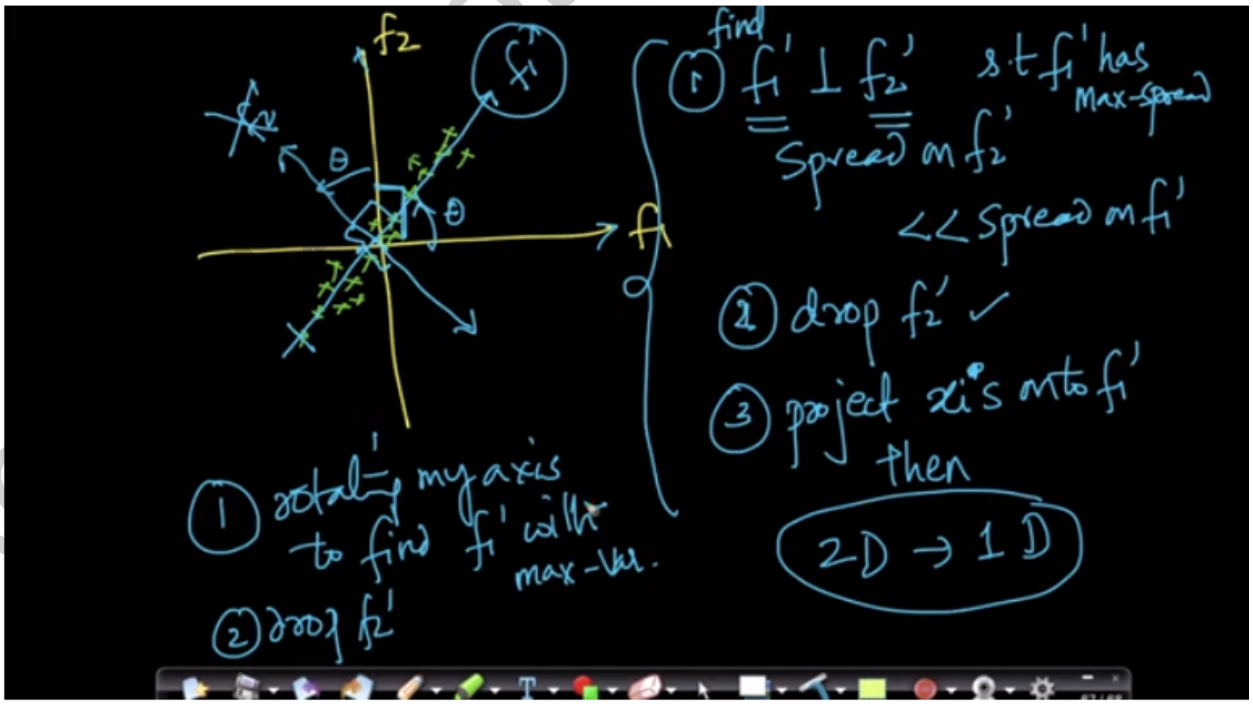
Timestamp: 5:24

As mentioned in the above figure, let's say that the feature f2 has more spread than feature f1, if we have to reduce the dimensions from 2-D to 1-D then we can pick up the feature f2 where the spread or variability of the data is much more. More the spread, the more we can visualize our data better. Hence we reduce our data which contains feature f1 and f2 to a data that contains only one feature f2 since it has more spread and preserve the direction in which the data has more spread.



Timestamp: 9:16

Suppose we have a 2-D dimensional dataset which is column standardized and let us consider the data is spread more not along the direction f1 or f2 but rather as shown in the image. Then we cannot simply just drop f1 or f2 because the spread is not in either of f1 or f2 direction. For this we have to do something smart as below.



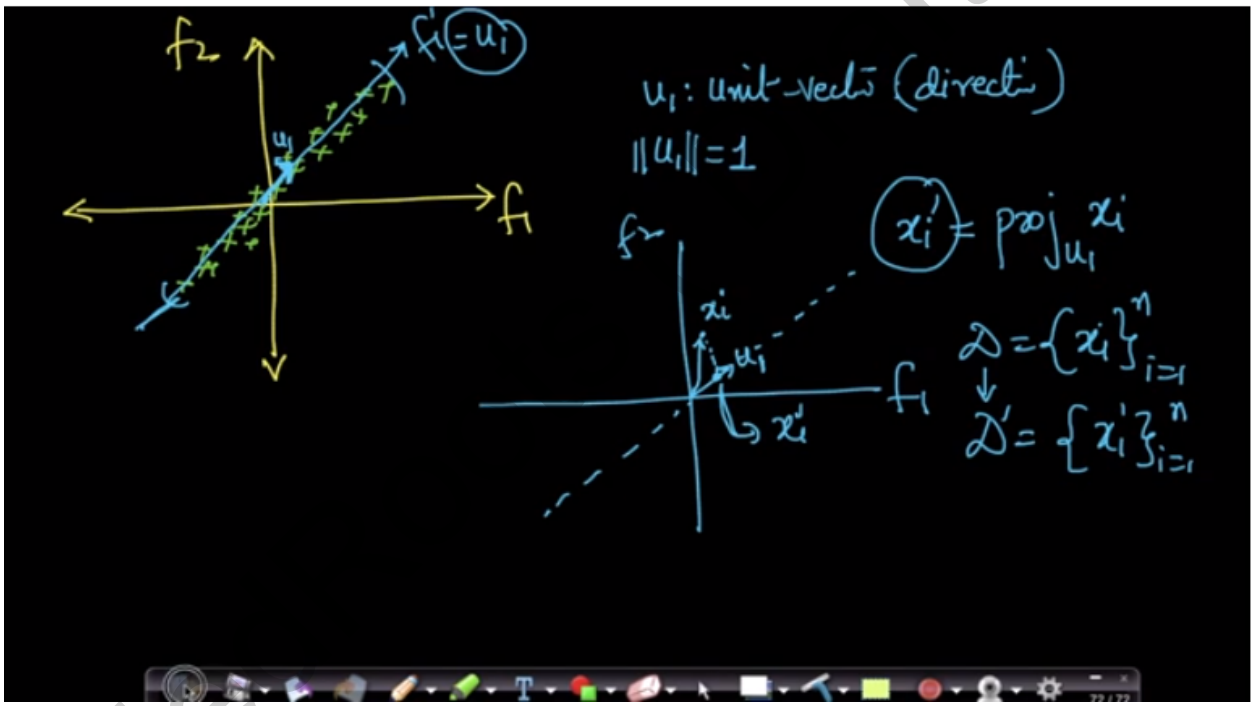
Timestamp: 13:43

As shown in the above figure, we will find f1' and f2' such that f1' is perpendicular to f2' such that the spread of f1' is much greater than the spread of f2'.

We will then drop f2' and project our data to f1', thereby reducing data from 2-D to 1-D.

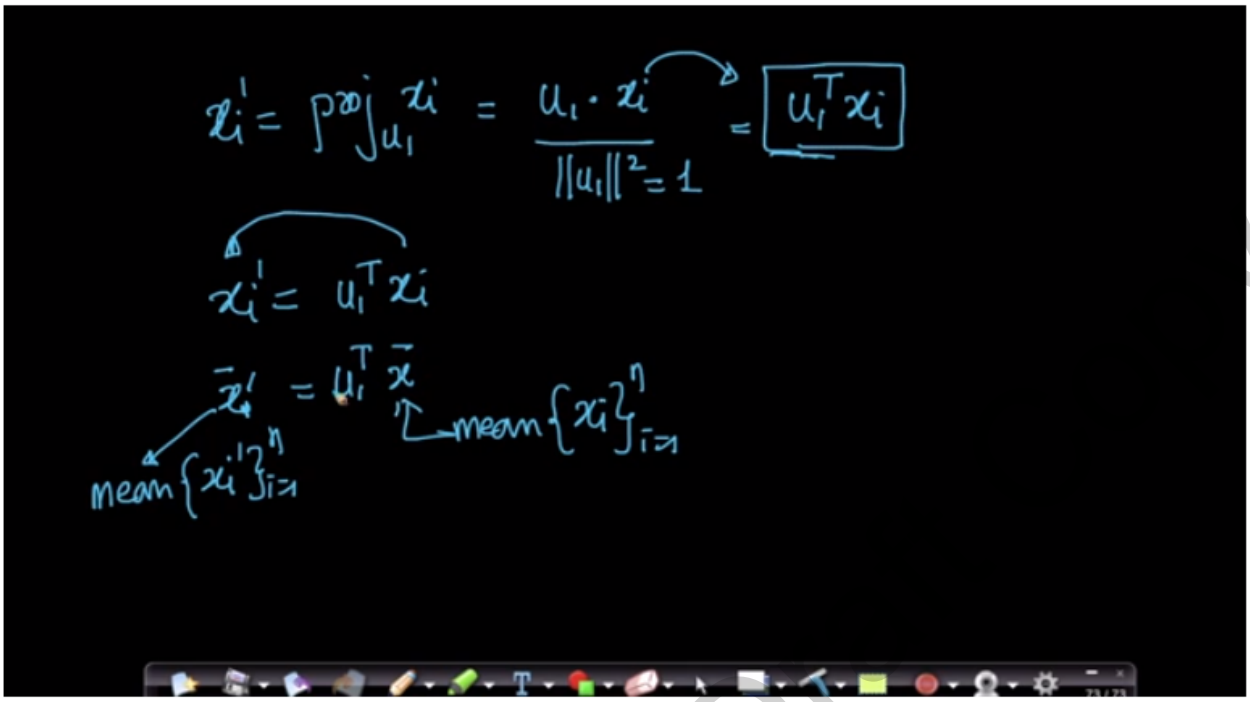
We can think of this as rotating the axis to find f1' that has the max-variance and dropping f2'.

26.3 Mathematical objective function of PCA



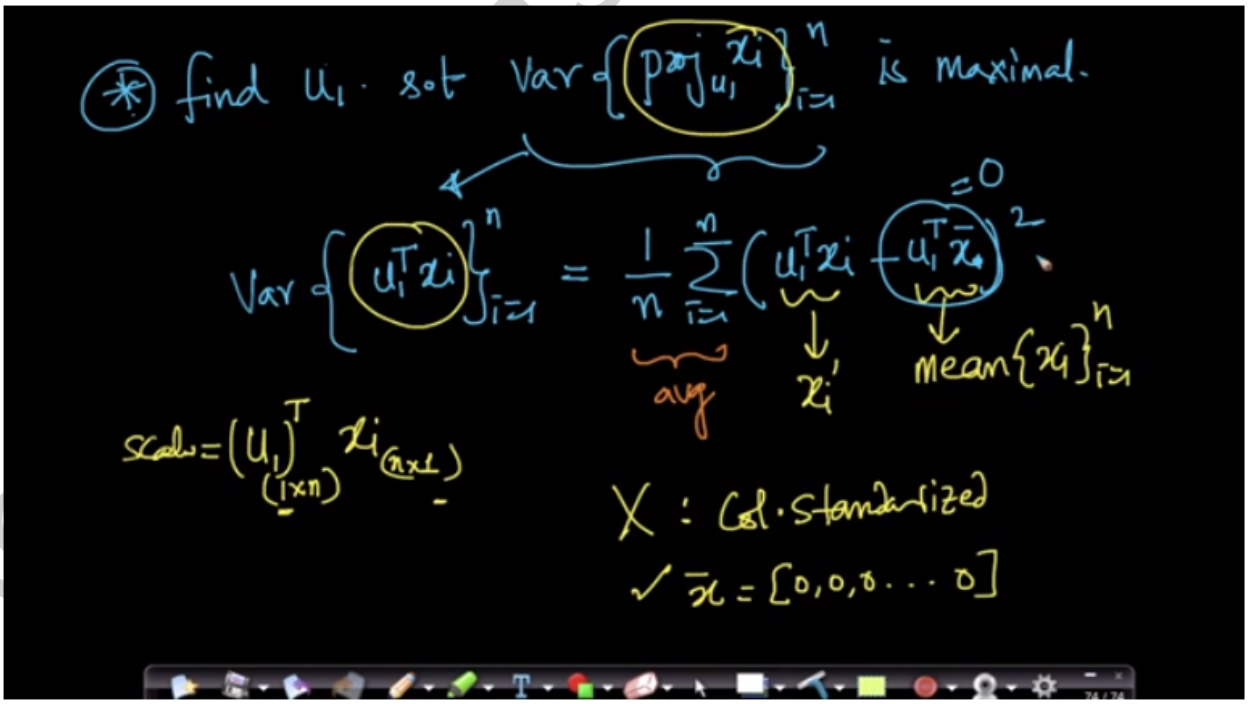
Timestamp: 2:34

As discussed before we want to find a direction where the spread of our data is more. Let us say u1 is a unit vector in that direction. Then to get the new dataset we will project each of our points on u1 to get the new dataset D' as shown in the image.



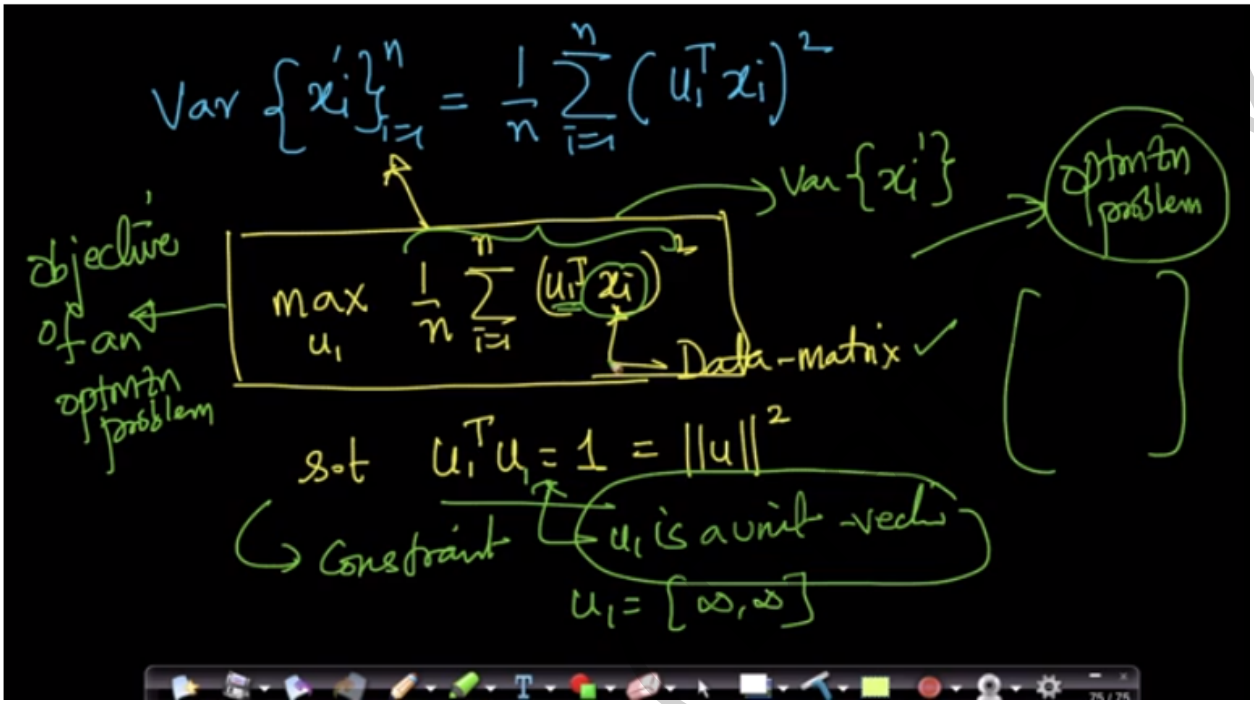
Timestamp: 4:19

Since vector u1 is a unit vector, the projection of point xi on u1 xi' will be just $u1^T xi$, notice that mean of xi' which is xi' bar , will be $u1^T xi$ bar. Meaning the mean of projections will be $u1^T$ multiplied by the mean of xis.



Timestamp: 7:28

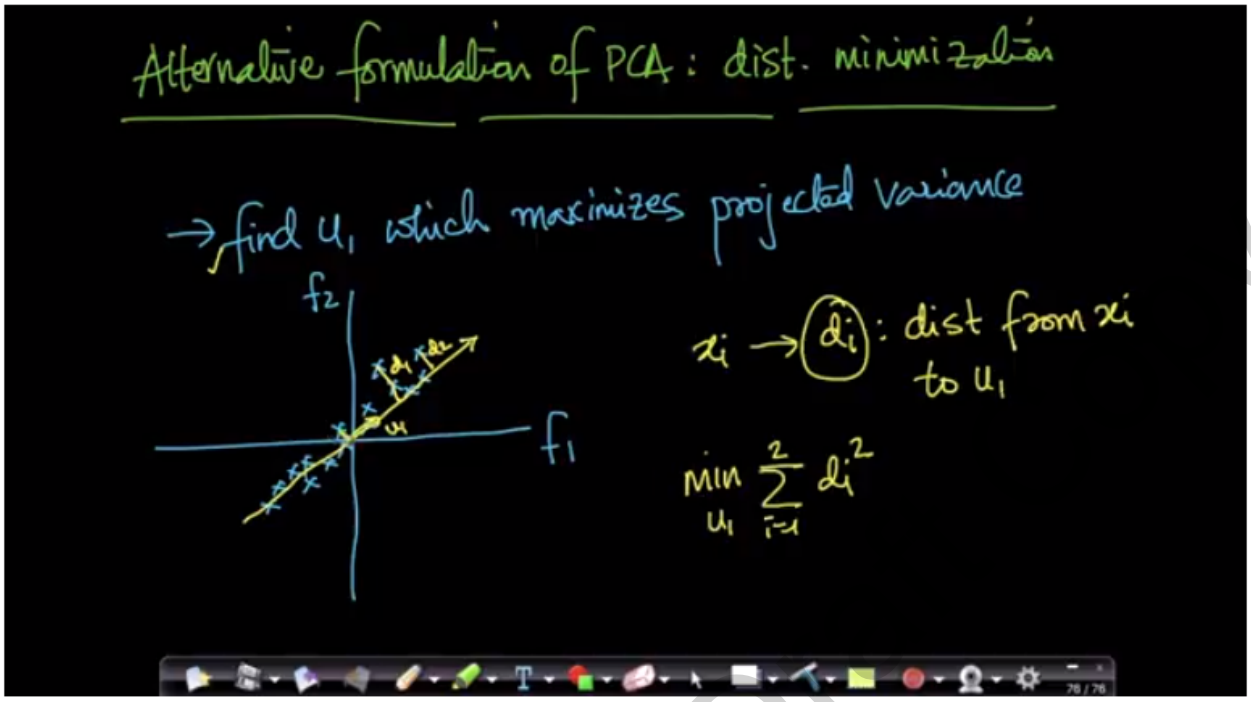
Our problem is to find u1 such that the variance of projections along u1 is maximal. So writing the variance of our projections, since we have column standardized the data, the mean of the original data points will be a zero vector. Hence our variance is reduced as below.



Timestamp: 11:23

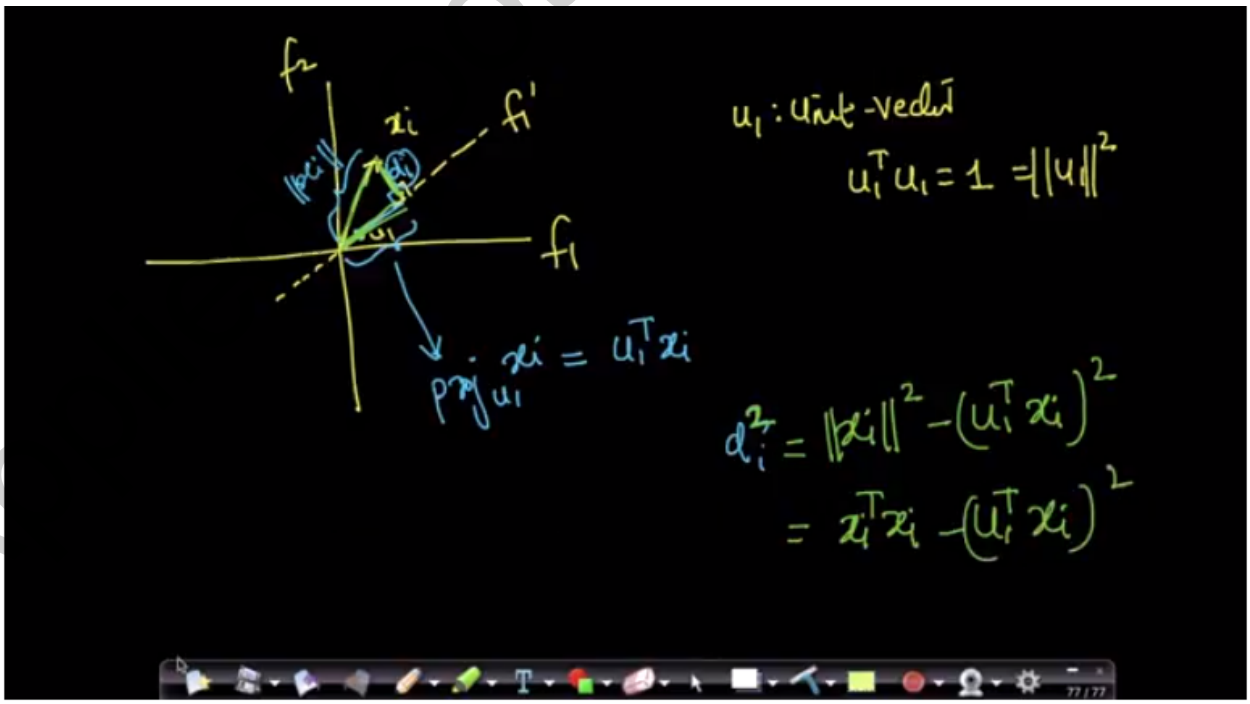
So the variance of the projections is reduced to the sum mentioned in the above figure. So now we have our optimization problem for PCA is to find u1 that maximizes the sum highlighted such that $u1^Tu1 = 1$. This constraint of making u1 a unit vector is important since otherwise any vector can satisfy the above equation by making its magnitude infinity. We will see how to solve these optimization problems in later chapters.

26.4 Alternative formulation of PCA: Distance minimization



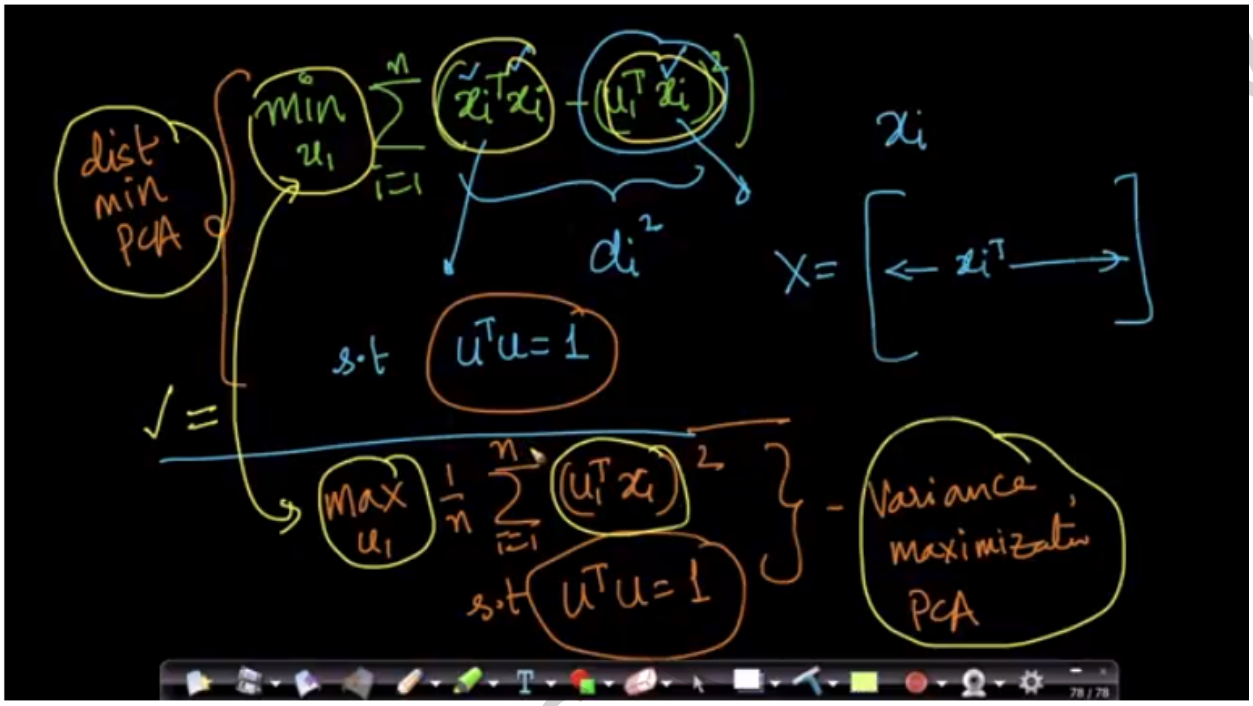
Timestamp: 2:47

Distance minimization is an alternative formulation of PCA, in which instead of choosing u1 that maximizes the variance of our projections, we want to choose u1 that minimizes the distance from xi to u1. Both are equivalent but just different formulations.



Timestamp: 5:36

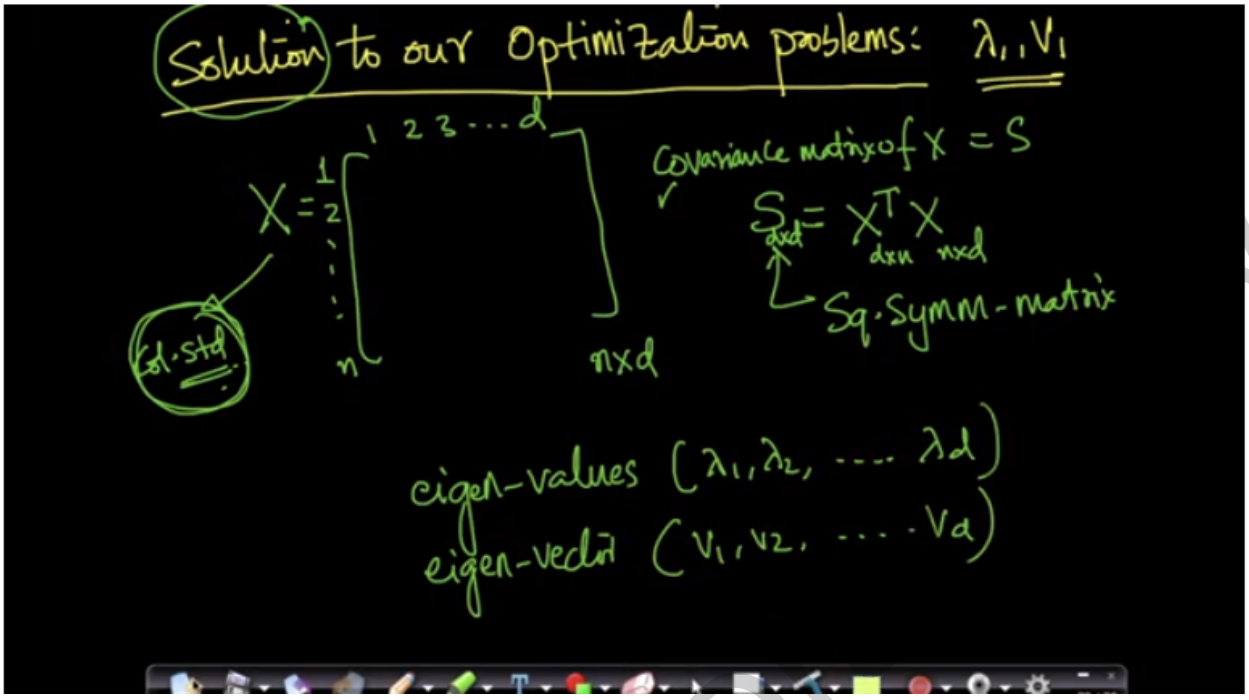
We find the distance di of our point xi from the direction u1, using pythagoras theorem as above. So our final objective function would look like below.



Timestamp: 9:42

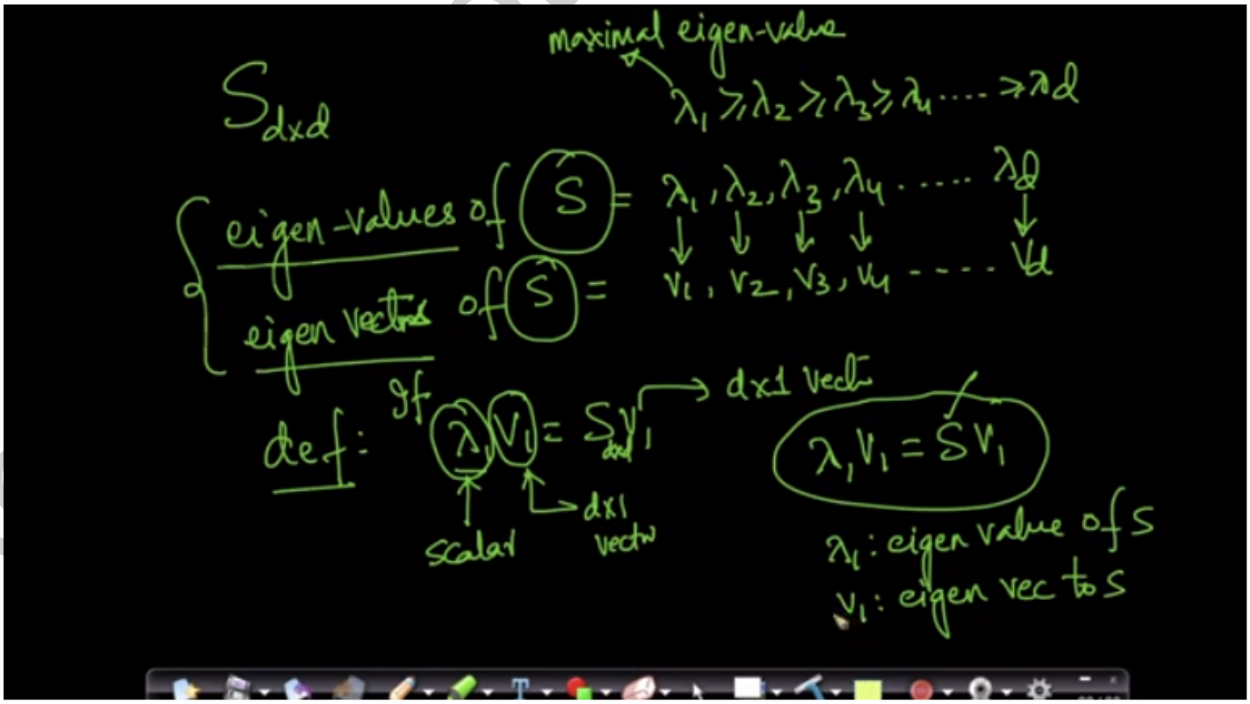
So our objective function according to distance formulation looks like above, it can be proved that it will be equal to the variance maximization formulation of PCA. (the proof will be understood when optimization is taught later in the course).

26.5 Eigenvalues and Eigenvectors(PCA): Dimensionality reduction



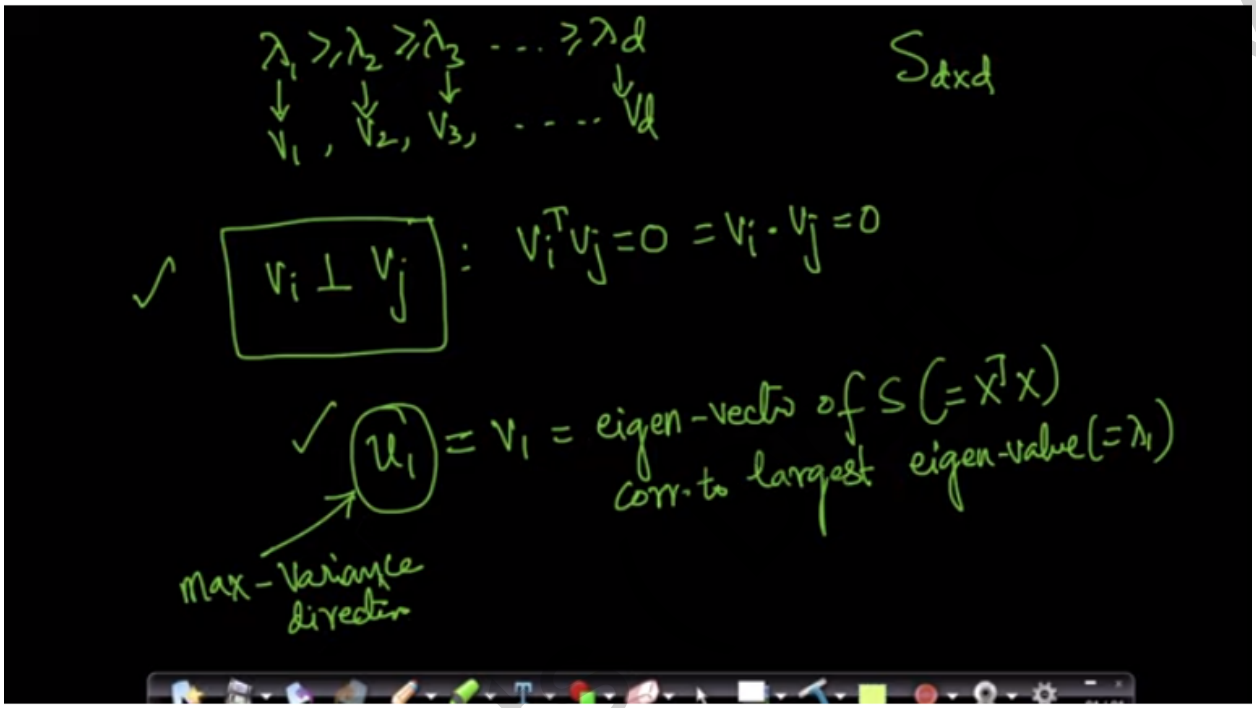
Timestamp: 3:10

Solution to our optimization problem will be to find the eigenvalues and eigenvectors of the covariance matrix S of our data matrix X. (Correction in image $S=(X^TX)/n$ [$S=(XTX)/(n-1)$ for unbiased estimate. What is meant by biased estimate and unbiased estimate will be covered in the upcoming live session]



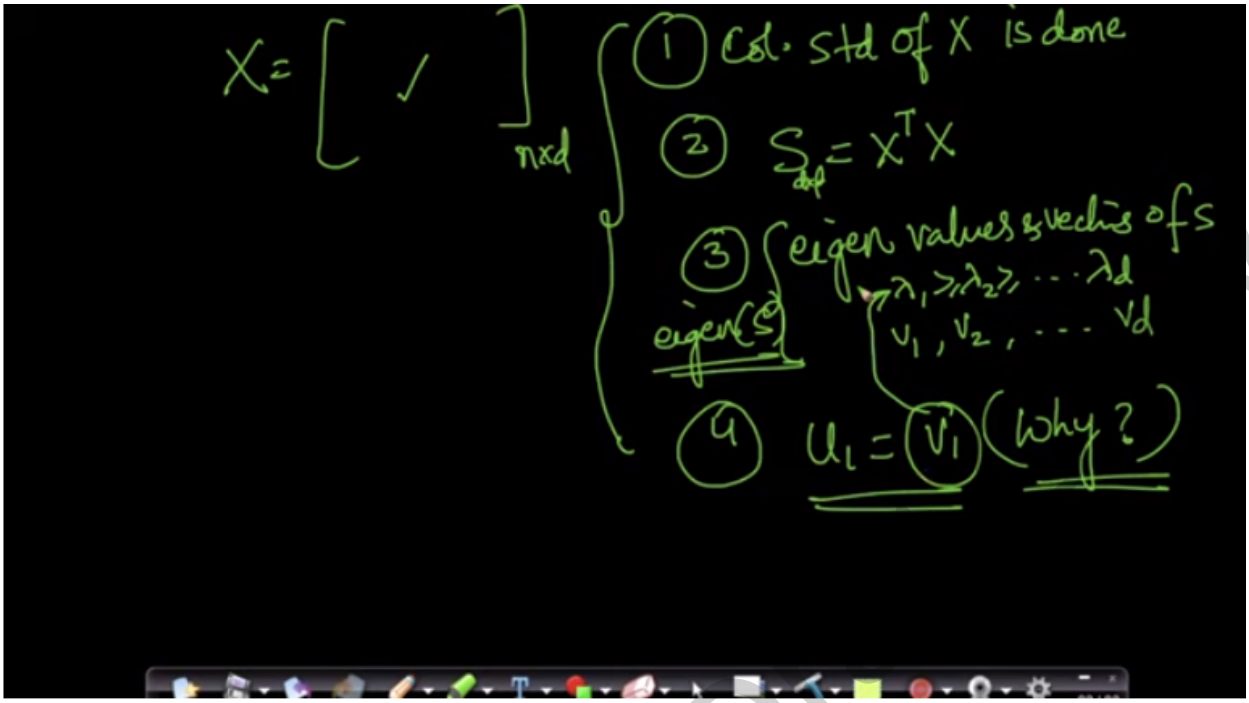
Timestamp: 6:33

Finding eigenvalues and eigen vectors of any matrix S is pretty straightforward and we might learnt earlier.We have to find eigen vectors and eigen values such that they satisfy the condition highlighted.



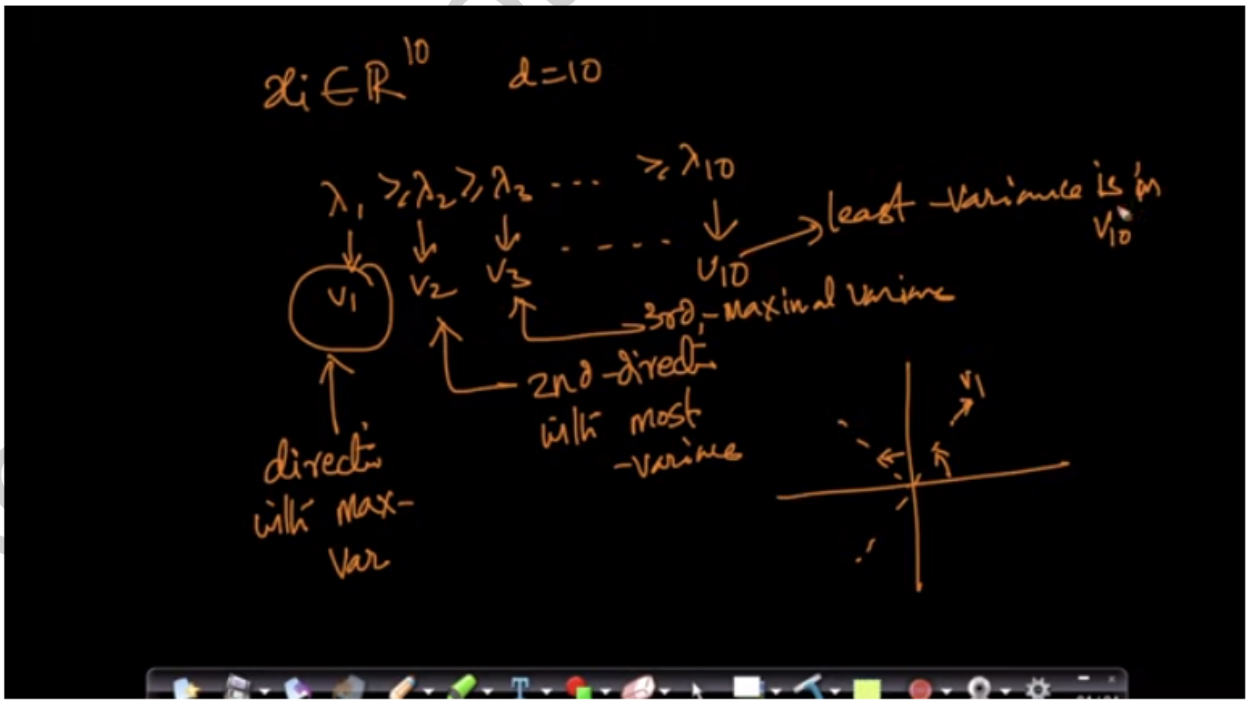
Timestamp: 9:10

Eigenvectors have a property that every eigen vector is perpendicular to every other eigenvector. If we can find the eigenvectors and eigenvalues of the covariance matrix S, then we can easily get the direction u1 which maximizes the variance as it is equal to the eigenvector of S that corresponds to the largest eigenvalue.



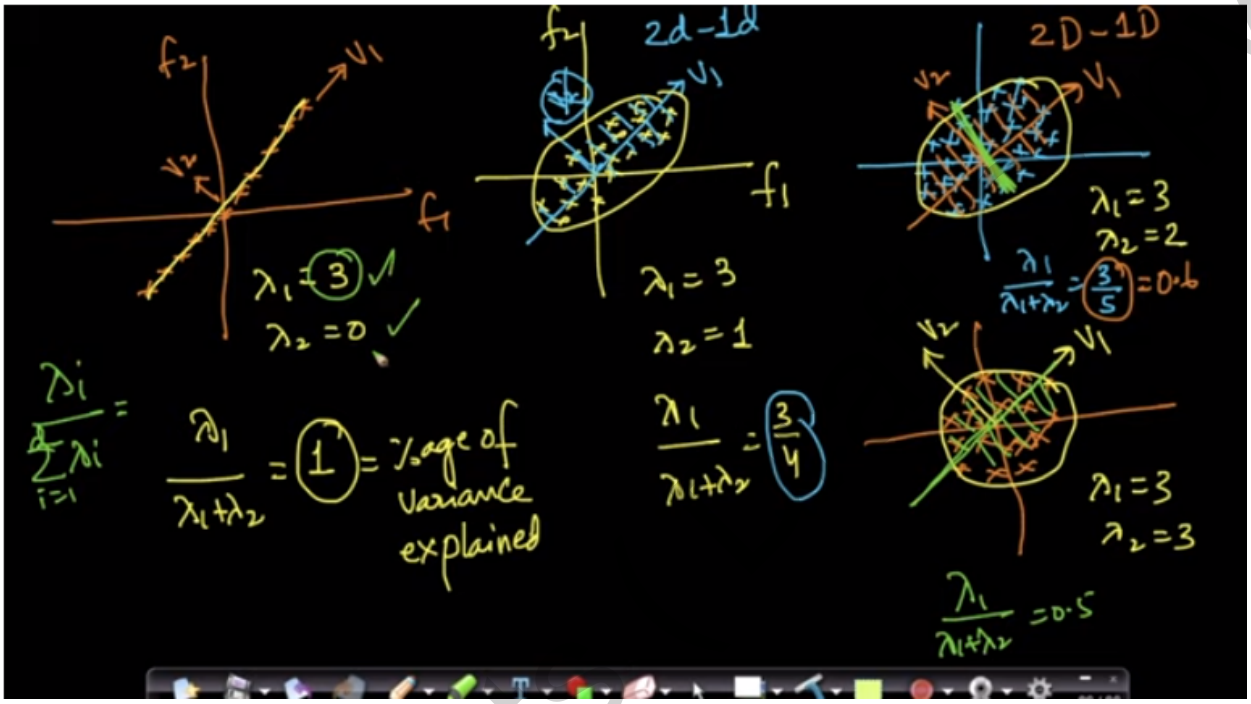
Timestamp: 11:12

So given a matrix X, in order to find u1, the max-variance direction, we need to ensure that X is column standardized, post this we can calculate the covariance matrix S. Following that you can just find the eigenvalues and eigenvectors of S and u1 will just be the eigenvector with max eigenvalue.



Timestamp: 15:33

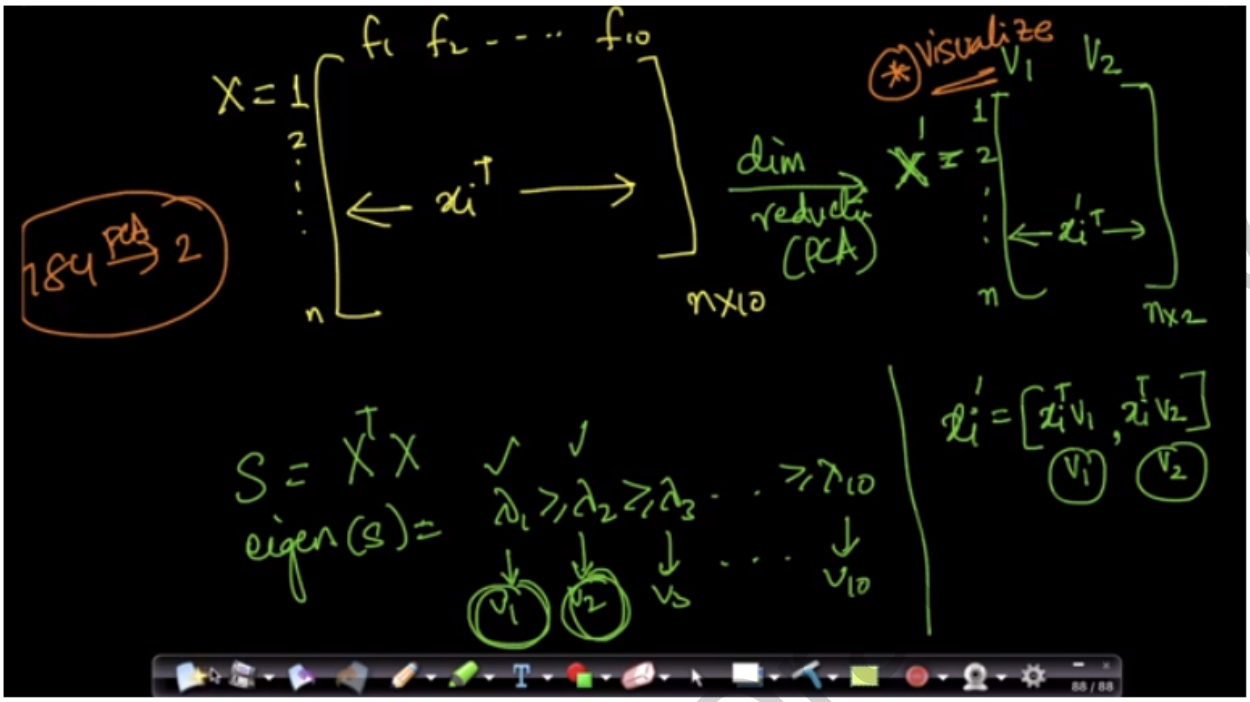
So if we sort of our eigenvectors of the covariance matrix S in the decreasing order of their corresponding eigenvalues, then v1 represents the direction with max-variance , v2 represents the direction with second max-variance and so on.



Timestamp: 22:36

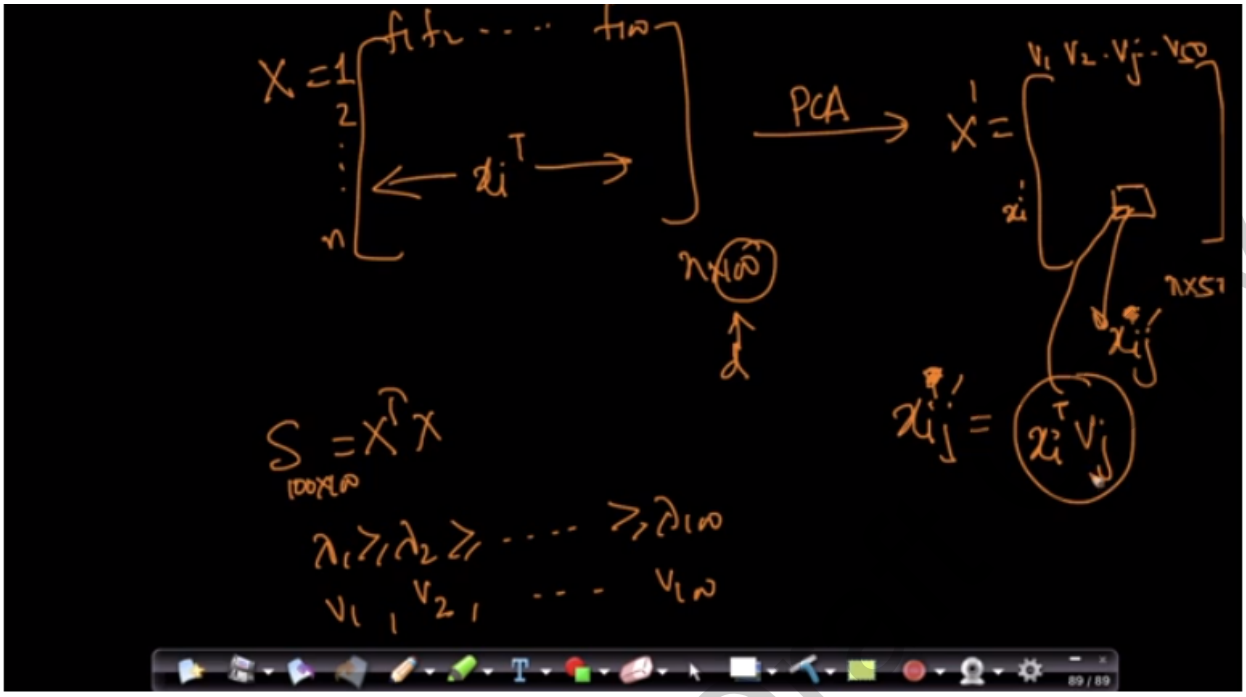
While eigenvectors v1,v2,etc tell us the directions of most-variance, second most variance etc. ƛ1,ƛ2,etc tell us the variance explained in that direction. As shown in the above figure ƛ1/(ƛ1+ƛ2) tells us the percentage of variance explained by the direction of v1.

26.6 PCA for Dimensionality Reduction and Visualization



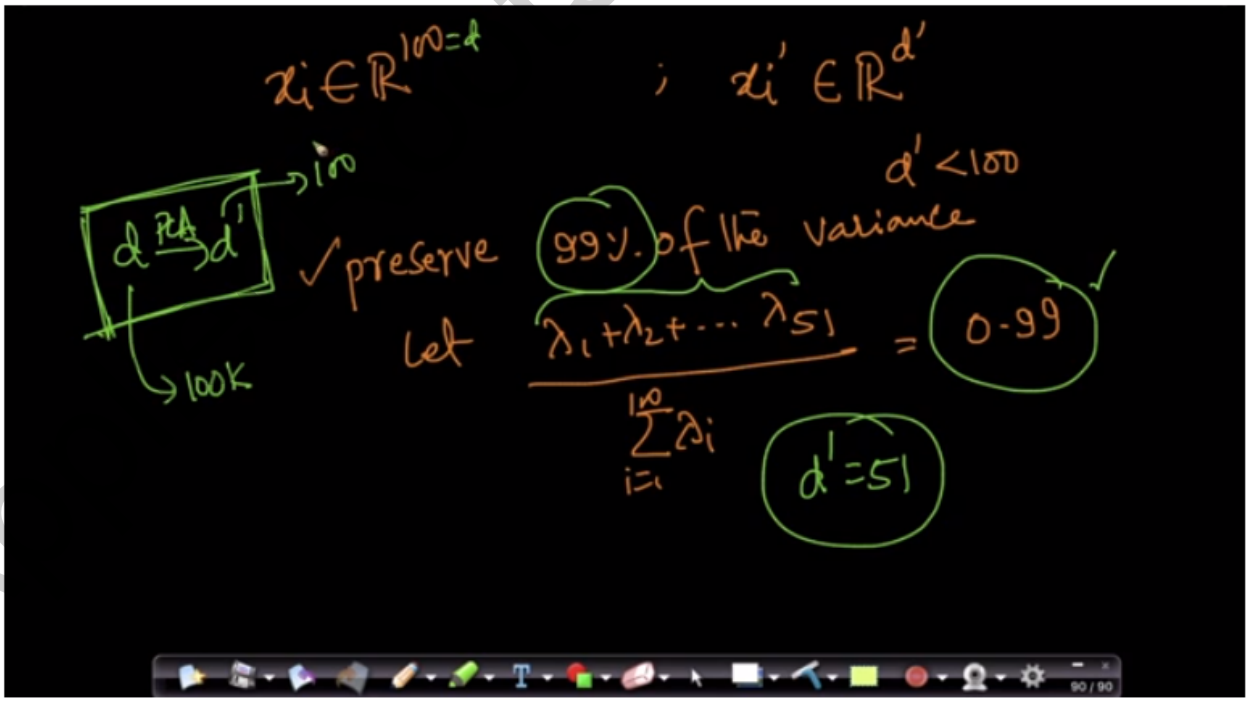
Timestamp: 5:32

Suppose we have data X of 784 dimensions, for us to visualize this data, we can take the covariance matrix S and find the eigenvalues and eigenvectors of matrix S and find the eigenvectors v1,v2 that have top two eigenvalues and transform this data to X' by taking each xi and transforming into xi' by doing xi^Tv1 and xi^Tv2. Since we can visualize 2-D, we can visualize X'.



Timestamp: 7:42

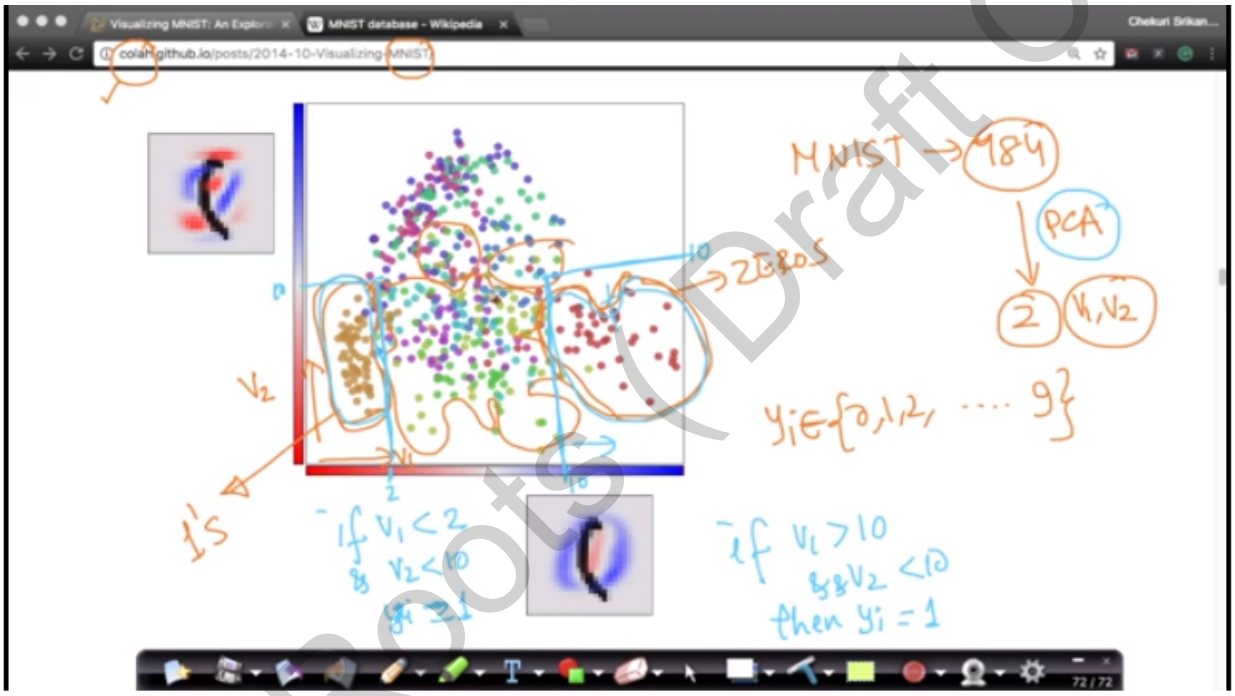
If we want to use PCA for just dimensionality reduction, to reduce our data from let's say 100 dimensions to 50 dimensions, then we can simply find the top50 eigenvectors and find xij' by just doing $x_i^T v_j$.



Timestamp: 9:42

Often it is the case that we don't pre-decide d' the dimensions to reduce the data to, often we need to preserve 99% of variance,etc. Then we just simply find the number of eigen vectors that needs to be taken to preserve at least 99% of variance and if 51 eigenvectors account for 99% of total variance then we just project our data onto this top 51 eigenvectors to get our dimension reduced data that has 99% of its variance preserved.

26.7 Visualize MNIST data

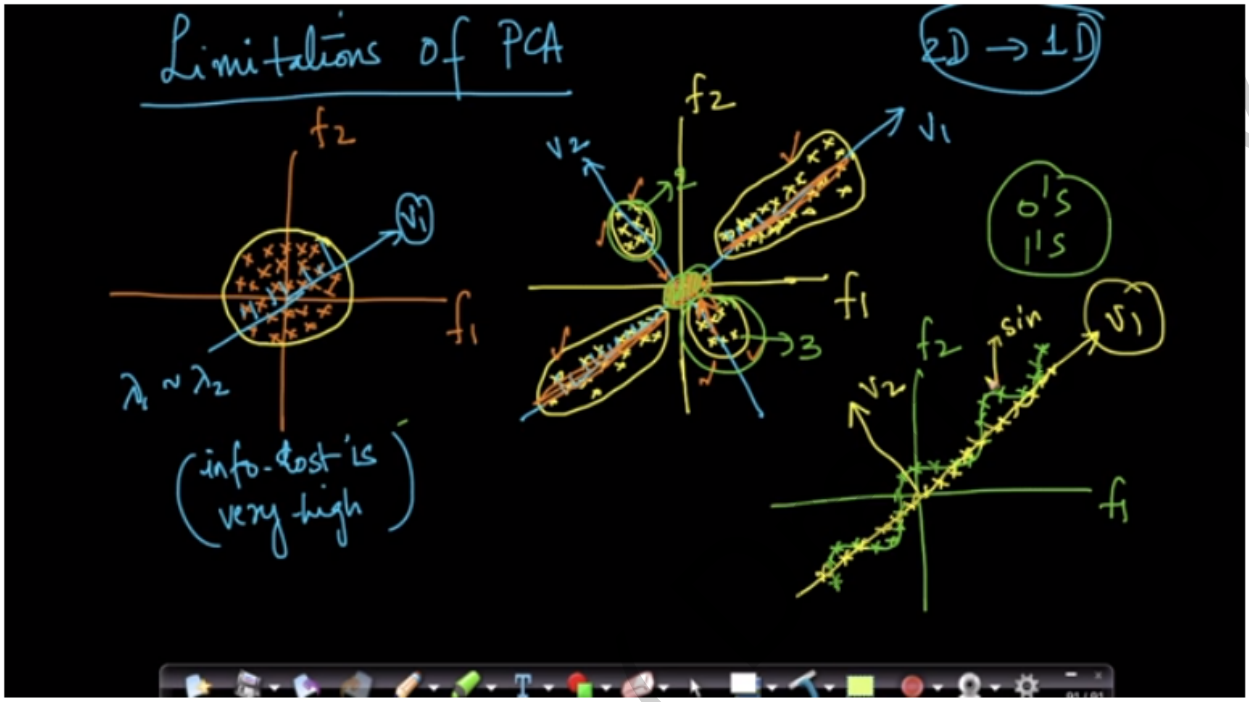


Minor Correction: @ 3:34min , it was mentioned as if V1>10 and V2<10, then yi = 1. But it is actually: if V1>10 and V2<10, then yi = 0

Timestamp: 4:44

When PCA is applied on MNIST data then we get the above visualization. From the above visualization we can see that for some of the classes like zeros and ones, we can easily separate them from the rest of the classes using simple rules such as v1 >10 and v2<10 then y=0,etc.

26.8 Limitations of PCA

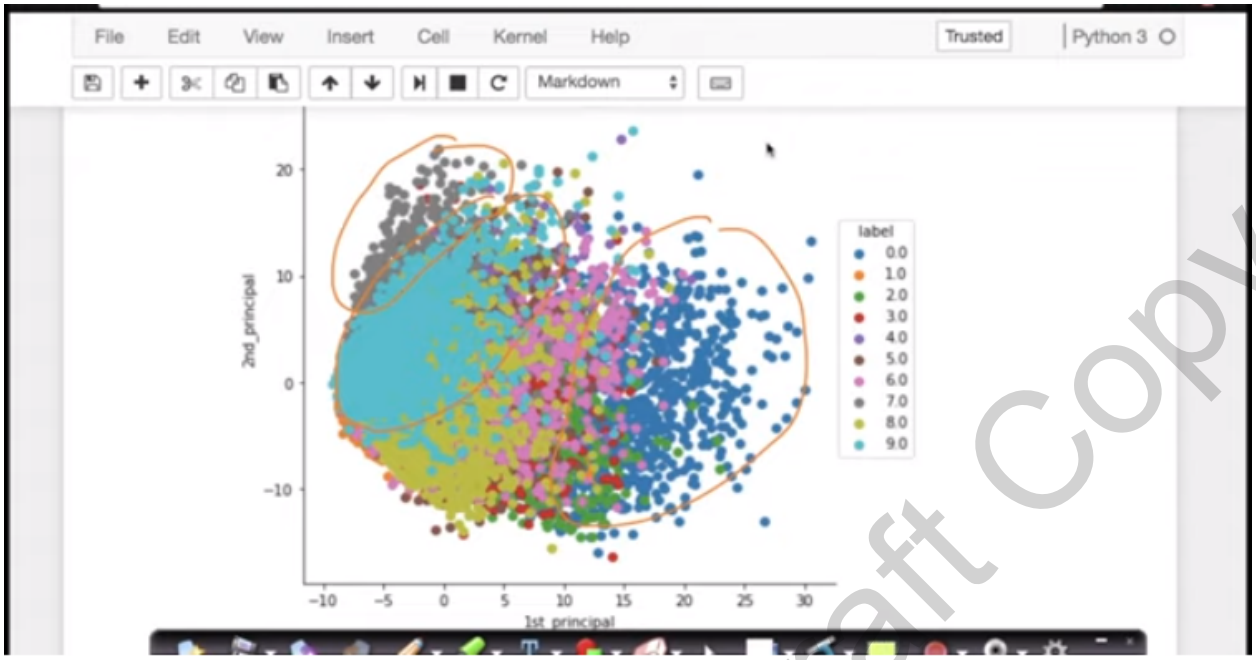


Timestamp: 4:47

There are certain limitations of PCA, as shown in the above images

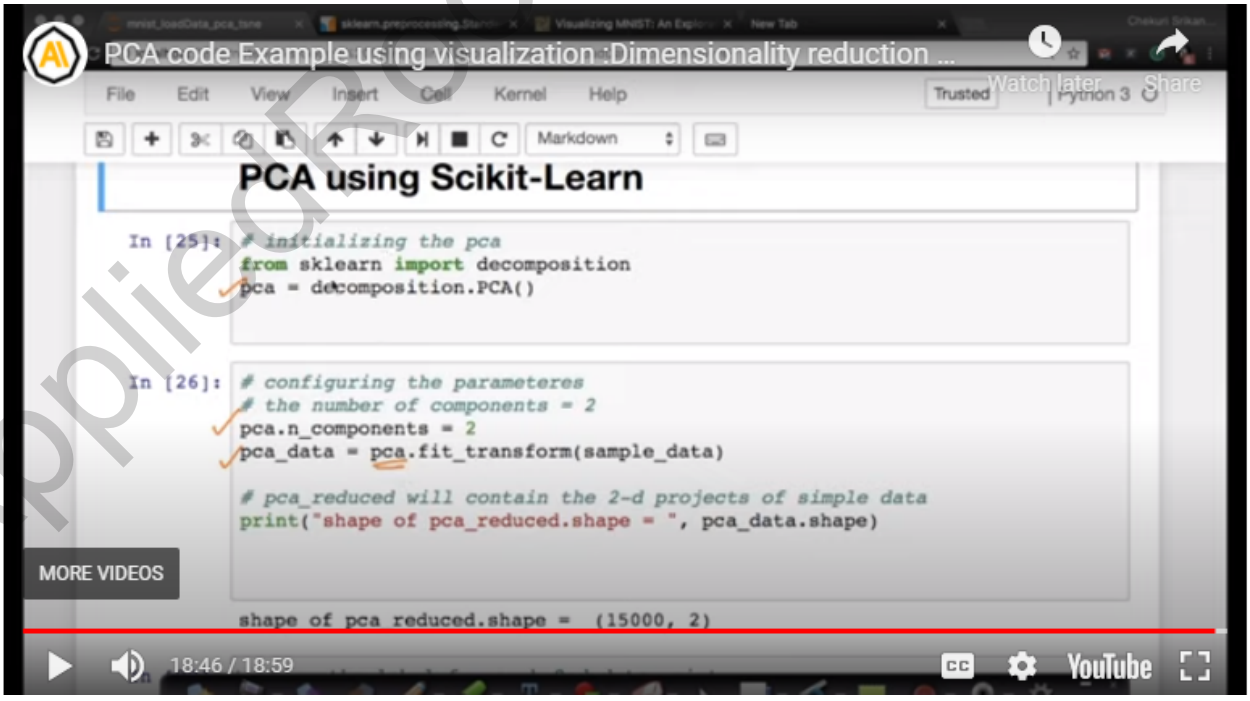
1) Suppose if the original data is like a circle, then the variance is spread among all the directions, but if we reduce the data to lower dimensions, we loose most of the useful information.
2) As shown in the second image, when initial the clusters of points are well separated in the original space but when the dimensionality of the data is reduced then points which are well separated might be projected together and hence difficult to separate.
3) If the original data has nice shapes like sin wave,etc when dimension is reduced we lose this information.

26.9 PCA code example



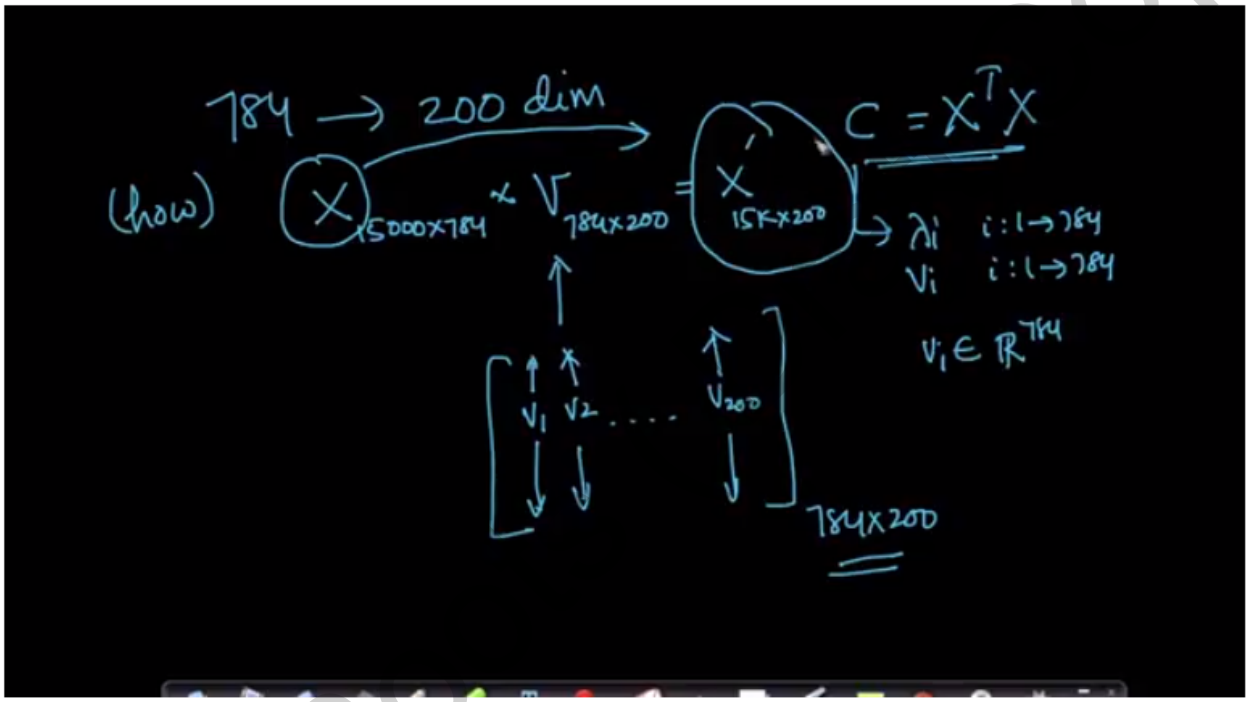
Timestamp: 18:15

In this video, we have implemented the code with both our custom code and with PCA provided by scikit-learn. Please refer to the notebook link provided in the video description for the complete code and explanation via comments. From the above plot we can see that some of the classes are well separated and others are not well separated when we visualize the data generated by PCA.



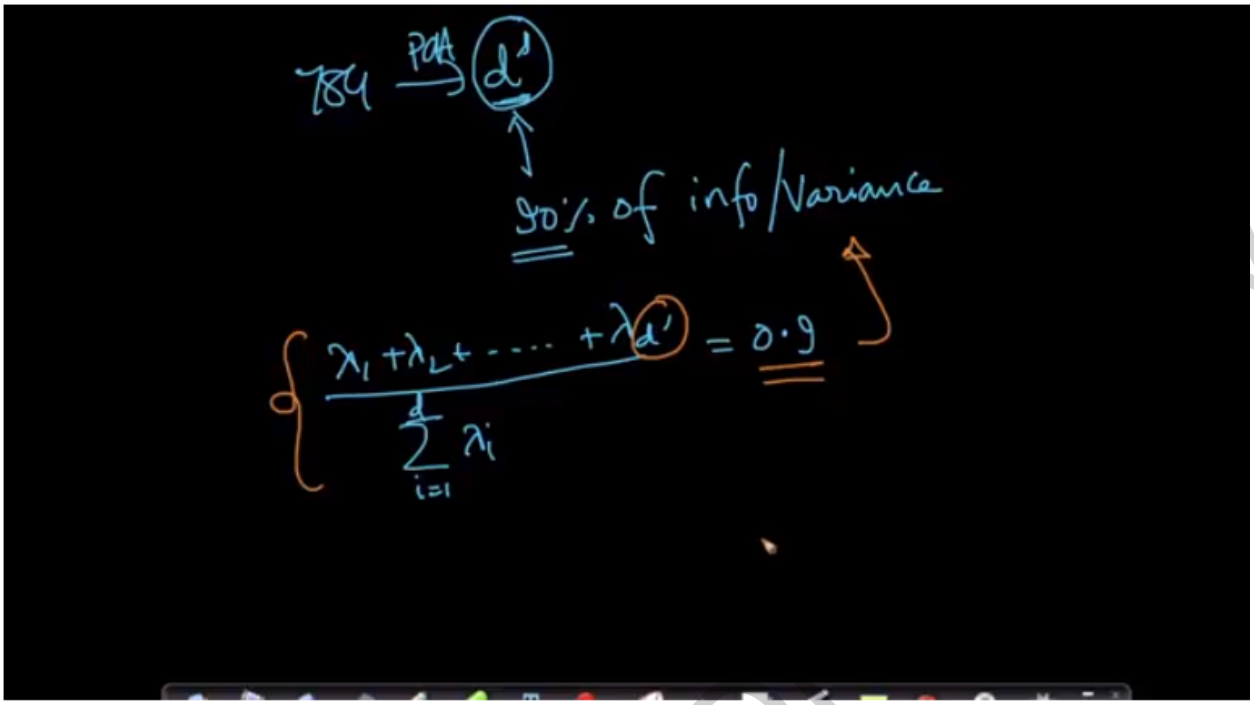
Timestamp: 18:46
The above is the simple code of scikit learn PCA, using which we can convert the sample data of MNIST of 784 dimensions to 2 dimensions.

26.10 PCA for dimensionality reduction (not-visualization)



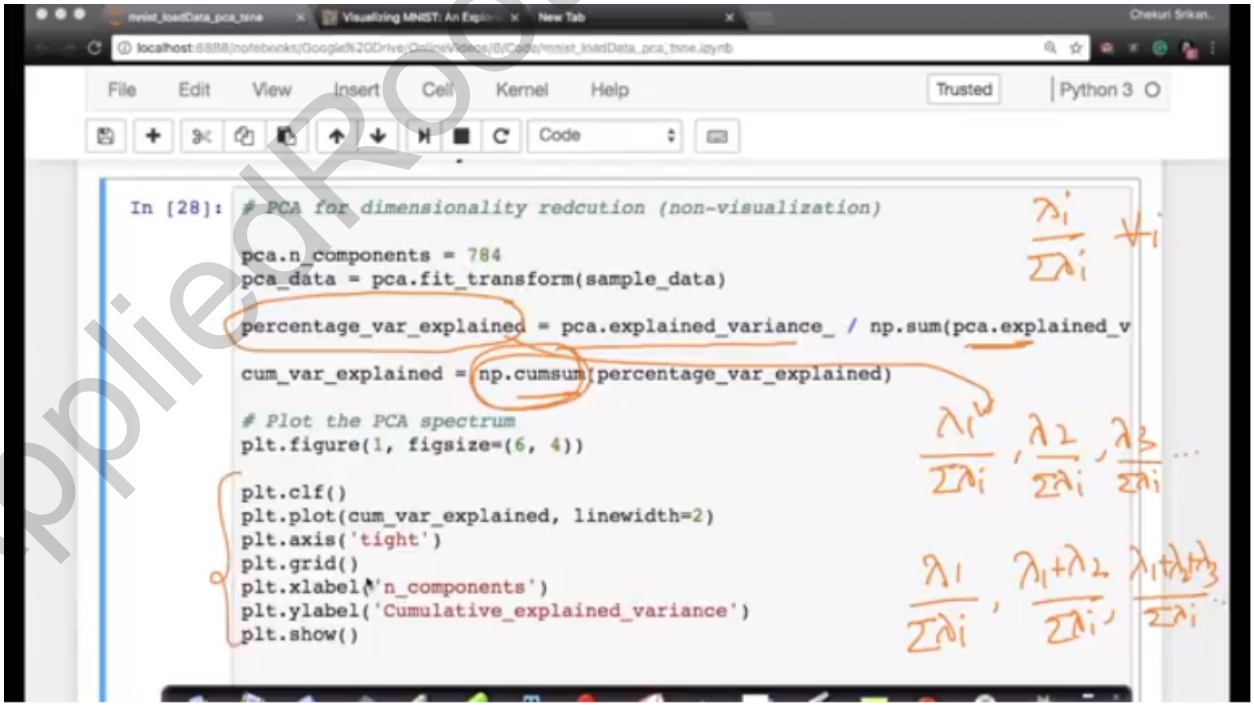
Timestamp: 4:21

Using PCA, to reduce the dimensions from 784 dimensions to 200 dimensions, we will just multiply the data X with the matrix of eigenvectors V to get the data of reduced dimension X'. Notice that our matrix of eigenvectors V is formed by placing the top 200 eigenvectors of the correlation matrix of X as a column vector.



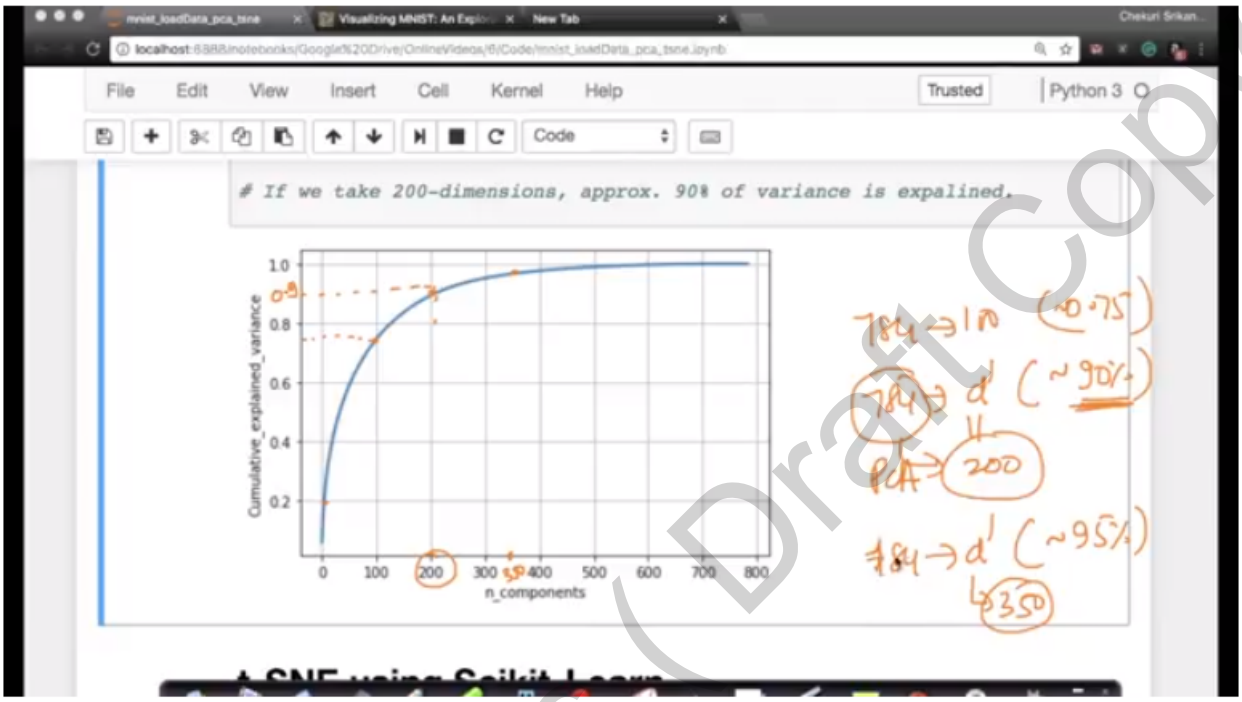
Timestamp: 9:46

As discussed before, people often doesn't predecide d' but rather decide on the amount of information/variance of the data they want to preserve and find such d' that preserves 90% of variance of our data and reduce our data from d to d' dimensions.



Timestamp: 12:31

The above is the code to plot the variance explained when we take different d', the dimensions to reduce the data into.



Timestamp: 14:36

From the above plot, we can see that about 75% of variance can be explained if we reduce the data to 100 dimensions, similarly around 90% of variance can be explained when we choose d' to be 200 dimensions. Hence if we decide that it is ok to preserve 90% variance of our original data, we can then choose d' to be 200 and can reduce the data to 200 dimensions using PCA.