Module 4

PCA in high dimension

In last session we've gone through the steps of the PCA algorithm.

In order to do PCA, we need to compute the data covariance matrix

In d dimension, the data co-variance matrix is a d×d matrix.

If d is very high, so in very high dimensions, then computing the eigenvalues and eigenvectors of this matrix can be quite expensive

It gets skewed cubically in number of rows and columns, in this case the number of dimensions

In this session we provide a solution to this problem, for the case that we have substantially fewer data point than dimensions

Assume we have data set, given as "X_1 ..... up to X_N in $\mathbb{R}^D$, and we assume that the data is centered, and has mean 0"

$$X_1, \ldots X_N \in \mathbb{R}^D$$

"Then the data covariance matrix, is given as S equals 1 of N, times X transpose times X"

$$S = \frac{1}{N} X^T X$$

"where we denote X to be the matrix that consist of X_1 transpose ..... to X_n transpose" which is in an N by D matrix"

$$X = \begin{bmatrix} X_1^T \\ \vdots \\ X_N^T \end{bmatrix} \in \mathbb{R}^{N \times D}$$

we now, assume that N is significantly smaller than D, which means that the number of data points is significantly smaller than the dimensionality of the data

"then the rank of the co-variance matrix is N"

$$S = \frac{1}{N} X^T X \Rightarrow \text{rank}(S) = N$$

that also means it has $D - N + 1$ many eigenvalue which are zero

that means that the matrix is not full rank, and rows and columns are linearly dependent.

i.e there are some redundancies.

In next few minutes, we'll exploit this and turn the D×D co-variance matrix S into a full rank N×N covariance matrix without eigenvalue 0.

$x_1, \ldots x_N \in \mathbb{R}^D$, $S = \frac{1}{N} X^T X$ (*)

just move the co-variance definition up here....

In PCA we ended up with the following eigen value / vector equation,

"we had S times b_i equals λ_i times b_i where b_i is a basis vector of the orthogonal complement of principal subspace"

$$S b_i = \lambda_i b_i$$

Lets rewrite the equation:
we now going to replace S with the definition up here (*)

So we'll get "1 over N times X transpose X, which is S, times b_i equals λ_i times b_i"

$$\underbrace{\frac{1}{N} X^T X}_{S} b_i = \lambda_i b_i$$

and now we multiply X from left hand side, so we will get:

" X time X transpose X b_i time 1 over N equals λ_i times X times b_i "

$$\frac{1}{N} X X^T \underbrace{X b_i}_{C_i} = \lambda_i \underbrace{X b_i}_{C_i}$$

Now we have a new eigen vector/value equation

So λ_i is still an eigen value, and now we have eigen vector $X b_i$, which we call " C_i " (see above) of the matrix $\frac{1}{N} X X^T$

This means that $\frac{1}{N} X X^T$ has the same nonzero eigenvalues values as the data covariance matrix,

But this is now an $N \times N$ matrix, so that we can compute the eigenvectors/values much quicker than for the original data covariance matrix.

$$\underbrace{\frac{1}{N} X X^T}_{\in \mathbb{R}^{N \times N}}$$

whereas S used to be a $D \times D$ matrix

So now we can compute the eigenvectors of this matrix $\frac{1}{N} X X^T$, and we use this to recover the original eigenvectors, ~~that~~ which we still need to do PCA

Currently we, we know the eigenvectors of $\frac{1}{N}XX^T$, 7
and we want to recover the eigenvectors of S.

If we left multiply our eigenvalue/vector equation, with X transpose we get the following:

"1 over N times X transpose times X transpose times c_i equals λ_i times X transpose times c_i"

$$\underbrace{\frac{1}{N}X^TX}_{S}c_i = \lambda_i X^T c_i$$

Now we find our S matrix again.

This is $S \Rightarrow \frac{1}{N}X^TX$, and this also means we recover X transpose times c_i as an eigenvector of S that belongs to the ~~same~~ eigenvalue λ_i

In this session we reformulated PCA, such that we can efficiently run PCA on datasets that the dimensionality of data is substantially larger than the number of datapoint