

# Principal Component Analysis II

CS771: Introduction to Machine Learning

Purushottam Kar

# Announcements

2

No class on Wednesday October 02 due to institute holiday

Several group mergers, migrations have taken place in the last week

*Nice that almost all assignment groups have at least 4 members*

*A few assignment groups dissolved since all members joined other groups*

*Will still allow mergers/migrations in case there is a pressing need, for example if a group member drops the course altogether*

*Group sizes should not exceed 6 members in any such mergers/migrations*

*Please focus on assignment 2 now (deadline October 26, 9:59PM IST)*



# Recap of Last Lecture

3

## Eigenpairs and their relation to singular triplets

*For any matrix  $X \in \mathbb{R}^{n \times d}$ , its right (resp. left) singular vectors are eigenvectors of  $X^T X$  (resp.  $XX^T$ ) and its non-zero singular values are square roots of the non-zero eigenvalues of either  $X^T X$  or  $XX^T$*

**Definiteness:** *a sq symm matrix is PSD iff all its eigenvalues are non-negative*  
*SVD always exists for every matrix, not true for eigendecomposition*

*Some matrices e.g. rotation matrices, may have no eigenvectors at all!*

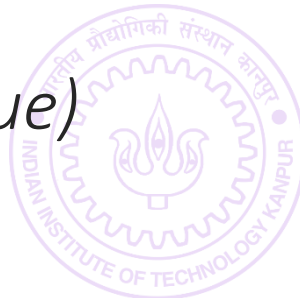
*Singular values are always non-negative, not true for eigenvalues*

**Principal Component Analysis:** process of extracting the leading singular triplets of  $X$  by extracting eigenpairs of  $X^T X$  or  $XX^T$

*Power method to extract the leading eigenpair (i.e. eigenvector + value)*

*Peeling method to extract the rest of the eigenpairs one by one*

*PCA gives us the best low-dimensional representation of our data*



# Applications of PCA – Space Savings

4

Given features  $X \in \mathbb{R}^{n \times d}$ , takes  $\mathcal{O}(nd)$  space to store, takes  $\mathcal{O}(nd)$  time to apply a linear model  $\mathbf{w} \in \mathbb{R}^d$  to all data points i.e. compute  $X\mathbf{w}$

*Perform PCA and approximate  $X$  using top  $k$  singular pairs*

*Find  $\hat{U} \in \mathbb{R}^{n \times k}$ ,  $\hat{V} \in \mathbb{R}^{d \times k}$ ,  $\hat{\Sigma} \in \mathbb{R}^{k \times k}$  using power + peeling method*

Approx  $X \approx \hat{X} = \hat{U}\hat{\Sigma}\hat{V}^\top$  as a rank  $k$  matrix. PCA gives the best such rank  $k$  approximation

*Takes  $\mathcal{O}((n + d) \cdot k) \ll nd$  space to store  $\hat{U}, \hat{V}, \hat{\Sigma}$*

**Warning:** the above benefit is lost if we compute and store  $\hat{X}$  (instead, store  $\hat{U}, \hat{V}, \hat{\Sigma}$ )

*Applying linear model  $\hat{X}\mathbf{w} = \left( \hat{U} \left( \hat{\Sigma}(\hat{V}^\top \mathbf{w}) \right) \right)$  takes  $\mathcal{O}((n + d) \cdot k) \ll nd$  time*

**Warning:** the above benefit is lost if we compute  $\hat{X}$  then compute  $\hat{X}\mathbf{w}$

How to choose  $k$ ? Various considerations that pose a tradeoff

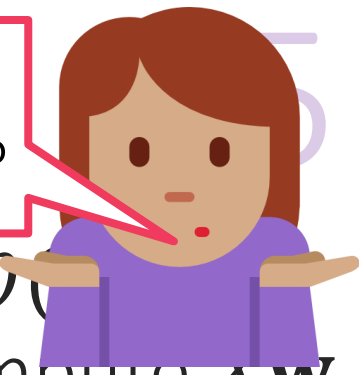
**Time/Space Budget:** choose  $k$  small enough s.t.  $\mathcal{O}((n + d) \cdot k)$  is manageable

**Error Tolerance:** choose  $k$  large enough s.t.  $\|X - \hat{X}\|_F$  is acceptable



# Applications of PCA – §

Is there a quick way to find whether  $\|X - \hat{X}\|_F$  is small or not?



Given features  $X \in \mathbb{R}^{n \times d}$ , takes  $\mathcal{O}(nd)$  space to store, takes  $\mathcal{O}(nd)$  time to apply a linear model  $\mathbf{w} \in \mathbb{R}^d$  to all data points i.e. compute  $X\mathbf{w}$

*Perform PCA and approximate  $X$  using top  $k$  singular pairs*

*Find  $\hat{U} \in \mathbb{R}^{n \times k}$ ,  $\hat{V} \in \mathbb{R}^{d \times k}$ ,  $\hat{\Sigma} \in \mathbb{R}^{k \times k}$  using power + peeling method*

Approx  $X \approx \hat{X} = \hat{U}\hat{\Sigma}\hat{V}^\top$  as a rank  $k$  matrix. PCA gives the best such rank  $k$  approximation

*Takes  $\mathcal{O}((n + d) \cdot k) \ll nd$  space to store  $\hat{U}, \hat{V}, \hat{\Sigma}$*

**Warning:** the above benefit is lost if we compute and store  $\hat{X}$  (instead, store  $\hat{U}, \hat{V}, \hat{\Sigma}$ )

*Applying linear model  $\hat{X}\mathbf{w} = \left( \hat{U} \left( \hat{\Sigma}(\hat{V}^\top \mathbf{w}) \right) \right)$  takes  $\mathcal{O}((n + d) \cdot k) \ll nd$  time*

**Warning:** the above benefit is lost if we compute  $\hat{X}$  then compute  $\hat{X}\mathbf{w}$

How to choose  $k$ ? Various considerations that pose a tradeoff

**Time/Space Budget:** choose  $k$  small enough s.t.  $\mathcal{O}((n + d) \cdot k)$  is manageable

**Error Tolerance:** choose  $k$  large enough s.t.  $\|X - \hat{X}\|_F$  is acceptable



# Finding the right value of $k$

6

For  $d < n$ , we have  $X = \sum_{j=1}^d \sigma_j \mathbf{u}^j (\mathbf{v}^j)^\top$  and  $\hat{X} = \sum_{j=1}^k \sigma_j \mathbf{u}^j (\mathbf{v}^j)^\top$

Thus, ignoring peeling errors, we get  $X - \hat{X} = \sum_{j=k+1}^d \sigma_j \mathbf{u}^j (\mathbf{v}^j)^\top$

**Fact:**  $\|A\|_F^2 = \text{tr}(A^\top A) = \text{tr}(AA^\top)$  where  $\text{tr}$  is the trace (sum of diagonal)

$$\|X - \hat{X}\|_F^2 = \text{tr} \left( \sum_{j=k+1}^d \sigma_j^2 \mathbf{v}^j (\mathbf{u}^j)^\top \mathbf{u}^j (\mathbf{v}^j)^\top + \sum_{j \neq l} \sigma_j \sigma_l \mathbf{v}^j (\mathbf{u}^j)^\top \mathbf{u}^l (\mathbf{v}^l)^\top \right)$$

Now  $\mathbf{u}^j$  are orthonormal and so are  $\mathbf{v}^j$  i.e.  $(\mathbf{u}^j)^\top \mathbf{u}^l = 0$  and  $(\mathbf{u}^j)^\top \mathbf{u}^j = 1$

$$\|X - \hat{X}\|_F^2 = \text{tr} \left( \sum_{j=k+1}^d \sigma_j^2 \mathbf{v}^j (\mathbf{v}^j)^\top \right) = \sum_{j=k+1}^d \sigma_j^2 \text{tr} \left( \mathbf{v}^j (\mathbf{v}^j)^\top \right) = \sum_{j=k+1}^d \sigma_j^2$$

Used linearity of trace and  $\text{tr} \left( \mathbf{v}^j (\mathbf{v}^j)^\top \right) = \text{tr} \left( (\mathbf{v}^j)^\top \mathbf{v}^j \right) = 1$  to show this

Similarly, can show that  $\|X\|_F^2 = \sum_{j=1}^d \sigma_j^2$

A prominent knee point, if one exists, gives us a good idea of the true intrinsic dimensionality of the data



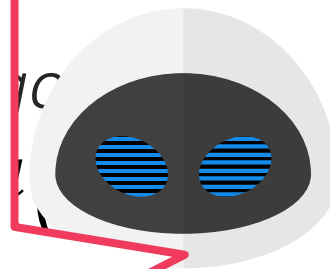
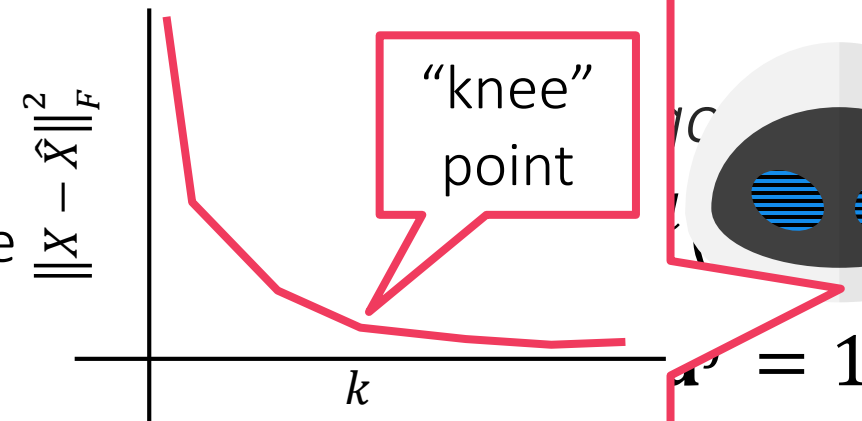
# Finding

For  $d < n$ ,

So, first find  $\|X\|_F^2 = \sum_{j=1}^d \sigma_j^2$ . Then, once you have a PCA for some value of  $k$ , compute the squared sum of singular values you have got i.e.  $\sum_{j=1}^k \sigma_j^2$ . Use this to get  $\sum_{j=k+1}^d \sigma_j^2 = \|X - \hat{X}\|_F^2$



Often, if you plot how the error  $\|X - \hat{X}\|_F^2$  goes down as  $k \uparrow$ , you will find that after some golden value of  $k$ , error drops much more slowly. This “knee” point is a good place to stop to get good bang-for-buck in terms of the accuracy-speed tradeoff



$$\|X - \hat{X}\|_F^2 = \text{tr} \left( \sum_{j=k+1}^d \sigma_j^2 \mathbf{v}^j (\mathbf{v}^j)^\top \right) = \sum_{j=k+1}^d \sigma_j^2 \text{tr} \left( \mathbf{v}^j (\mathbf{v}^j)^\top \right) = \sum_{j=k+1}^d \sigma_j^2$$

Used linearity of trace and  $\text{tr} \left( \mathbf{v}^j (\mathbf{v}^j)^\top \right) = \text{tr} \left( (\mathbf{v}^j)^\top \mathbf{v}^j \right) = 1$  to show this

Similarly, can show that  $\|X\|_F^2 = \sum_{j=1}^d \sigma_j^2$

*A prominent knee point, if one exists, gives us a good idea of the true intrinsic dimensionality of the data*



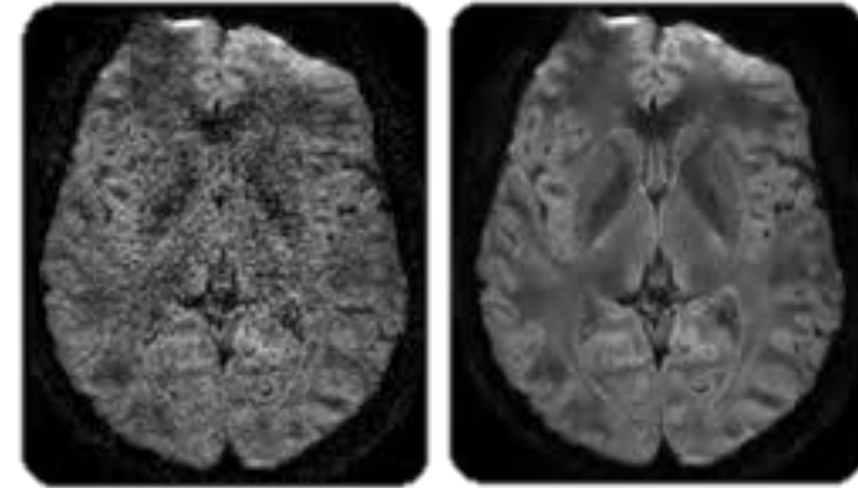


# Applications of PCA – Noise Removal

8

If data features are  $d$ -dim but data is really lying close to some  $k \ll d$  dim subspace then it may be noise that is making the data  $d$ -dim

PCA can help extract only the important (hidden)  $k$  features in the data which we can then use to classify/do other ML stuff



Given  $X \in \mathbb{R}^{n \times d}$ , compute PCA  $\hat{X} = \hat{U}\hat{\Sigma}\hat{V}^T$  and use  $\tilde{X} = \hat{U}\hat{\Sigma} \in \mathbb{R}^{n \times k}$  as a set of new  $k$ -dimensional features for the  $n$  data points

*Training algos would get sped up if used with  $k$ -dim features instead of  $d$ -dim*

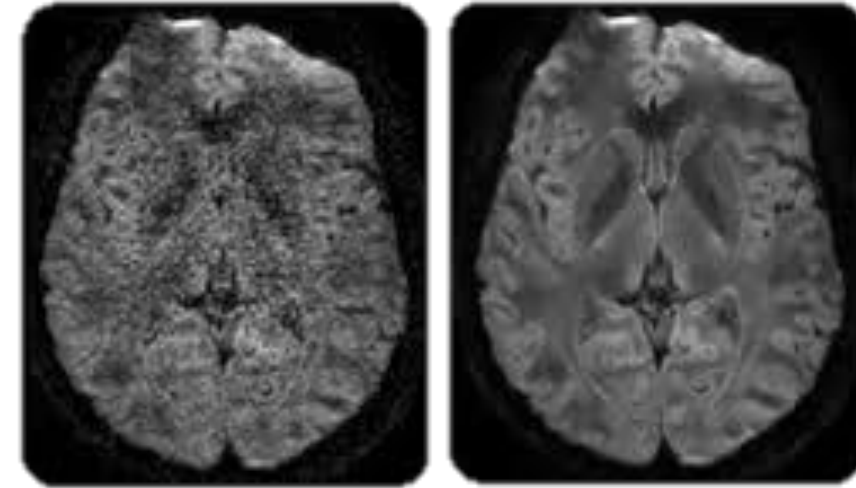
*Testing may not get sped up because for a given test point  $\mathbf{x}^t \in \mathbb{R}^d$ , we will first have to find out its  $k$ -dim representation – would need to compute  $\hat{V}^T \mathbf{x}^t$*

*Notice that we have  $X\hat{V} = U\Sigma V^T \hat{V} = \hat{U}\hat{\Sigma} = \tilde{X}$  (since  $V$  is orthonormal) so even for training features we can compute  $k$ -dim rep by just hitting with  $\hat{V}$*



If data features are  $d$ -dim but data is really lying close to some  $k \ll d$  dim subspace then it may be noise that is making the data  $d$ -dim

PCA can help extract only the important (hidden)  $k$  features in the data which we can then use to classify/do other ML stuff



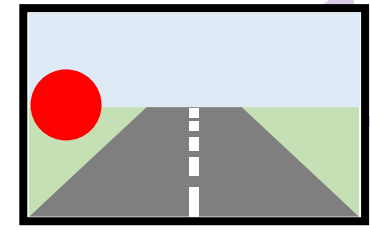
Given  $X \in \mathbb{R}^{n \times d}$ , compute PCA  $\hat{X} = \hat{U}\hat{\Sigma}\hat{V}^T$  and use  $\tilde{X} = \hat{U}\hat{\Sigma} \in \mathbb{R}^{n \times k}$  as a set of new  $k$ -dimensional features for the  $n$  data points

*Training algos would get sped up if used with  $k$ -dim features instead of  $d$ -dim*

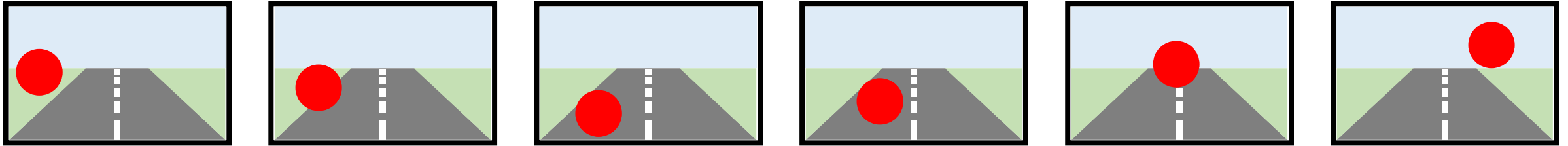
*Testing may not get sped up because for a given test point  $\mathbf{x}^t \in \mathbb{R}^d$ , we will first have to find out its  $k$ -dim representation – would need to compute  $\hat{V}^T \mathbf{x}^t$*

*Notice that we have  $X\hat{V} = U\Sigma V^T \hat{V} = \hat{U}\hat{\Sigma} = \tilde{X}$  (since  $V$  is orthonormal) so even for training features we can compute  $k$ -dim rep by just hitting with  $\hat{V}$*

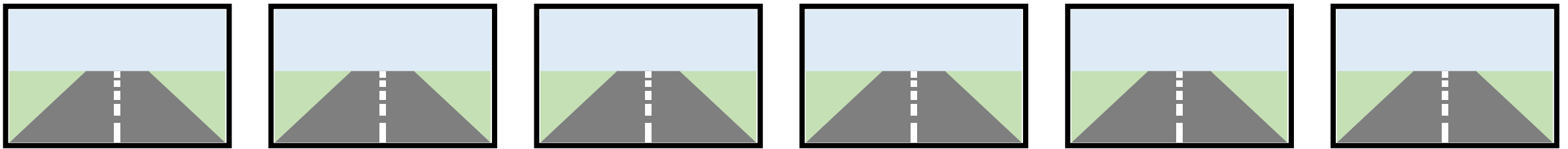
# Foreground Background Separation



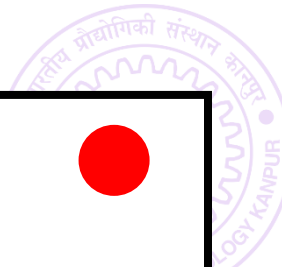
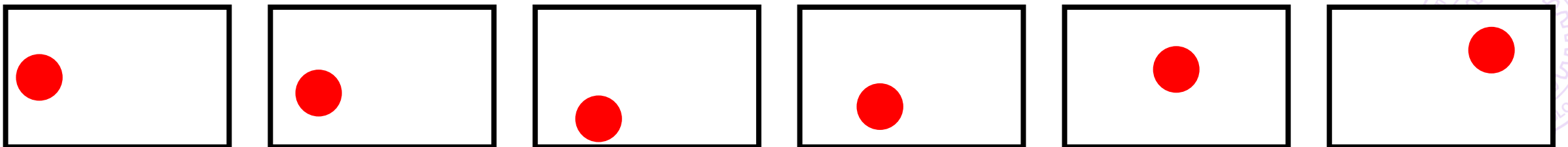
10



=



+



# Denoising, Foreground Extraction

11

Make every frame a vector  $\mathbf{x}^i \in \mathbb{R}^d$  where  $d$  is number of pixels

*Thus, we have  $n$  frames represented as a matrix  $X = [\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^n]^\top \in \mathbb{R}^{n \times d}$*

*Background is a constant vector  $\mathbf{b} \in \mathbb{R}^d$ . Let  $\mathbf{1} = [1, 1, 1, \dots, 1]^\top \in \mathbb{R}^n$*

*Foreground treated as noise  $\mathbf{f}^i \in \mathbb{R}^d$ . Let  $F = [\mathbf{f}^1, \mathbf{f}^2, \dots, \mathbf{f}^n]^\top \in \mathbb{R}^{n \times d}$*

This gives us  $X = \mathbf{1} \cdot \mathbf{b}^\top + F$  i.e. if noise is not too much then  $X$  is approximately rank 1 😊

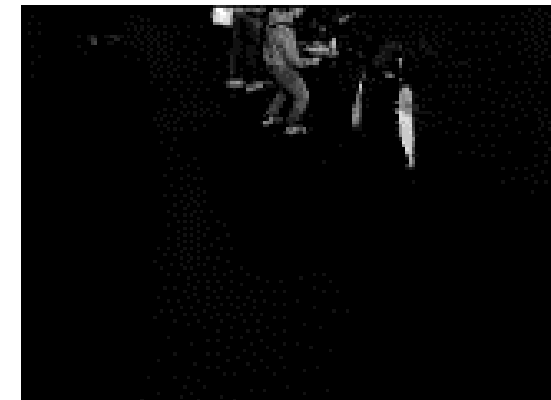
*We can do PCA and recover  $F$  as noise*



=



+



**Note:** here we are treating images as vectors and a group of images (say a video) is treated as a matrix – this is different from the noise removal example where every image was treated as a matrix itself

Make every frame a vector  $\mathbf{x}^i \in \mathbb{R}^d$  where  $d$  is number of pixels

Thus, we have  $n$  frames represented as a matrix  $X = [\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^n]^\top \in \mathbb{R}^{n \times d}$

Background is a constant vector  $\mathbf{b} \in \mathbb{R}^d$ . Let  $\mathbf{1} = [1, 1, 1, \dots, 1]^\top \in \mathbb{R}^n$

Foreground treated as noise  $\mathbf{f}^i \in \mathbb{R}^d$ . Let  $F = [\mathbf{f}^1, \mathbf{f}^2, \dots, \mathbf{f}^n]^\top \in \mathbb{R}^{n \times d}$

This gives us  $X = \mathbf{1} \cdot \mathbf{b}^\top + F$  i.e. if noise is not too much then  $X$  is approximately rank 1 😊

We can do PCA and recover  $F$  as noise



=



+



# Applications of PCA – Learning Prototypes<sup>13</sup>

Given  $X \in \mathbb{R}^{n \times d}$ , compute PCA  $\hat{X} = \hat{U}\hat{\Sigma}\hat{V}^\top$

*Notice that  $\hat{V} \in \mathbb{R}^{k \times d}$  i.e. we can think of  $V$  as a dataset of  $k$  prototypes*

*All points in dataset  $X$  can be approximated well as a linear combination of these  $k$  prototypes – the linear combinations are given by  $\hat{U}\hat{\Sigma}$*

*Specifically, the  $i$ -th data point (the  $i$ -th row of  $X$ ) can be approximated as*

$$\mathbf{x}^i \approx \sum_{j=1}^k \hat{U}_{ij} \sigma_j \cdot \mathbf{v}^j$$

Thus, PCA gives us a new way to get good prototypes to explain data

GMMs earlier had given us one way to get good prototypes

*However, GMM did not assure us that data features could be approximated as linear combination of the prototypes it learnt. PCA does give us that assurance*





# Eigenfaces [Sirovich and Kirby, Turk and Pentland]

14

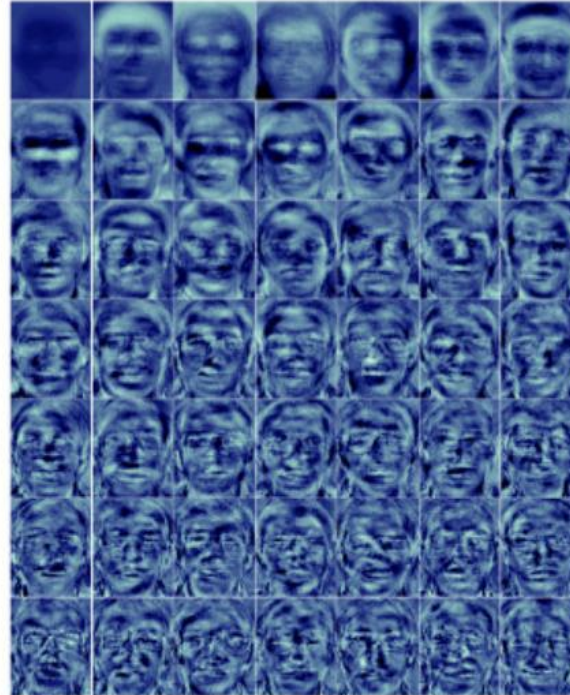
An iconic application of PCA – given face images, the prototypes given by the leading few right singular vectors are called “eigenfaces”

*Images are treated as vectors for this (and many other) application*

Original Collection of Images



K=49 Eigenvectors  
("eigenfaces") learned  
by PCA on this data



Each image's reconstructed version





E **Note:** here again, we are treating images as vectors and a group of images is treated as a matrix [and Kirby, Turk and Pentland]

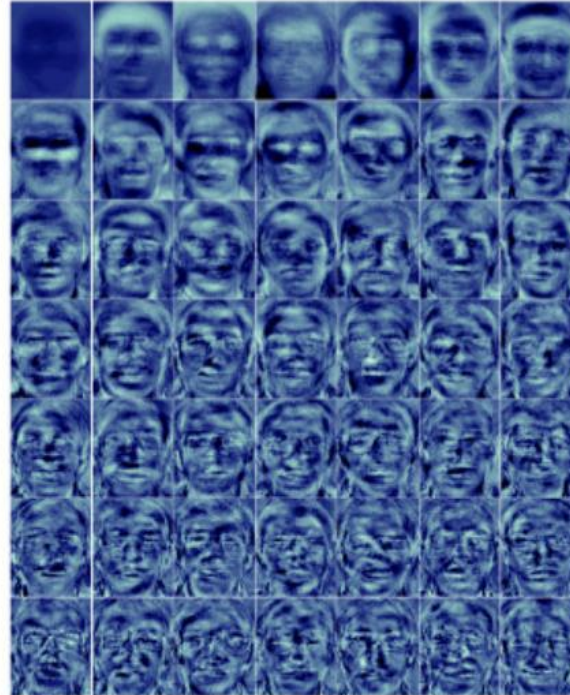
An iconic application of PCA – given face images, the prototypes given by the leading few right singular vectors are called “eigenfaces”

*Images are treated as vectors for this (and many other) application*

Original Collection of Images



K=49 Eigenvectors  
("eigenfaces") learned  
by PCA on this data



Each image's reconstructed version





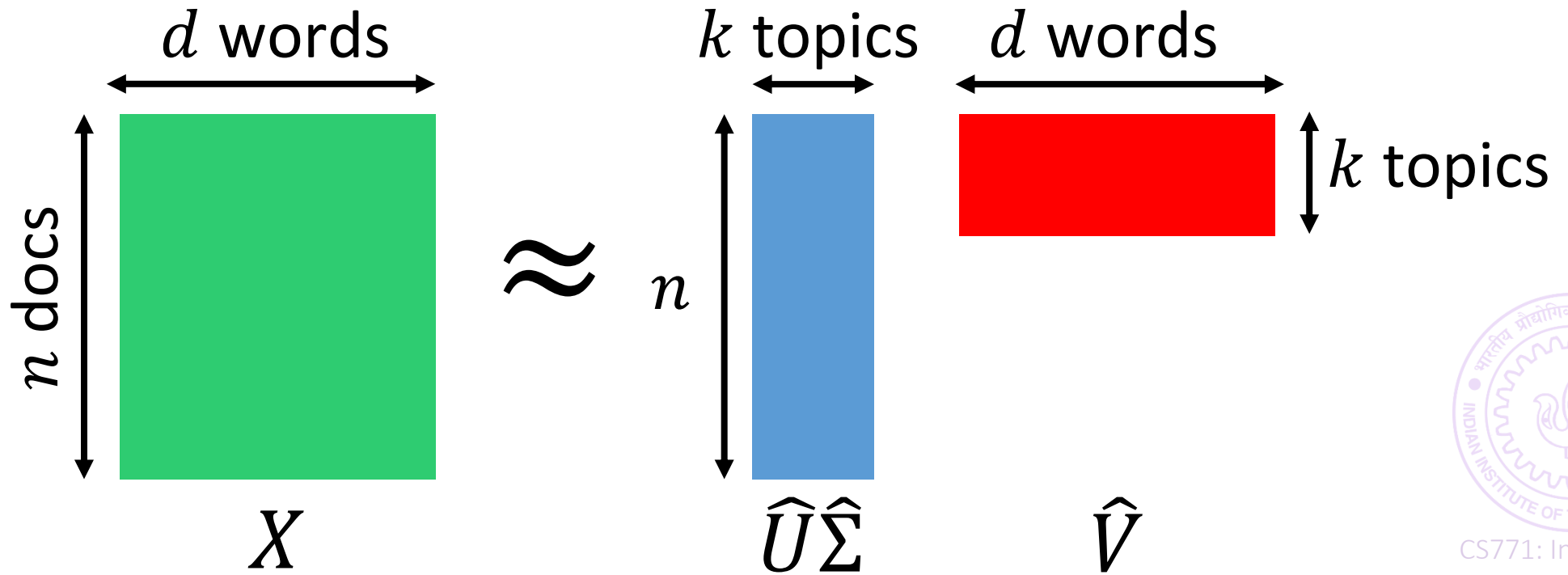
# Latent Semantic Analysis (LSA/LSI)

16

Used to be a very popular course project topic 😊

Given  $n$  documents, each as bag-of-words representation with dictionary size  $d$  as very large  $d \approx 10^6$ , discover  $k \ll d$  “topics” about which the documents are talking

*Topics could be sports, education, politics, science, entertainment*



# Latent

Used to

Given  $n$

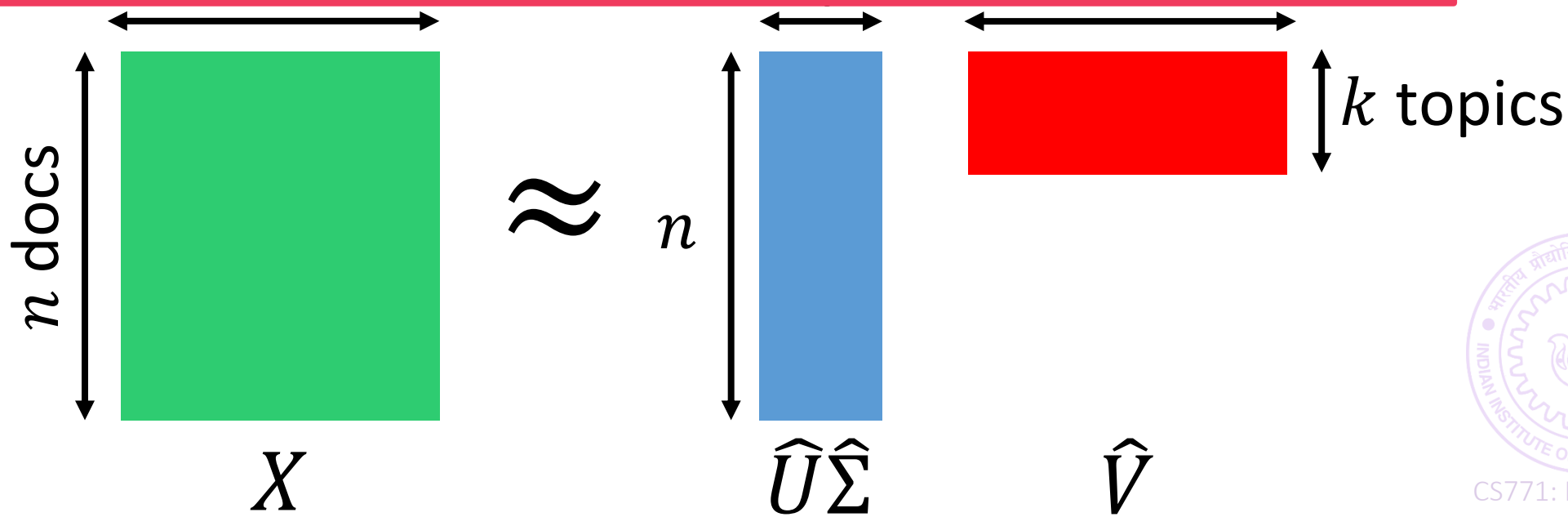
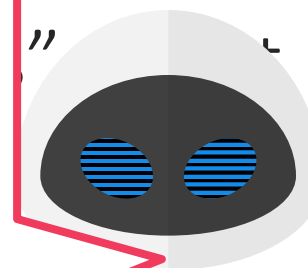
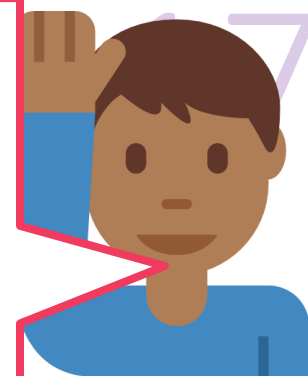
dictiona

which th

Topics

Each topic here is represented by a prototypical document for that topic. All documents are linear representations of the topics. The amount of weight a document places on a certain topic in its representation tells us a lot about what is that document talking about e.g. if  $|U_{ij}| \gg 1$  then the  $j$ -th topic is really core to the  $i$ -th document

**Word of caution:** PCA itself will not tell you whether blah prototype is about sports or bleh prototype is about politics. You have to take a look at the prototype vector, also look at documents that place lots of weight on that prototype and make these deductions separately



# Recommendation Systems

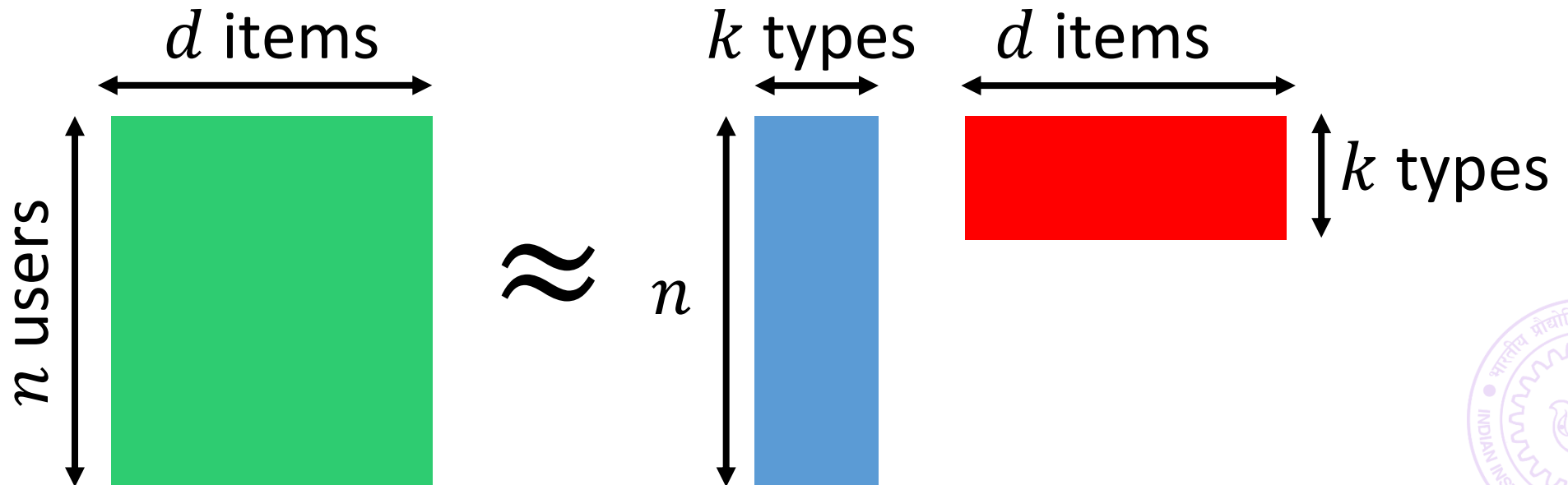
18

A popular technique in RecSys is **Collaborative Filtering**

*Have data for  $n$  users and their interactions with  $d$  items*

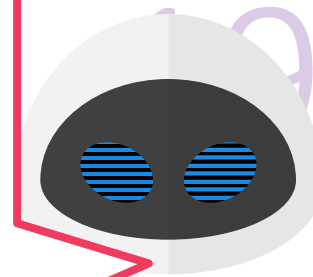
*For these  $n$  users, predict other items that they would also like*

*Done by discovering  $k$  “user types” and representing each user as a combination of these user types*



# Recomm

**Word of caution:** This is a very different problem setting than the one in assignment 2. Assignment 2, where users have features, is called *content-based filtering*. In the setting in this slide, users have no features. A drawback of collaborative filtering is that it gets more difficult to add new users to the system

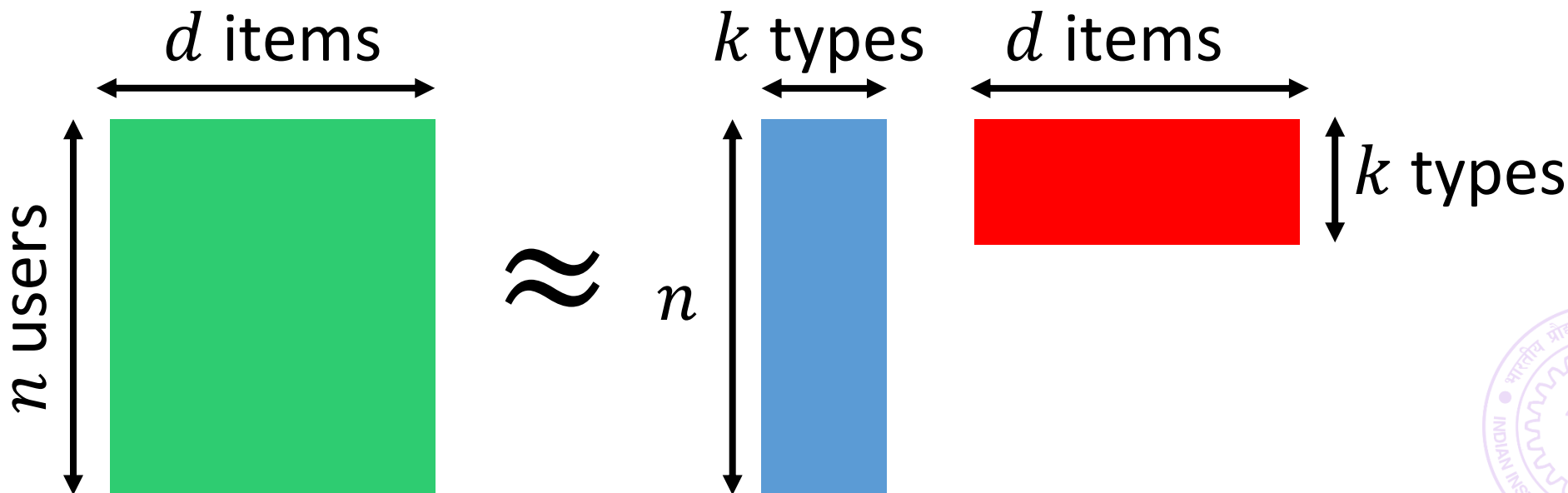


A popular technique

Have data for

*For these  $n$  users, predict other items that they would also like*

*Done by discovering  $k$  “user types” and representing each user as a combination of these user types*



# The Many Faces of PCA

20

Has been “discovered” several times  
[Pearson, 1901; Hotelling, 1930]

*Gives us best possible (in terms of Frob.  
norm error) low-rank approx of our data*

*Can be thought of as giving low-dim reps  
of our feature vectors so that pairwise L2  
distances among them is preserved – see  
multidimensional scaling (MDS)*

*Also, gives us new basis with smaller dim  
( $\hat{V}$  is orthonormal) s.t. in that basis, data  
can be reconstructed with little error*

*Can also be thought of as giving us the  
directions along which the data has  
maximum variance*



# The Many Faces of PCA

20

Has been “discovered” several times  
[Pearson, 1901; Hotelling, 1930]

*Gives us best possible (in terms of Frob.  
norm error) low-rank approx of our data*

*Can be thought of as giving low-dim reps  
of our feature vectors so that pairwise L2  
distances among them is preserved – see  
multidimensional scaling (MDS)*

*Also, gives us new basis with smaller dim  
( $\hat{V}$  is orthonormal) s.t. in that basis, data  
can be reconstructed with little error*

*Can also be thought of as giving us the  
directions along which the data has  
maximum variance*



# The Many Faces of PCA

20

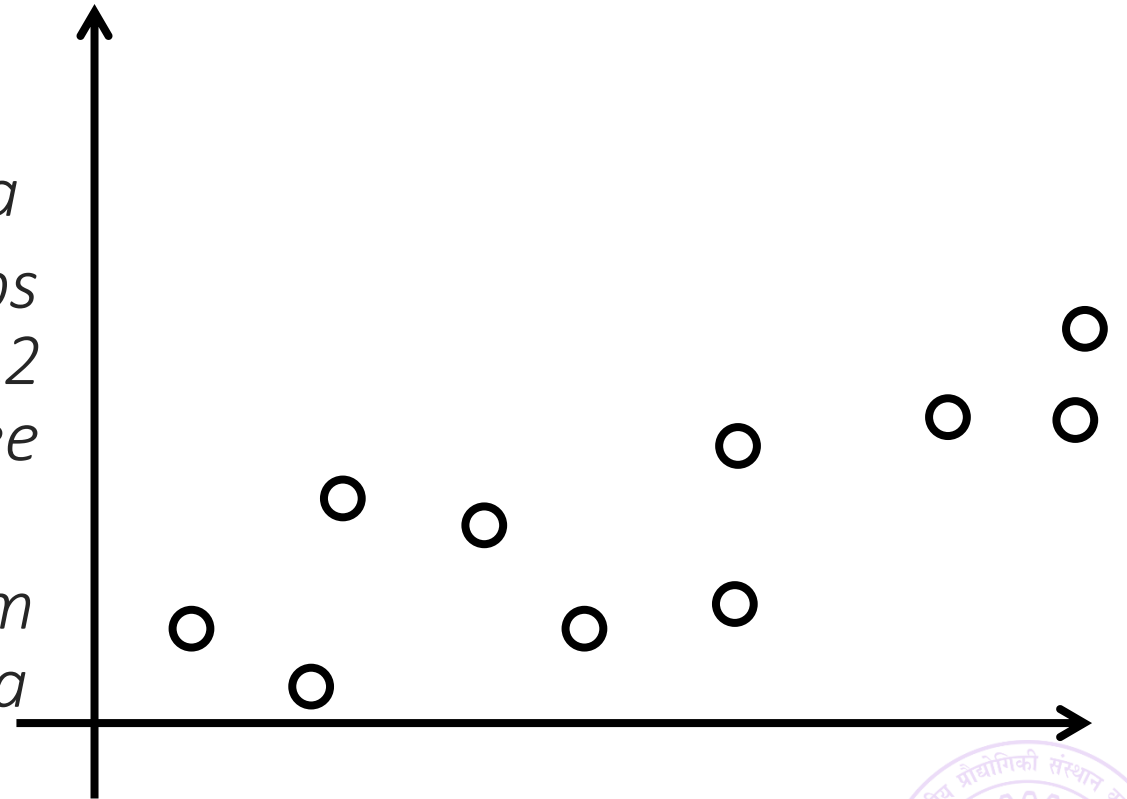
Has been “discovered” several times  
[Pearson, 1901; Hotelling, 1930]

*Gives us best possible (in terms of Frob. norm error) low-rank approx of our data*

*Can be thought of as giving low-dim reps of our feature vectors so that pairwise L2 distances among them is preserved – see multidimensional scaling (MDS)*

*Also, gives us new basis with smaller dim ( $\hat{V}$  is orthonormal) s.t. in that basis, data can be reconstructed with little error*

*Can also be thought of as giving us the directions along which the data has maximum variance*





# The Many Faces of PCA

20

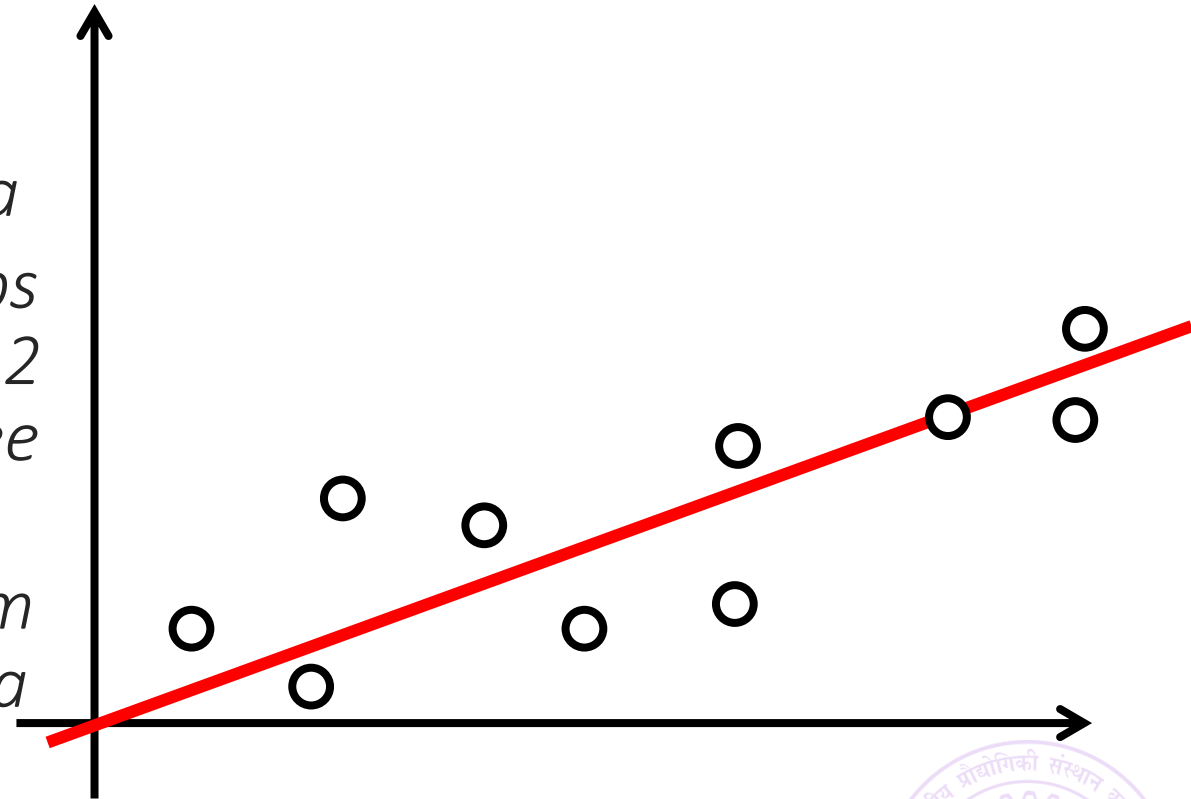
Has been “discovered” several times  
[Pearson, 1901; Hotelling, 1930]

*Gives us best possible (in terms of Frob. norm error) low-rank approx of our data*

*Can be thought of as giving low-dim reps of our feature vectors so that pairwise L2 distances among them is preserved – see multidimensional scaling (MDS)*

*Also, gives us new basis with smaller dim ( $\hat{V}$  is orthonormal) s.t. in that basis, data can be reconstructed with little error*

*Can also be thought of as giving us the directions along which the data has maximum variance*



# The Many Faces of PCA

20

Has been “discovered” several times  
[Pearson, 1901; Hotelling, 1930]

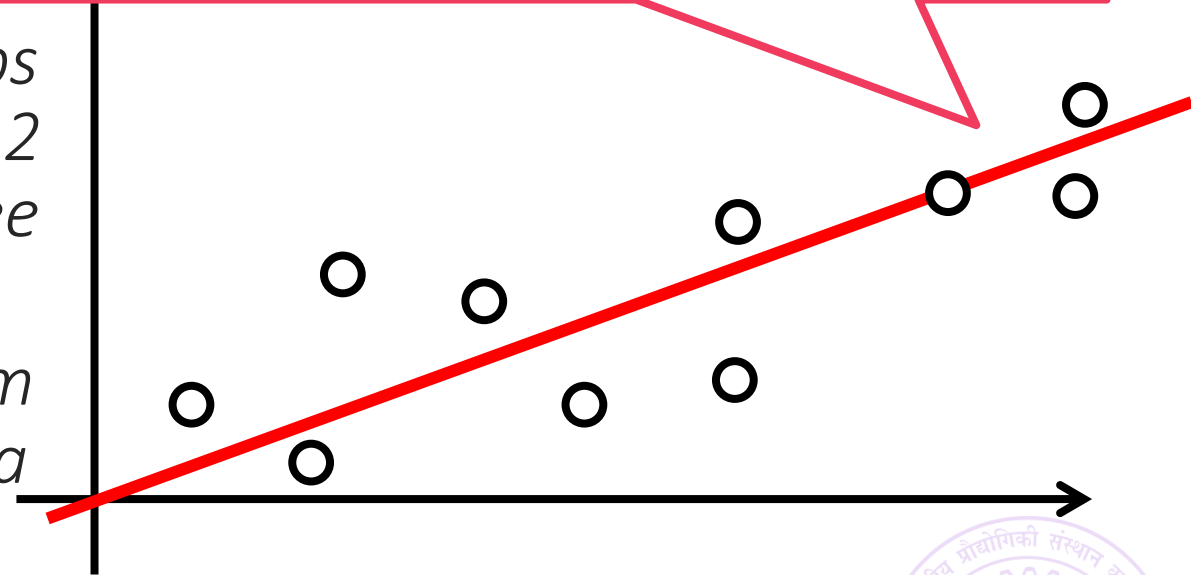
*Gives us best possible (in terms of Frobenius norm error) low-rank approx of our data*

*Can be thought of as giving low-dim reps of our feature vectors so that pairwise L2 distances among them is preserved – see multidimensional scaling (MDS)*

*Also, gives us new basis with smaller dim ( $\hat{V}$  is orthonormal) s.t. in that basis, data can be reconstructed with little error*

*Can also be thought of as giving us the directions along which the data has maximum variance*

This line captures most of the information in the data Why not get rid of the other information? Save space, reduce noise!



# The Many Faces of PCA

20

Has been “discovered” several times  
[Pearson, 1901; Hotelling, 1930]

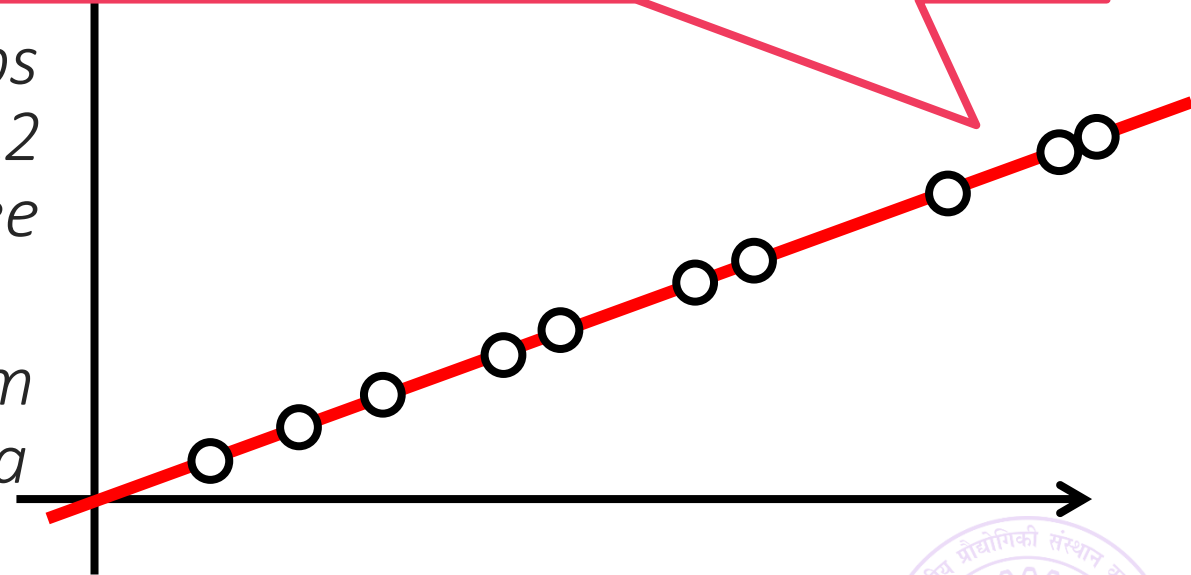
*Gives us best possible (in terms of Frobenius norm error) low-rank approx of our data*

*Can be thought of as giving low-dim reps of our feature vectors so that pairwise L2 distances among them is preserved – see multidimensional scaling (MDS)*

*Also, gives us new basis with smaller dim ( $\hat{V}$  is orthonormal) s.t. in that basis, data can be reconstructed with little error*

*Can also be thought of as giving us the directions along which the data has maximum variance*

This line captures most of the information in the data Why not get rid of the other information? Save space, reduce noise!



# PCA Minimizes Reconstruction Error

26

Already seen that for any matrix  $X \in \mathbb{R}^{n \times d}$ , its best rank  $k$  approx. is obtained by  $\hat{X} = \hat{U}\hat{\Sigma}\hat{V}^T$  by taking leading  $k$  singular triplets

**Proof for  $k = 1$ :** want best rank 1 approx. for  $X$  i.e. want to fit all data points on a 1D subspace i.e. find a unit vector  $\mathbf{w} \in \mathbb{R}^d$  s.t. data is best represented along subspace spanned by  $\mathbf{w}$  than any other vector  $\mathbf{v}$

**Claim:** any vector  $\mathbf{x} \in \mathbb{R}^d$  is best represented in  $\text{span}(\mathbf{w})$  as  $\mathbf{w}^T \mathbf{x} \cdot \mathbf{w}$

*Proof:* we want  $\arg \min_{t \in \mathbb{R}} \|\mathbf{x} - t \cdot \mathbf{w}\|_2^2$ . Apply first order optimality now

Thus, reconstruction error for entire dataset is  $\sum_{i=1}^n \|\mathbf{x}^i - \mathbf{w}^T \mathbf{x}^i \cdot \mathbf{w}\|_2^2$

$$\operatorname{argmin}_{\|\mathbf{w}\|_2=1} \sum_{i=1}^n \|\mathbf{x}^i - \mathbf{w} \mathbf{w}^T \mathbf{x}^i\|_2^2 = \operatorname{argmin}_{\|\mathbf{w}\|_2=1} \sum_{i=1}^n -(\mathbf{w}^T \mathbf{x}^i)^2$$

$$\operatorname{argmax}_{\|\mathbf{w}\|_2=1} \mathbf{w}^T \left( \sum_{i=1}^n \mathbf{x}^i (\mathbf{x}^i)^T \right) \mathbf{w} = \operatorname{argmax}_{\|\mathbf{w}\|_2=1} \mathbf{w}^T (X^T X) \mathbf{w}$$

*This matches exactly the definition of the leading eigenvector of  $X^T X$*



# PCA Minimize

We can show that for  $k > 1$  as well, PCA offers the best reconstruction error. In that case, instead of optimizing over a unit vector  $\mathbf{w} \in \mathbb{R}^d$ , we would have to optimize over an orthonormal matrix  $V \in \mathbb{R}^{d \times k}$  by taking leading  $k$  singular triplets



Already seen that for  $k=1$ , the best approximation is obtained by  $\hat{X} = \hat{U}\hat{\Sigma}\hat{V}^T$  by taking leading  $k$  singular triplets

**Proof for  $k = 1$ :** want best rank 1 approx. for  $X$  i.e. want to fit all data points on a 1D subspace i.e. find a unit vector  $\mathbf{w} \in \mathbb{R}^d$  s.t. data is best represented along subspace spanned by  $\mathbf{w}$  than any other vector  $\mathbf{v}$

**Claim:** any vector  $\mathbf{x} \in \mathbb{R}^d$  is best represented in  $\text{span}(\mathbf{w})$  as  $\mathbf{w}^T \mathbf{x} \cdot \mathbf{w}$

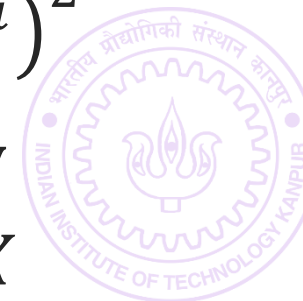
*Proof:* we want  $\arg \min_{t \in \mathbb{R}} \|\mathbf{x} - t \cdot \mathbf{w}\|_2^2$ . Apply first order optimality now

Thus, reconstruction error for entire dataset is  $\sum_{i=1}^n \|\mathbf{x}^i - \mathbf{w}^T \mathbf{x}^i \cdot \mathbf{w}\|_2^2$

$$\operatorname{argmin}_{\|\mathbf{w}\|_2=1} \sum_{i=1}^n \|\mathbf{x}^i - \mathbf{w} \mathbf{w}^T \mathbf{x}^i\|_2^2 = \operatorname{argmin}_{\|\mathbf{w}\|_2=1} \sum_{i=1}^n -(\mathbf{w}^T \mathbf{x}^i)^2$$

$$\operatorname{argmax}_{\|\mathbf{w}\|_2=1} \mathbf{w}^T \left( \sum_{i=1}^n \mathbf{x}^i (\mathbf{x}^i)^T \right) \mathbf{w} = \operatorname{argmax}_{\|\mathbf{w}\|_2=1} \mathbf{w}^T (X^T X) \mathbf{w}$$

*This matches exactly the definition of the leading eigenvector of  $X^T X$*



# PCA Preserves Maximum Data Variance

28

Data may take similar values along one direction, varied values along another

**Directional variance:** *given a unit vector  $\mathbf{u} \in \mathbb{R}^d$ , directional var. along  $\mathbf{u}$  is defined as variance of the r.v.  $z = \mathbf{u}^\top \mathbf{x}$  where we choose each data point  $\mathbf{x}^i$  w.p.  $1/n$*

Assume data is centered i.e.  $\sum_{i=1}^n \mathbf{x}^i = \mathbf{0}$

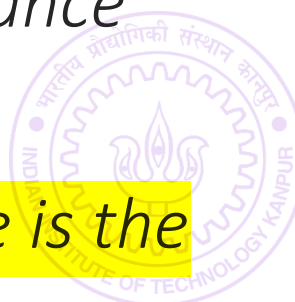
Thus,  $\mathbb{E}z = 0$  i.e.  $\mathbb{V}z = \mathbb{E}[z^2]$

Directions with more directional variance preserve more info

*PCA does give us orthonormal directions with largest directional variance*

$$\mathbb{E}[z^2] = \frac{1}{n} \sum_{i=1}^n (\mathbf{u}^\top \mathbf{x}^i)^2 = \frac{1}{n} \mathbf{u}^\top (X^\top X) \mathbf{u}$$

*Thus, finding the direction offering the maximum directional variance is the same as finding the leading right singular vector*



# PCA Preserves Maximum Data Variance

28

Data may take similar values along one direction, varied values along another

**Directional variance:** given a unit vector  $\mathbf{u} \in \mathbb{R}^d$ , directional var. along  $\mathbf{u}$  is defined as variance of the r.v.  $z = \mathbf{u}^\top \mathbf{x}$  where we choose each data point  $\mathbf{x}^i$  w.p.  $1/n$

Assume data is centered i.e.  $\sum_{i=1}^n \mathbf{x}^i = \mathbf{0}$

Thus,  $\mathbb{E}z = 0$  i.e.  $\mathbb{V}z = \mathbb{E}[z^2]$

Directions with more directional variance preserve more info

PCA does give us orthonormal directions with largest directional variance

$$\mathbb{E}[z^2] = \frac{1}{n} \sum_{i=1}^n (\mathbf{u}^\top \mathbf{x}^i)^2 = \frac{1}{n} \mathbf{u}^\top (X^\top X) \mathbf{u}$$

Thus, finding the direction offering the maximum directional variance is the same as finding the leading right singular vector





# PCA Preserves Maximum Data Variance

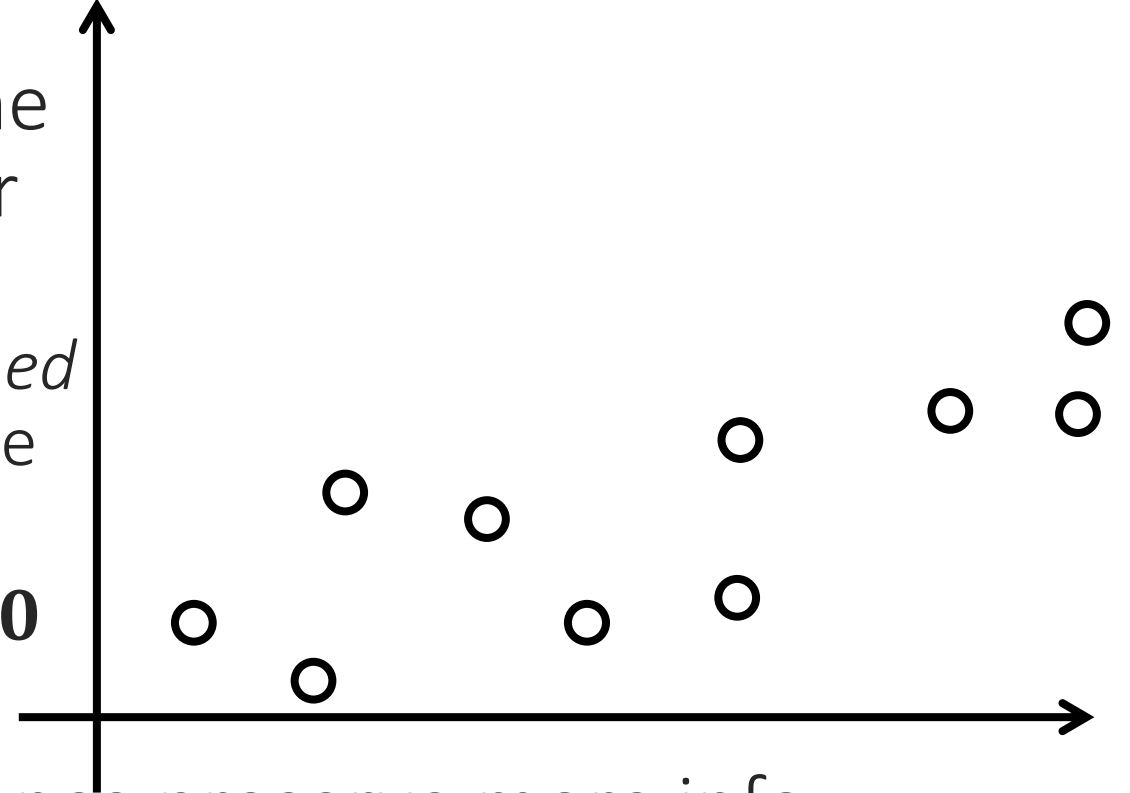
28

Data may take similar values along one direction, varied values along another

**Directional variance:** given a unit vector  $\mathbf{u} \in \mathbb{R}^d$ , directional var. along  $\mathbf{u}$  is defined as variance of the r.v.  $z = \mathbf{u}^\top \mathbf{x}$  where we choose each data point  $\mathbf{x}^i$  w.p.  $1/n$

Assume data is centered i.e.  $\sum_{i=1}^n \mathbf{x}^i = \mathbf{0}$

Thus,  $\mathbb{E}z = 0$  i.e.  $\mathbb{V}z = \mathbb{E}[z^2]$



Directions with more directional variance preserve more info

PCA does give us orthonormal directions with largest directional variance

$$\mathbb{E}[z^2] = \frac{1}{n} \sum_{i=1}^n (\mathbf{u}^\top \mathbf{x}^i)^2 = \frac{1}{n} \mathbf{u}^\top (X^\top X) \mathbf{u}$$

Thus, finding the direction offering the maximum directional variance is the same as finding the leading right singular vector



# PCA Preserves Maximum Data Variance

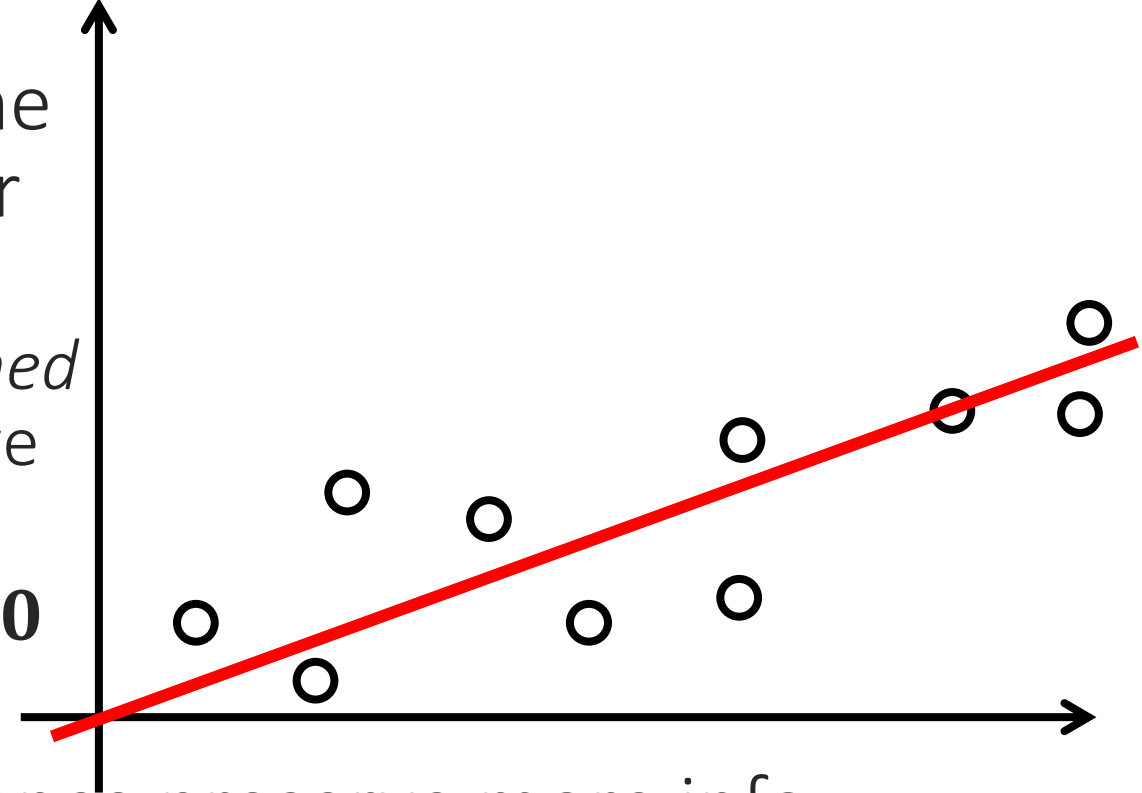
28

Data may take similar values along one direction, varied values along another

**Directional variance:** given a unit vector  $\mathbf{u} \in \mathbb{R}^d$ , directional var. along  $\mathbf{u}$  is defined as variance of the r.v.  $z = \mathbf{u}^\top \mathbf{x}$  where we choose each data point  $\mathbf{x}^i$  w.p.  $1/n$

Assume data is centered i.e.  $\sum_{i=1}^n \mathbf{x}^i = \mathbf{0}$

Thus,  $\mathbb{E}z = 0$  i.e.  $\mathbb{V}z = \mathbb{E}[z^2]$



Directions with more directional variance preserve more info

PCA does give us orthonormal directions with largest directional variance

$$\mathbb{E}[z^2] = \frac{1}{n} \sum_{i=1}^n (\mathbf{u}^\top \mathbf{x}^i)^2 = \frac{1}{n} \mathbf{u}^\top (X^\top X) \mathbf{u}$$

Thus, finding the direction offering the maximum directional variance is the same as finding the leading right singular vector



# PCA Preserves Maximum Data Variance

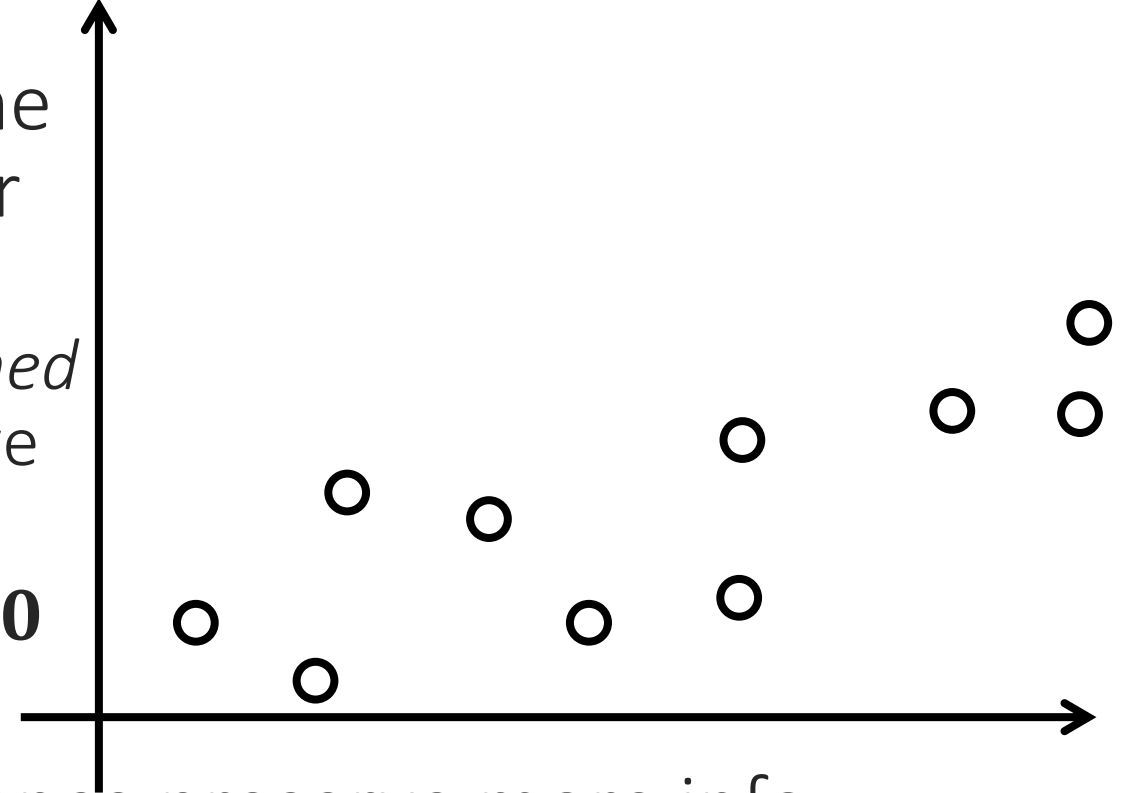
28

Data may take similar values along one direction, varied values along another

**Directional variance:** given a unit vector  $\mathbf{u} \in \mathbb{R}^d$ , directional var. along  $\mathbf{u}$  is defined as variance of the r.v.  $z = \mathbf{u}^\top \mathbf{x}$  where we choose each data point  $\mathbf{x}^i$  w.p.  $1/n$

Assume data is centered i.e.  $\sum_{i=1}^n \mathbf{x}^i = \mathbf{0}$

Thus,  $\mathbb{E}z = 0$  i.e.  $\mathbb{V}z = \mathbb{E}[z^2]$



Directions with more directional variance preserve more info

PCA does give us orthonormal directions with largest directional variance

$$\mathbb{E}[z^2] = \frac{1}{n} \sum_{i=1}^n (\mathbf{u}^\top \mathbf{x}^i)^2 = \frac{1}{n} \mathbf{u}^\top (X^\top X) \mathbf{u}$$

Thus, finding the direction offering the maximum directional variance is the same as finding the leading right singular vector



# PCA Preserves Maximum Data Variance

28

Data may take similar values along one direction, varied values along another

**Directional variance:** given a unit vector  $\mathbf{u} \in \mathbb{R}^d$ , directional var. along  $\mathbf{u}$  is defined as variance of the r.v.  $z = \mathbf{u}^\top \mathbf{x}$  where we choose each data point  $\mathbf{x}^i$  w.p.  $1/n$

Assume data is centered i.e.  $\sum_{i=1}^n \mathbf{x}^i = \mathbf{0}$

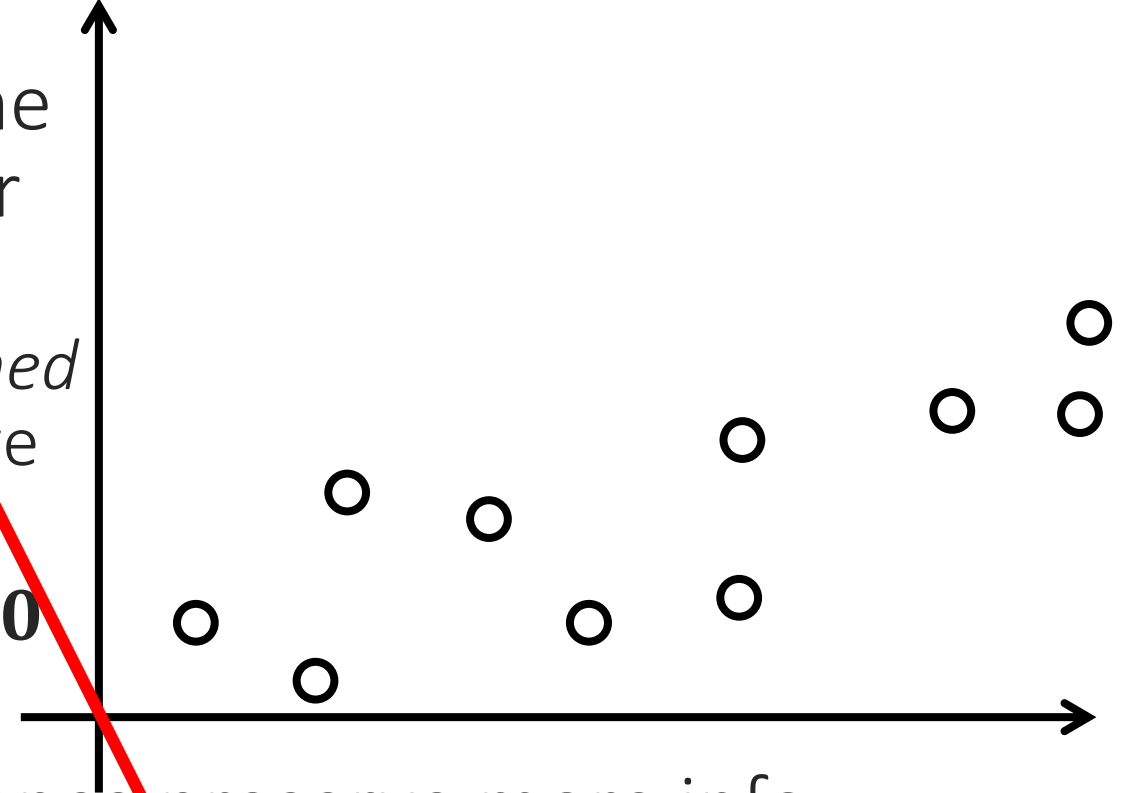
Thus,  $\mathbb{E}z = 0$  i.e.  $\mathbb{V}z = \mathbb{E}[z^2]$

Directions with more directional variance preserve more info

PCA does give us orthonormal directions with largest directional variance

$$\mathbb{E}[z^2] = \frac{1}{n} \sum_{i=1}^n (\mathbf{u}^\top \mathbf{x}^i)^2 = \frac{1}{n} \mathbf{u}^\top (X^\top X) \mathbf{u}$$

Thus, finding the direction offering the maximum directional variance is the same as finding the leading right singular vector



# PCA Preserves Maximum Data Variance

28

Data may take similar values along one direction, varied values along another

**Directional variance:** given a unit vector  $\mathbf{u} \in \mathbb{R}^d$ , directional var. along  $\mathbf{u}$  is defined as variance of the r.v.  $z = \mathbf{u}^T \mathbf{x}$  where we choose each data point  $\mathbf{x}^i$  w.p.  $1/n$

Assume data is centered i.e.  $\sum_{i=1}^n \mathbf{x}^i = \mathbf{0}$

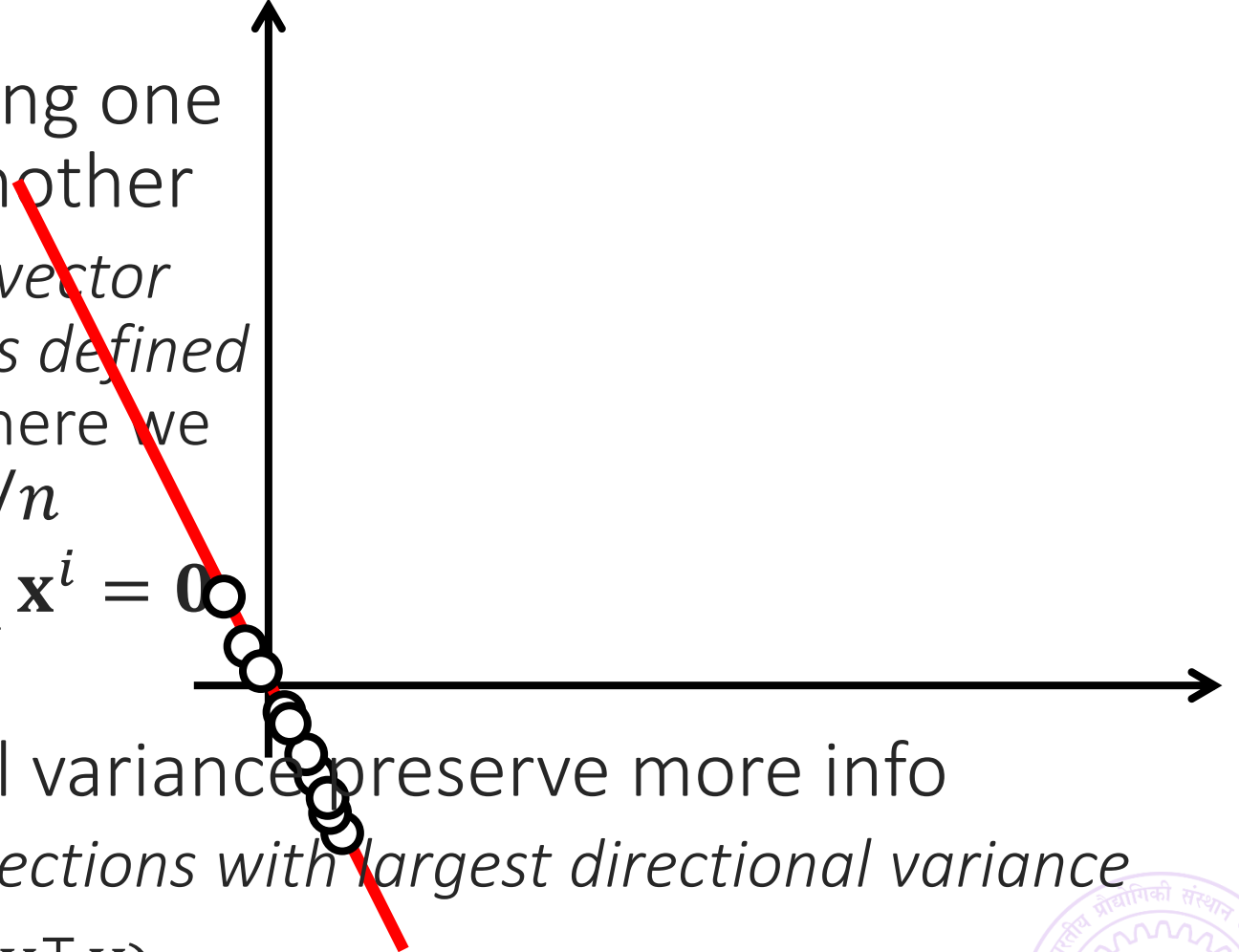
Thus,  $\mathbb{E}z = 0$  i.e.  $\mathbb{V}z = \mathbb{E}[z^2]$

Directions with more directional variance preserve more info

PCA does give us orthonormal directions with largest directional variance

$$\mathbb{E}[z^2] = \frac{1}{n} \sum_{i=1}^n (\mathbf{u}^T \mathbf{x}^i)^2 = \frac{1}{n} \mathbf{u}^T (X^T X) \mathbf{u}$$

Thus, finding the direction offering the maximum directional variance is the same as finding the leading right singular vector



# PCA Preserves Maximum Data Variance

28

Data may take similar values along one direction, varied values along another

**Directional variance:** given a unit vector  $\mathbf{u} \in \mathbb{R}^d$ , directional var. along  $\mathbf{u}$  is defined as variance of the r.v.  $z = \mathbf{u}^T \mathbf{x}$  where we choose each data point  $\mathbf{x}^i$  w.p.  $1/n$

Assume data is centered i.e.  $\sum_{i=1}^n \mathbf{x}^i = \mathbf{0}$

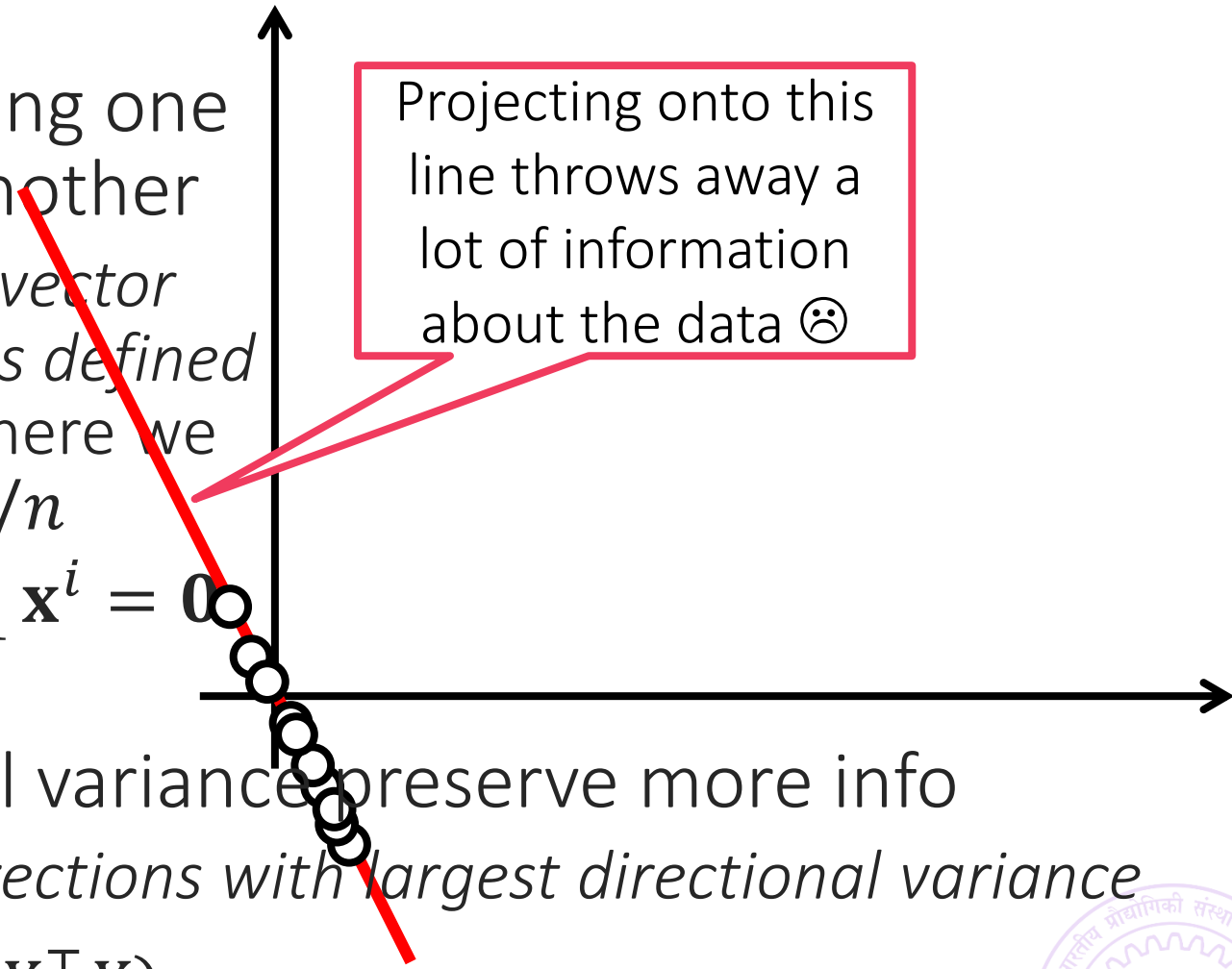
Thus,  $\mathbb{E}z = 0$  i.e.  $\mathbb{V}z = \mathbb{E}[z^2]$

Directions with more directional variance preserve more info

PCA does give us orthonormal directions with largest directional variance

$$\mathbb{E}[z^2] = \frac{1}{n} \sum_{i=1}^n (\mathbf{u}^T \mathbf{x}^i)^2 = \frac{1}{n} \mathbf{u}^T (X^T X) \mathbf{u}$$

Thus, finding the direction offering the maximum directional variance is the same as finding the leading right singular vector



Projecting onto this line throws away a lot of information about the data 😞



# PCA Preserves Maximum Data Variance

28

Data may take similar values along one direction, varied values along another

**Directional variance:** given a unit vector  $\mathbf{u} \in \mathbb{R}^d$ , directional var. along  $\mathbf{u}$  is defined as variance of the r.v.  $z = \mathbf{u}^T \mathbf{x}$  where we choose each data point  $\mathbf{x}^i$  w.p.  $1/n$

Assume data is centered i.e.  $\sum_{i=1}^n \mathbf{x}^i = \mathbf{0}$

Thus,  $\mathbb{E}z = 0$  i.e.  $\mathbb{V}z = \mathbb{E}[z^2]$

Directions with more directional variance preserve more info

We can show that for  $k > 1$  as well, PCA offers a set of orthonormal directions such that the total directional variance captured along those directions is the maximum.

the maximum directional variance is the same as finding the leading right singular vector

Projecting onto this line throws away a lot of information about the data ☹️

