Principal Component Analysis (PCA)

CSC411/2515 Tutorial

Slides by Wenjie Luo, Ladislav Rampasek

Lagrange Multipliers

- If we want to find stationary point of a function of multiple variables $f(\mathbf{x})$ subject to one or more constraints $g(\mathbf{x}) = 0$

1. Introduce Lagrangian function:

$$L(\mathbf{x}, \lambda) \equiv f(\mathbf{x}) + \lambda g(\mathbf{x})$$

2. and find it's stationary point w.r.t. both $\mathbf{x}$ and λ

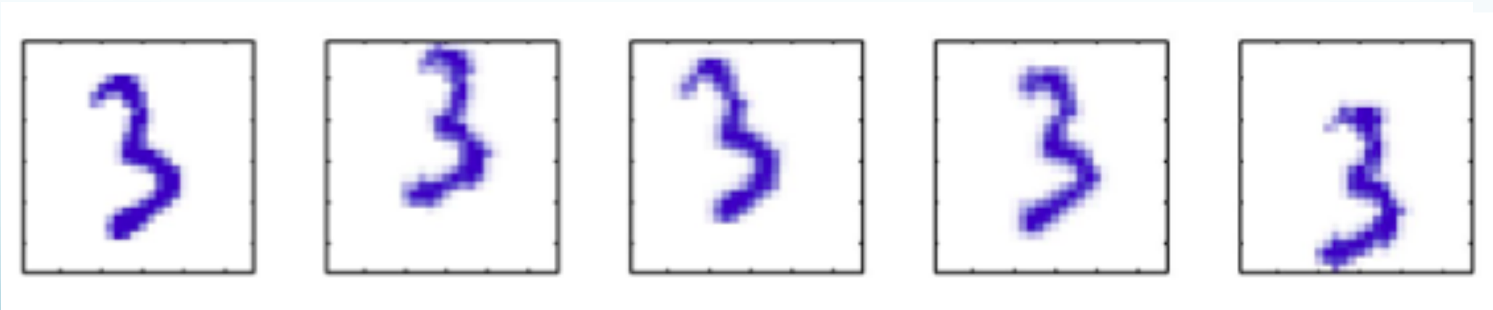
- If you are not familiar with it, check out Appendix E in Bishop's book

Dimensionality Reduction

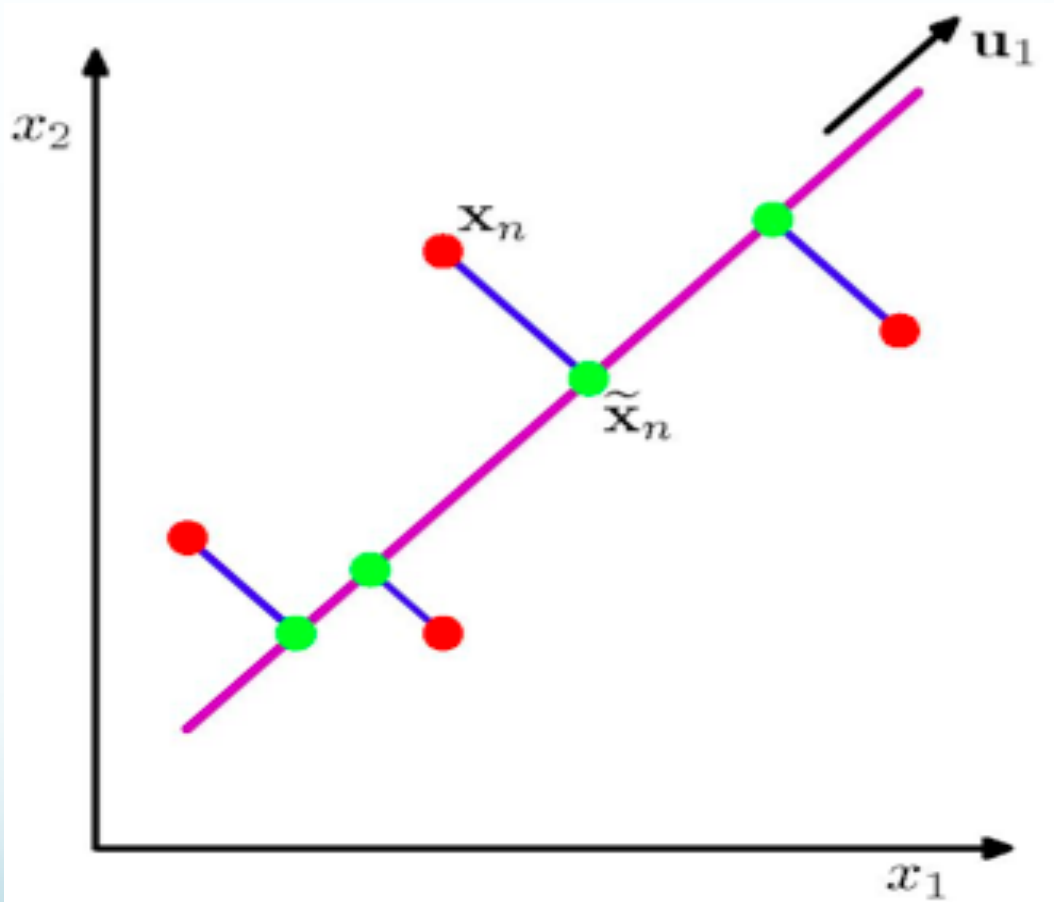
- We have some data $X \in \mathbb{R}^{N \times D}$
- D may be huge, etc.
- We would like to find a new representation $Z \in \mathbb{R}^{N \times K}$ where K << D.
  - For computational reasons.
  - To better understand (e.g., visualize) the data.
  - For compression.
  - ...
- We will restrict ourselves to linear transformations for the time being.

Example

- In this dataset, there are only 3 degrees of freedom: horizontal and vertical translations, and rotations.
- Yet each image contains 784 pixels, so X will be 784 elements wide.

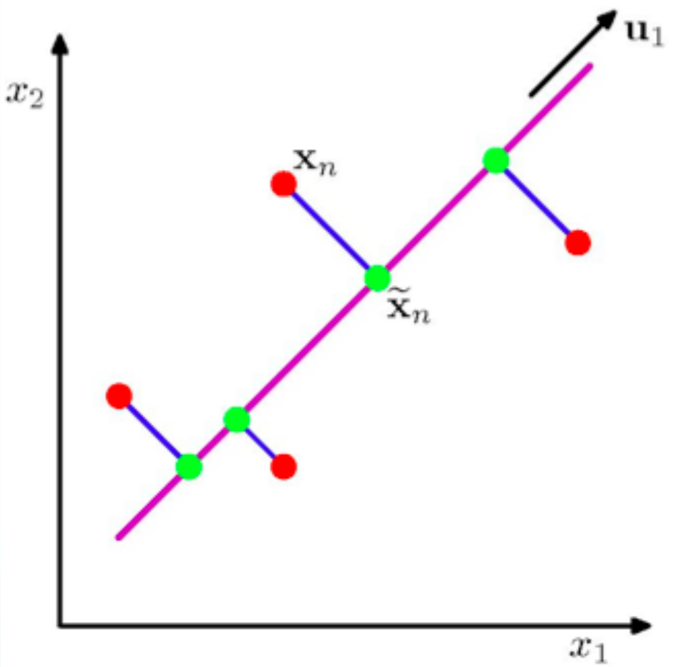


Abstract Visualization



What is a Good Transformation?

- Goal is to find good directions u that preserves "important" aspects of the data.
- In a linear setting: $z = x^T u$
- This will turn out to be the top-K eigenvalues of the data covariance.
- Two ways to view this:
    1. Find directions of *maximum variation*
    2. Find projections that *minimize reconstruction error*



Principal Component Analysis (Maximum Variance)

$$\text{maximize} \frac{1}{2N} \sum_{n=1}^{N} (u_1^T x_n - u_1^T \bar{x}_n)^2$$
$$= u_1^T S u_1$$

i.e., variance of the projected data

where the sample mean and covariance are given by:

$$\bar{x} = \frac{1}{N} \sum_{n=1}^{N} x_n$$

$$S = \frac{1}{N} \sum_{n=1}^{N} (x_n - \bar{x})(x_n - \bar{x})^T$$

Finding u1

- We want to maximize $u_1^T S u_1$

  subject to $||u_1|| = 1$
  (since we are finding a direction)

- Use Lagrange multiplier α_1 to express this as

$$u_1^T S u_1 + \alpha_1 (1 - u_1^T u_1)$$

Finding u1

- Take derivative and set to 0

$$
\begin{aligned}
Su_1 - \alpha_1 u_1 &= 0 \\
Su_1 &= \alpha_1 u_1
\end{aligned}
$$

- So u_1 is an eigenvector of S with eigenvalue α_1
- In fact it must be the eigenvector with maximum eigenvalue, since this maximizes the objective.

Finding u2

$$\begin{aligned} \text{maximize} \quad & u_2^T S u_2 \\ \text{subject to} \quad & ||u_2|| = 1 \\ & u_2^T u_1 = 0 \end{aligned}$$

Lagrange form: $u_2^T S u_2 + \alpha_2(1 - u_2^T u_2) - \beta u_2^T u_1$

Finding β: $\frac{\partial}{\partial u_2} = S u_2 - \alpha_2 u_2 - \beta u_1 = 0$

$$\implies u_1^T S u_2 - \alpha_2 u_1^T u_2 - \beta u_1^T u_1 = 0$$

$$\implies \alpha_1 u_1^T u_2 - \alpha_2 u_1^T u_2 - \beta u_1^T u_1 = 0$$

$$\implies \alpha_1 \cdot 0 - \alpha_2 \cdot 0 - \beta \cdot 1 = 0$$

$$\implies \beta = 0$$

Finding u2

$$\begin{aligned} \text{maximize} \quad & u_2^T S u_2 \\ \text{subject to} \quad & ||u_2|| = 1 \\ & u_2^T u_1 = 0 \end{aligned}$$

Lagrange form: $$u_2^T S u_2 + \alpha_2(1 - u_2^T u_2) - \beta u_2^T u_1$$

0

Finding α_2: $$\frac{\partial}{\partial u_2} = S u_2 - \alpha_2 u_2 = 0$$

$$\Longrightarrow S u_2 = \alpha_2 u_2$$

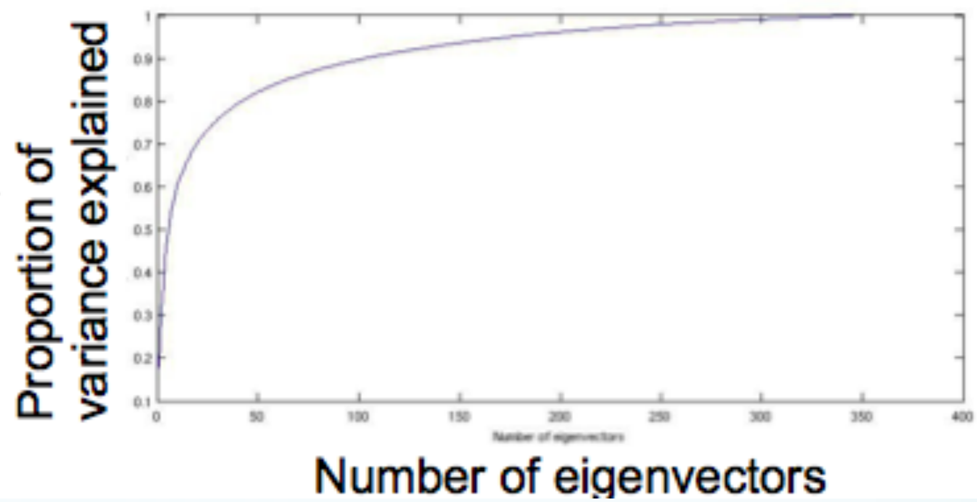
So α_2 must be the second largest eigevalue of S.

PCA in General

- We can compute the entire PCA solution by just computing the eigenvectors with the top-k eigenvalues.
- These can be found using the singular value decomposition of S.

- How do we choose the number of components?

$$\frac{\sum_{i=1}^{M} \alpha_i}{\sum_{i=1}^{N} \alpha_i}$$



- Look at the spectrum of covariance, pick K to capture most of the variation.
- More principled: Bayesian treatment (beyond this course).

Demo

- Eigenfaces

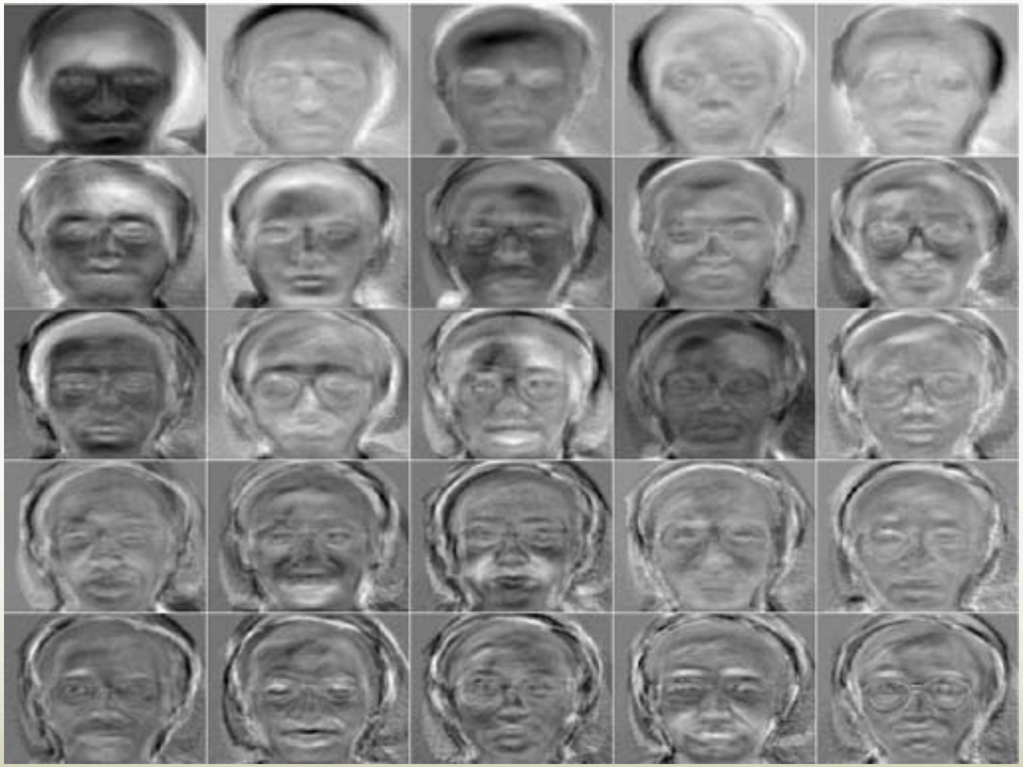
PCA for face recognition

- *Goal:*
  Face recognition by similarity in principal subspace
- Learn the PCA projection on train set of 319x242 face images
- Reparameterize a query picture to a basis of "eigenfaces"
- Eigenvectors of the data covariance matrix can be rearrainged into a 2D image --> has the appearance of a ghostly face

adapted from http://www.cs.princeton.edu/~cdecoro/eigenfaces/

Eigenfaces

Eigenfaces = principal components of a dataset of face images



adapted from http://www.cs.princeton.edu/~cdecoro/eigenfaces/

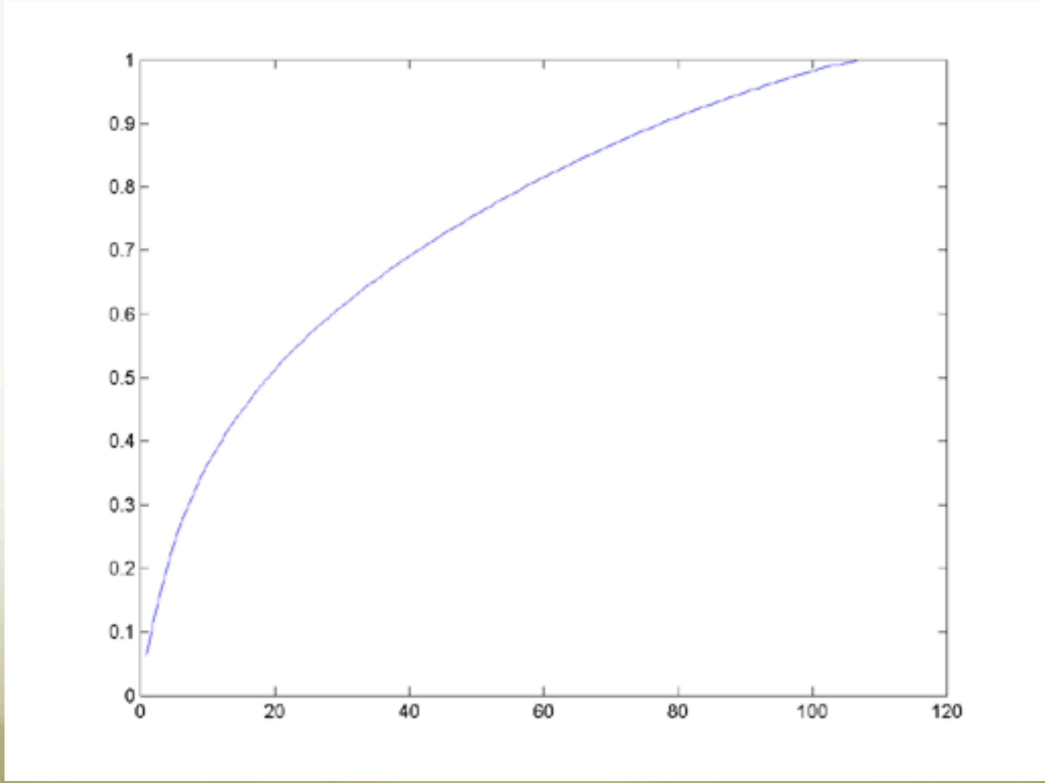
Face recognition results

- Trained on 70% of the data set with K=25
- Includes faces with glasses or different lighting conditions

adapted from http://www.cs.princeton.edu/~cdecoro/eigenfaces/

Proportion of covariance explained

- How much of the variation is captured by the first K principal components?
- K=10 => variance=0.363; K=25 => variance=0.566



adapted from http://www.cs.princeton.edu/~cdecoro/eigenfaces/

Eigenfaces for reconstruction

- Using **K=1** to **K=25** principal components

adapted from http://www.cs.princeton.edu/~cdecoro/eigenfaces/

Eigenfaces for reconstruction

- Using **K=1** to **K=97** principal components (with steps of 8 PC)

adapted from http://www.cs.princeton.edu/~cdecoro/eigenfaces/

Eigenfaces for reconstruction

- Removing faces with glasses from data set helps to reduce the K needed for good reconstruction
- But not as much as removing faces with different lighting conditions
- => lighting conditions create a lot of variance in the data, thus they are captured by PCs before capturing detail features of a face

adapted from http://www.cs.princeton.edu/~cdecoro/eigenfaces/

Eigenfaces for reconstruction

- Using **K=1** to **K=25** principal components when faces with different lighting conditions or glasses are removed from training set

adapted from http://www.cs.princeton.edu/~cdecoro/eigenfaces/

Principal Component Analysis (Minimum Reconstruction Error)

- We can also think of PCA as minimizing the *reconstruction error* of the compressed data.

$$\text{minimize} = \frac{1}{2N}\sum_{n=1}^{N}||x_n - \hat{x}_n||^2$$

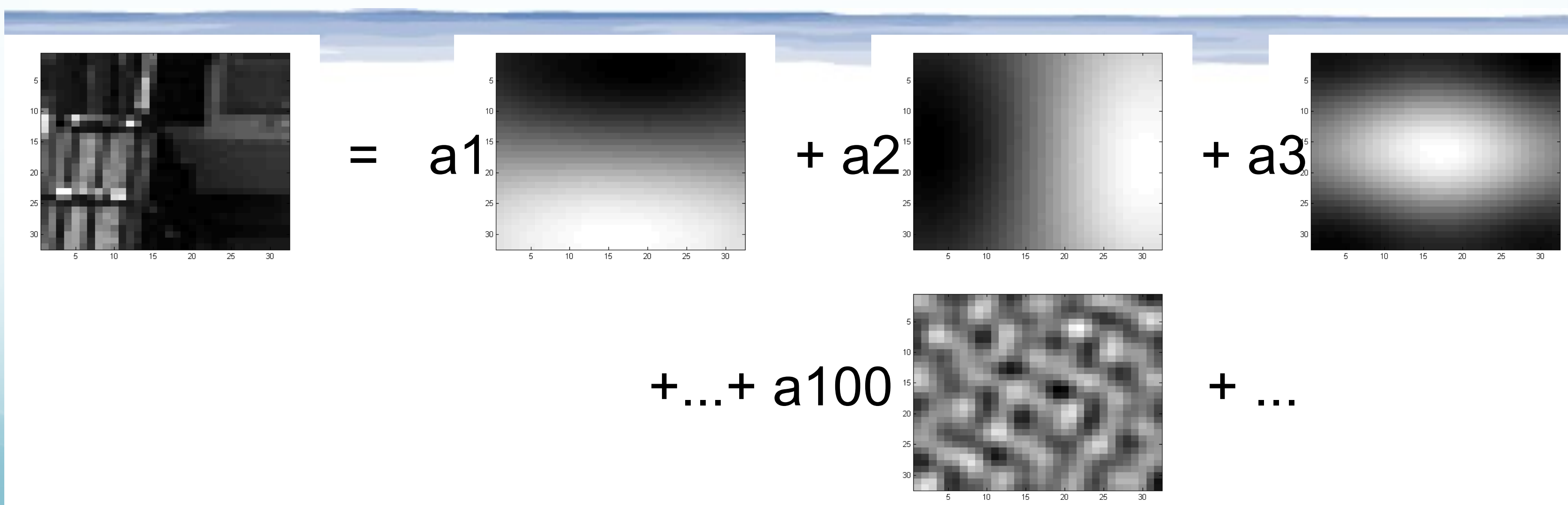
- We will omit the details for now, but the key is that we define some K-dimensional basis such that:

$$\hat{x} = Wx + const$$

- The solution will turn out to be the same as the minimum variance formulation.

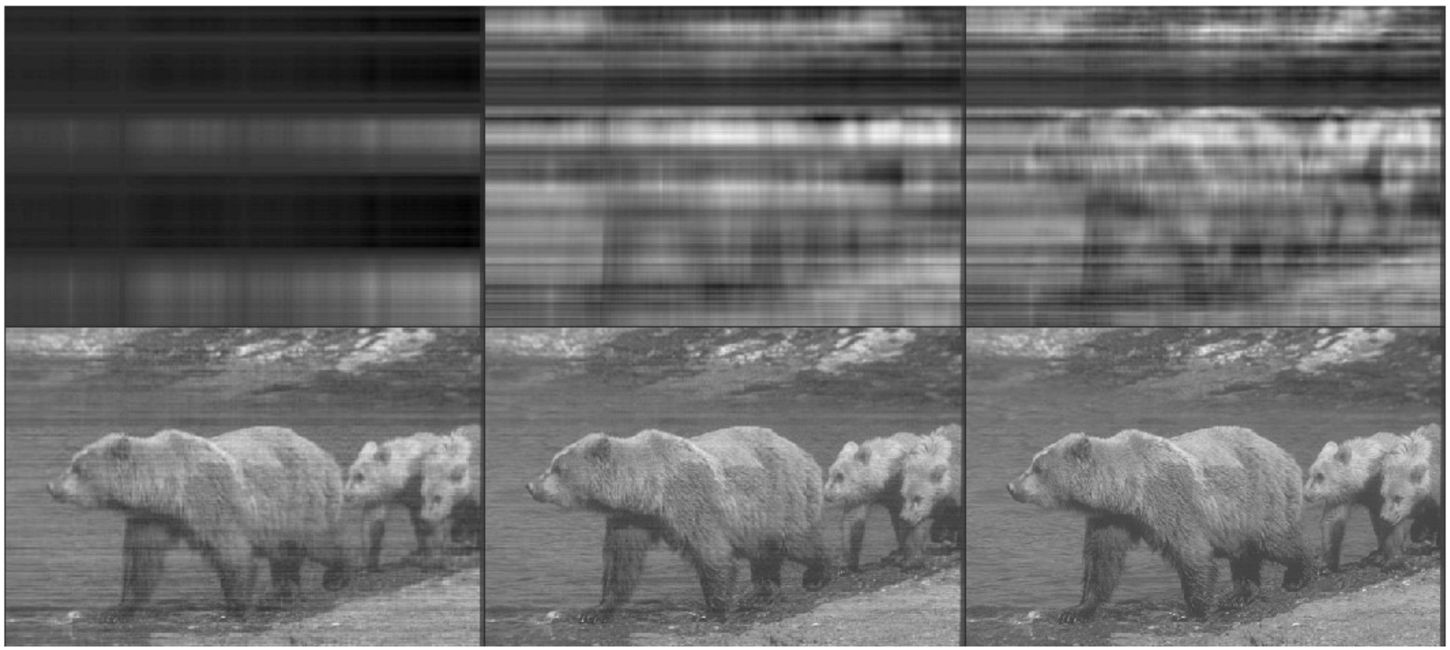
Reconstruction

- PCA learns to represent vectors in terms of sums of basis vectors.
- For images, e.g.,



PCA for Compression

D=1 D=5 D=10



D=50 D=100 D=200

321x481 image, D is the number of basis vectors used

D in this slide is the same as K in the previous slides

Summary (1)

- PCA is a linear projection of D-dimensional $\{\mathbf{x}_n\}$ to K≤D vector space given by $\{\mathbf{u}_k\}$ basis vectors such that it:
  - maximizes variance
  - minimizes projection error (square loss)
  - $\{\mathbf{u}_k\}$ are orthonormal
  - $\{\mathbf{u}_k\}$ turn out to be first K eigenvectors of the data covariance matrix with K larges eigenvalues
  - can be computed in $O(KD^2)$

Summary (2)

- PCA is good for:
  - Dimensionality reduction
  - Visualization
  - Compression (with loss)
  - Denoising (by removing small variance in the data)
  - Can be used for data **whitening** = decorrelation, so that features have unit covariance
- Caution! In classification task, if the class labels' signal in the data has small variance, PCA may remove it completely

Thank You ;-)