

ITCS 6156/8156 Fall 2024  
Machine Learning

# Principal Components Analysis & Autoencoders

Instructor: Hongfei Xue

Email: [hongfei.xue@charlotte.edu](mailto:hongfei.xue@charlotte.edu)

Class Meeting: Tue & Thu, 4:00 PM – 5:15 PM, WWH 130



Some content in the slides is based on Dr. Raquel Urtasun's lecture

# Example

- What are the intrinsic latent dimensions in these two datasets?



- How can we find these dimensions from the data?

# PCA Toy Example

Consider the following 3D points

1	2	4	3	5	6
2	4	8	6	10	12
3	6	12	9	15	18

If each component is stored in a byte,  
we need  $18 = 3 \times 6$  bytes

# PCA Toy Example

Looking closer, we can see that all the points are related geometrically: they are all the same point, scaled by a factor:

1		1		2		1		4		1
2	$= 1 *$	2		4	$= 2 *$	2		8	$= 4 *$	2
3		3		6		3		12		3
3		1		5		1		6		1
6	$= 3 *$	2		10	$= 5 *$	2		12	$= 6 *$	2
9		3		15		3		18		3

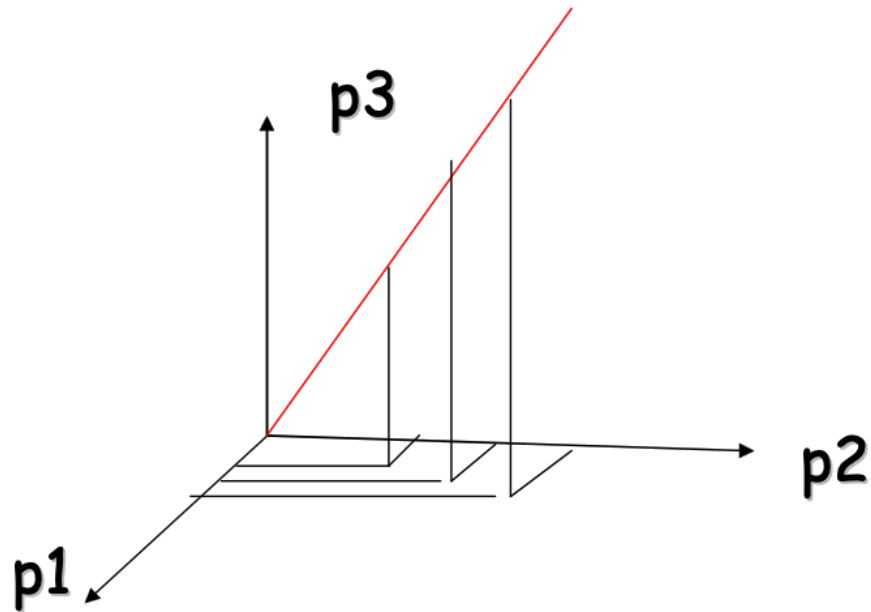
# PCA Toy Example

1		1		2		1		4		1
2	= 1 *	2		4	= 2 *	2		8	= 4 *	2
3		3		6		3		12		3
3		1		5		1		6		1
6	= 3 *	2		10	= 5 *	2		12	= 6 *	2
9		3		15		3		18		3

They can be stored using only 9 bytes (50% savings!):  
Store one point (3 bytes) + the multiplying constants (6 bytes)

# Geometrical Interpretation

View each point in 3D space.

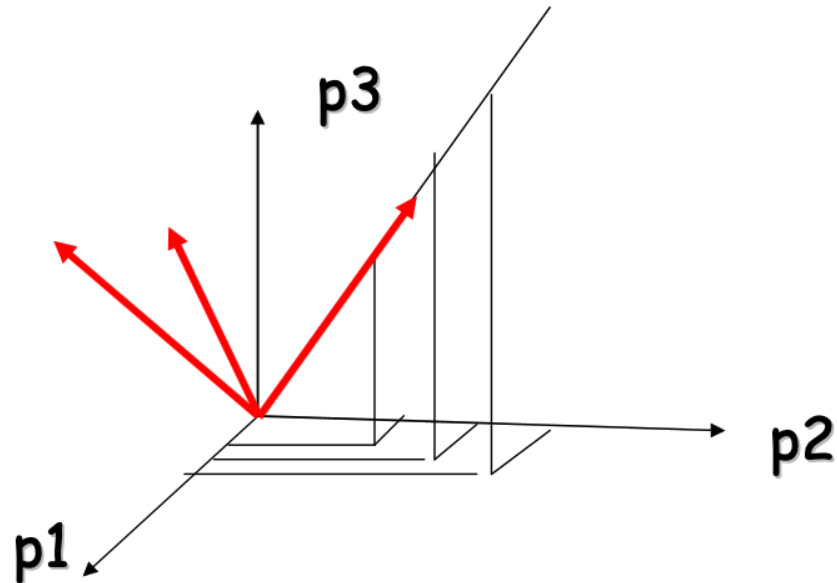


But in this example, all the points happen to belong to a line: a 1D subspace of the original 3D space.



# Geometrical Interpretation

Consider a new coordinate system where one of the axes is along the direction of the line:



In this coordinate system, every point has only one non-zero coordinate: we only need to store the direction of the line (a 3 bytes image) and the non-zero coordinate for each of the points (6 bytes).

# Principal Components Analysis

- Given a set of points, how do we know if they can be compressed like in the previous example?
  - The answer is to look into the correlation between the points
  - The tool for doing this is called Principal Components Analysis (PCA)



# Principal Components Analysis

- An exploratory technique used to reduce the dimensionality of the data set to 2D or 3D
- **Aim:** find a small number of “directions” in input space that explain variation in input data; re-represent data by projecting along those directions
- **Important assumption:** variation contains information
- Can be used to:
  - Reduce number of dimensions in data
  - Find patterns in high-dimensional data
  - Visualize data of high dimensionality
- Example **applications:**
  - Face recognition
  - Image compression
  - Gene expression analysis

# Covariance

- Variance and Covariance are a measure of the “spread” of a set of points around their center of mass (mean)
- Variance - measure of the deviation from the mean for points in one dimension e.g. heights
- Covariance as a measure of how much **each of the dimensions** vary from the mean **with respect to each other**.
- Covariance is measured between 2 dimensions to see if there is a relationship between the 2 dimensions e.g. number of hours studied & marks obtained.
- The covariance between one dimension and itself is the variance

# Covariance

$$\text{covariance } (X,Y) = \frac{\sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y})}{(n-1)}$$

- So, if you had a 3-dimensional data set  $(x,y,z)$ , then you could measure the covariance between the x and y dimensions, the y and z dimensions, and the x and z dimensions. Measuring the covariance between x and x , or y and y , or z and z would give you the variance of the x , y and z dimensions respectively.

# Covariance Matrix

- Representing Covariance between dimensions as a matrix e.g. for 3 dimensions:

$$C = \begin{bmatrix} \text{cov}(x,x) & \text{cov}(x,y) & \text{cov}(x,z) \\ \text{cov}(y,x) & \text{cov}(y,y) & \text{cov}(y,z) \\ \text{cov}(z,x) & \text{cov}(z,y) & \text{cov}(z,z) \end{bmatrix}$$

**Variances**

- Diagonal is the **variances** of x, y and z
- $\text{cov}(x,y) = \text{cov}(y,x)$  hence matrix is **symmetrical** about the diagonal
- N-dimensional data will result in **NxN covariance matrix**

# Covariance

- What is the interpretation of covariance calculations?

e.g.: 2 dimensional data set

x: number of hours studied for a subject

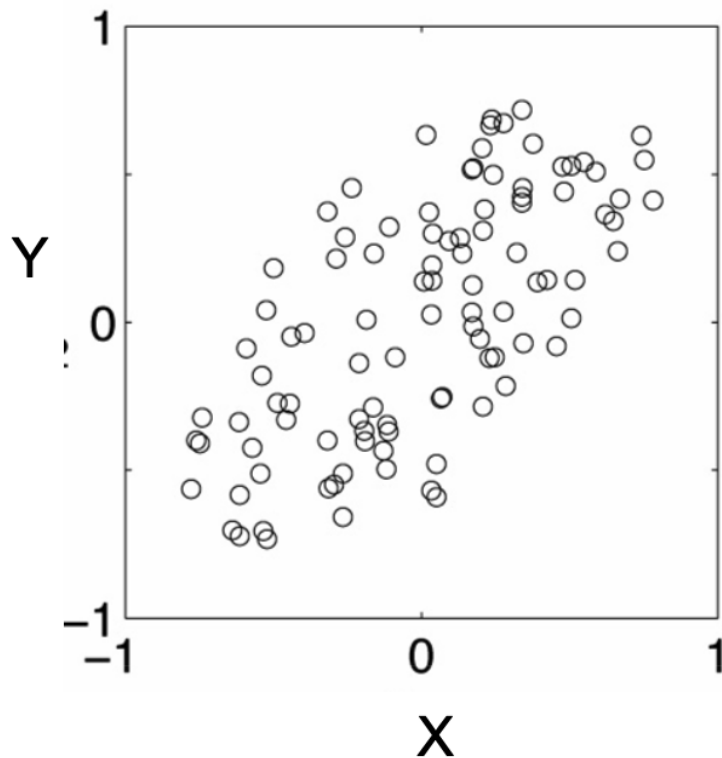
y: marks obtained in that subject

covariance value is say: 104.53

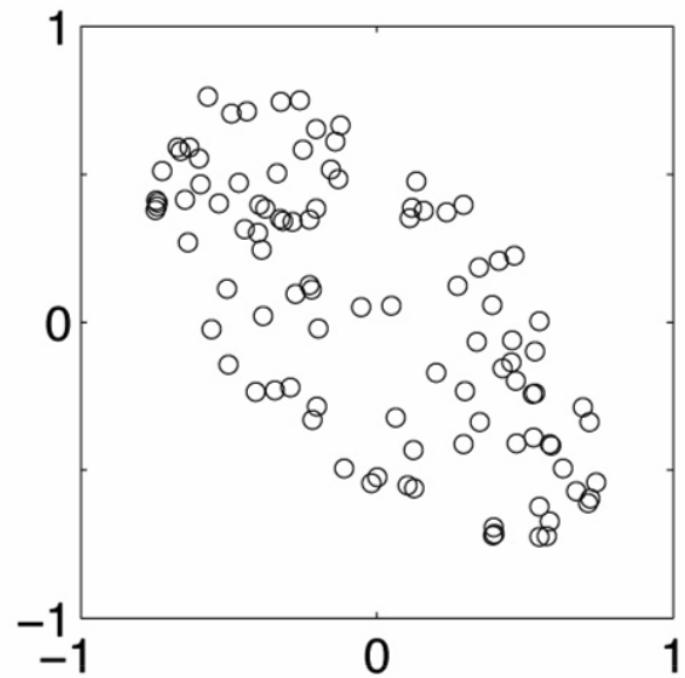
what does this value mean?

# Covariance

positive covariance



negative covariance



# Covariance

- Exact value is not as important as it's sign.
- A positive value of covariance indicates **both dimensions increase or decrease together** e.g. as the number of hours studied increases, the marks in that subject increase.
- A negative value indicates **while one increases the other decreases**, or vice-versa e.g. active social life at UNCC vs performance in CS dept.
- If covariance is zero: the two dimensions are **independent** of each other e.g. heights of students vs the marks obtained in a subject



# Covariance

- Why bother with calculating covariance when we could just plot the 2 values to see their relationship?

Covariance calculations are used to find relationships between dimensions in high dimensional data sets (usually greater than 3) where visualization is difficult.

# Eigenvalues & eigenvectors

- $A\mathbf{x}=\lambda\mathbf{x} \Leftrightarrow (A-\lambda I)\mathbf{x}=0$
- How to calculate  $\mathbf{x}$  and  $\lambda$ :
  - Calculate  $\det(A-\lambda I)$ , yields a polynomial (degree  $n$ )
  - Determine roots to  $\det(A-\lambda I)=0$ , roots are eigenvalues  $\lambda$
  - Solve  $(A-\lambda I)\mathbf{x}=0$  for each  $\lambda$  to obtain eigenvectors  $\mathbf{x}$

# Eigenvalues & eigenvectors

- Vectors  $\mathbf{x}$  having same direction as  $A\mathbf{x}$  are called *eigenvectors* of  $A$  ( $A$  is an  $n$  by  $n$  matrix).
- In the equation  $A\mathbf{x}=\lambda\mathbf{x}$ ,  $\lambda$  is called an *eigenvalue* of  $A$ .

$$\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} x \begin{pmatrix} 3 \\ 2 \end{pmatrix} = \begin{pmatrix} 12 \\ 8 \end{pmatrix} = 4x \begin{pmatrix} 3 \\ 2 \end{pmatrix}$$

# PCA

- By finding the **eigenvalues and eigenvectors** of the covariance matrix, we find that the eigenvectors with the largest eigenvalues correspond to the dimensions that have the strongest correlation in the dataset.
- This is the **principal component**.

# Principle of Maximal Variance

- Least loss of information
- Best capture the “spread”
- What is the direction of maximal variance?
- Given any direction ( $\hat{\mathbf{u}}$ ), the projection of  $\mathbf{x}$  on  $\hat{\mathbf{u}}$  is given by:

The diagram illustrates the projection of centered data onto a unit vector. On the left, a box labeled "Unit vector" has an arrow pointing to the term  $\hat{\mathbf{u}}$  in the expression  $\mathbf{x}_i^\top \hat{\mathbf{u}}$ . On the right, a box labeled "Centered data" has an arrow pointing to the term  $\mathbf{x}_i$  in the same expression. The expression  $\mathbf{x}_i^\top \hat{\mathbf{u}}$  is centered between the two boxes.

- Direction of maximal variance can be obtained by maximizing

$$\begin{aligned} \frac{1}{N} \sum_{i=1}^N (\mathbf{x}_i^\top \hat{\mathbf{u}})^2 &= \frac{1}{N} \sum_{i=1}^N \hat{\mathbf{u}}^\top \mathbf{x}_i \mathbf{x}_i^\top \hat{\mathbf{u}} \\ &= \hat{\mathbf{u}}^\top \left( \frac{1}{N} \sum_{i=1}^N \mathbf{x}_i \mathbf{x}_i^\top \right) \hat{\mathbf{u}} \end{aligned}$$

# Finding Direction of Maximal Variance

Find:

$$\max_{\hat{\mathbf{u}}: \hat{\mathbf{u}}^\top \hat{\mathbf{u}}=1} \hat{\mathbf{u}}^\top \mathbf{S} \hat{\mathbf{u}}$$

where:

$$\mathbf{S} = \frac{1}{N} \sum_{i=1}^N \mathbf{x}_i \mathbf{x}_i^\top$$

- $\mathbf{S}$  is the sample (empirical) covariance matrix of the mean-centered data

# Solution

The solution to the above constrained optimization problem may be obtained using the Lagrange multipliers method. We maximize the following w.r.t.  $\hat{\mathbf{u}}$ :

$$(\hat{\mathbf{u}}^\top \mathbf{S} \hat{\mathbf{u}}) - \lambda(\hat{\mathbf{u}}^\top \hat{\mathbf{u}} - 1)$$

to get:

$$\begin{aligned} \frac{d}{d\hat{\mathbf{u}}}(\hat{\mathbf{u}}^\top \mathbf{S} \hat{\mathbf{u}}) - \lambda(\hat{\mathbf{u}}^\top \hat{\mathbf{u}} - 1) &= 0 \\ \mathbf{S} \hat{\mathbf{u}} - \lambda \hat{\mathbf{u}} &= 0 \\ \mathbf{S} \hat{\mathbf{u}} &= \lambda \hat{\mathbf{u}} \end{aligned}$$

Obviously, the solution to the above equation is an eigen vector of the matrix  $\mathbf{S}$ . But which  $\mathbf{S}$ ? Note that for the optimal solution:

$$\hat{\mathbf{u}}^\top \mathbf{S} \hat{\mathbf{u}} = (\hat{\mathbf{u}}^\top \lambda \hat{\mathbf{u}}) = \lambda$$

Thus we should choose the largest possible  $\lambda$  which means that the first solution is the eigen vector of  $\mathbf{S}$  with largest eigen value.



# PCA Algorithm

1. Center  $\mathbf{X}$

$$\mathbf{X} = \mathbf{X} - \hat{\boldsymbol{\mu}}$$

2. Compute sample covariance matrix:

$$\mathbf{S} = \frac{1}{N-1} \mathbf{X}^\top \mathbf{X}$$

3. Find eigen vectors and eigen values for  $\mathbf{S}$
4.  $\mathbf{W}$  consists of first  $L$  eigen vectors as columns
  - Ordered by decreasing eigen-values
  - $\mathbf{W}$  is  $D \times L$

5. Let  $\mathbf{Z} = \mathbf{XW}$

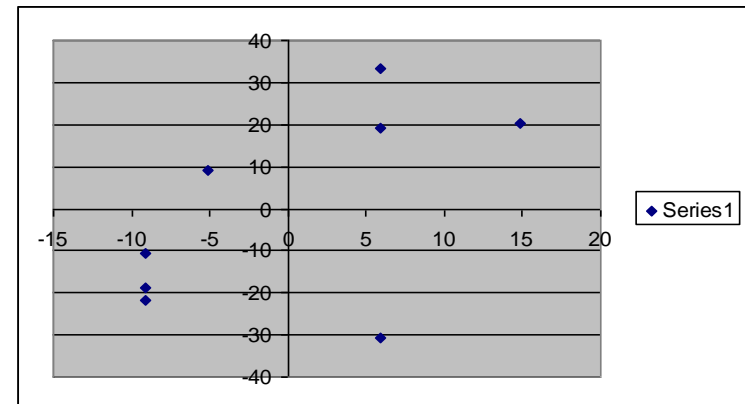
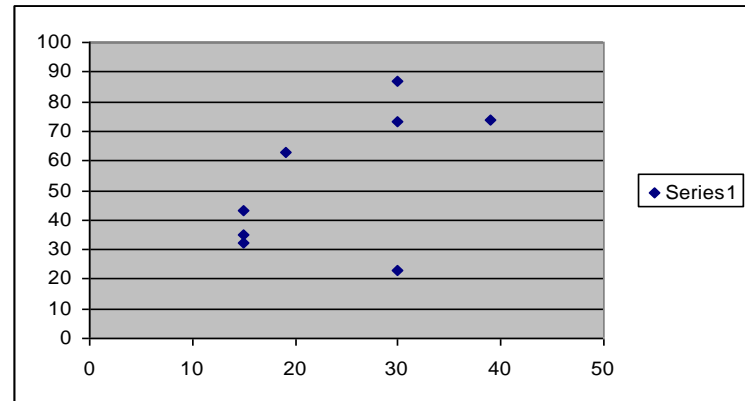
6. Each row in  $\mathbf{Z}$  (or  $\mathbf{z}_i^\top$ ) is the lower dimensional embedding of  $\mathbf{x}_i$

# An Example

X1	X2	X1'	X2'
19	63	-5.1	9.25
39	74	14.9	20.25
30	87	5.9	33.25
30	23	5.9	-30.75
15	35	-9.1	-18.75
15	43	-9.1	-10.75
15	32	-9.1	-21.75
30	73	5.9	19.25

Mean1=24.1

Mean2=53.8



# An Example

- $C =$ 

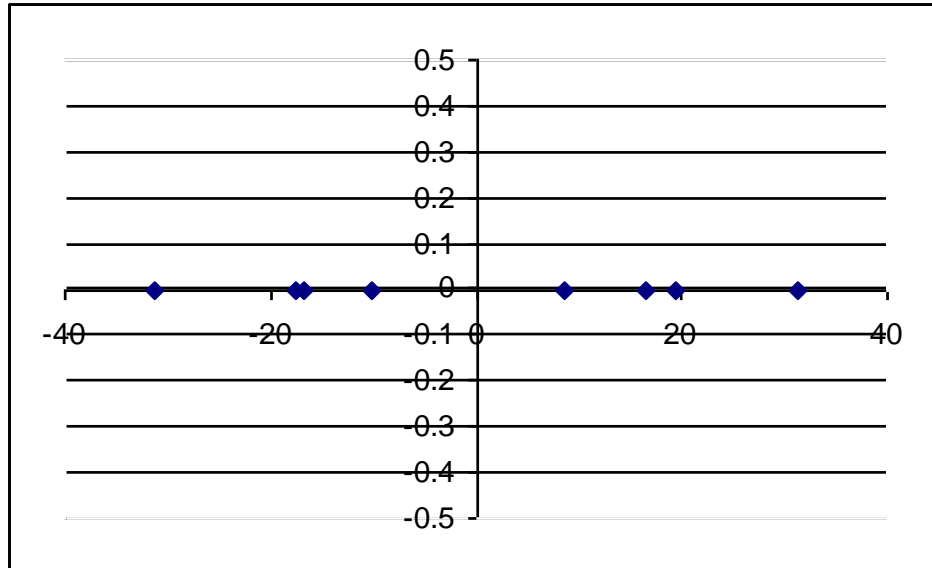
75	106
106	482

- Using MATLAB, we find out:
  - Eigenvectors:
    - $e_1 = (-0.98, -0.21)$ ,  $\lambda_1 = 51.8$
    - $e_2 = (0.21, -0.98)$ ,  $\lambda_2 = 560.2$
    - Thus the second eigenvector is more important!

# An Example

- If we only keep one dimension: e2

- We keep the dimension of  $e2=(0.21,-0.98)$
- We can obtain the final data as

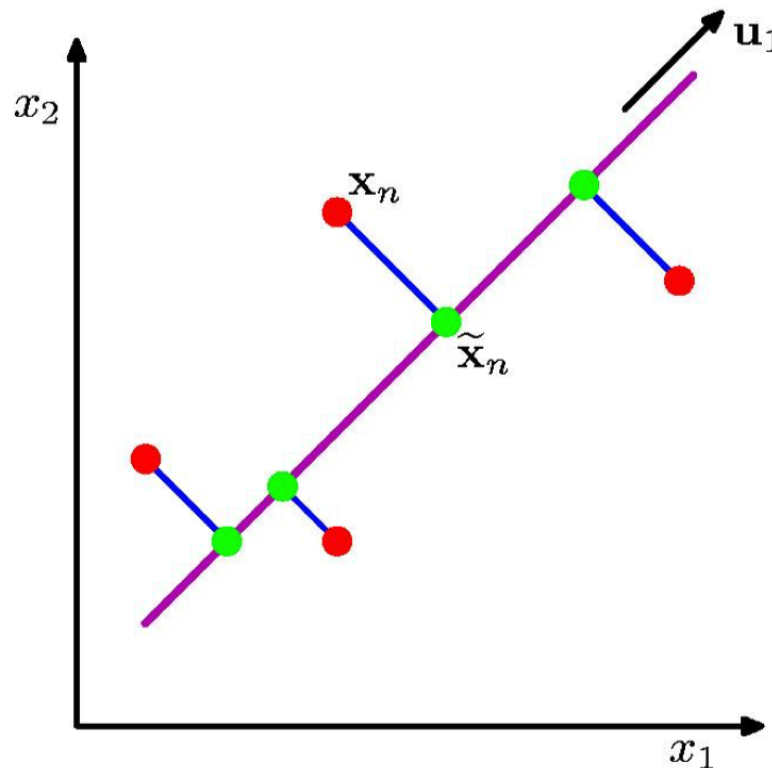


$y_i$
-10.14
-16.72
-31.35
31.374
16.464
8.624
19.404
-17.63

$$y_i = (0.21 \quad -0.98) \begin{pmatrix} x_{i1} \\ x_{i2} \end{pmatrix} = 0.21 * x_{i1} - 0.98 * x_{i2}$$

# Two Derivations of PCA

- Two views/derivations:
  - ▶ Maximize variance (scatter of green points)
  - ▶ Minimize error (red-green distance per datapoint)



# Applying PCA to faces

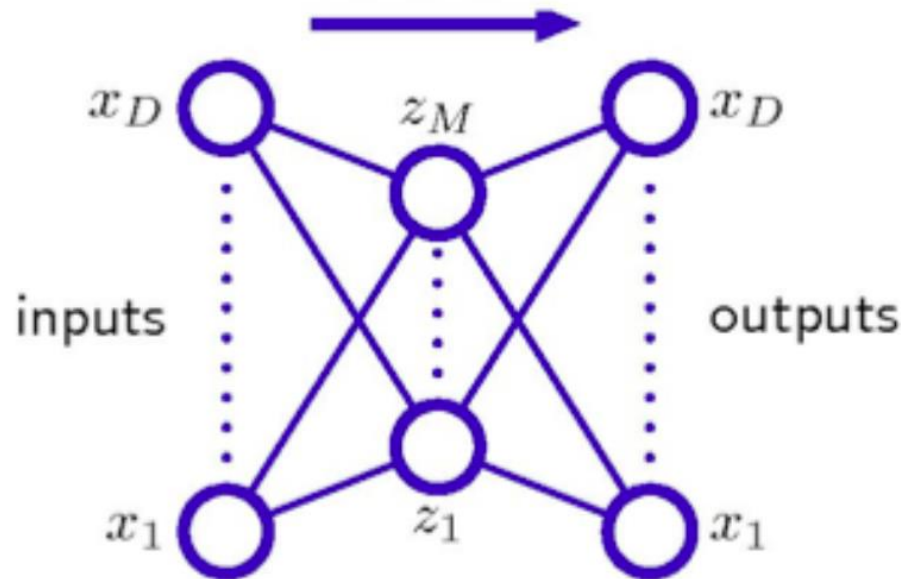
- Run PCA on 2429 19x19 grayscale images (CBCL data)
- Compresses the data: can get good reconstructions with only 3 components



- PCA for pre-processing: can apply classifier to latent representation
  - ▶ PCA with 3 components obtains 79% accuracy on face/non-face discrimination on test data vs. 76.8% for GMM with 84 states
- Can also be good for visualization

# Relation to Neural Networks

- PCA is closely related to a particular form of neural network
- An **autoencoder** is a neural network whose outputs are its own inputs



- The goal is to minimize **reconstruction error**



# Autoencoders

- Define

$$\mathbf{z} = f(W\mathbf{x}); \quad \hat{\mathbf{x}} = g(V\mathbf{z})$$

- Goal:

$$\min_{W, V} \frac{1}{2N} \sum_{n=1}^N \|\mathbf{x}^{(n)} - \hat{\mathbf{x}}^{(n)}\|^2$$

- If  $g$  and  $f$  are linear

$$\min_{W, V} \frac{1}{2N} \sum_{n=1}^N \|\mathbf{x}^{(n)} - VW\mathbf{x}^{(n)}\|^2$$

- In other words, the optimal solution is PCA.

# Autoencoders: Nonlinear PCA

- What if  $g()$  is not linear?
- Then we are basically doing **nonlinear PCA**
- Some subtleties but in general this is an accurate description

# Comparing Reconstructions



Real data

30-d deep autoencoder

30-d logistic PCA

30-d PCA

# Questions?