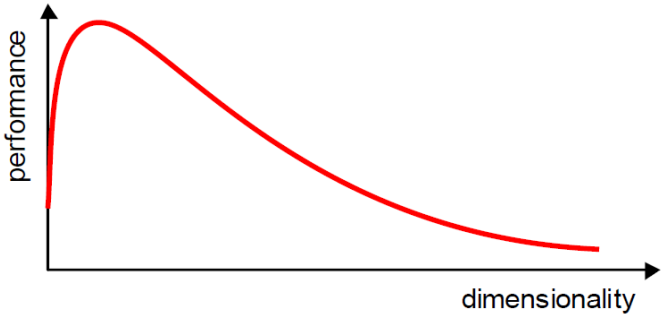
Dimensionality Reduction (PCA ..)

Working with High-Dimensional Data

- High-dimensional data
  - Many applications: text documents, DNA micro-array data
  - Major challenges:
    - Many irrelevant dimensions may mask class and clusters
    - Measure becomes meaningless—e.g. distance ➔ equi-distance
    - Issues like Clusters may exist only in some subspaces
- Methods
  - Feature transformation: only effective if most dimensions are relevant
    - PCA & SVD useful only when features are highly correlated/redundant
  - Feature selection: wrapper or filter approaches
    - useful to find a subspace where the data have nice clusters
  - Subspace-clustering: find clusters in all the possible subspaces
    - CLIQUE, ProClus, and frequent pattern-based clustering

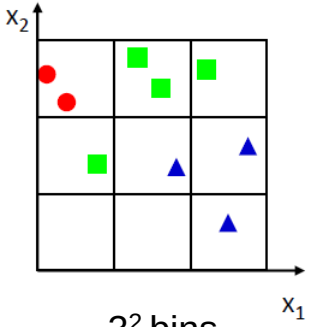
Curse of Dimensionality

- Increasing the number of features will not always improve classification accuracy.
- In practice, the inclusion of more features might actually lead to worse performance.
- The number of training examples required increases exponentially with dimensionality **d** (i.e., $k^{\mathbf{d}}$).

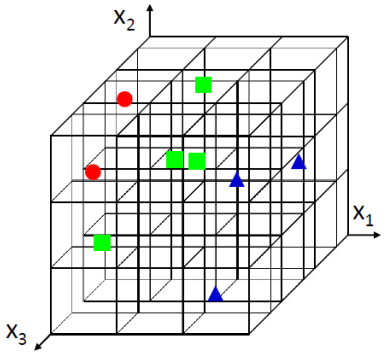


3^1 bins

k=3



3^2 bins

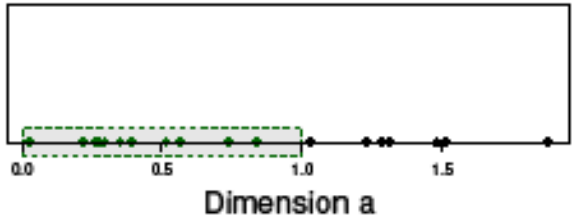


3^3 bins

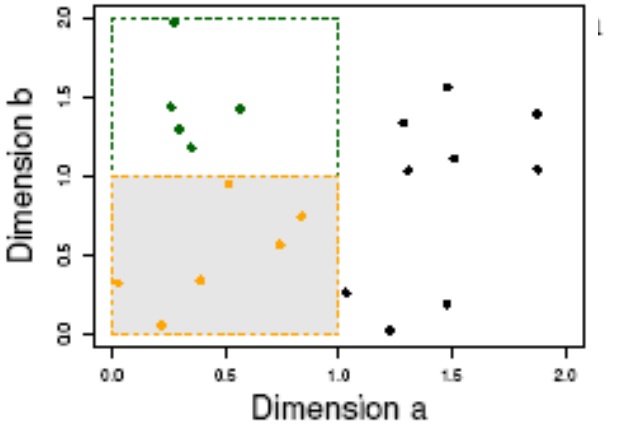
k: number of bins per feature

Curse of Dimensionality

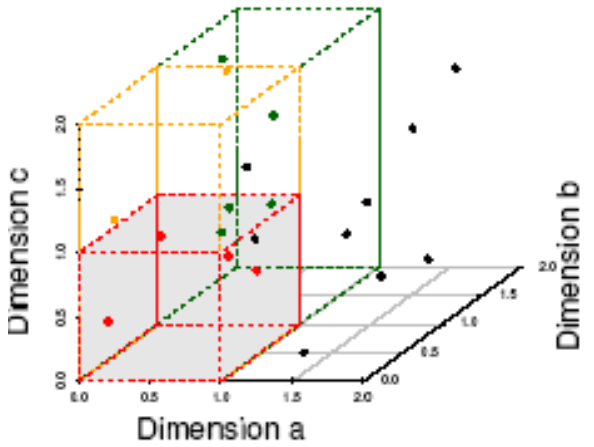
(graphs adapted from Parsons et al. KDD Explorations 2004)



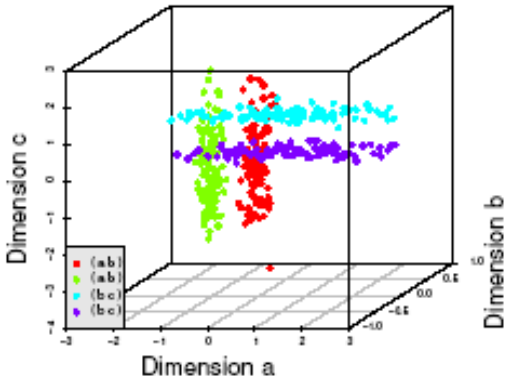
- Data in only one dimension is relatively packed
- Adding a dimension “stretch” the points across that dimension, making them further apart
- Adding more dimensions will make the points further apart—high dimensional data is extremely sparse
- Distance measure becomes meaningless—due to equi-distance



(b) 6 Objects in One Unit Bin



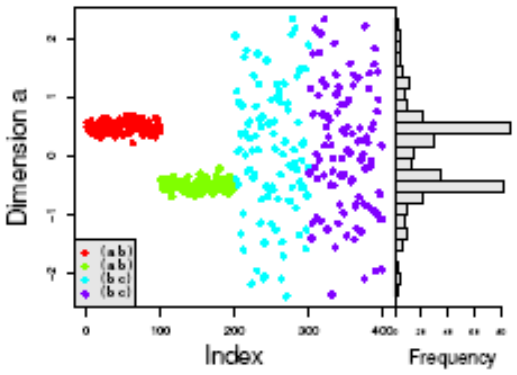
(c) 4 Objects in One Unit Bin



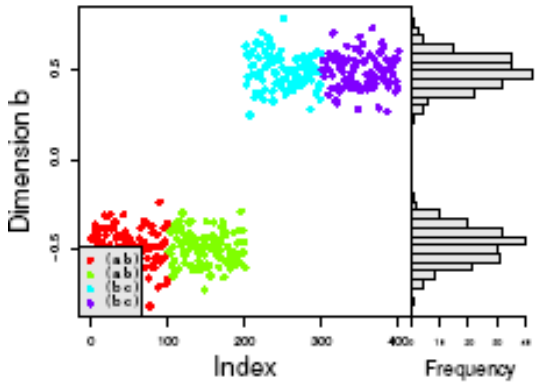
Why Subspace Clustering?

(adapted from Parsons et al. SIGKDD Explorations 2004)

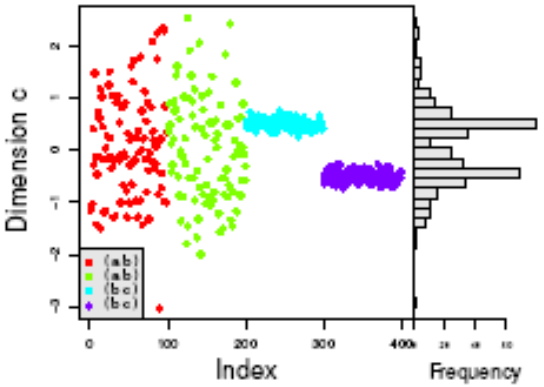
- Clusters may exist only in some subspaces
- Subspace-clustering: find clusters in all the subspaces



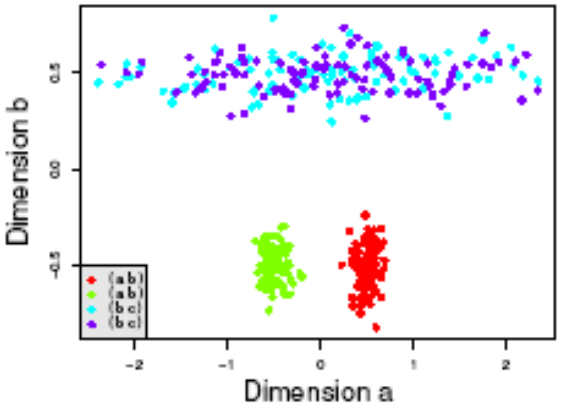
(a) Dimension a



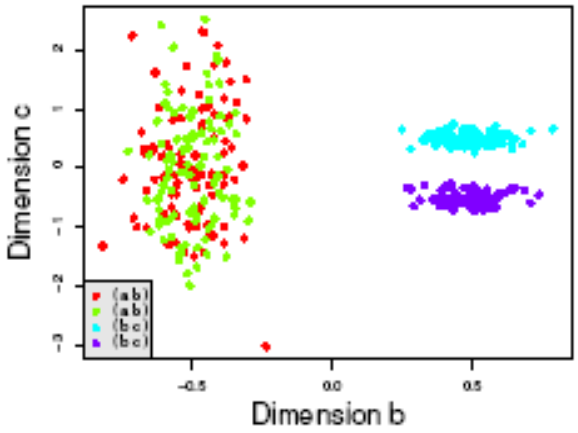
(b) Dimension b



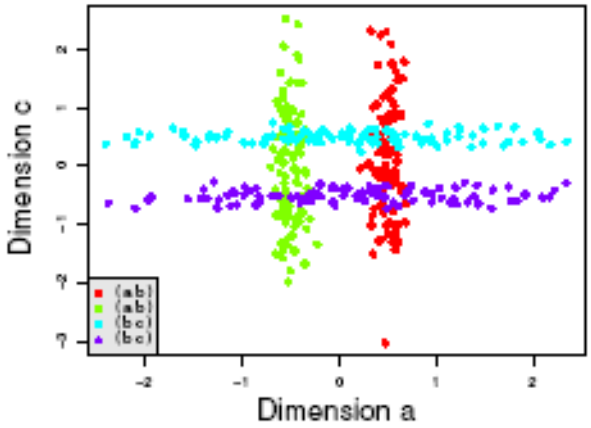
(c) Dimension c



(a) Dims a & b



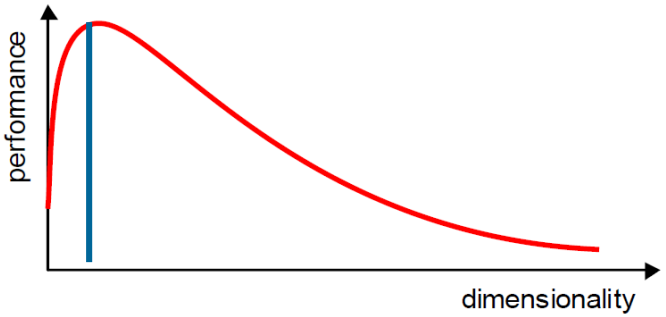
(b) Dims b & c



(c) Dims a & c

Dimensionality Reduction

- What is the objective?
  - Choose an optimum set of features of lower dimensionality to improve classification accuracy.



- Different methods can be used to reduce dimensionality:
  - Feature extraction
  - Feature selection

Dimensionality Reduction (cont'd)

Feature extraction: finds a set of new features (i.e., through some mapping f()) from the existing features.

Feature selection: chooses a subset of the original features.

The mapping f() could be linear or non-linear

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ x_N \end{bmatrix} \xrightarrow{f(\mathbf{x})} \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ y_K \end{bmatrix}$$

K<<N

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ x_N \end{bmatrix} \rightarrow \mathbf{y} = \begin{bmatrix} x_{i_1} \\ x_{i_2} \\ \cdot \\ \cdot \\ x_{i_K} \end{bmatrix}$$

K<<N

Feature Extraction

- Linear combinations are particularly attractive because they are simpler to compute and analytically tractable.

- Given $x \in R^N$, find an K x N matrix T such that:

$$y = Tx \in R^K \text{ where } K<<N$$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ x_N \end{bmatrix} \xrightarrow{T,\ f(\mathbf{x})} \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ y_K \end{bmatrix}$$

This is a projection from the N-dimensional space to a K-dimensional space.

Feature Extraction (cont'd)

- From a mathematical point of view, finding an optimum mapping $\mathbf{y}=f(\mathbf{x})$ is equivalent to optimizing an **objective** criterion.

- Different methods use different objective criteria, e.g.,

  - Minimize Information Loss: represent the data as accurately as possible in the lower-dimensional space.

  - Maximize Discriminatory Information: enhance the class-discriminatory information in the lower-dimensional space.

Feature Extraction (cont'd)

- Popular linear feature extraction methods:
  - Principal Components Analysis (PCA): Seeks a projection that **preserves** as much **information** in the data as possible.
  - Linear Discriminant Analysis (LDA): Seeks a projection that **best discriminates** the data.

- Many other methods:
  - Making features as independent as possible (Independent Component Analysis or ICA).
  - Retaining interesting directions (Projection Pursuit).
  - Embedding to lower dimensional manifolds (Isomap, Locally Linear Embedding or LLE).

Vector Representation

- A vector **x** ϵ R^n can be represented by n components:

$$\mathbf{x}: \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ x_N \end{bmatrix}$$

- Assuming the standard base $<v_1, v_2, \ldots, v_N>$ (i.e., unit vectors in each dimension), x_i can be obtained by projecting **x** along the direction of v_i:

$$x_i = \frac{\mathbf{x}^T v_i}{v_i^T v_i} = \mathbf{x}^T v_i$$

- **x** can be "reconstructed" from its projections as follows:

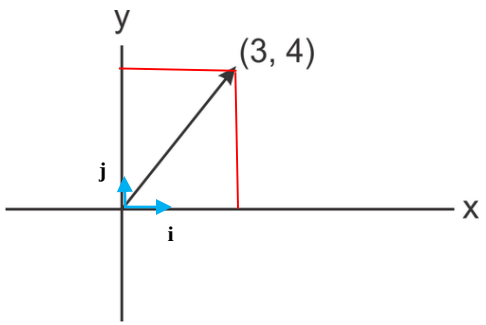
$$\mathbf{x} = \sum_{i=1}^{N} x_i v_i = x_1 v_1 + x_2 v_2 + \ldots + x_N v_N$$

- Since the basis vectors are the same for all x ϵ R^n (standard basis), we typically represent them as a n-component vector.

Vector Representation (cont'd)

- **Example** assuming n=2:

$$\mathbf{x}: \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$$



- Assuming the standard base <v_1=i, v_2=j>, x_i can be obtained by projecting x along the direction of v_i:

$$x_1 = \mathbf{x}^T i = \begin{bmatrix} 3 & 4 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = 3$$

$$x_2 = \mathbf{x}^T j = \begin{bmatrix} 3 & 4 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = 4$$

- **x** can be "reconstructed" from its projections as follows:

$$\mathbf{x} = 3i + 4j$$

Principal Component Analysis (PCA)

- If $\mathbf{x} \in R^N$, then it can be written a linear combination of an orthonormal set of N basis vectors $<v_1, v_2, \ldots, v_N>$ in R^N (e.g., using the standard base):

$$v_i^T v_j = \begin{cases} 1 & if \ \ i = j \\ 0 & otherwise \end{cases}$$

$$\mathbf{x} = \sum_{i=1}^{N} x_i v_i = x_1 v_1 + x_2 v_2 + \ldots + x_N v_N$$

$$where \quad x_i = \frac{\mathbf{x}^T v_i}{v_i^T v_i} = \mathbf{x}^T v_i$$

$$\mathbf{x}: \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ . \\ . \\ x_N \end{bmatrix}$$

- PCA seeks to approximate **x** in a subspace of R^N using a new set of K<<N basis vectors $<u_1, u_2, \ldots, u_K>$ in R^N:

$$\hat{\mathbf{x}} = \sum_{i=1}^{K} y_i u_i = y_1 u_1 + y_2 u_2 + \ldots + y_K u_K \quad \text{(reconstruction)} \qquad where \quad y_i = \frac{\mathbf{x}^T u_i}{u_i^T u_i} = \mathbf{x}^T u_i$$

$$\hat{\mathbf{x}}: \begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ y_K \end{bmatrix}$$

such that $\| \mathbf{x} - \hat{\mathbf{x}} \|$ is minimized!

(i.e., minimize information loss)

Principal Component Analysis (PCA)

- The "optimal" set of basis vectors $<u_1, u_2, \ldots, u_K>$ can be found as follows (we will see why):

(1) Find the eigenvectors u_i of the covariance matrix of the (training) data $\Sigma_{\mathbf{x}}$

$$\Sigma_{\mathbf{x}}\, u_i = \lambda_i\, u_i$$

(2) Choose the K "largest" eigenvectors u_i (i.e., corresponding to the K "largest" eigenvalues λ_i)

$<u_1, u_2, \ldots, u_K>$ correspond to the "optimal" basis!

We refer to the "largest" eigenvectors u_i as principal components.

PCA - Steps

- Suppose we are given $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_M$ (N x 1) vectors

N: # of features

M: # data

Step 1: compute sample mean

$$\bar{\mathbf{x}} = \frac{1}{M}\sum_{i=1}^{M}\mathbf{x}_i$$

Step 2: subtract sample mean (i.e., center data at zero)

$$\Phi_i = \mathbf{x}_i - \bar{\mathbf{x}}$$

Step 3: compute the sample covariance matrix Σ_x

$$\Sigma_{\mathbf{x}} = \frac{1}{M}\sum_{i=1}^{M}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T = \frac{1}{M}\sum_{i=1}^{M}\Phi_i\Phi_i^T = \frac{1}{M}AA^T$$

where $A=[\Phi_1\ \Phi_2\ ...\ \Phi_M]$

i.e., the columns of A are the Φ_i

(N x M matrix)

PCA - Steps

Step 4: compute the eigenvalues/eigenvectors of Σ_x

$$\Sigma_x u_i = \lambda_i u_i$$

where we assume $\lambda_1 > \lambda_2 > \ldots > \lambda_N$

Note : most software packages return the eigenvalues (and corresponding eigenvectors) is decreasing order – if not, you can explicitly put them in this order)

Since Σ_x is symmetric, $\langle u_1, u_2, \ldots, u_N \rangle$ form an orthogonal basis in R^N and we can represent any $\mathbf{x} \in R^N$ as:

$$\mathbf{x} - \bar{\mathbf{x}} = \sum_{i=1}^{N} y_i u_i = y_1 u_1 + y_2 u_2 + \ldots + y_N u_N$$

$$y_i = \frac{(\mathbf{x} - \bar{\mathbf{x}})^T u_i}{u_i^T u_i} = (\mathbf{x} - \bar{\mathbf{x}})^T u_i \qquad if \quad \| u_i \| = 1$$

i.e., this is just a "change" of basis!

$$\mathbf{x} - \bar{\mathbf{x}}: \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ x_N \end{bmatrix} \rightarrow \begin{bmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ y_N \end{bmatrix}$$

Note : most software packages normalize u_i to unit length to simplify calculations; if not, you can explicitly normalize them)

PCA - Steps

Step 5: dimensionality reduction step – approximate **x** using only the first K eigenvectors (K<<N) (i.e., corresponding to the K largest eigenvalues where K is a parameter):

$$\mathbf{x} - \bar{\mathbf{x}} = \sum_{i=1}^{N} y_i u_i = y_1 u_1 + y_2 u_2 + ... + y_N u_N$$

approximate $\mathbf{x}$ *by* $\hat{\mathbf{x}}$
using first K eigenvectors only

$$\hat{\mathbf{x}} - \bar{\mathbf{x}} = \sum_{i=1}^{K} y_i u_i = y_1 u_1 + y_2 u_2 + ... + y_K u_K$$

(reconstruction)

$$\mathbf{x} - \bar{\mathbf{x}}: \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ x_N \end{bmatrix} \rightarrow \begin{bmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ y_N \end{bmatrix} \rightarrow \hat{\mathbf{x}} - \bar{\mathbf{x}}: \begin{bmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ y_K \end{bmatrix}$$

note that if K=N, then [redacted]
(i.e., zero reconstruction error)

What is the Linear Transformation implied by PCA?

- The linear transformation **y = Tx** which performs the dimensionality reduction in PCA is:

$$\hat{\mathbf{x}} - \bar{\mathbf{x}} = \sum_{i=1}^{K} y_i u_i = y_1 u_1 + y_2 u_2 + \ldots + y_K u_K$$

$$(\hat{\mathbf{x}} - \bar{\mathbf{x}}) = U \begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ y_K \end{bmatrix}$$

$$where\ U = [u_1\, u_2 \ldots u_K]\quad N\,x\,K \text{ matrix}$$

i.e., the columns of U are the the first K eigenvectors of Σ_x

$$\begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ y_K \end{bmatrix} = U^T (\hat{\mathbf{x}} - \bar{\mathbf{x}})$$

$T = U^T$ K x N matrix

i.e., the rows of T are the first K eigenvectors of Σ_x

What is the form of Σ_y ?

$$\Sigma_{\mathbf{x}} = \frac{1}{M}\sum_{i=1}^{M}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T = \frac{1}{M}\sum_{i=1}^{M}\Phi_i\Phi_i^T$$

Using diagonalization:

$$\Sigma_{\mathbf{x}} = P\Lambda P^T$$

The columns of P are the eigenvectors of $\Sigma_{\mathbf{x}}$

The diagonal elements of Λ are the eigenvalues of $\Sigma_{\mathbf{x}}$ or the variances

$$\mathbf{y}_i = U^T(\mathbf{x}_i - \bar{\mathbf{x}}) = P^T\Phi_i$$

$$\Sigma_{\mathbf{y}} = \frac{1}{M}\sum_{i=1}^{M}(\mathbf{y}_i - \bar{\mathbf{y}})(\mathbf{y}_i - \bar{\mathbf{y}})^T = \frac{1}{M}\sum_{i=1}^{M}(\mathbf{y}_i)(\mathbf{y}_i)^T = \frac{1}{M}\sum_{i=1}^{M}(P^T\Phi_i)(P^T\Phi_i)^T =$$

$$\frac{1}{M}\sum_{i=1}^{M}(P^T\Phi_i)(\Phi_i^T P) = P^T\left(\frac{1}{M}\sum_{i=1}^{M}\Phi_i\Phi_i^T\right)P = P^T\Sigma_{\mathbf{x}}P = P^T(P\Lambda P^T)P = \Lambda$$

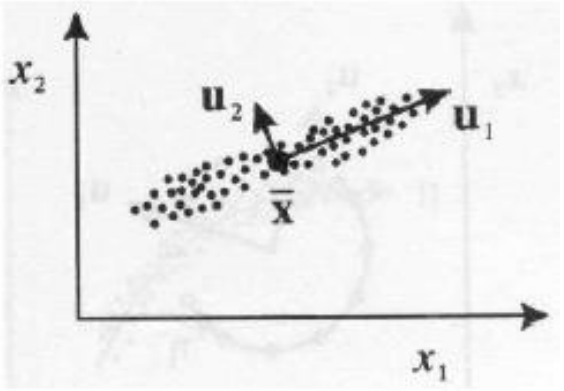
$$\boxed{\Sigma_{\mathbf{y}} = \Lambda}$$

PCA de-correlates the data!

Preserves original variances!

Interpretation of PCA

- PCA chooses the eigenvectors of the covariance matrix corresponding to the largest eigenvalues.
- The eigenvalues correspond to the variance of the data along the eigenvector directions.
- Therefore, PCA projects the data along the directions where the data varies most.
- PCA preserves as much information in the data by preserving as much variance in the data.



u_1: direction of max variance
u_2: orthogonal to u_1

Example

- Compute the PCA of the following dataset:

(1,2),(3,3),(3,5),(5,4),(5,6),(6,5),(8,7),(9,8)

- Compute the sample covariance matrix is:

$$\hat{\Sigma} = \frac{1}{n}\sum_{k=1}^{n}(\mathbf{x}_k - \hat{\boldsymbol{\mu}})(\mathbf{x}_k - \hat{\boldsymbol{\mu}})^t$$

$$\Sigma_x = \begin{bmatrix} 6.25 & 4.25 \\ 4.25 & 3.5 \end{bmatrix}$$

- The eigenvalues can be computed by finding the roots of the characteristic polynomial:

$$\Sigma_x v = \lambda v \Rightarrow |\Sigma_x - \lambda I| = 0$$
$$\Rightarrow \begin{vmatrix} 6.25 - \lambda & 4.25 \\ 4.25 & 3.5 - \lambda \end{vmatrix} = 0$$
$$\Rightarrow \boldsymbol{\lambda_1 = 9.34; \lambda_2 = 0.41}$$

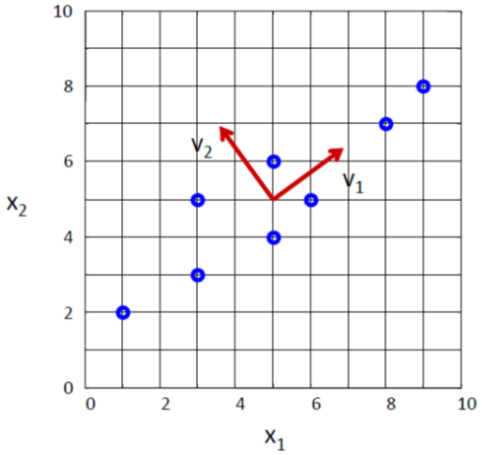
Example (cont'd)

- The eigenvectors are the solutions of the systems:

$$\Sigma_{\mathbf{x}} u_i = \lambda_i u_i$$

$$\begin{bmatrix} 6.25 & 4.25 \\ 4.25 & 3.5 \end{bmatrix} \begin{bmatrix} v_{11} \\ v_{12} \end{bmatrix} = \begin{bmatrix} \lambda_1 v_{11} \\ \lambda_1 v_{12} \end{bmatrix} \Rightarrow \begin{bmatrix} v_{11} \\ v_{12} \end{bmatrix} = \begin{bmatrix} 0.81 \\ 0.59 \end{bmatrix}$$

$$\begin{bmatrix} 6.25 & 4.25 \\ 4.25 & 3.5 \end{bmatrix} \begin{bmatrix} v_{21} \\ v_{22} \end{bmatrix} = \begin{bmatrix} \lambda_2 v_{21} \\ \lambda_2 v_{22} \end{bmatrix} \Rightarrow \begin{bmatrix} v_{21} \\ v_{22} \end{bmatrix} = \begin{bmatrix} -0.59 \\ 0.81 \end{bmatrix}$$



Note: if u_i is a solution, then cu_i is also a solution where $c \neq 0$.

Eigenvectors can be normalized to unit-length using:

$$\hat{v}_i = \frac{v_i}{\| v_i \|}$$

How do we choose K ?

- *K* is typically chosen based on how much information (variance) we want to preserve:

Choose the smallest K that satisfies the following inequality:

$$\frac{\sum_{i=1}^{K} \lambda_i}{\sum_{i=1}^{N} \lambda_i} > T \quad where\ T\ is\ a\ threshold\ (e.g., 0.9)$$

- If T=0.9, for example, we "preserve" 90% of the information (variance) in the data.

- If K=N, then we "preserve" 100% of the information in the data (i.e., just a "change" of basis and $\hat{\mathbf{x}} = \mathbf{x}$)

Approximation Error

- The approximation error (or reconstruction error) can be computed by:

$$\| \mathbf{x} - \hat{\mathbf{x}} \|$$

where $\hat{\mathbf{x}} = \sum_{i=1}^{K} y_i u_i + \bar{\mathbf{x}} = y_1 u_1 + y_2 u_2 + \ldots + y_K u_K + \bar{\mathbf{x}}$

(reconstruction)

- It can also be shown that the approximation error can be computed as follows:

$$\| \mathbf{x} - \hat{\mathbf{x}} \| = \frac{1}{2} \sum_{i=K+1}^{N} \lambda_i$$

Data Normalization

- The principal components are dependent on the ***units*** used to measure the original variables as well as on the ***range*** of values they assume.

- Data should always be normalized prior to using PCA.

- A common normalization method is to transform all the data to have zero mean and unit standard deviation:

$$\frac{x_i - \mu}{\sigma}$$

where μ and σ are the mean and standard deviation of the i-th feature x_i

Application to Images

- The goal is to represent images in a space of lower dimensionality using PCA.
  - Useful for various applications, e.g., face recognition, image compression, etc.
- Given M images of size N x N, first represent each image as a 1D vector (i.e., by stacking the rows together).
  - Note that for face recognition, faces must be centered and of the same size.