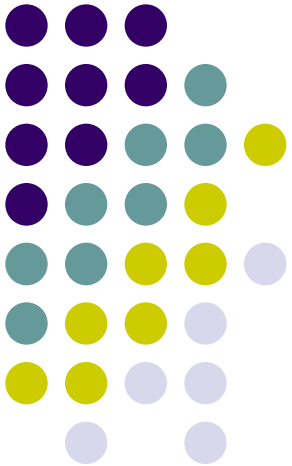
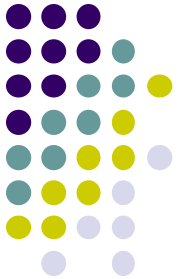
Component Analysis

Hongxin Zhang
zhx@cad.zju.edu.cn

State Key Lab of CAD&CG, ZJU
2021-03-09

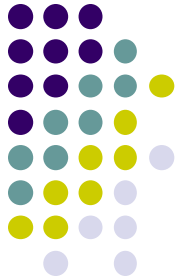


What do you have to know in last lesson?



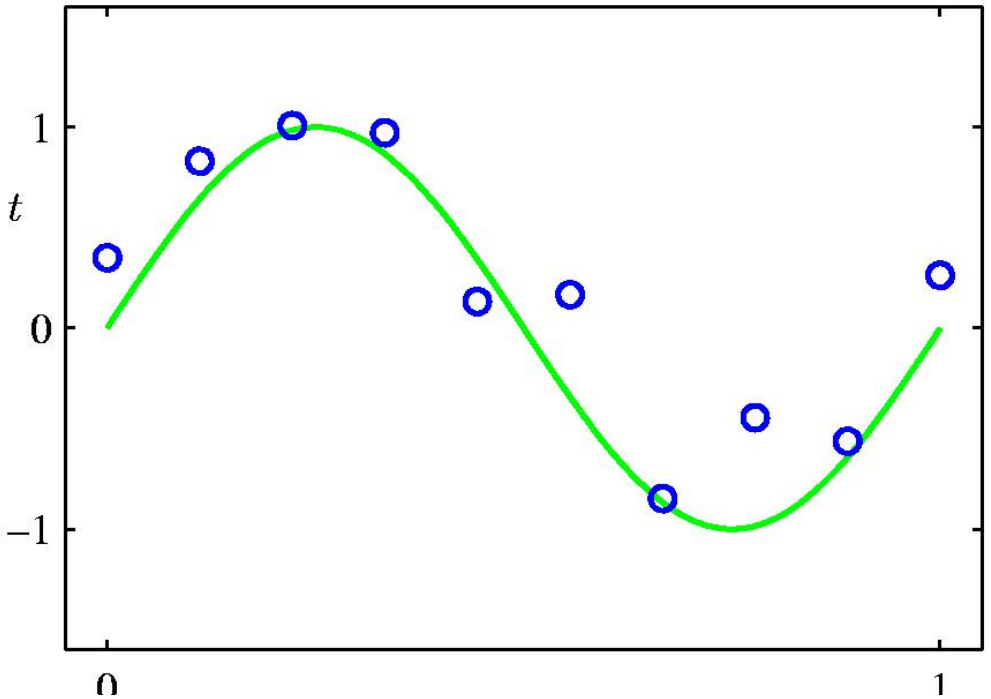
- Concepts
  - Random variable: x
  - (Bayesian) Probability: $P(x)$
    - Condition ~, Joint ~, and Marginal Probability
  - Density function $f(x)$, Distribution, Gaussian (normal) distribution
  - Expectation, Mean, Variance, Moments
  - Likelihood, Prior, Posterior

What do you have to know in last lesson?



- MLE, Bayesian reasoning, Bayes law, MAP
  - Conjugate distribution, beta distribution, gamma function
- Regression
  - Over fitting
  - Regularization

Regression revisit: Polynomial Curve Fitting



$$\boldsymbol{p}(x) = \begin{pmatrix} x^0 \\ x^1 \\ \vdots \\ x^M \end{pmatrix}$$

$$\mathrm{w} = \begin{pmatrix} w^0 \\ w^1 \\ \vdots \\ w^M \end{pmatrix} \qquad \mathrm{y} = \begin{pmatrix} y^0 \\ y^1 \\ \vdots \\ y^N \end{pmatrix}$$

$$y_j = \mathbf{p}(x_j) \cdot \mathbf{w}$$

$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

$$\mathbf{y} = \mathbf{P}(\mathbf{x}) \cdot \mathbf{w}$$

Matrix form

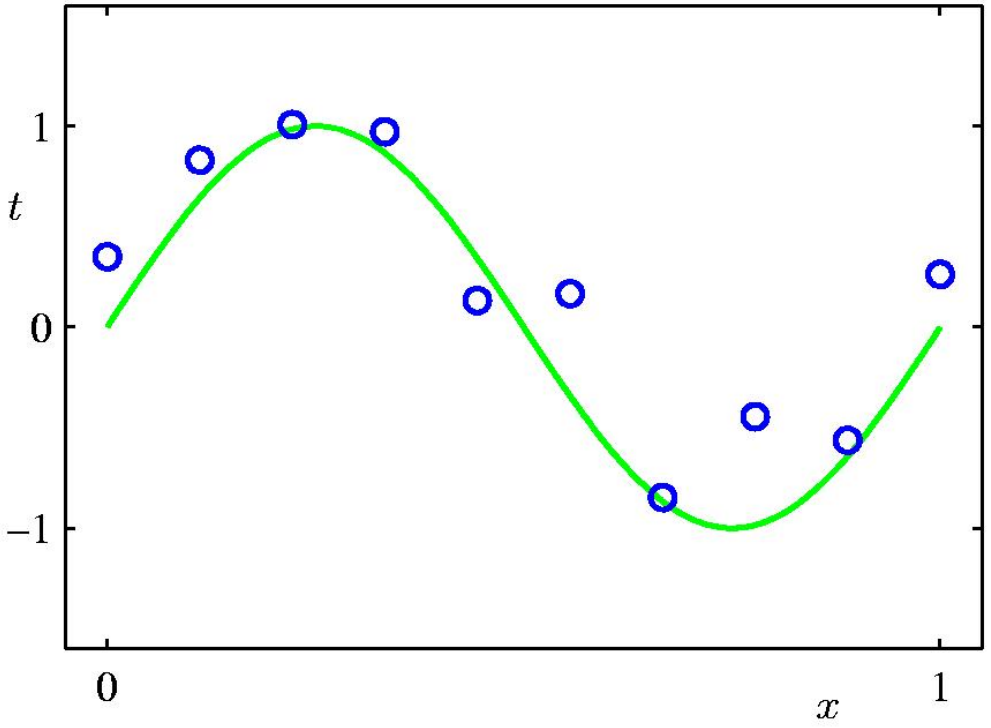
$$\arg\min_{\mathbf{w}} \left\| \mathbf{y} - \mathbf{P}(\mathbf{x}) \cdot \mathbf{w} \right\|^2$$

Least squares

$$\mathbf{w} = (\mathbf{P}^{\mathrm{T}}\mathbf{P})^{-1}\mathbf{P}^{\mathrm{T}}\mathbf{y}$$

Normal equation

Regression revisit: Polynomial Curve Fitting



$$\boldsymbol{h}(x) = \begin{pmatrix} h_0(x) \\ h_1(x) \\ \vdots \\ h_M(x) \end{pmatrix}$$

$$\mathbf{w} = \begin{pmatrix} w^0 \\ w^1 \\ \vdots \\ w^M \end{pmatrix} \qquad \mathbf{y} = \begin{pmatrix} y^0 \\ y^1 \\ \vdots \\ y^M \end{pmatrix}$$

$$y = \mathbf{h}(x) \cdot \mathbf{w}$$

$$\mathbf{w} = (\mathbf{H}^{\mathrm{T}}\mathbf{H})^{-1}\mathbf{H}^{\mathrm{T}}\mathbf{y}$$

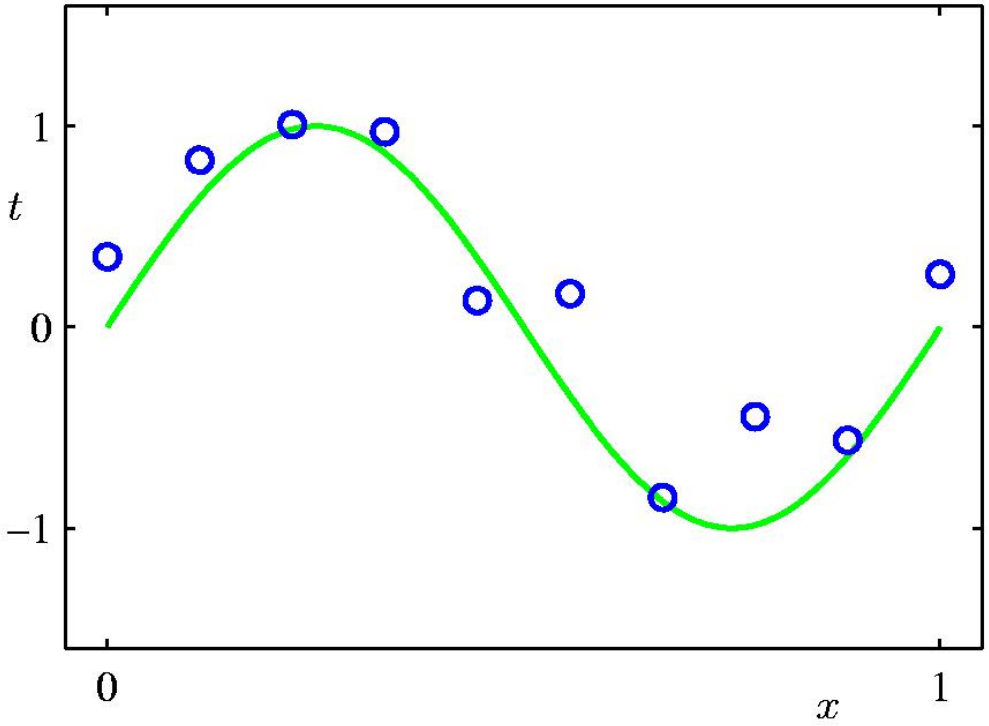
Normal equation

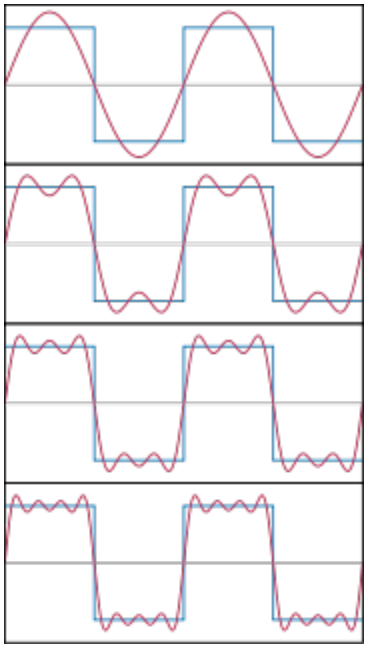
$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

$$y(x, \mathbf{w}) = w_0 + w_1 h_1(x) + w_2 h_2(x) + \ldots + w_M h_M(x) = \sum_{j=0}^{M} w_j h_j(x)$$

Basis function

Regression revisit: Polynomial Curve Fitting





$$y(x,\mathbf{w}) = w_0 + w_1 h_1(x) + w_2 h_2(x) + \ldots + w_M h_M(x) = \sum_{j=0}^{M} w_j h_j(x)$$

Fourier series (orthogonal decomposition)

$$h_j(x) = \begin{cases} \cos(jx/2) & j \text{ is even} \\ \sin((j+1)x/2) & j \text{ is odd} \end{cases}$$

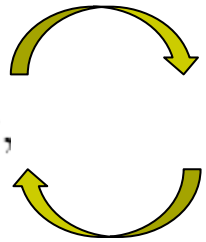
$$\mathbf{w} = (\mathbf{H}^{\mathrm{T}}\mathbf{H})^{-1}\mathbf{H}^{\mathrm{T}}\mathbf{y} = \mathbf{H}^{\mathrm{T}}\mathbf{y}$$

Normal equation

Fourier Transform

- A mathematical operation
  - decomposes a signal (data sequence) into its constituent frequencies

$$\hat{f}(\xi) = \int_{-\infty}^{\infty} f(x)\, e^{-2\pi i x \xi}\, dx, \qquad f(x) = \int_{-\infty}^{\infty} \hat{f}(\xi)\, e^{2\pi i x \xi}\, d\xi,$$



  - Related techniques: Different basis functions
  - (discrete) cosine transform, wavelet transform

- Image / Video compression:
  - JPEG/JPEG 2000, MPEG (1/2/4), H.263/264

JPEG compression: main idea

http://zh.wikipedia.org/zh-cn/JPEG

$$\begin{bmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 55 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\ 79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{bmatrix}$$

RGB Image

→ YCbCr Color Model

$$\begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

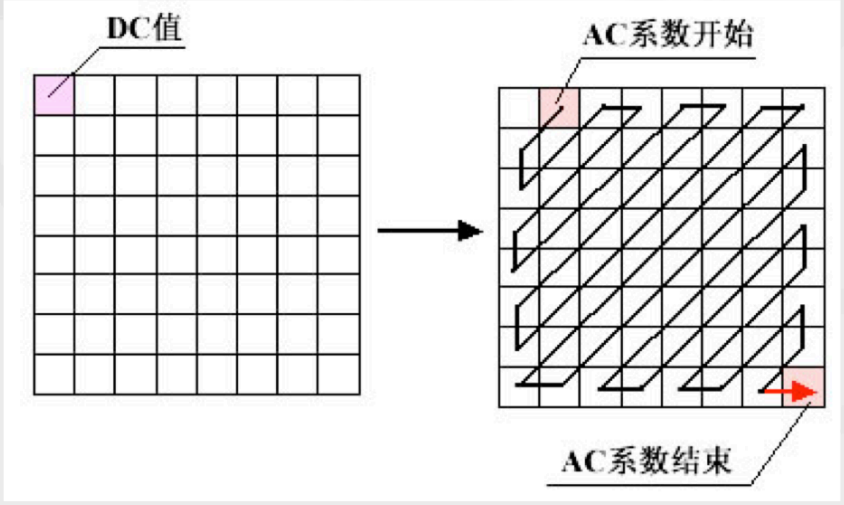
→ 8x8 image blocks

DCT

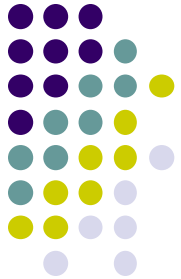
→ Quantization

$$\begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -4 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

frequency-domain representation



Data compression = spectral transforms?

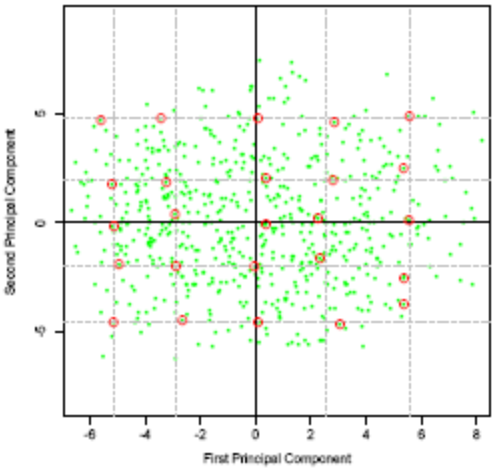


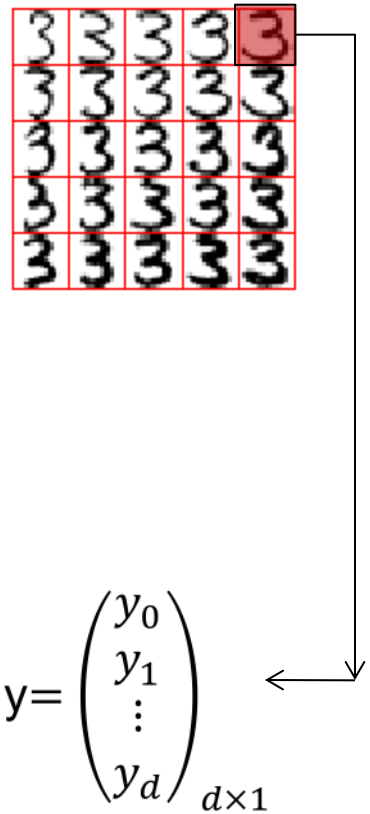
- Goal: choosing suitable transforms, so as to obtain high "information packing".
  - Raw data => Meaningful features.
  - Unsupervised/Automatic methods.

- To exploit and remove information redundancies via transform.

Feature extraction

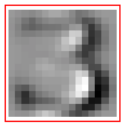
- Data independent
  - DFT, DWT, DCT
    - A single piece of signal
- Spectral analysis





Two-component model has the form

$$\hat{f}(\lambda) = \bar{x} + \lambda_1 v_1 + \lambda_2 v_2$$

$$= \text{  } + \lambda_1 \cdot \text{  } + \lambda_2 \cdot \text{  }.$$

$$y = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_d \end{pmatrix}_{d \times 1}$$

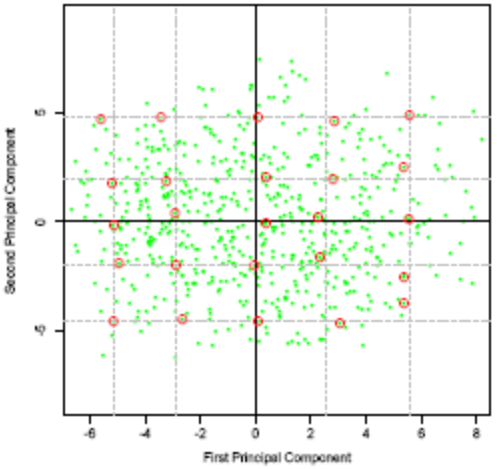
constant

Low frequency component

High frequency component

Feature extraction

- Data independent
  - DFT, DWT, DCT
    - A single piece of signal

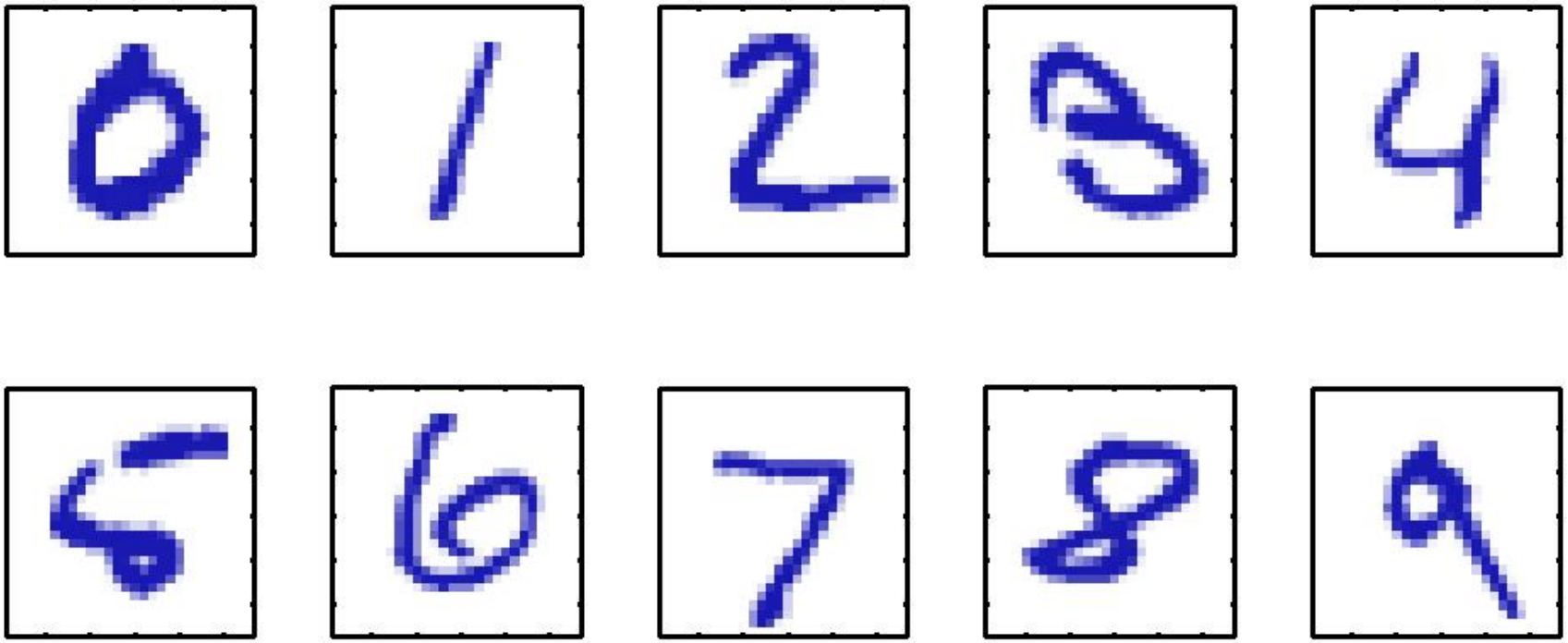


- Data dependent
  - PCA, K-PCA, R-PCA, Factor Analysis, LDA, MDS, …
    - A set of signals (images, motion data, shapes,…)

- Key: define desirable transforms
  - Data driven
  - Raw data => Feature space

Example

Handwritten Digit Recognition

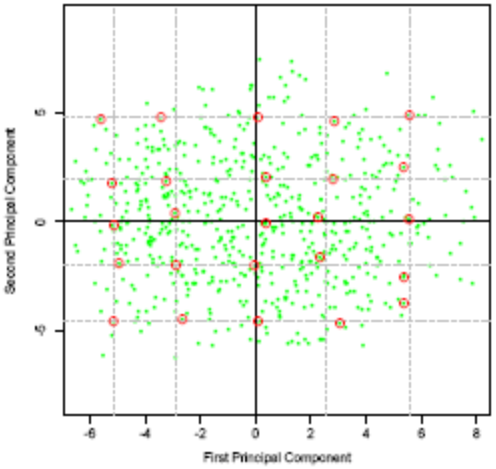


Digit data

$$X = \begin{pmatrix} x_{0,0} & x_{1,0} & x_{2,0} & \cdots & x_{N-1,0} \\ x_{0,1} & x_{1,1} & x_{2,1} & \cdots & x_{N-1,1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{0,p-1} & x_{1,p-1} & x_{2,p-1} & \cdots & x_{N-1,p-1} \end{pmatrix}_{p \times N}$$

130 threes, a subset of 638 such threes and part of the handwritten digit dataset. Each "three" is a 16×16 grayscale image, and the variables x_j, $j = 1, \ldots, 256$ are the grayscale values for each pixel.

Digit: rank-2 model for threes



Two-component model has the form

$$\hat{f}(\lambda) = \bar{x} + \lambda_1 v_1 + \lambda_2 v_2$$

$$= \text{3} + \lambda_1 \cdot \text{3} + \lambda_2 \cdot \text{3}.$$

Here we have displayed the first two principal component directions, v_1 and v_2, as images.

$$\boldsymbol{y} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_d \end{pmatrix}_{d\times 1} \approx \begin{pmatrix} \bar{x}_0 \\ \bar{x}_1 \\ \vdots \\ \bar{x}_d \end{pmatrix}_{d\times 1} + w_1 \begin{pmatrix} h_1(x_0) \\ h_1(x_1) \\ \vdots \\ h_1(x_d) \end{pmatrix}_{d\times 1} + w_2 \begin{pmatrix} h_2(x_0) \\ h_2(x_1) \\ \vdots \\ h_2(x_d) \end{pmatrix}_{d\times 1}$$

w and **X** are both unknown !

$$\boldsymbol{y} \approx \bar{\boldsymbol{x}} + w_1 \boldsymbol{x}_1 + w_2 \boldsymbol{x}_2 =: \boldsymbol{X}^{\mathrm{T}} \boldsymbol{w}$$

$$\arg\min_{\boldsymbol{X},\boldsymbol{w}} \left\| \boldsymbol{y} - \boldsymbol{X}^{\mathrm{T}} \boldsymbol{w} \right\|$$

Apply to data set

$$\arg\min_{\mathbf{X},\mathbf{w}} \left\| \mathbf{y} - \mathbf{X}^{\mathrm{T}}\mathbf{w} \right\|$$

$$\arg\min_{\mathbf{X},\mathbf{W}} \left\| \mathbf{Y} - \mathbf{X}^{\mathrm{T}}\mathbf{W} \right\|$$

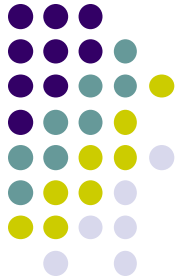
$$\boldsymbol{Y} = \left(\boldsymbol{y}_1 \quad \boldsymbol{y}_2 \quad \cdots \quad \boldsymbol{y}_N \right)_{d\times N} \qquad \boldsymbol{y}_i \in E^d$$

$$\boldsymbol{W} = \left(\boldsymbol{w}_1 \quad \boldsymbol{w}_2 \quad \cdots \quad \boldsymbol{w}_N \right)_{p\times N} \qquad \boldsymbol{w}_i \in E^p$$

$$\boldsymbol{X} = \left(\boldsymbol{x}_1 \quad \boldsymbol{x}_2 \quad \cdots \quad \boldsymbol{x}_d \right)_{p\times d}$$

- d: data dimension
- p: feature dimension $\qquad d \gg p$
- N: number of data examples

A data driven problem

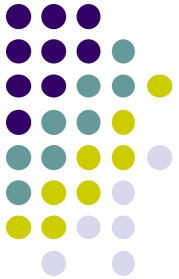


- Given data Y
  - find transform x as well as feature w

$$\arg\min_{X,W} \left\| Y - X^{\mathrm{T}} W \right\|_F$$

  - Straightforward solution:
    - Fix w, solve X by LSQ; then fix X, solve w LSQ …
    - Not good!

Solution:
Singular Value Decomposition



Let $\hat{Y}$ be the centered $d \times N$ data matrix (assume $N > d$).

$$\sum_i \mathbf{y}_i = \mathbf{0} \qquad \mathbf{Y} = \begin{pmatrix} y_{1,1} & y_{2,1} & y_{3,1} & \cdots & y_{N,1} \\ y_{1,2} & y_{2,2} & y_{3,2} & \cdots & y_{N,2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ y_{1,d} & y_{2,d} & y_{3,d} & \cdots & y_{N,d} \end{pmatrix}_{d \times N} = \mathbf{USV}^{\mathrm{T}}$$

$\boldsymbol{y}_2$

Singular values

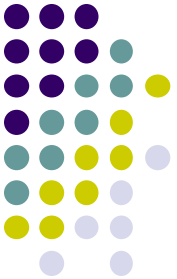
Unitary Matrices

is the SVD of $\hat{\mathbf{Y}}$, where

- ***U*** **is *d*×*d* orthogonal, the left singular vectors.**
- ***V*** **is *N*×*N* orthogonal, the right singular vectors.**
- ***S*** **is *d*×*N* diagonal, with $s_1 \geq s_2 \geq \ldots \geq s_d \geq 0$, the singular values.**

- ✓ **The SVD always exists, and is unique up to signs.**
- ✓ **The columns of V are the principal components**

Solution:
Singular Value Decomposition



Let $\hat{Y}$ be the centered $d \times N$ data matrix (assume $N > d$).

$$\sum_i \mathbf{y}_i = \mathbf{0}$$

$$\hat{\mathbf{Y}} = \begin{pmatrix} y_{1,1} & y_{2,1} & y_{3,1} & \cdots & y_{N,1} \\ y_{1,2} & y_{2,2} & y_{3,2} & \cdots & y_{N,2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ y_{1,d} & y_{2,d} & y_{3,d} & \cdots & y_{N,d} \end{pmatrix}_{d \times N} = \mathbf{USV}^{\mathrm{T}}$$

$\mathbf{y}_2$

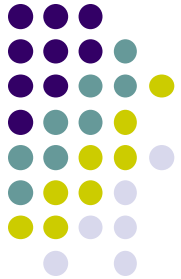
Singular values

Unitary Matrices

$$\arg\min_{\mathbf{X},\mathbf{W}} \left\| \mathbf{Y} - \mathbf{X}^{\mathrm{T}}\mathbf{W} \right\|$$

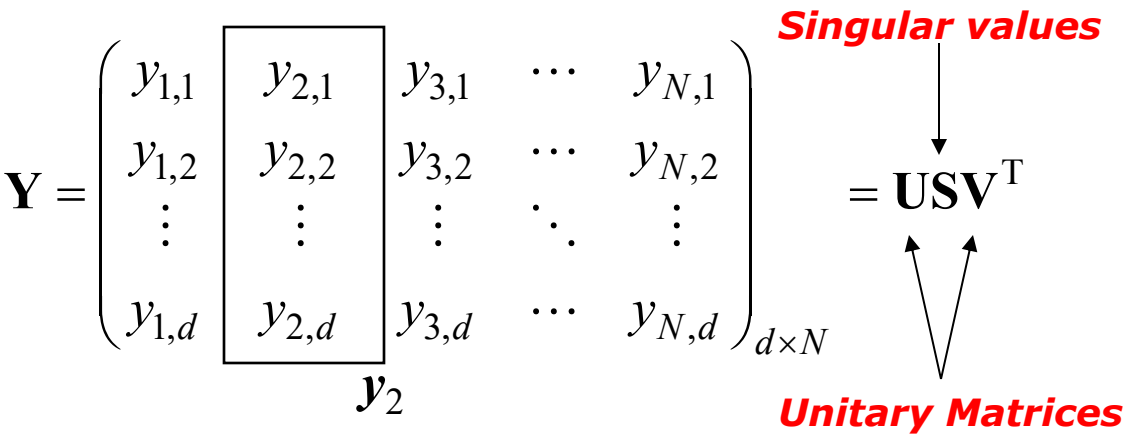
$$\mathbf{X} = \mathbf{U}^{\mathbf{T}}, \mathbf{W} = \mathbf{SV}^{\mathbf{T}}$$

Simple example: Singular Value Decomposition



- From wiki
  - http://en.wikipedia.org/wiki/Singular_value_decomposition

Why SVD works?

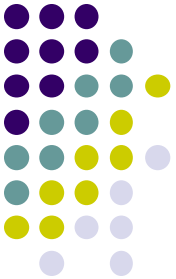


Eckart–Young theorem

Let s_p be s with all but the first p diagonal elements set to zero. Then $\hat{\mathbf{Y}}_p = \mathbf{U}\mathbf{S}_p\mathbf{V}^T$ solves

$$\min_{rank(\hat{\mathbf{Y}}_p)=p} \left\| \hat{\mathbf{Y}} - \hat{\mathbf{Y}}_q \right\|$$

Why SVD works? (cont.)

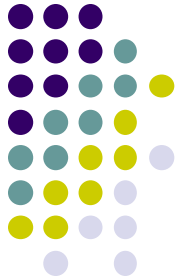


- Low-rank matrix approximation
  - Find $\tilde{\mathbf{Y}}$, $\min \left\| \mathbf{Y} - \tilde{\mathbf{Y}} \right\|_F$ s.t. $rank(\tilde{\mathbf{Y}}) = p$
  - Quick proof:
    - Equivalent to $\min \left\| \mathbf{S} - \mathbf{U}^{\mathrm{T}} \tilde{\mathbf{Y}} \mathbf{V} \right\|_F$
    - Matrix $\mathbf{T} = \mathbf{U}^{\mathrm{T}} \tilde{\mathbf{Y}} \mathbf{V} = \mathrm{diag}(t_1, ..., t_p)$ must be diagonal.

$$\min \left\| \mathbf{S} - \mathbf{T} \right\|_F^2 = \sum_{i=1}^{p} (s_i - t_i)^2 + \sum_{i=p+1}^{d} s_i^2$$

    - It follows that $t_i = s_i, i = 1, ..., p$, $\tilde{\mathbf{Y}} = \mathbf{U} \tilde{\mathbf{S}} \mathbf{V}^{\mathrm{T}}$

Why it works? (cont.)

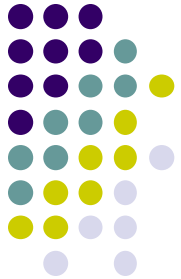


- Matrix decomposition (the inductive method)
  - When p=1

$$\tilde{\mathbf{Y}} = s_1 \mathbf{u}_1 \mathbf{v}_1^{\mathrm{T}} = s_1 \begin{pmatrix} u_{11} \\ u_{12} \\ \vdots \\ u_{1d} \end{pmatrix} \begin{pmatrix} v_{11} & v_{12} & \cdots & v_{1N} \end{pmatrix}$$

  - In general: $\tilde{\mathbf{Y}} = \sum_{i=1}^{p} s_i \mathbf{u}_i \mathbf{v}_i^{\mathrm{T}}$

How to compute



- Matrix decomposition:
  - Mainly used in matlab, clapack:

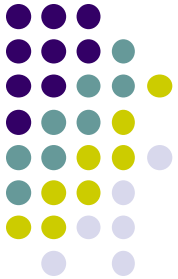
- Relation to eigenvalue decomposition:

$$\mathbf{Z} := \mathbf{Y}^T\mathbf{Y} = \mathbf{V}\mathbf{S}\mathbf{U}^T\mathbf{U}\mathbf{S}\mathbf{V}^T = \mathbf{V}\mathbf{S}^2\mathbf{V}^T$$

$$\mathbf{Z}\,\mathbf{V} = \mathbf{V}\,\mathbf{S}^2$$

- The columns of V (right singular vectors) are eigenvectors of Z

Compute eigen~ (vectors and values)

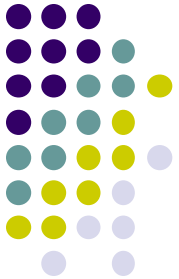


- Eigen problem $\mathbf{ZV} = \lambda \mathbf{V}$
- Characteristic polynomial

$$\det(\mathbf{Z} - \lambda \mathbf{I}) = 0$$

- Iterative method (when matrix is very huge)
  - Simplest method: $\mathbf{v}^{(n+1)} = \mathbf{Z}\ \mathbf{v}^{(n)}$
  - Mostly used method: Lanczos method
    - http://en.wikipedia.org/wiki/Lanczos_algorithm

Principle component analysis



- Given data $\mathbf{Y}$
  - find transform $\mathbf{X}$ as well as feature $\mathbf{W}$

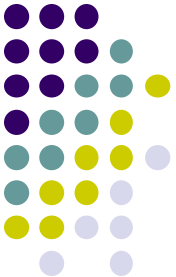
$$\arg\min_{\mathbf{X},\mathbf{W}} \left\| \mathbf{Y} - \mathbf{X}^{\mathrm{T}}\mathbf{W} \right\|_F \qquad \mathbf{X} = \mathbf{U}^{\mathrm{T}}, \mathbf{W} = \mathbf{S}\mathbf{V}^{\mathrm{T}}$$

  - Given a new data $\mathbf{y}_{\mathrm{new}}$ we fix transform $\mathbf{X}$, then:

$$\mathbf{w}_{new} = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_p \end{pmatrix} = \begin{pmatrix} \mathbf{u}_1 \cdot \mathbf{y}_{new} \\ \mathbf{u}_2 \cdot \mathbf{y}_{new} \\ \vdots \\ \mathbf{u}_p \cdot \mathbf{y}_{new} \end{pmatrix}$$

$\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_p$ are principle components (rows of $\mathbf{X}$)

PCA: An Intuitive Approach



Let us say we have $\mathbf{x}_i$, i=1…N data points in d dimensions (d is large)

If we want to represent the data set by a single point $\mathbf{x}_0$, then

$$\mathbf{x}_0 = \mathbf{m} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i$$ ← Sample mean

Can we justify this choice mathematically?

$$J_0(\mathbf{x}_0) = \sum_{i=1}^{N}\|\mathbf{x}_i - \mathbf{x}_0\|^2$$

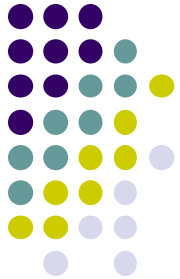
通过该案例
理解变分的概念

Variational Methods

It turns out that if you minimize J_0, you get the above solution, *viz*., sample mean

Source: Chapter 3 of [DHS]

PCA: An Intuitive Approach…



Representing the data set $\mathbf{x}_i$, i=1…N by its mean is quite uninformative

So let's try to represent the data by a straight line of the form:

$$\mathbf{x} = \mathbf{m} + w\mathbf{e}$$

This is equation of a straight line that says that it passes through **m**

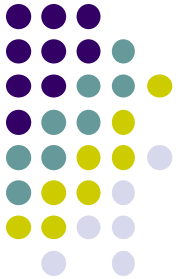
e is a unit vector along the straight line

And the signed distance of a point **x** from **m** is w

The training points projected on this straight line would be

$$\mathbf{x}_i = \mathbf{m} + w_i\mathbf{e}, \quad i = 1...N$$

PCA: An Intuitive Approach…



Let's now determine w_i's

$$J_1(w_1, w_2, \ldots, w_N, \mathbf{e}) = \sum_{i=1}^{N} \left\| \mathbf{m} + w_i \mathbf{e} - \mathbf{x}_i \right\|^2$$

$$= \sum_{i=1}^{N} w_i^2 \, \| \mathbf{e} \|^2 - 2 \sum_{i=1}^{N} w_i \mathbf{e}^T (\mathbf{x}_i - \mathbf{m}) + \sum_{i=1}^{N} \| \mathbf{x}_i - \mathbf{m} \|^2$$

$$= \sum_{i=1}^{N} w_i^2 - 2 \sum_{i=1}^{N} w_i \mathbf{e}^T (\mathbf{x}_i - \mathbf{m}) + \sum_{i=1}^{N} \| \mathbf{x}_i - \mathbf{m} \|^2$$

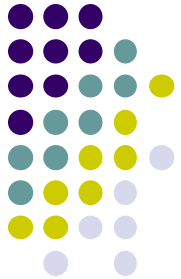
Partially differentiating with respect to w_i we get: $w_i = \mathbf{e}^{\mathrm{T}} (\mathbf{x}_i - \mathbf{m})$

Plugging in this expression for w_i in J_1 we get:

$$J_1(\mathbf{e}) = -\sum_{i=1}^{N} \mathbf{e}^T (\mathbf{x}_i - \mathbf{m})(\mathbf{x}_i - \mathbf{m})^T \mathbf{e} + \sum_{i=1}^{N} \| \mathbf{x}_i - \mathbf{m} \|^2 = -\mathbf{e}^T S \mathbf{e} + \sum_{i=1}^{N} \| \mathbf{x}_i - \mathbf{m} \|^2$$

where $S = \sum_{i=1}^{N} (\mathbf{x}_i - \mathbf{m})(\mathbf{x}_i - \mathbf{m})^T$ is called the scatter matrix

PCA: An Intuitive Approach…



So minimizing J_1 is equivalent to maximizing: $\mathbf{e}^T S\mathbf{e}$

Subject to the constraint that **e** is a unit vector: $\mathbf{e}^T\mathbf{e} = 1$

Use Lagrange multiplier method to form the objective function:

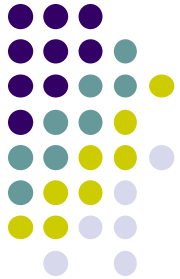
$$\mathbf{e}^T S\mathbf{e} - \lambda(\mathbf{e}^T\mathbf{e} - 1)$$

Differentiate to obtain the equation:

$$2S\mathbf{e} - 2\lambda\mathbf{e} = \mathbf{0} \; or \; \mathbf{Se} = \lambda\mathbf{e}$$

Solution is that **e** is the eigenvector of S corresponding to the largest eigenvalue

PCA: An Intuitive Approach…



The preceding analysis can be extended in the following way.

Instead of projecting the data points on to a straight line, we may now want to project them on a d-dimensional plane of the form:

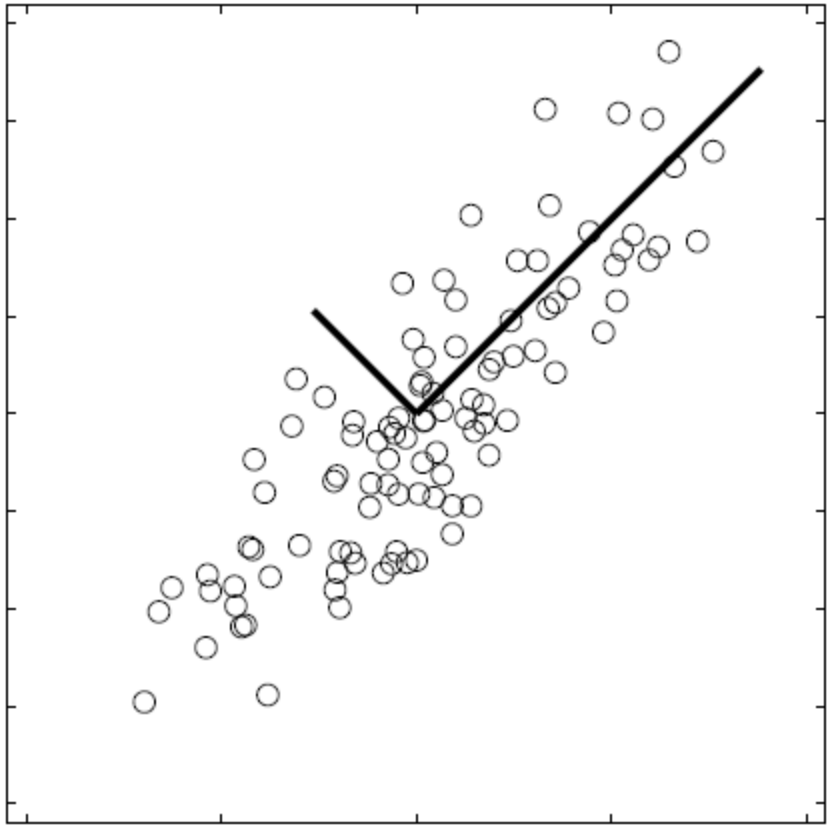
$$\mathbf{x} = \mathbf{m} + w_1\mathbf{e}_1 + \cdots + w_p\mathbf{e}_p$$

d is much smaller than the original dimension *p*

In this case one can form the objective function: $J_p = \sum_{i=1}^{N} \| (\mathbf{m} + \sum_{k=1}^{p} w_{ik}\mathbf{e}_k) - \mathbf{x}_i \|^2$

It can also be shown that the vectors $\mathbf{e}_1$, $\mathbf{e}_2$, …, $\mathbf{e}_p$ are *p* eigenvectors corresponding to p largest eigen values of the scatter matrix $S = \sum_{i=1}^{N} (\mathbf{x}_i - \mathbf{m})(\mathbf{x}_i - \mathbf{m})^T$

PCA: Visualization



Data points are represented in a rotated orthogonal coordinate system: the origin is the mean of the data points and the axes are provided by the eigenvectors.

Computation of PCA

- In practice we compute PCA via SVD (singular value decomposition)
- Form the centered data matrix:

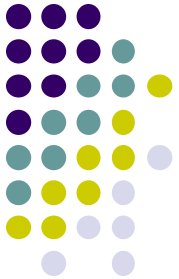
$$\mathbf{X}_{d\times N} = \left[(\mathbf{x}_1 - \mathbf{m}) \ldots\ldots (\mathbf{x}_N - \mathbf{m})\right]$$

- Compute its SVD:

$$\mathbf{X} = \mathbf{U}_{d,d}\mathbf{D}_{d,d}(\mathbf{V}_{N,d})^{\mathrm{T}}$$

- **U** and **V** are orthogonal (unitary) matrices,
- **D** is a diagonal (singular value) matrix

Computation of PCA…



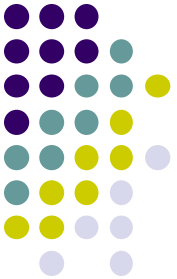
- Note that the scatter matrix can be written as:

$$\mathbf{S} = (\mathbf{X} - \mathbf{m})(\mathbf{X} - \mathbf{m})^{\mathrm{T}} = \mathbf{U}\mathbf{D}^{2}\mathbf{U}^{\mathrm{T}}$$

- So the eigenvectors of *S* are the columns of *U* and the eigenvalues are the diagonal elements of D^2
- Take only a few significant eigenvalue-eigenvector pairs *p*<<d; The new reduced dimension representation becomes:

$$\tilde{\mathbf{x}}_i = \mathbf{m} + \mathbf{U}_{d,p}(\mathbf{U}_{d,p})^{\mathrm{T}}(\mathbf{x}_i - \mathbf{m})$$

Computation of PCA…



- Sometimes we are given only a few high dimensional data points, *i.e.*, $d >> N$ (mostly in image processing)
- In such cases compute the SVD of X^T:

$$\mathbf{X}^T = \mathbf{V}_{N,N}\mathbf{D}_{N,N}(\mathbf{U}_{p,N})^T$$

- So that we get:

$$\mathbf{X} = \mathbf{U}_{p,N}\mathbf{D}_{N,N}(\mathbf{V}_{N,N})^T$$

- Then, proceed as before, choose only $p < N$ significant eigenvalues for data representation:

$$\tilde{\mathbf{x}}_i = \mathbf{m} + \mathbf{U}_{p,d}(\mathbf{U}_{p,d})^T(\mathbf{x}_i - \mathbf{m})$$

PCA: A Gaussian Viewpoint

$$\mathbf{x} \sim \frac{1}{\sqrt{(2\pi)^d \,|\Sigma|}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\right) = \prod_{i=1}^{d} \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left(-\frac{(\mathbf{u}_i^T(\mathbf{x}-\boldsymbol{\mu}))^2}{2\sigma_i^2}\right),$$

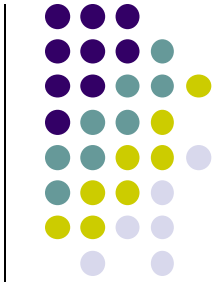
where the covariance matrix Σ estimated from the scatter matrix as $(1/N)S$
$\mathbf{u}$'s and sigma's are respectively eigenvectors and eigenvalues of S.

If *d* is large, then we need a even larger number of data points to estimate the covariance matrix. So, when a limited number of training data points is available the estimation of the covariance matrix goes quite wrong. This is known as curse of dimensionality in this context.

To combat curse of dimensionality, we discard smaller eigenvalues and be content with:

$$\mathbf{x} \sim \prod_{i=1}^{p} \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left(-\frac{(\mathbf{u}_i^T(\mathbf{x}-\boldsymbol{\mu}))^2}{2\sigma_i^2}\right), \text{ where } p < \min(d, N)$$

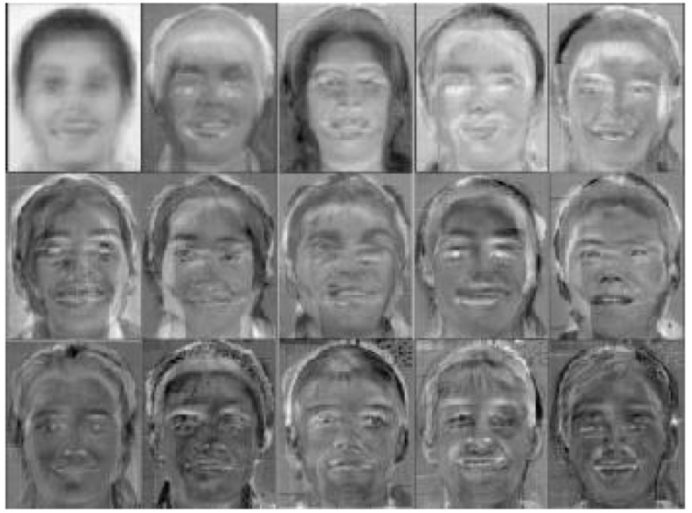
PCA Examples



- Image compression example
- Novelty detection example
- Face recognition

PCA: example Eigenfaces

- G. D. Finlayson, B. Schiele & J. Crowley. Comprehensive colour image normalization. ECCV 98 pp. 475~490.



- Eigen-X, ☺

Far beyond PCA

- Human bodies in 3D
- Human body representation in image

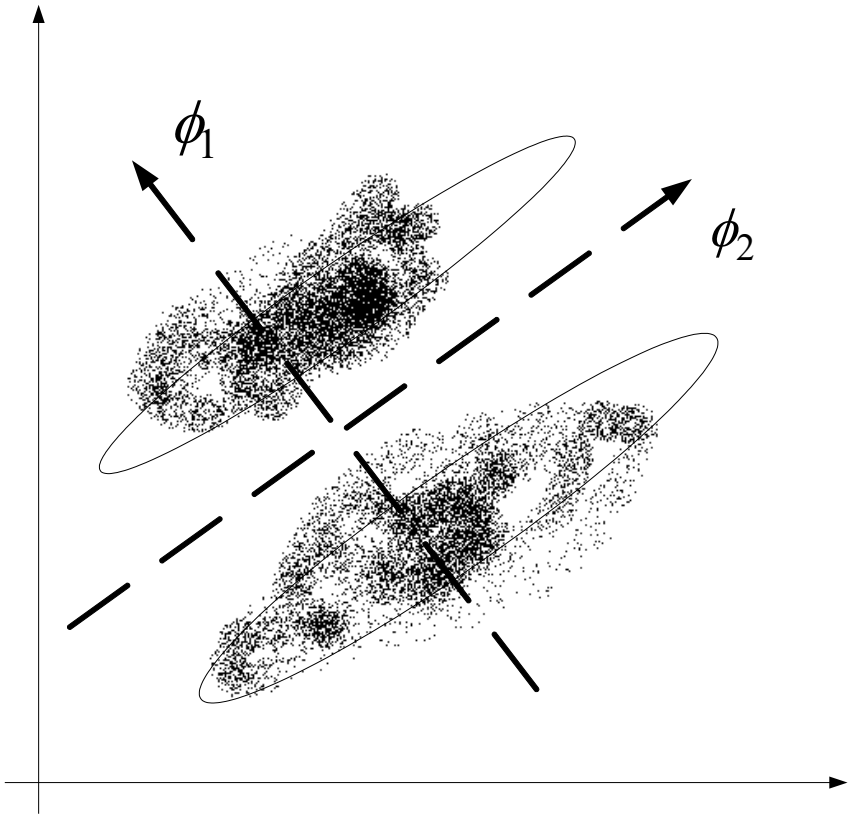
http://grail.cs.washington.edu/projects/digital-human/pub/allen03space.avi

PCA and dimensional reduction

- Space transform via SVD
  - $\mathbf{Y} \rightarrow \mathbf{W}$
- Dimension:
  - $d, N >> p$

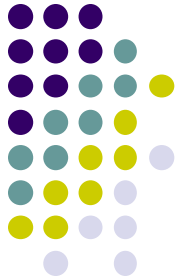
- Representation
- Errors …

Problems of PCA



- Only suitable for normal distributed data
- More consideration
  - ICA: Independent components.
  - K-PCA: Nonlinear
  - NMF: Non-negative
  - Tensor …

Kernel PCA



- Assumption behind PCA is that the data points **x** are multivariate Gaussian
- Often this assumption does not hold
- However, it may still be possible that a transformation *phi*(**x**) is still Gaussian, then we can perform PCA in the space of *phi*(**x**)
- Kernel PCA performs this PCA; however, because of “kernel trick,” it never computes the mapping *phi*(**x**) explicitly!

KPCA: Basic Idea

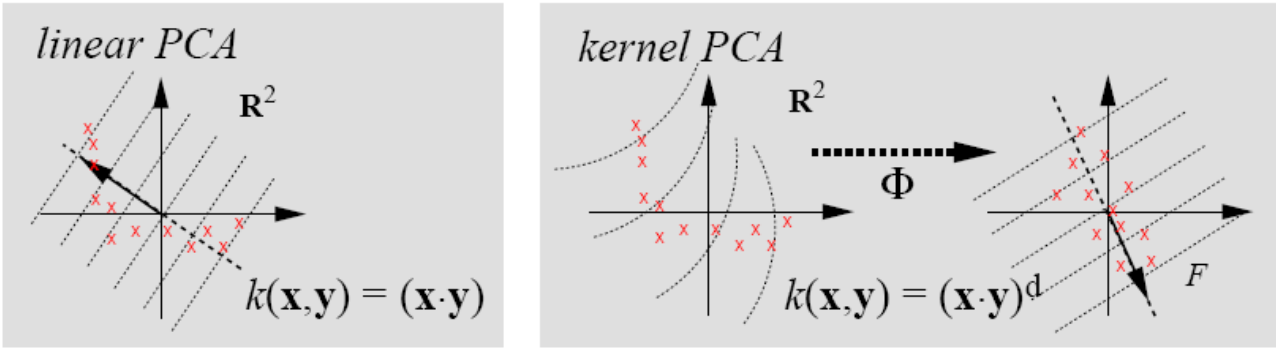
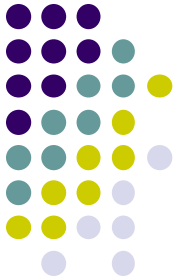


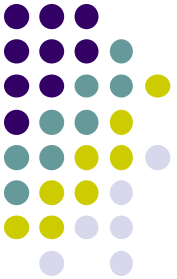
Fig. 1. Basic idea of kernel PCA: by using a nonlinear kernel function k instead of the standard dot product, we implicitly perform PCA in a possibly high–dimensional space F which is nonlinearly related to input space. The dotted lines are contour lines of constant feature value.

Kernel PCA Formulation



- We need the following fact:
- Let **v** be an eigenvector of the scatter matrix: $S = \sum_{i=1}^{N} \mathbf{x}_i \mathbf{x}_i^T$
- Then **v** belongs to the linear space spanned by the data points $\mathbf{x}_i \ \ (i=1, 2, \ldots N)$.
- Proof: $S\mathbf{v} = \lambda \mathbf{v} \Rightarrow \mathbf{v} = \frac{1}{\lambda} \sum_{i=1}^{N} \mathbf{x}_i (\mathbf{x}_i^T \mathbf{v}) = \sum_{i=1}^{N} w_i \mathbf{x}_i$

Kernel PCA Formulation…



- Let *C* be the scatter matrix of the centered mapping phi(**x**):

$$C = \sum_{i=1}^{N} \varphi(\mathbf{x}_i)\varphi(\mathbf{x}_i)^{\mathrm{T}} \qquad S = \sum_{i=1}^{N} \mathbf{x}_i \mathbf{x}_i^T$$

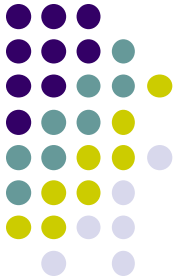
- Let **L** be an eigenvector of *C*, then **L** can be written as a linear combination:

$$\mathbf{L} = \sum_{k=1}^{N} w_k \varphi(\mathbf{x}_k)$$

- Also, we have: $\mathbf{CL} = \lambda \mathbf{L}$

- Combining, we get: $(\sum_{i=1}^{N} \varphi(\mathbf{x}_i)\varphi(\mathbf{x}_i)^T)(\sum_{k=1}^{N} w_k \varphi(\mathbf{x}_k)) = \lambda \sum_{k=1}^{N} w_k \varphi(\mathbf{x}_k)$

Kernel PCA Formulation…



$$(\sum_{i=1}^{N}\varphi(\mathbf{x}_i)\varphi(\mathbf{x}_i)^T)(\sum_{k=1}^{N}w_k\varphi(\mathbf{x}_k)) = \lambda\sum_{k=1}^{N}w_k\varphi(\mathbf{x}_k) \Rightarrow$$

$$\sum_{i=1}^{N}\sum_{k=1}^{N}\varphi(\mathbf{x}_i)\varphi(\mathbf{x}_i)^T\varphi(\mathbf{x}_k)w_k = \lambda\sum_{k=1}^{N}w_k\varphi(\mathbf{x}_k) \Rightarrow$$

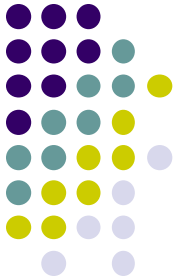
$$\sum_{i=1}^{N}\sum_{k=1}^{N}\varphi(\mathbf{x}_l)^T\varphi(\mathbf{x}_i)\varphi(\mathbf{x}_i)^T\varphi(\mathbf{x}_k)w_k = \lambda\sum_{k=1}^{N}w_k\varphi(\mathbf{x}_l)^T\varphi(\mathbf{x}_k),\ l = 1,2,\ldots,N \Rightarrow$$

$$K^2\mathbf{w} = \lambda K\mathbf{w} \Rightarrow$$

$$K\mathbf{w} = \lambda\mathbf{w},\ \text{where}\ K_{ij} = \varphi(\mathbf{x}_i)^T\varphi(\mathbf{x}_j).$$

Kernel or Gram matrix

Kernel PCA Formulation…



From the eigen equation $K\mathbf{w} = \lambda\mathbf{w}$

And the fact that the eigenvector $\mathbf{L}$ is normalized to 1, we obtain:

$$\| \mathbf{L} \|^2 = (\sum_{i=1}^{N} w_i \varphi(\mathbf{x}_i))^T (\sum_{i=1}^{N} w_i \varphi(\mathbf{x}_i)) = \mathbf{w}^T K \mathbf{w} = 1 \Rightarrow$$

$$\mathbf{w}^T \mathbf{w} = \frac{1}{\lambda}$$

KPCA Algorithm

Step 1: Compute the Gram matrix: $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j),\ i, j = 1, \ldots, N$

Step 2: Compute (eigenvalue, eigenvector) pairs of *K:* $(\mathbf{w}^l, \lambda_l), l = 1, \ldots, M$

Step 3: Normalize the eigenvectors: $\mathbf{w}^l \leftarrow \frac{\mathbf{w}^l}{\lambda_l}$

Thus, an eigenvector $\mathbf{w}^l$ of *C* is now represented as: $\mathbf{L}^l = \sum_{k=1}^{N} w_k^l \varphi(\mathbf{x}_k)$

To project a test feature ($\mathbf{x}$) onto $\mathbf{L}^l$ we need to compute:

$$\varphi(\mathbf{x})^T L^l = \varphi(\mathbf{x})^T \left(\sum_{k=1}^{N} w_k^l \varphi(\mathbf{x}_k)\right) = \sum_{k=1}^{N} w_k^l k(\mathbf{x}_k, \mathbf{x})$$

So, we never need compute *phi* explicitly

Feature Map Centering

So far we assumed that the feature map $\phi(\mathbf{x})$ is centered for the data points $\mathbf{x}_1, \ldots \mathbf{x}_N$

Actually, this centering can be done on the Gram matrix without ever explicitly computing the feature map $\phi(\mathbf{x})$.

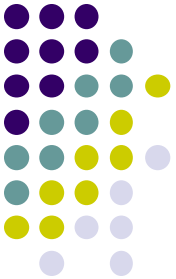
$$\widetilde{K} = (I - 11^T / N)K(I - 11^T / N)$$

is the kernel matrix for centered features, *i.e.*, $\sum_{i=1}^{N} \phi(\mathbf{x}_i) = 0$

A similar expression exist for projecting test features on the feature eigenspace

Scholkopf, Smola, Muller, "Nonlinear component analysis as a kernel eigenvalue problem," Technical report #44, Max Plank Institute, 1996.

KPCA: USPS Digit Recognition



# of components	Test Error Rate for degree (*d*)						
	1	2	3	4	5	6	7
32	9.6	8.8	8.1	8.5	9.1	9.3	10.8
64	8.8	7.3	6.8	6.7	6.7	7.2	7.5
128	8.6	5.8	5.9	6.1	5.8	6.0	6.8
256	8.7	5.5	5.3	5.2	5.2	5.4	5.4
512	n.a.	4.9	4.6	4.4	5.1	4.6	4.9
1024	n.a.	4.9	4.3	4.4	4.6	4.8	4.6
2048	n.a.	4.9	4.2	4.1	4.0	4.3	4.4

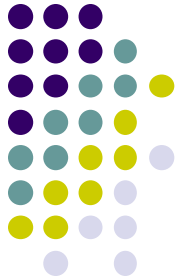
Linear PCA

Kernel function: $k(x, y) = (\mathbf{x}^T \mathbf{y})^d$

Classier: Linear SVM with features as kernel principal components
N = 3000, p = 16-by-16 image

Scholkopf, Smola, Muller, “Nonlinear component analysis as a kernel eigenvalue problem,” Technical report #44, Max Plank Institute, 1996.

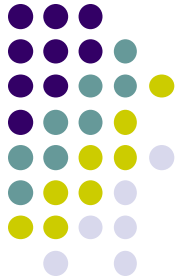
Robust-
Principal Component Analysis



- reference

1. Chandrasekharan, V., Sanghavi, S., Parillo, P., Wilsky, A.: Rank-sparsity incoherence for matrix decomposition. preprint 2009.
2. Wright, J., Ganesh, A., Rao, S., Peng, Y., Ma, Y.: Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization. In: NIPS 2009.
3. X. Yuan and J. Yang. Sparse and low-rank matrix decomposition via alternating direction methods. preprint, 2009.
4. Z. Lin, M. Chen, L. Wu, and Y. Ma. The augmented Lagrange multiplier method for exact recovery of a corrupted low-rank matrices. Mathematical Programming, submitted, 2009.
5. E. J. Candès, X. Li, Y. Ma, and J. Wright. Robust Principal Component Analysis? Submitted for publication, 2009.

research trends

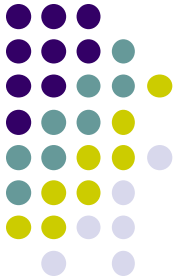


- Appear in the latest 2008-2009
- Theories are guaranteed and still refining; numerical algorithms are practical for 1000×1000 matrix (12 second) and still improving; applications not yet expand
- Research background: comes from

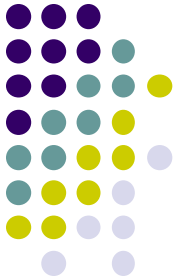
① matrix completion problem

② L1 norm and nuclear norm convex optimization

RPCA: outlines

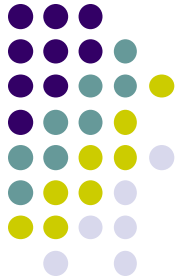


- Part I: theory
- Part II: numerical algorithm
- Part III: applications



- Part I: theory

PCA



- Given a data matrix M, assume $M = L_0 + N_0$

  L_0 is a Low-rank matrix

  N_0 is a small and i.i.d. Gaussian noise matrix

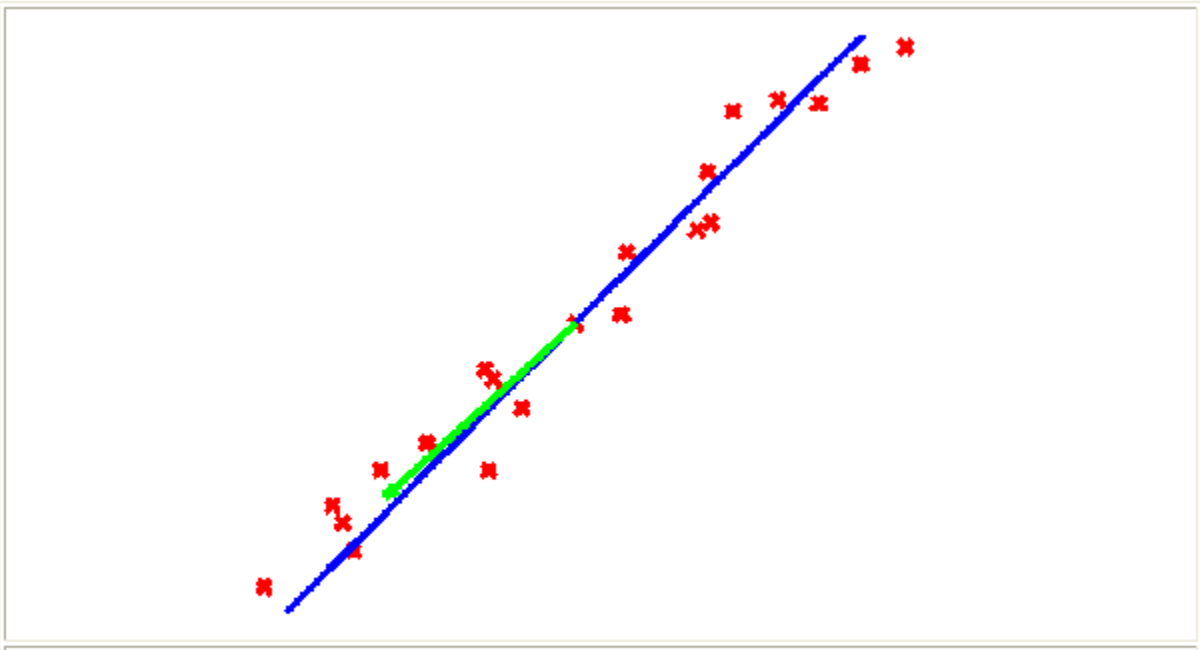
- Classical PCA seeks the best (in an L2 norm sense) rank-k estimate of L_0 by solving

$$\begin{array}{ll} \text{minimize} & \|M - L\|^2 \\ \text{subject to} & \text{rank}(L) \leq k \end{array}$$

- It can be solved by SVD

PCA example

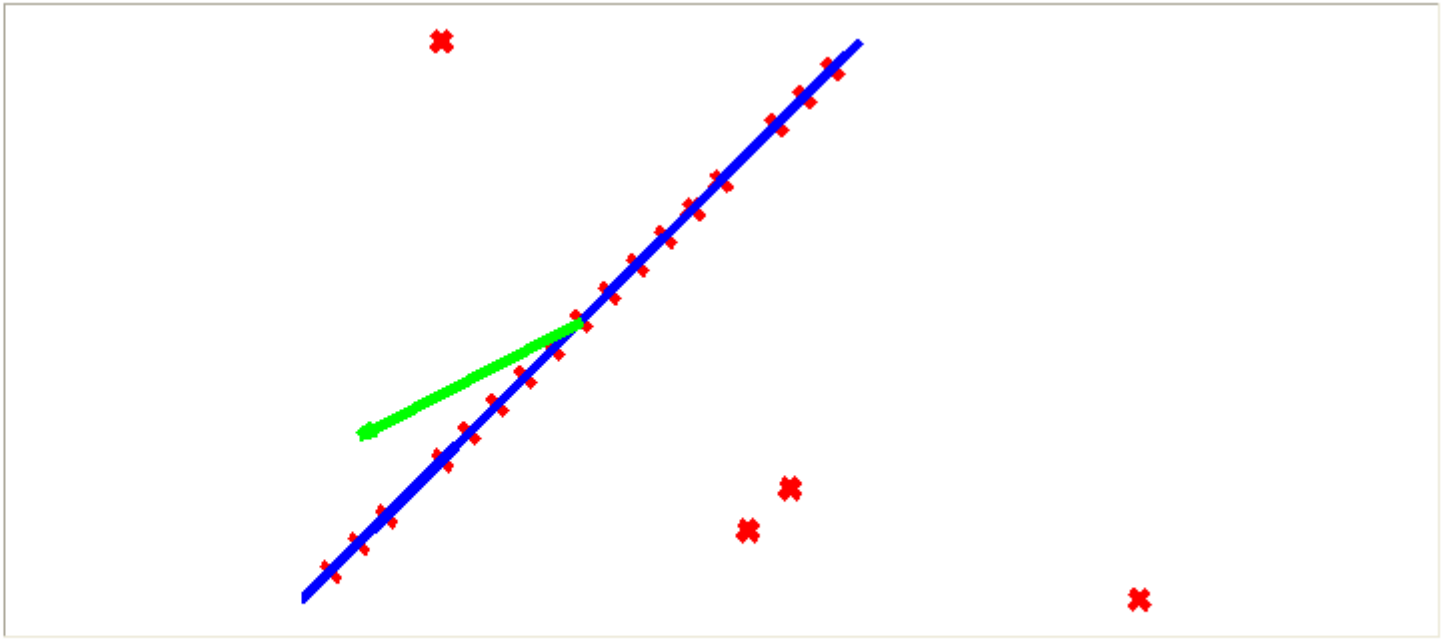
- When noise are small Gaussian, PCA does well



Samples (red) from a one-dimensional subspace (blue) corrupted by small Gaussian noise. The output of classical PCA (green) is very close to the true subspace despite all samples being noisy.

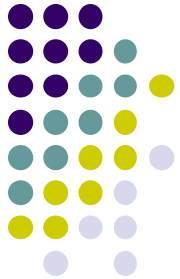
Defect of PCA

- When noise are not Gaussian, but appear like spike, i.e. data contains outliers, PCA fails



Samples (red) from a one-dimensional subspace (blue) corrupted by sparse, large errors. The principal component (green) is quite far from the true subspace even when over three-fourths of the samples are uncorrupted.

RPCA



- When noise are sparse spikes, another robust model (RPCA) should be built
- Assume $M = L_0 + S_0$

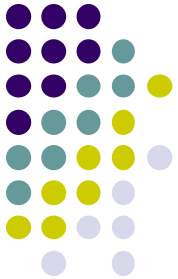
  L_0 is a Low-rank matrix

  S_0 is a Sparse spikes noise matrix

- Problem: we know M is composed by a low rank and a sparse matrix. Now, we are given M and asked to recover its original two components

  It's purely a matrix decomposition problem

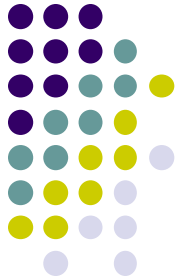
ill-posed problem



- We only observe M, it's impossible to know which two matrices add up to be it. So without further assumptions, it can't be solved:

1. let $A^\star$ be any sparse matrix and let $B^\star = e_i e_j^T$, another valid sparse-plus-low-rank decomposition might be $\hat{A} = A^\star + e_i e_j^T$ and $\hat{B} = 0$ Thus, the low-rank matrix should be assumed to be not too sparse
2. $B^\star$ is any low-rank matrix and $A^\star = -ve_1^T$, with v being the first column of $B^\star$. A reasonable sparse-plus-low-rank decomposition in this case might be $\hat{B} = B^\star + A^\star$ and $\hat{A} = 0$ Thus, the sparse matrix should be assumed to not be low-rank

Assumptions about how L and S are generated



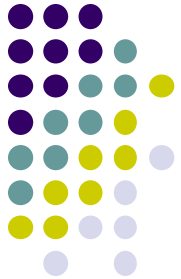
1. Low-rank matrix L:

Random orthogonal model . A rank-k matrix $B^\star \in \mathbb{R}^{n \times n}$ with SVD $B^\star = U\Sigma V'$ is constructed as follows: The singular vectors $U, V \in \mathbb{R}^{n \times k}$ are drawn *uniformly* at random from the collection of rank-k partial isometries in $\mathbb{R}^{n \times k}$. The choices of U and V need not be mutually independent. No restriction is placed on the singular values.

2. Sparse matrix S:

Random sparsity model. The matrix $A^\star$ is such that support$(A^\star)$ is chosen uniformly at random from the collection of all support sets of size m. There is no assumption made about the values of $A^\star$ at locations specified by support$(A^\star)$.

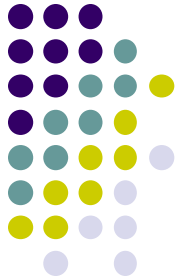
Under what conditions can M be correctly decomposed ?



1. Let the matrices with rank ≤ r(L) and with either the same row-space or column-space as L live in a matrix space denoted by T(L)
2. Let the matrices with the same support as S and number of nonzero entries ≤ those of S live in a matrix space denoted by O(S)

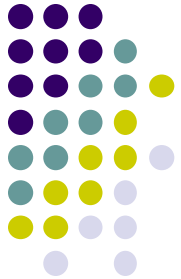
- Then, if T(L) ∩O(S)=null, M can be correctly decomposed.

Detailed conditions



- Various work in 2009 proposed different detailed conditions. They improved on each other, being more and more relaxed.
- Under each of these conditions, they proved that matrix can be precisely or even exactly decomposed.

Conditions involving probability distributions



COROLLARY 4. *Suppose that a rank-k matrix* $B^\star \in \mathbb{R}^{n\times n}$ *is drawn from the random orthogonal model, and that* $A^\star \in \mathbb{R}^{n\times n}$ *is drawn from the random sparsity model with m non-zero entries. Given* $C = A^\star + B^\star$*, there exists a range of values for* γ *(given by (4.8)) so that* $(\hat{A}, \hat{B}) = (A^\star, B^\star)$ *is the unique optimum of the SDP (1.3) with high probability provided*

$$m \lesssim \frac{n^{1.5}}{\log n \sqrt{\max(k, \log n)}}.$$

- for B with rank k (smaller than n), exact recovery is possible with high probability even when m is super-linear in n

the latest condition developed

- The work of [1] and [2] are parallel, latest [5] improved on them and yields the 'best' condition

$$\begin{aligned} &\text{minimize} && \|L\|_* + \lambda \|S\|_1 \\ &\text{subject to} && L + S = M \end{aligned}$$

$$\max_i \|U^* e_i\|^2 \leq \frac{\mu r}{n_1}, \quad \max_i \|V^* e_i\|^2 \leq \frac{\mu r}{n_2}, \tag{1.2}$$

$$\|UV^*\|_\infty \leq \sqrt{\frac{\mu r}{n_1 n_2}}. \tag{1.3}$$

Theorem 1.1 *Suppose L_0 is $n \times n$, obeys (1.2)–(1.3), and that the support set of S_0 is uniformly distributed among all sets of cardinality m. Then there is a numerical constant c such that with probability at least $1 - cn^{-10}$ (over the choice of support of S_0), Principal Component Pursuit (1.1) with $\lambda = 1/\sqrt{n}$ is* exact, *i.e. $\hat{L} = L_0$ and $\hat{S} = S_0$, provided that*

$$\text{rank}(L_0) \leq \rho_r n \mu^{-1} (\log n)^{-2} \quad \text{and} \quad m \leq \rho_s n^2. \tag{1.4}$$

Above, ρ_r and ρ_s are positive numerical constants. In the general rectangular case where L_0 is

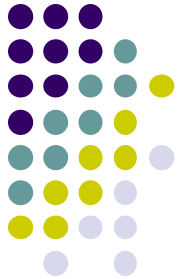
Brief remarks

- in [5], they prove even if:

1. the rank of L grows proportional to $O(n/\log^2 n)$
2. noise in S are of order $O(n^2)$

exact decomposition is feasible

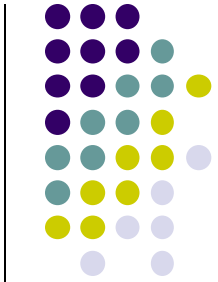
Norm and Matrix Norm



- L2 (Frobenius)
- LP, L1, L∞, L0
- Nuclear Norm (the sum of all singular values)

- http://en.wikipedia.org/wiki/L0_norm#Zero_norm
- http://en.wikipedia.org/wiki/Matrix_norm

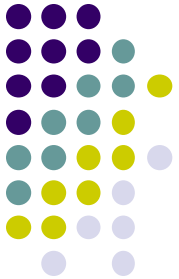
Norm and Matrix Norm



$$\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2} = \sqrt{\operatorname{trace}(A^*A)} = \sqrt{\sum_{i=1}^{\min\{m,\,n\}}\sigma_i^2}$$

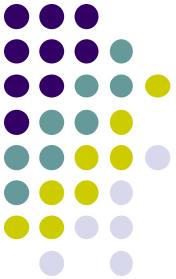
$$\|A\|_p = \|\operatorname{vec}(A)\|_p = \left(\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^p\right)^{1/p}$$

$$\|A\|_* = \operatorname{trace}\left(\sqrt{A^*A}\right) = \sum_{i=1}^{\min\{m,\,n\}}\sigma_i.$$



- Part II: numerical algorithm

Convex optimization



- In order to solve the original problem, it is reformulated into optimization problem.
- A straightforward propose is

$$\min_{A,E} \ \mathrm{rank}(A) + \gamma \|E\|_0 \quad \text{subj} \quad A + E = D$$

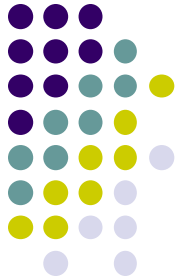
but it's not convex and intractable

- Recent advances in understanding of the nuclear norm heuristic for low-rank solutions and the L1 heuristic for sparse solutions suggest

$$\min_{A,E} \ \|A\|_* + \lambda \|E\|_1 \quad \text{subj} \quad A + E = L$$

which is convex, i.e. exists a unique minima

numerical algorithm



- During just two years, a series of algorithms have been proposed, [4] provides all comparisons, and most codes available at

  http://watt.csl.illinois.edu/~perceive/matrix-rank/sample_code.html

- They include:

1. Interior point method [1]
2. iterative thresholding algorithm
3. Accelerated Proximal Gradient (APG) [2]
4. A dual approach [4]
5. (latest & best) Augmented Lagrange Multiplier (ALM) [3,4]or Alternating Directions Method (ADM) [3,5]

ADM

- Problem $\begin{array}{ll}\min_{A,B} & \gamma\|A\|_{l_1}+\|B\|_* \\ s.t. & A+B=C,\end{array}$
- The corresponding Augmented Lagrangian function is

$$L(A,B,Z):=\gamma\|A\|_{l_1}+\|B\|_*-\langle Z,A+B-C\rangle+\frac{\beta}{2}\|A+B-C\|^2$$

- $Z\in\mathcal{R}^{m\times n}$ is the multiplier of the linear constraint. < > is trace inner product for matrix <X,Y>=trace(X^TY)
- Then, the iterative scheme of ADM is

$$\begin{cases} A^{k+1}\in \operatorname{argmin}_{A\in R^{m\times n}}\{L(A,B^k,Z^k)\}, \\ B^{k+1}\in \operatorname{argmin}_{B\in\mathcal{R}^{m\times n}}\{L(A^{k+1},B,Z^k)\}, \\ Z^{k+1}=Z^k-\beta(A^{k+1}+B^{k+1}-C), \end{cases}$$

Two established facts

- To approach the optimization, two well known facts is needed

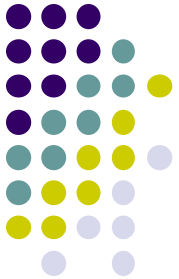
1. $\mathcal{S}_\varepsilon[W] = \arg\min_X \varepsilon\|X\|_1 + \frac{1}{2}\|X - W\|_F^2$
2. $U\mathcal{S}_\varepsilon[S]V^T = \arg\min_X \varepsilon\|X\|_* + \frac{1}{2}\|X - W\|_F^2$

$\mathcal{S}_\varepsilon$ is the soft thresholding operator

$$\mathcal{S}_\varepsilon[x] \doteq \begin{cases} x - \varepsilon, & \text{if } x > \varepsilon, \\ x + \varepsilon, & \text{if } x < -\varepsilon, \\ 0, & \text{otherwise,} \end{cases}$$

USV^T is SVD of W

Optimization solution



- Sparse A with L_1 norm

$$A^{k+1} = \frac{1}{\beta} Z^k - B^k + C - P_{\Omega_\infty^{\gamma/\beta}}[\frac{1}{\beta} Z^k - B^k + C]$$

$$\Omega_\infty^{\gamma/\beta} := \{X \in \mathbf{R}^{n \times n} \mid -\gamma/\beta \leq X_{ij} \leq \gamma/\beta\}$$

- Low-rank B with nuclear norm. Reformulate the objective so that previous fact can be used:

$$B^{k+1} = \text{argmin}_{B \in R_{m \times n}} \{\|B\|_* + \frac{\beta}{2} \|B - [C - A^{k+1} + \frac{1}{\beta} Z^k]\|^2\}$$

$$B^{k+1} = U^{k+1} \, \text{diag}(\max\{\sigma_i^{k+1} - \frac{1}{\beta}, 0\}) (V^{k+1})^T$$

$$C - A^{k+1} + \frac{1}{\beta} Z^k = U^{k+1} \Sigma^{k+1} (V^{k+1})^T \quad \text{with} \quad \Sigma^{k+1} = \text{diag}(\{\sigma_i^{k+1}\}_{i=1}^r)$$

Final algorithm of ADM

Algorithm: the ADM for SLRMD problem:

Step 1. Generate A^{k+1}:

$$A^{k+1} = \frac{1}{\beta}Z^k - B^k + C - P_{\Omega_\infty^{\gamma/\beta}}[\frac{1}{\beta}Z^k - B^k + C].$$

Step 2 Generate B^{k+1}:

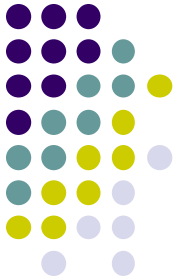
$$B^{k+1} = U^{k+1}\,\mathrm{diag}(\max\{\sigma_i^{k+1} - \frac{1}{\beta}, 0\})(V^{k+1})^T,$$

where U^{k+1}, V^{k+1} and $\{\sigma_i^{k+1}\}$ are generated by the singular values decomposition of $C - A^{k+1} + \frac{1}{\beta}Z^k$, i.e.,

$$C - A^{k+1} + \frac{1}{\beta}Z^k = U^{k+1}\Sigma^{k+1}(V^{k+1})^T, \text{ with } \Sigma^{k+1} = \mathrm{diag}(\{\sigma_i^{k+1}\}_{i=1}^r).$$

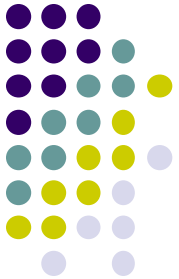
Step 3. Update the multiplier:

$$Z^{k+1} = Z^k - \beta(A^{k+1} + B^{k+1} - C).$$



- Part III: application

Applications [5]



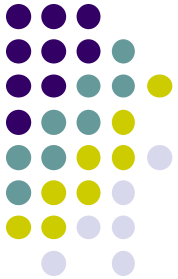
(1) background modeling from surveillance videos

① Airport video

② Lobby video with varying illumination

(2) removing shadows and specularities from face images

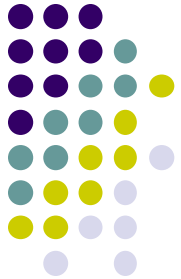
Airport video



- a video of 200 frames (resolution $176 \times 144 = 25344$ pixels) has a static background, but significant foreground variations
- reshape each frame as a column vector (25344×1) and stack them into a matrix M (25344×200)
- Objective: recover the low-rank and sparse components of M

(a) Original frames (b) Low-rank $\hat{L}$ (c) Sparse $\hat{S}$

Lobby video



- a video of 250 frames (resolution $168 \times 120 = 20160$ pixels) with several drastic illumination changes
- reshape each frame as a column vector (20160×1) and stack them into a matrix M (20160×250)
- Objective: recover the low-rank and sparse components of M

(a) Original frames (b) Low-rank $\hat{L}$ (c) Sparse $\hat{S}$

The End

新浪微博：@浙大张宏鑫

微信公众号：