



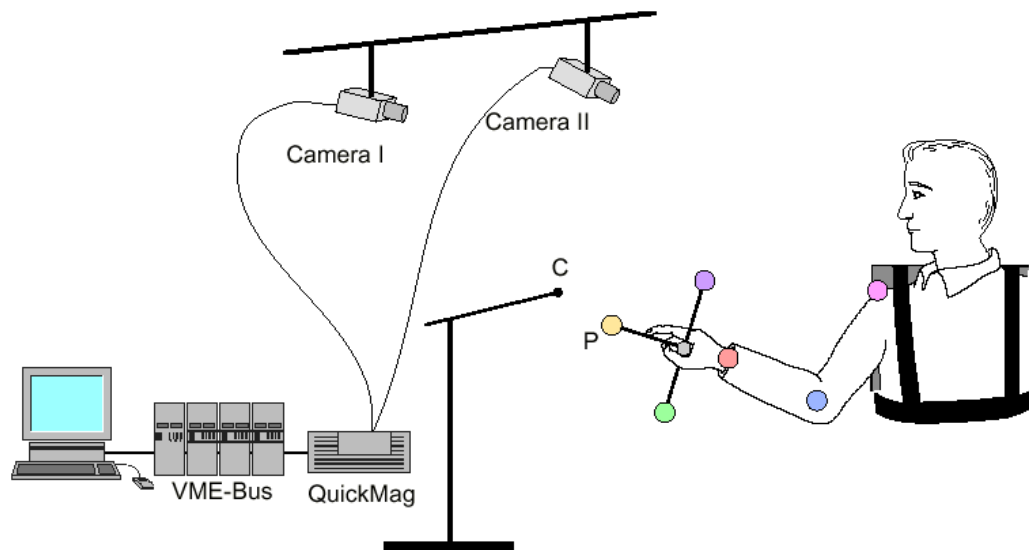
# Lecture 8– Dimensionality Reduction

## Contents:

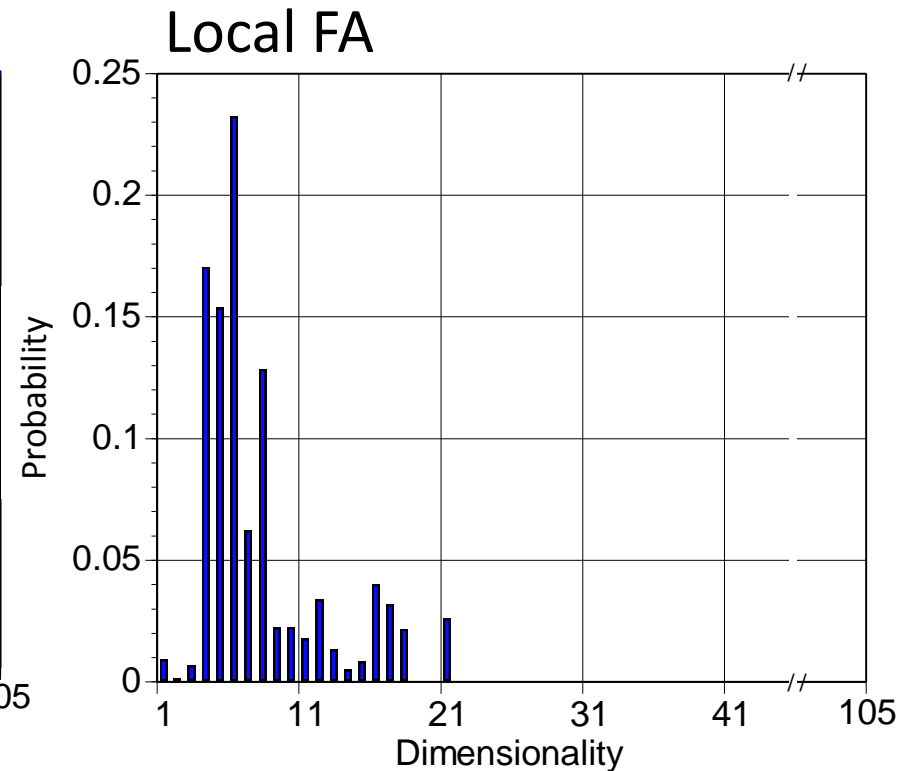
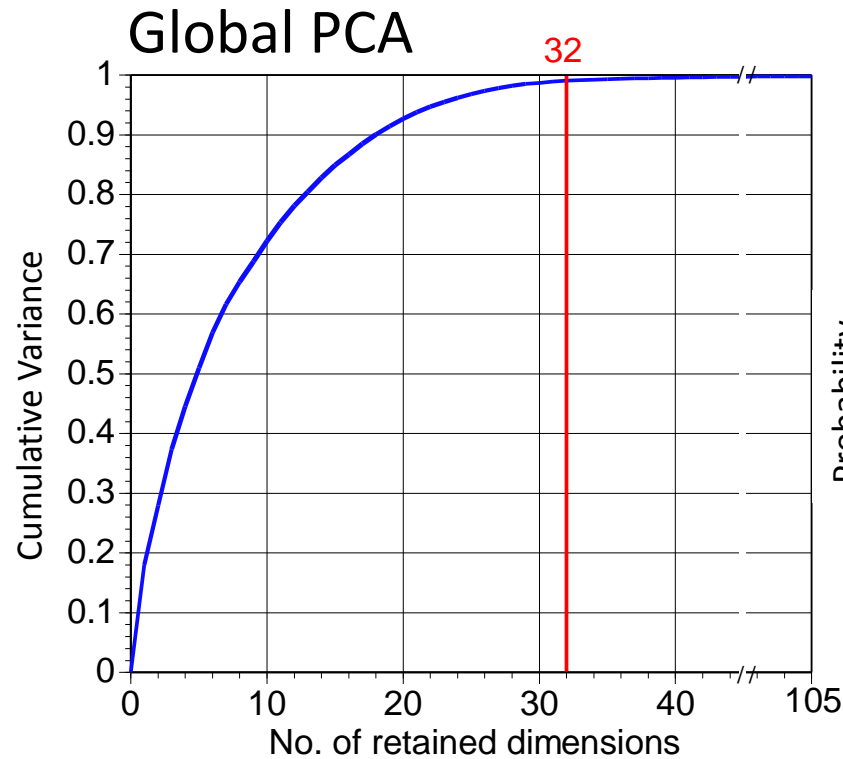
- Subset Selection & Shrinkage
  - Ridge regression, Lasso
- PCA, PCR, PLS
- Comparison of Methods

# Data From Human Movement

- Measure arm movement and full-body movement of humans and anthropomorphic robots
- Perform local dimensionality analysis with a growing variational mixture of factor analyzers



# Dimensionality of Full Body Motion



**About 8 dimensions in the space formed by joint positions, velocities, and accelerations are needed to model an inverse dynamics model**

# Dimensionality Reduction

## Goals:

- fewer dimensions for subsequent processing
- better numerical stability due to removal of correlations
- simplify post processing due to “advanced statistical properties” of pre-processed data
- don't lose important information, only redundant information or irrelevant information
- perform dimensionality reduction spatially localized for nonlinear problems (e.g., each RBF has its own local dimensionality reduction)

# Subset Selection & Shrinkage Methods

## Subset Selection

Refers to methods which selects a set of variables to be included and discards other dimensions based on some optimality criterion. Does regression on this reduced dimensional input set. **Discrete method** –variables are either selected or discarded

- leaps and bounds procedure (Furnival & Wilson, 1974)
- Forward/Backward stepwise selection

## Shrinkage Methods

Refers to methods which reduces/shrinks the redundant or irrelevant variables in a more continuous fashion.

- Ridge Regression
- Lasso
- Derived Input Direction Methods – PCR, PLS

# Ridge Regression

**Ridge regression** shrinks the coefficients by imposing a penalty on their size. They minimize a penalized residual sum of squares :

$$\hat{w}^{ridge} = \arg \min_w \left\{ \sum_{i=1}^N (t_i - w_0 - \sum_{j=1}^M x_{ij} w_j)^2 + \lambda \sum_{j=1}^M w_j^2 \right\}$$

Complexity parameter controlling amount of shrinkage

Equivalent representation:

$$\hat{w}^{ridge} = \arg \min_w \left\{ \sum_{i=1}^N (t_i - w_0 - \sum_{j=1}^M x_{ij} w_j)^2 \right\}$$

subject to  $\sum_{j=1}^M w_j^2 \leq s$

# Ridge Regression (cont'd)

Some notes:

- when there are many correlated variables in a regression problem, their coefficients can be poorly determined and exhibit high variance e.g. a wildly large positive variable can be canceled by a similarly largely negative coefficient on its correlated cousin.
- The bias term is not included in the penalty.
- When the inputs are orthogonal, the ridge estimates are just a scaled version of the least squares estimate

Matrix representation of criterion and it's solution:

$$J(\mathbf{w}, \lambda) = \frac{1}{2} (\mathbf{t} - \mathbf{X}\mathbf{w})^T (\mathbf{t} - \mathbf{X}\mathbf{w}) + \lambda \mathbf{w}^T \mathbf{w}$$
$$\hat{\mathbf{w}}^{ridge} = (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^T \mathbf{t}$$

# Ridge Regression (cont'd)

Ridge regression shrinks the dimension with least variance the most.

## Shrinkage Factor :

*Each direction is shrunk by  $\frac{d_j^2}{(d_j^2 + \lambda)}$ , [Page 62: Elem. Stat. Learning]*

*where  $d_j^2$  refers to the corresponding eigen value.*

## Effective degree of freedom :

$$\text{df}(\lambda) = \text{tr}[\mathbf{X}(\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^T] = \sum_{j=1}^M \frac{d_j^2}{(d_j^2 + \lambda)} \quad [\text{Page 63: Elem. Stat. Learning}]$$

*Note :  $\text{df}(\lambda) = M$  if  $\lambda = 0$  (no regularization)*



# Lasso

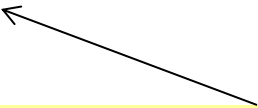
**Lasso** is also a shrinkage method like ridge regression. It minimizes a penalized residual sum of squares :

$$\hat{w}^{lasso} = \arg \min_w \left\{ \sum_{i=1}^N (t_i - w_0 - \sum_{j=1}^M x_{ij} w_j)^2 + \lambda \sum_{j=1}^M |w_j| \right\}$$

Equivalent representation:

$$\hat{w}^{lasso} = \arg \min_w \left\{ \sum_{i=1}^N (t_i - w_0 - \sum_{j=1}^M x_{ij} w_j)^2 \right\}$$

subject to  $\sum_{j=1}^M |w_j| \leq s$



The  $L_2$  ridge penalty is replaced by the  $L_1$  lasso penalty. This makes the solution non-linear in  $\mathbf{t}$ .

# Derived Input Direction Methods

These methods essentially involves transforming the input directions to some low dimensional representation and using these directions to perform the regression.

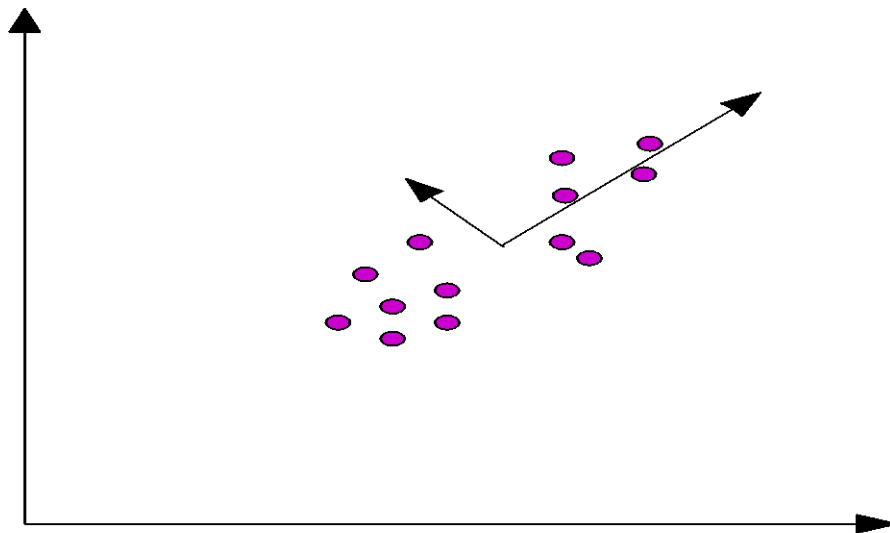
## Example Methods

- **Principal Components Regression (PCR)**
  - Based on input variance only
- **Partial Least Squares (PLS)**
  - Based on input-output correlation and output variance

# Principal Component Analysis

From earlier discussions:

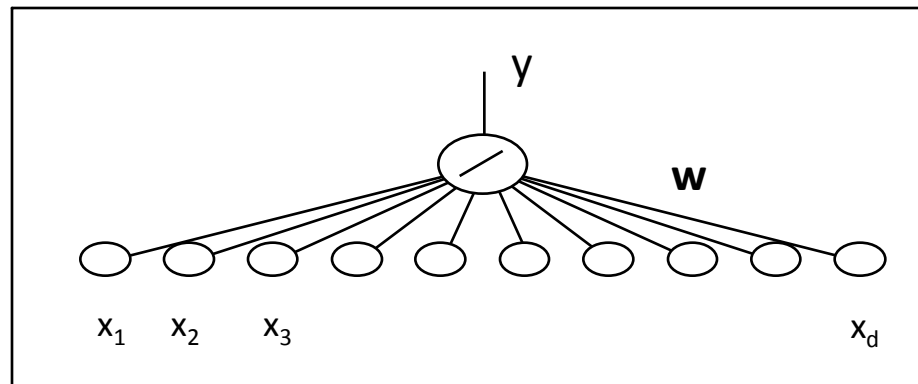
- PCA finds the eigenvectors of the correlation (covariance) matrix of the the data
- Here, we show how PCA can be learned by bottle-neck neural networks (auto-associator) and Hebbian learning frameworks



# PCA Implementation

## Hebbian Learning

- obtain a measure of familiarity of a new data point
  - ◆ the more familiar a data point, the larger the output



$$\mathbf{w}^{n+1} = \mathbf{w}^n + \alpha y \mathbf{x}$$

# Fixing the Instability (Oja's rule)

◆ Renormalization  $\mathbf{w}^{n+1} = \frac{\mathbf{w}^n + \alpha \mathbf{x} y}{\|\mathbf{w}^n + \alpha \mathbf{x} y\|}$

◆ Oja's Rule  $\Delta \mathbf{w} = \mathbf{w}^{n+1} - \mathbf{w}^n = \alpha y (\mathbf{x} - y \mathbf{w}) = \alpha (y \mathbf{x} - y^2 \mathbf{w})$

◆ Verify Oja's Rule (does it do the right thing ??)

$$\begin{aligned} E\{\Delta \mathbf{w}\} &= E\{\alpha (\mathbf{x} y - y^2 \mathbf{w})\} \\ &= \alpha E\{\mathbf{x} \mathbf{x}^T \mathbf{w} - \mathbf{w}^T \mathbf{x} \mathbf{x}^T \mathbf{w}\} \\ &= \alpha (\mathbf{C} \mathbf{w} - \mathbf{w}^T \mathbf{C} \mathbf{w}) \end{aligned}$$

After convergence:

$$E\{\Delta \mathbf{w}\} = 0 \Rightarrow$$

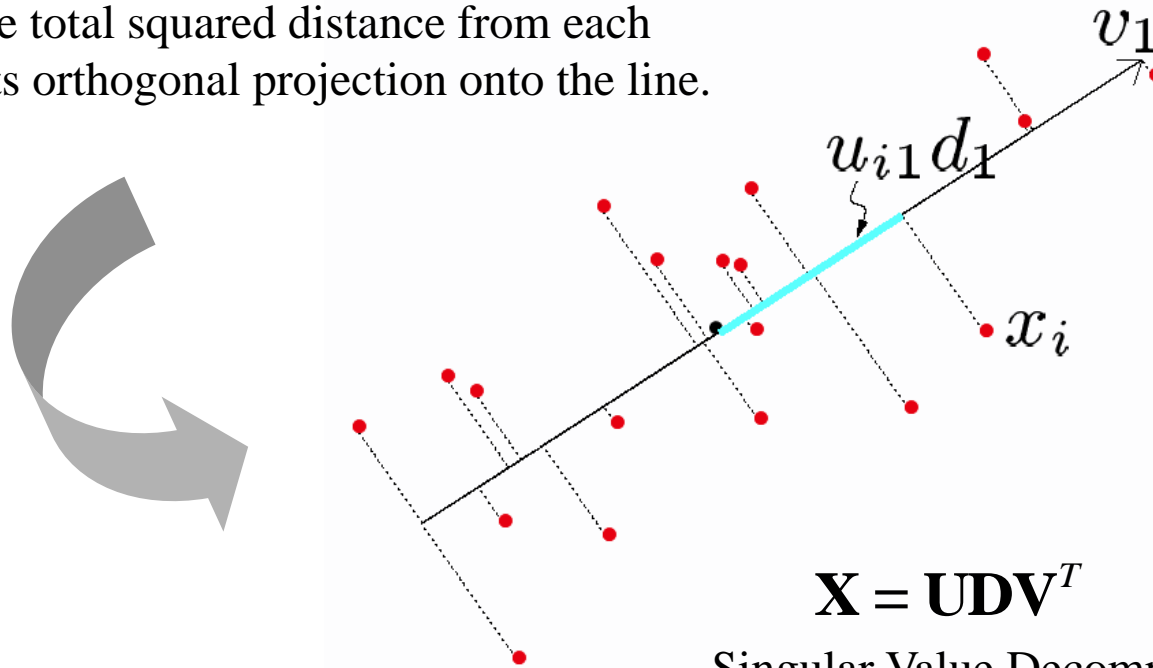
$$\mathbf{C} \mathbf{w} = (\mathbf{w}^T \mathbf{C} \mathbf{w}) \mathbf{w} = \lambda \mathbf{w}, \text{ thus } \mathbf{w} \text{ is eigenvector}$$

$$\mathbf{w}^T \mathbf{C} \mathbf{w} = \mathbf{w}^T (\mathbf{w}^T \mathbf{C} \mathbf{w}) \mathbf{w} = (\mathbf{w}^T \mathbf{C} \mathbf{w}) \mathbf{w}^T \mathbf{w}$$

$$\text{thus, } \mathbf{w}^T \mathbf{w} = 1$$

# Application Examples of Oja's Rule

Finds the direction (line) which minimizes the sum of the total squared distance from each point to its orthogonal projection onto the line.



$$\mathbf{X} = \mathbf{U}\mathbf{D}\mathbf{V}^T$$

Singular Value Decomposition (SVD)

# PCA : Batch vs Stochastic

## PCA in Batch Update:

- just subtract the mean from the data
- calculate eigenvectors to obtain principle components (Matlab function “eig”)

## PCA in Stochastic Update:

- stochastically estimate the mean and subtract it from input data

$$\bar{\mathbf{x}}^{n+1} = \frac{\bar{\mathbf{x}}^n n + \mathbf{x}}{n+1} = \bar{\mathbf{x}}^n + \frac{1}{n+1} (\mathbf{x} - \bar{\mathbf{x}}^n)$$

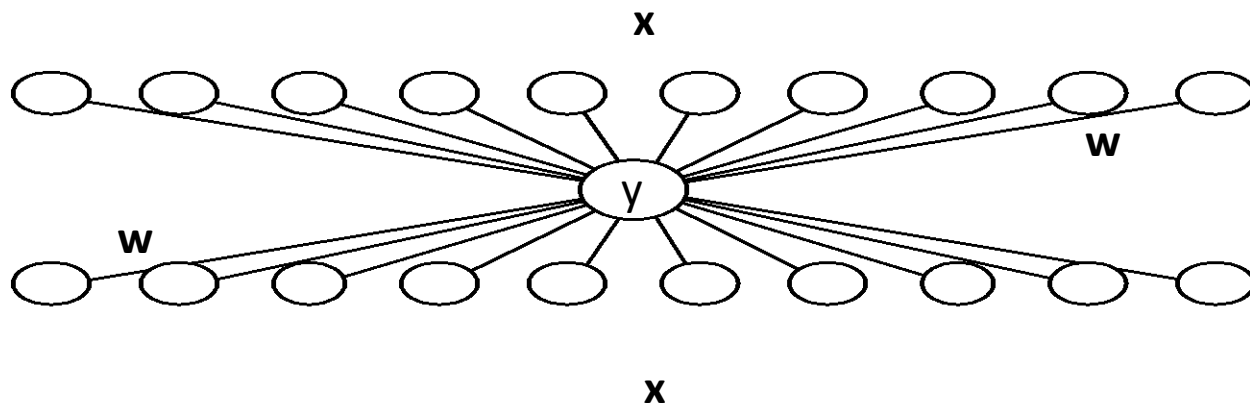
$$\bar{\mathbf{x}}^{n+1} = \bar{\mathbf{x}}^n + \alpha (\mathbf{x} - \bar{\mathbf{x}}^n)$$

- use Oja’s rule on mean subtracted data

# Autoencoder as motivation for Oja's Rule

- ◆ Note that Oja's rule looks like a supervised learning rule
  - the update looks like a reverse delta-rule: it depends on the difference between the actual input and the back-propagated output

$$\Delta \mathbf{w} = \mathbf{w}^{n+1} - \mathbf{w}^n = \alpha y (\mathbf{x} - y \mathbf{w})$$



Convert unsupervised learning into a supervised learning problem by trying to reconstruct the inputs from few features!



# Learning in the Autoencoder

◆ Minimize the cost  $J = \frac{1}{2}(\mathbf{x} - \hat{\mathbf{x}})^T(\mathbf{x} - \hat{\mathbf{x}}) = \frac{1}{2}(\mathbf{x} - y\mathbf{w})^T(\mathbf{x} - y\mathbf{w})$

$$\begin{aligned}\frac{\partial J}{\partial \mathbf{w}} &= -\left(\frac{\partial y}{\partial \mathbf{w}} \mathbf{w} + y \frac{\partial \mathbf{w}}{\partial \mathbf{w}}\right)^T (\mathbf{x} - y\mathbf{w}) \\ &= -\left(\frac{\partial(\mathbf{w}^T \mathbf{x})}{\partial \mathbf{w}} \mathbf{w} + y \frac{\partial \mathbf{w}}{\partial \mathbf{w}}\right)^T (\mathbf{x} - y\mathbf{w}) \\ &= -(\mathbf{x}^T \mathbf{w} + y)(\mathbf{x} - y\mathbf{w}) \\ &= -(y + y)(\mathbf{x} - y\mathbf{w}) \\ &= -2y(\mathbf{x} - y\mathbf{w})\end{aligned}$$

$$\Delta \mathbf{w} = -\alpha \frac{\partial J}{\partial \mathbf{w}} = \alpha' y(\mathbf{x} - y\mathbf{w})$$

This is Oja's rule!

# Discussion about PCA

- ◆ If data is noisy, we may represent the noise instead of the data
  - The way out: Factor Analysis (handles noise in input dimension also)
- ◆ PCA has no data generating model
- ◆ PCA has no probabilistic interpretation (*not quite true !!*)
- ◆ PCA ignores possible influence of subsequent (e.g., supervised) learning steps
- ◆ PCA is a linear method
  - Way out: Nonlinear PCA
- ◆ PCA can converge very slowly
  - Way out: EM versions of PCA
- ◆ But PCA is a very reliable method for dimensionality reduction if it is appropriate!

# PCA preprocessing for Supervised Learning

In Batch Learning Notation for a Linear Network:

$$\mathbf{U} = \left[ \text{eigenvectors} \left( \frac{\sum_{n=1}^N (\mathbf{x}^n - \bar{\mathbf{x}})(\mathbf{x}^n - \bar{\mathbf{x}})^T}{N-1} \right) \right]_{\max(1:k)}$$
$$= \left[ \text{eigenvectors} \left( \frac{\tilde{\mathbf{X}}^T \tilde{\mathbf{X}}}{N-1} \right) \right]_{\max(1:k)} \quad \text{where } \tilde{\mathbf{X}} \text{ contains mean-zero data}$$

Subsequent Linear Regression for Network Weights

$$\mathbf{W} = (\mathbf{U}^T \tilde{\mathbf{X}}^T \tilde{\mathbf{X}} \mathbf{U})^{-1} \mathbf{U}^T \tilde{\mathbf{X}}^T \mathbf{Y}$$

**NOTE: Inversion of the above matrix is very cheap since it is diagonal! No numerical problems!**

**Problems of this pre-processing:**

Important regression data might have been clipped!

# Principal Component Regression (PCR)

Build the matrix  $\mathbf{X}$  and vector  $\mathbf{y}$

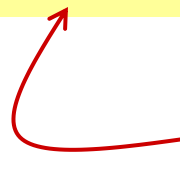
$$\mathbf{X} = (\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, \dots, \tilde{\mathbf{x}}_n)^T$$
$$\mathbf{t} = (t_1, t_2, \dots, t_n)^T$$

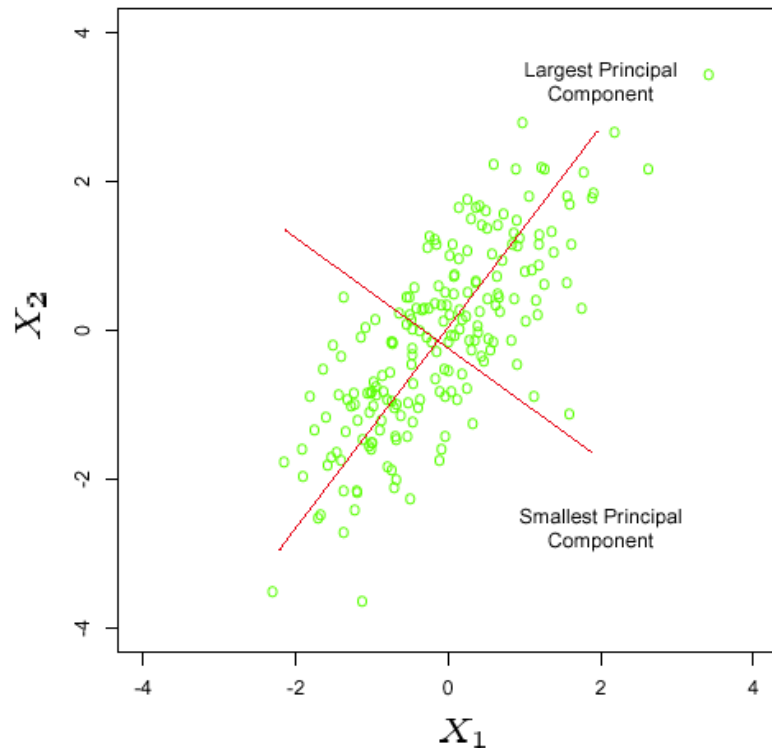
Eigen Decomposition

$$\mathbf{X}^T \mathbf{X} = \mathbf{V} \mathbf{D}^2 \mathbf{V}^T$$

Compute the linear model

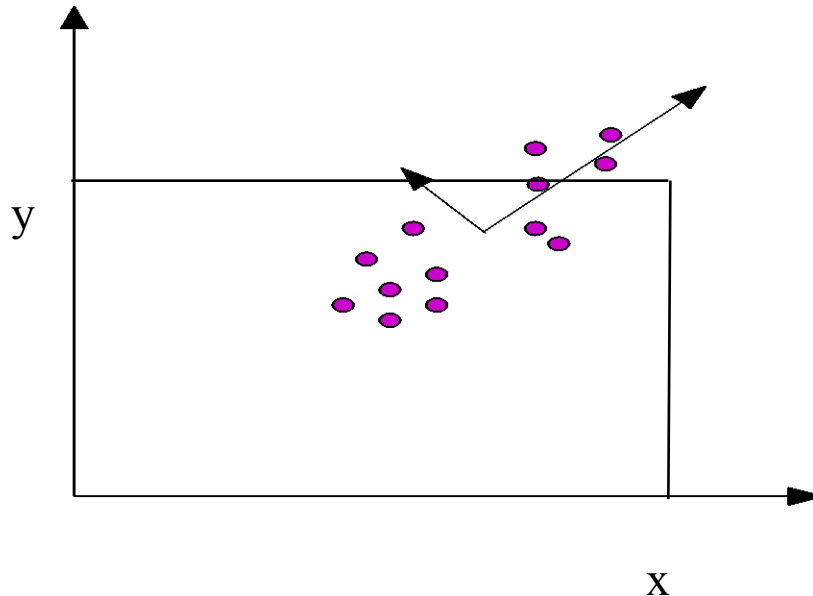
$$\mathbf{U} = \max_{\text{eigenvalues}} [\mathbf{v}_1 \mathbf{v}_2 \dots \mathbf{v}_k]$$
$$\hat{\mathbf{W}}^{pcr} = (\mathbf{U}^T \mathbf{X}^T \mathbf{X} \mathbf{U})^{-1} \mathbf{U}^T \mathbf{X}^T \mathbf{t}$$

 **Data projection**



# PCA in Joint Data Space

- ◆ A straightforward extension to take supervised learning step into account:
  - perform PCA in joint data space
  - extract linear weight parameters from PCA results



# PCA in Joint Data Space: Formalization

$$\mathbf{Z} = \begin{bmatrix} \mathbf{x} - \bar{\mathbf{x}} \\ \mathbf{y} - \bar{\mathbf{y}} \end{bmatrix}$$
$$\mathbf{U} = \left[ \text{eigenvectors} \left( \frac{\sum_{n=1}^N (\mathbf{z}^n - \bar{\mathbf{z}})(\mathbf{z}^n - \bar{\mathbf{z}})^T}{N-1} \right) \right]_{\max(1:k)}$$
$$= \left[ \text{eigenvectors} \left( \frac{\mathbf{Z}^T \mathbf{Z}}{N-1} \right) \right]_{\max(1:k)}$$

The Network Weights become:

$$\mathbf{W} = \mathbf{U}_x \left( \mathbf{U}_y^T - \mathbf{U}_y^T (\mathbf{U}_y \mathbf{U}_y^T - \mathbf{I})^{-1} \mathbf{U}_y \mathbf{U}_y^T \right), \text{ where } \mathbf{U} = \begin{bmatrix} \mathbf{U}_x (= d \times k) \\ \mathbf{U}_y (= c \times k) \end{bmatrix}$$

**Note:** this new kind of linear network can actually tolerate noise in the input data!  
But only the same noise level in all (joint) dimensions!

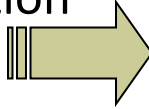
# Partial Least Squares (PLS)

Partial Least Squares is a linear regression methods that includes dimensionality reduction

**Build the matrix  $\mathbf{X}$  and vector  $\mathbf{y}$**

$$\mathbf{X} = (\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, \dots, \tilde{\mathbf{x}}_n)^T$$

$$\mathbf{t} = (t_1, t_2, \dots, t_n)^T$$

Projection direction based on input output correlation. 

**Projection direction**

**Univariate regressions**

**residuals**

**Recursively compute the linear model**

Initialize :  $\mathbf{X}_{\text{res}} = \mathbf{X}$ ,  $\mathbf{y} = \mathbf{y}_{\text{res}}$   
For  $i = 1 : r$  (# of projections)

$$\mathbf{u}_i = \mathbf{X}_{\text{res}}^T \mathbf{y}_{\text{res}}$$

$$\mathbf{s} = \mathbf{X}_{\text{res}} \mathbf{u}_i$$

$$w_i^{pls} = \frac{\mathbf{s}^T \mathbf{y}_{\text{res}}}{\mathbf{s}^T \mathbf{s}}$$

$$\mathbf{p}_i = \frac{\mathbf{s}^T \mathbf{X}_{\text{res}}}{\mathbf{s}^T \mathbf{s}}$$

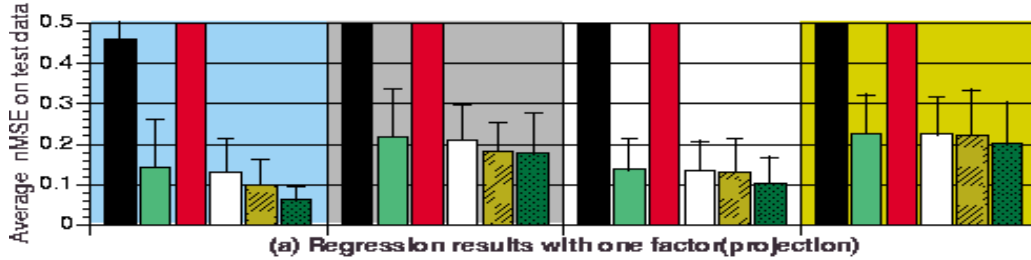
$$\mathbf{y}_{\text{res}} = \mathbf{y}_{\text{res}} - \mathbf{s} \beta_i$$

$$\mathbf{X}_{\text{res}} = \mathbf{X}_{\text{res}} - \mathbf{s} \mathbf{p}_i$$

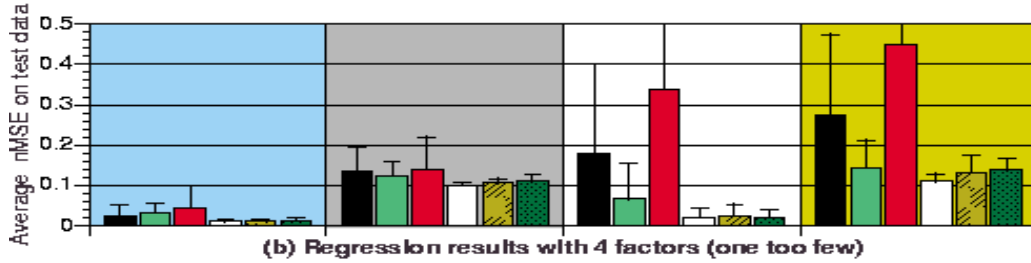
**Projected data**

# Comparing Dimensionality Red. Methods

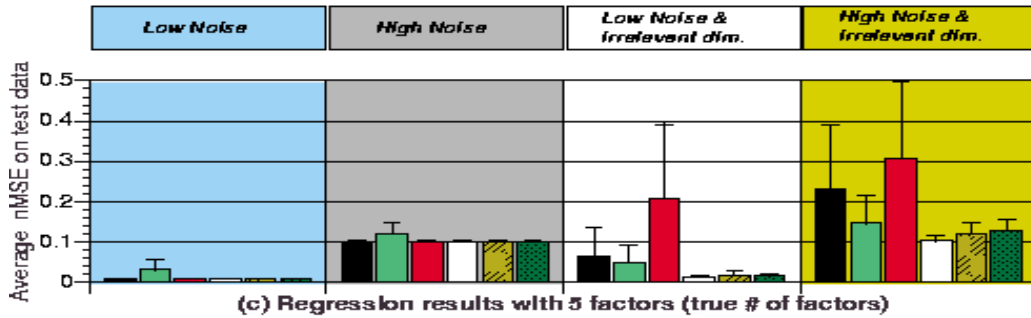
- FA
- PCR
- PCA\_J
- PLS



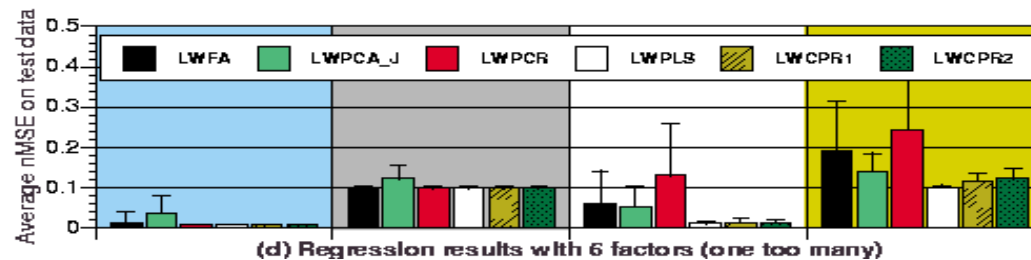
One Projection



Four Projections



Five Projections  
- Correct



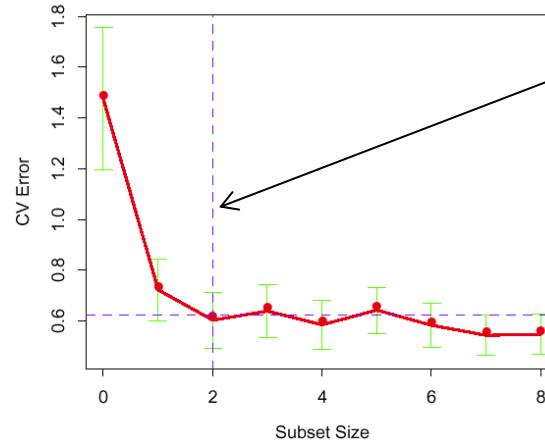
Six Projections



# Comparison of Methods (I)

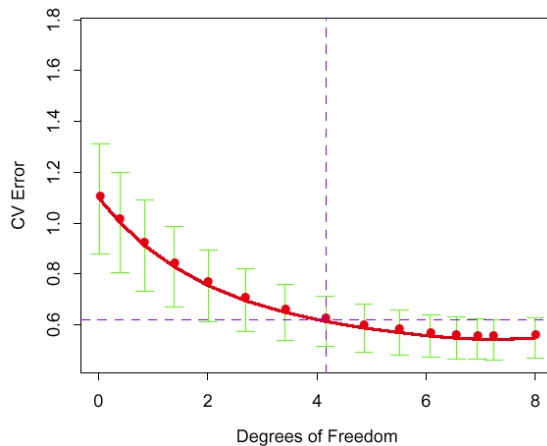
Prostate Cancer Data Example  
(pg. 57, *Elem. Stat. Analysis*)

All Subsets

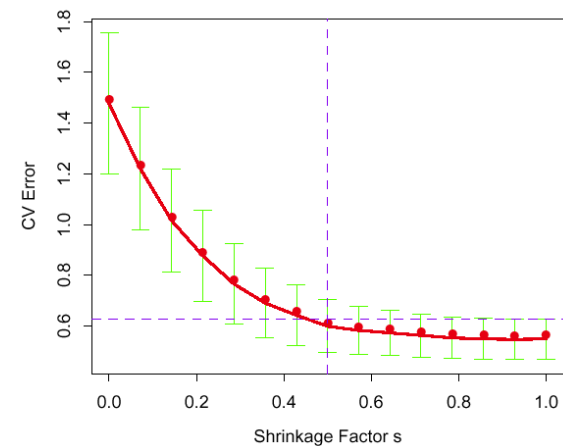


Least complex model within one S.D. of the best

Ridge Regression



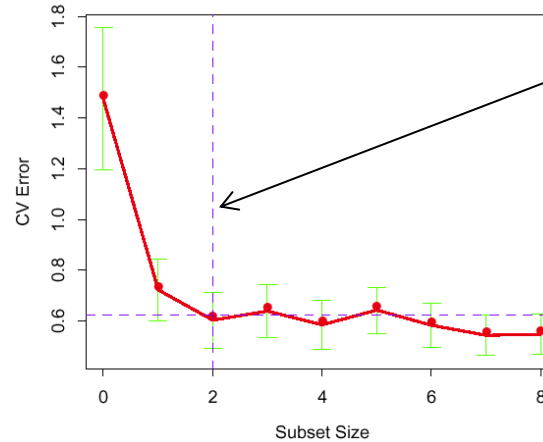
Lasso



# Comparison of Methods (II)

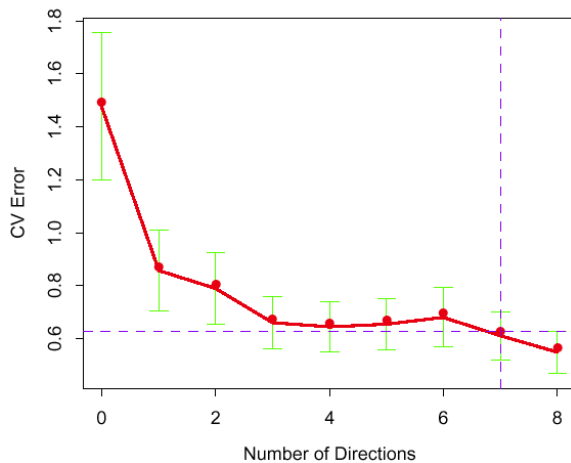
Prostate Cancer Data Example  
(pg. 57, *Elem. Stat. Analysis*)

All Subsets

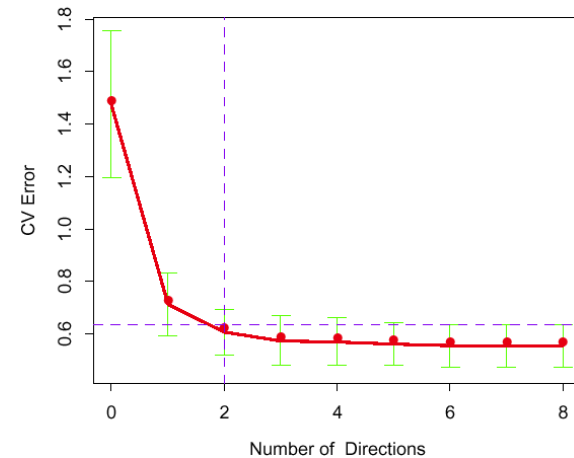


Least complex model within one S.D. of the best

Principal Components Regression



Partial Least Squares



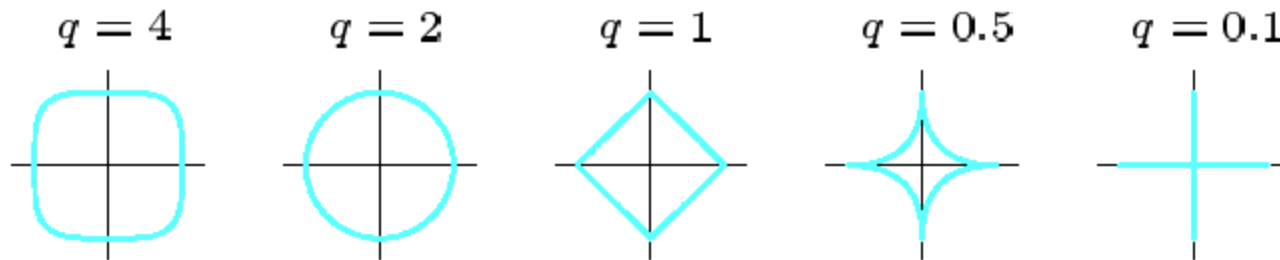
# Generalization of Shrinkage Methods

$$\hat{w}^{gen} = \arg \min_w \left\{ \sum_{i=1}^N (t_i - w_0 - \sum_{j=1}^M x_{ij} w_j)^2 + \lambda \sum_{j=1}^M |w_j|^q \right\} \quad \text{for } q \geq 0.$$

$q = 0$ : variable subset selection

$q = 1$ : lasso

$q = 2$ : ridge regression



Contours of constant value of  $\sum_{j=1}^M |w_j|^q$  for given value of  $q$ .

# Error- Regularization Tradeoff

