Rensselaer

why not change the world?®

Projection-based Chemometrics and Deep Reconstruction

Dr. Uwe Kruger

Department of Biomedical Engineering
Jonsson Engineering Center
Rensselaer Polytechnic Institute

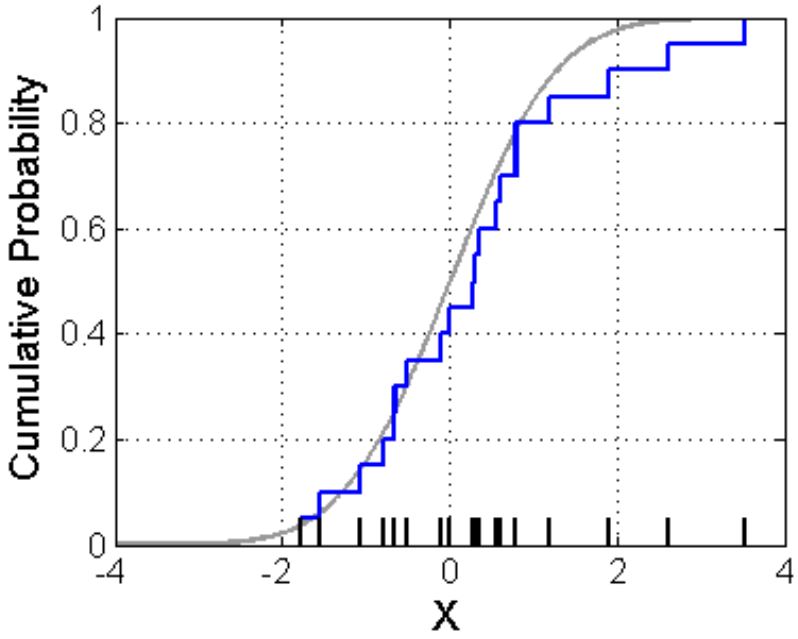
Presentation Outline

- **Motivation for kernel-based methods (kernel density estimation)**
- **Principal Component Analysis (PCA) and Kernel principal component analysis (KPCA)**
- **Partial Least Squares (PLS) and Kernel partial least squares (KPLS)**
- **Some ideas on how to integrate nonlinear projection-based methods for network pruning and detecting/diagnosing anomalies.**

Motivation for Kernel-Based methods

- Let's examine a very simple approach to motivate Cover's theorem and the idea behind reproducing kernels:
- How can we estimate the cumulative distribution function of a random variable X using a set of n observations drawn from the distribution of X?
- Let's try the following naïve estimator:

$$\hat{F}(x) = \frac{\#\, x_i \leq x}{n} = \frac{S(x)}{n}$$



Motivation for Kernel-Based methods

- OK, the n observations, if assumed to be drawn independently, can be used to formulate a total of n Bernoulli trials (like flipping a coin)
  - two outcomes, the value can be larger or smaller than x;
  - the probability to be smaller then x (success) is equal to the cumulative probability distribution function for x, i.e. $F(x)$; and
  - for the *i*th draw (drawing the *i*th value of the random variable X), the probability that x_i is smaller than or equal to x is $F(x)$ for $1 \leq i \leq n$.
- Under these assumptions, $S(x)$ has a binomial distribution with *n* degrees of freedom and the probability of success is $F(x)$:

$$S(x) \sim B\{n, p = F(x)\} \qquad E\{S(x)\} = np = nF(x)$$

$$f(x) = \binom{n}{x} p^x (1-p)^{n-x} \qquad V\{S(x)\} = np(1-p) = nF(x)(1-F(x))$$

Motivation for Kernel-Based methods

- OK, this implies that the naïve estimator is unbiased:

$$E\left\{\hat{F}(x)\right\} = \frac{E\{S(x)\}}{n} = \frac{nF(x)}{n} = F(x)$$

$$V\left\{\hat{F}(x)\right\} = \frac{V\{S(x)\}}{n^2} = \frac{nF(x)(1-F(x))}{n^2} = \frac{F(x)(1-F(x))}{n}$$

$$\lim_{n\to\infty} V\left\{\hat{F}(x)\right\} \to 0$$

$$\lim_{n\to\infty} \hat{F}(x) = \lim_{n\to\infty} F(x)$$

- This follows from simple asymptotics!
- We can develop this one step further by utilizing the fact that the Binomial distribution can be approximated by a normal distribution with a reasonable degree of accuracy, meaning a large enough sample size: $np > 5$ and $n(1-p) > 5$!

Motivation for Kernel-Based methods

- Let's define a new random variable first:

$$Z(x) = \frac{S(x) - nF(x)}{\sqrt{nF(x)(1-F(x))}} \sim N\{0,1\}$$

$$Z(x) = \frac{(\#\, x_i \leq x) - nF(x)}{\sqrt{nF(x)(1-F(x))}}$$

$$-1.96 \leq \frac{(\#\, x_i \leq x) - nF(x)}{\sqrt{nF(x)(1-F(x))}} \leq 1.96$$

$$nF(x) - 1.96\sqrt{nF(x)(1-F(x))} \leq (\#\, x_i \leq x) \leq nF(x) + 1.96\sqrt{nF(x)(1-F(x))}$$

- The above confidence interval is computed for a significance of α=0.05!
- OK, let's move on and convert this into an integral equation, one second…

Motivation for Kernel-Based methods

$$nF(x)-1.96\sqrt{nF(x)(1-F(x))} \leq \int_{-\infty}^{x}\sum_{i=1}^{n}\delta(\xi-x_i)\,\mathrm{d}\,\xi \leq nF(x)+1.96\sqrt{nF(x)(1-F(x))}$$

$$\int_{-\infty}^{x}\delta(\xi-x_i)\,\mathrm{d}\,\xi = \begin{cases}1 \text{ if } x_i \leq x\\ 0 \text{ if } x_i > x\end{cases}$$

$$F(x)-1.96\sqrt{\frac{F(x)(1-F(x))}{n}} \leq \int_{-\infty}^{x}\frac{1}{n}\sum_{i=1}^{n}\delta(\xi-x_i)\,\mathrm{d}\,\xi \leq F(x)+1.96\sqrt{\frac{F(x)(1-F(x))}{n}}$$

$$F(x)-1.96\sqrt{\frac{F(x)(1-F(x))}{n}} \leq \int_{-\infty}^{x}\frac{1}{n}\sum_{i=1}^{n}\underbrace{K(\xi-x_i)}_{\substack{\text{slightlyless "spiky"}\\ \text{Dirac delta function}}}\,\mathrm{d}\,\xi \leq F(x)+1.96\sqrt{\frac{F(x)(1-F(x))}{n}}$$

$$f(x)-1.96\frac{\mathrm{d}\left(\sqrt{\frac{F(x)(1-F(x))}{n}}\right)}{\mathrm{d}\,x} \leq \frac{1}{n}\sum_{i=1}^{n}K(x-x_i) \leq f(x)+1.96\frac{\mathrm{d}\left(\sqrt{\frac{F(x)(1-F(x))}{n}}\right)}{\mathrm{d}\,x}$$

Motivation for Kernel-Based methods

- So what have we got?

$$f(x)-1.96\frac{\mathrm{d}\left(\sqrt{\frac{F(x)(1-F(x))}{n}}\right)}{\mathrm{d}\,x}\leq\frac{1}{n}\sum_{i=1}^{n}K(x-x_i)\leq f(x)+1.96\frac{\mathrm{d}\left(\sqrt{\frac{F(x)(1-F(x))}{n}}\right)}{\mathrm{d}\,x}$$

$$\lim_{n\to\infty} f(x)\pm 1.96\frac{\mathrm{d}\left(\sqrt{\frac{F(x)(1-F(x))}{n}}\right)}{\mathrm{d}\,x}=\lim_{n\to\infty} f(x)\pm\frac{1}{\sqrt{n}}\frac{\mathrm{d}\left(\sqrt{F(x)(1-F(x))}\right)}{\mathrm{d}\,x}\to f(x)$$

$$\lim_{n\to\infty}\frac{1}{n}\sum_{i=1}^{n}K(x-x_i)\to f(x)$$

- All we said about the slightly less spiky Dirac delta function is that its integral must be equal to one, so how about defining it as follows:

$$K(x-x_i)=\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{1}{2}\left(\frac{x-x_i}{\sigma}\right)^2}\Rightarrow\lim_{\sigma\to 0}K(x-x_i)\to\delta(x-x_i)$$

Kernel Density Estimation

- The function $K(x - x_i)$ is referred to as a kernel function and the derivative shows that, asymptotically, the estimate:

$$\frac{1}{n}\sum_{i=1}^{n} K(x - x_i)$$

  converges to the true probability density function for any value of x. The above estimator is defined as a kernel density estimator.

- Along the same lines, we can also develop an approach to develop nonlinear counterpart of data-driven chemometric modeling techniques, such as principal component analysis (PCA) and partial least squares (PLS).
- Essentially, an artificial neural network can be seen as a kernel-based nonlinear modeling technique, *i.e*. the neurons are, effectively, small kernels.
- Let's start with PCA first, after some more discussions on kernels.

Kernel Density Estimation

- Theoretically, kernel functions other than the Gaussian kernel:

$$K(x - x_i) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{x - x_i}{\sigma}\right)^2}$$

  can be considered if their area is equal to 1 and include the Epanechnikov, the triangular and the uniform kernel among others.

- Theoretically, the derivative showed that the shape of the kernel function does not influence the estimate in an asymptotic sense.

- Practically, however, the shape of the kernel function does influence the accuracy of the estimate. This yields the following general form of the kernel density estimator:

$$\frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right), \quad K\left(\frac{x - x_i}{h}\right) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x - x_i}{h}\right)^2}, \quad h \rightarrow \text{bandwidth}$$

Kernel Principal Component Analysis - Introduction

- Kernel PCA is a generic nonlinear extension to linear PCA (Kruger *et al*., 2008).
- Let's look at some basics before we go into the kernel stuff.

$$\mathbf{z} = \mathbf{A}\mathbf{s} \qquad \dim\{\mathbf{z}\} > \dim\{\mathbf{s}\} \qquad E\{\mathbf{z}\} = \mathbf{A}E\{\mathbf{s}\} = \mathbf{0}$$

$$\mathbf{Z} = \begin{bmatrix} \mathbf{z}_1^T \\ \mathbf{z}_2^T \\ \vdots \\ \mathbf{z}_n^T \end{bmatrix} = \begin{bmatrix} \mathbf{s}_1^T \\ \mathbf{s}_2^T \\ \vdots \\ \mathbf{s}_n^T \end{bmatrix} \mathbf{A}^T = \mathbf{U}\mathbf{L}\mathbf{P}^T \Rightarrow \text{singular value decomposition}$$

- Next, let's define the following two matrices:

$$\boldsymbol{\Sigma}_z = \tfrac{1}{n}\mathbf{Z}^T\mathbf{Z} = \mathbf{P}\left[\tfrac{1}{n}\mathbf{L}^2\right]\mathbf{P}^T \rightarrow \text{data covariance matrix and its eigendecom position}$$

$$\boldsymbol{\Phi}_z(\mathbf{Z},\mathbf{Z}) = \mathbf{Z}\mathbf{Z}^T = \mathbf{U}\left[\mathbf{L}^2\right]\mathbf{U}^T \rightarrow \text{Gram matrix and its eigendecom position}$$

Kernel Principal Component Analysis - Introduction

- Let's see how we can determine the unknown source variables (up to a similarity transformation) – which are the principal components:

$$\begin{bmatrix} \mathbf{s}_1^T \\ \mathbf{s}_2^T \\ \vdots \\ \mathbf{s}_n^T \end{bmatrix} \mathbf{A}^T = \mathbf{U}\mathbf{L}\mathbf{P}^T \Rightarrow \mathbf{S} \propto \mathbf{U}\mathbf{L} = \mathbf{T}, \mathbf{A} \propto \mathbf{P}$$

$$\mathbf{t} = \mathbf{P}^T\mathbf{z} = \mathbf{L}^{-1}\mathbf{U}^T\mathbf{z}\mathbf{z} = \mathbf{L}^{-1}\mathbf{U}^T\boldsymbol{\Phi}_z(\mathbf{Z}, \mathbf{z}) \text{ given that } \mathbf{Z} = \mathbf{U}\mathbf{L}\mathbf{P}^T \Rightarrow \mathbf{P}^T = \mathbf{L}^{-1}\mathbf{U}^T\mathbf{Z}$$

- Let's make the relationship between the source variables and the measured variables nonlinear, *i.e.*:

$$\mathbf{z} = \boldsymbol{\theta}(\mathbf{s}), \quad \mathbf{f} = \boldsymbol{\psi}(\mathbf{z}), \quad \mathbf{t} = \mathbf{P}^T\mathbf{f}, \quad \mathbf{F} = \begin{bmatrix} \boldsymbol{\psi}^T(\mathbf{z}_1) \\ \boldsymbol{\psi}^T(\mathbf{z}_2) \\ \vdots \\ \boldsymbol{\psi}^T(\mathbf{z}_n) \end{bmatrix}$$ which we assume to be bijective!

Kernel Principal Component Analysis - Introduction

- Let's define the Gram matrix

$$\boldsymbol{\Phi}_z(\mathbf{Z},\mathbf{Z}) = \underbrace{\left[\mathbf{I} - \tfrac{1}{n}\mathbf{1}\mathbf{1}^T\right]}_{\text{incorporating mean centering}} \underbrace{\mathbf{F}\mathbf{F}^T}_{\text{defined as the kernel matrix}} \underbrace{\left[\mathbf{I} - \tfrac{1}{n}\mathbf{1}\mathbf{1}^T\right]^T}_{\text{incorporating mean centering}}$$

$$\mathbf{F}\mathbf{F}^T = \begin{bmatrix} \boldsymbol{\psi}^T(\mathbf{z}_1)\boldsymbol{\psi}(\mathbf{z}_1) & \boldsymbol{\psi}^T(\mathbf{z}_1)\boldsymbol{\psi}(\mathbf{z}_2) & \cdots & \boldsymbol{\psi}^T(\mathbf{z}_1)\boldsymbol{\psi}(\mathbf{z}_n) \\ \boldsymbol{\psi}^T(\mathbf{z}_2)\boldsymbol{\psi}(\mathbf{z}_1) & \boldsymbol{\psi}^T(\mathbf{z}_2)\boldsymbol{\psi}(\mathbf{z}_2) & \cdots & \boldsymbol{\psi}^T(\mathbf{z}_2)\boldsymbol{\psi}(\mathbf{z}_n) \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{\psi}^T(\mathbf{z}_n)\boldsymbol{\psi}(\mathbf{z}_1) & \boldsymbol{\psi}^T(\mathbf{z}_n)\boldsymbol{\psi}(\mathbf{z}_2) & \cdots & \boldsymbol{\psi}^T(\mathbf{z}_n)\boldsymbol{\psi}(\mathbf{z}_n) \end{bmatrix} = \mathbf{K}(\mathbf{Z},\mathbf{Z})$$

$$\mathbf{K}(\mathbf{Z},\mathbf{Z}) = \begin{bmatrix} k_{11} & k_{12} & \cdots & k_{1n} \\ k_{21} & k_{22} & \cdots & k_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ k_{n1} & k_{n2} & \cdots & k_{nn} \end{bmatrix}$$

Kernel Principal Component Analysis - Introduction

- Let's repeat the "trick" we did when estimating the probability density function using the kernel density estimator using kernels:

$$k_{ij} = \boldsymbol{\psi}^T(\mathbf{z}_i)\boldsymbol{\psi}(\mathbf{z}_j) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{\|\mathbf{z}_i - \mathbf{z}_j\|}{\sigma}\right)^2}$$

$$\mathbf{K}(\mathbf{Z},\mathbf{Z}) = \frac{1}{\sqrt{2\pi}\sigma}\begin{bmatrix} 1 & e^{-\frac{1}{2}\left(\frac{\|\mathbf{z}_1 - \mathbf{z}_2\|}{\sigma}\right)^2} & \cdots & e^{-\frac{1}{2}\left(\frac{\|\mathbf{z}_1 - \mathbf{z}_n\|}{\sigma}\right)^2} \\ e^{-\frac{1}{2}\left(\frac{\|\mathbf{z}_2 - \mathbf{z}_1\|}{\sigma}\right)^2} & 1 & \cdots & e^{-\frac{1}{2}\left(\frac{\|\mathbf{z}_2 - \mathbf{z}_n\|}{\sigma}\right)^2} \\ \vdots & \vdots & \ddots & \vdots \\ e^{-\frac{1}{2}\left(\frac{\|\mathbf{z}_n - \mathbf{z}_1\|}{\sigma}\right)^2} & e^{-\frac{1}{2}\left(\frac{\|\mathbf{z}_n - \mathbf{z}_2\|}{\sigma}\right)^2} & \cdots & 1 \end{bmatrix}$$

Kernel Principal Component Analysis - Introduction

- Let's finalize the definition of the Gram matrix:

$$\mathbf{\Phi}_z(\mathbf{Z},\mathbf{Z}) = \mathbf{K}(\mathbf{Z},\mathbf{Z}) - \tfrac{1}{n}\mathbf{K}(\mathbf{Z},\mathbf{Z})\mathbf{1}\mathbf{1}^T - \tfrac{1}{n}\mathbf{1}\mathbf{1}^T\mathbf{K}(\mathbf{Z},\mathbf{Z}) + \tfrac{1}{n^2}\mathbf{1}\mathbf{1}^T\mathbf{K}(\mathbf{Z},\mathbf{Z})\mathbf{1}\mathbf{1}^T$$

- Next, we carry out the eigendecomposition of the Gram matrix:

$$\mathbf{\Phi}_z(\mathbf{Z},\mathbf{Z}) = \mathbf{U}\mathbf{L}\mathbf{U}^T$$

- In a similar fashion to PCA, we can now determine the principal components:

$$\mathbf{t} = \underbrace{\mathbf{L}^{-1}\mathbf{U}^T\left[\mathbf{I} - \tfrac{1}{n}\mathbf{1}\mathbf{1}^T\right]}_{\mathbf{A}^T}\mathbf{F}\left(\boldsymbol{\psi}(\mathbf{z}) - \underbrace{\overline{\boldsymbol{\psi}}}_{\frac{1}{n}\mathbf{F}^T\mathbf{1}}\right) = \underbrace{\mathbf{A}^T\left(\mathbf{k}(\mathbf{Z},\mathbf{z}) - \tfrac{1}{n}\mathbf{K}(\mathbf{Z},\mathbf{Z})\mathbf{1}\right)}_{\text{like a neural network, this is a weighted sum of basis functions}}$$

$$\mathbf{t} = \mathbf{A}^T\left(\mathbf{k}(\mathbf{Z},\mathbf{z}) - \overline{\mathbf{k}}\right)$$

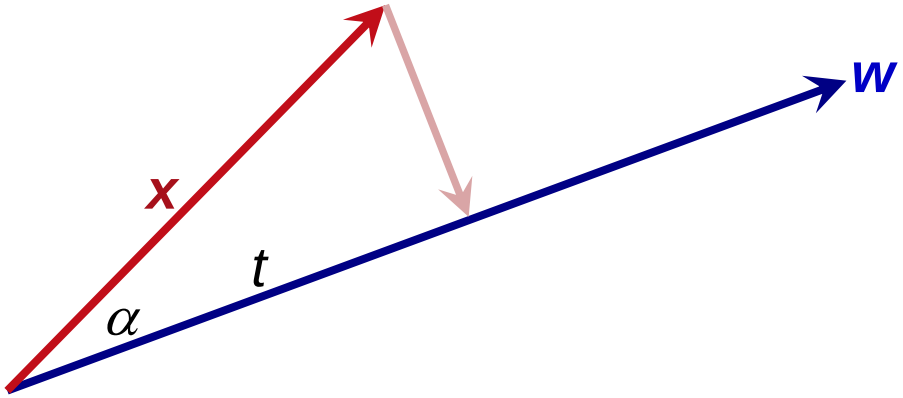
Kernel Principal Component Analysis - Introduction

- Asymptotically, $n \rightarrow \infty$, the shape of the basic kernel function is not important.
- Theoretically, and this follows from the properties of reproducing kernels, any function can be constructed in the feature space that maps the nonlinear surface in the data space to become a plane (subspace) in the feature space.
- The projection in the feature space then yields linear principal components in the feature space that are related to the source variables in the original variable space – connected through the following mappings:

$$\mathbf{z} = \boldsymbol{\theta}(\mathbf{s}), \quad \mathbf{f} = \boldsymbol{\psi}(\mathbf{z}), \quad \mathbf{t} = \mathbf{A}^T(\mathbf{f} - \bar{\mathbf{f}}), \quad \mathbf{z} = \tilde{\boldsymbol{\theta}}(\mathbf{t})$$

Partial Least Squares – Introduction

- Let's examine the geometric framework that underpins the partial least squares concept: orthogonally projecting the data points onto directions for the predictor space:



$$\cos(\alpha) = \frac{\boldsymbol{x}^T\boldsymbol{w}}{\|\boldsymbol{x}\|\|\boldsymbol{w}\|}$$

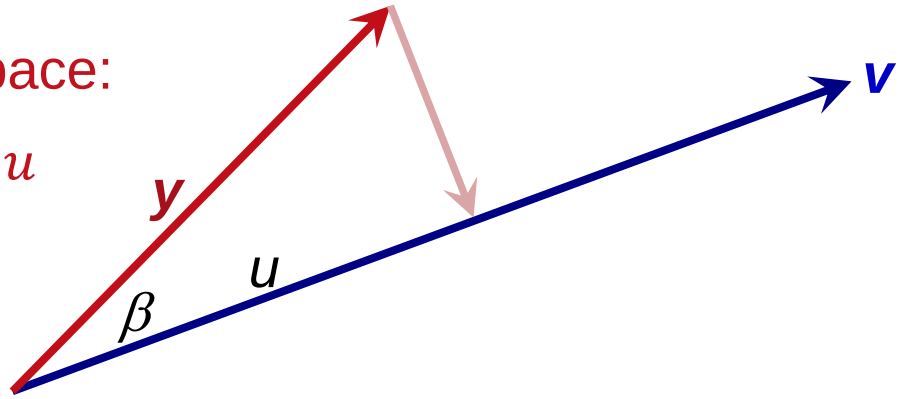
with $\|\boldsymbol{w}\| = 1$, we get

$$\cos(\alpha)\|\boldsymbol{x}\| = \boldsymbol{x}^T\boldsymbol{w} = t$$

and the response space:

$$\cos(\beta)\|\boldsymbol{y}\| = \boldsymbol{y}^T\boldsymbol{v} = u$$

if $\|\boldsymbol{v}\| = 1$



Partial Least Squares – Introduction

- Let's examine the random vectors – these can be of a considerable dimension – ***X*** and ***Y*** describing the predictor and response sets that are related as follows:

  $\boldsymbol{Y} = \boldsymbol{B}\boldsymbol{X} + \boldsymbol{E}$ - ***E*** being a random vector describing uncertainty

- We could use ordinary least squares to determine the parameter matrix ***B***:

  $\boldsymbol{B} = \boldsymbol{S}_{YX}\boldsymbol{S}_{XX}^{-1}$

- The problem is that if X has a very large dimension, the inverse of the covariance matrix $\boldsymbol{S}_{XX}$ may not exist or is badly conditioned!

- Here is where PLS comes in! Using the projections we discussed before:

  $T = \boldsymbol{X}^T\boldsymbol{w}$ and $U = \boldsymbol{Y}^T\boldsymbol{v}$

- Now, we select the random variables *T* and *U* such that they maximize their covariance!

Partial Least Squares – Introduction

- This yields the following objective function:

$$J = E\{TU\} - \lambda_1(\boldsymbol{w}^T\boldsymbol{w} - 1) - \lambda_2(\boldsymbol{v}^T\boldsymbol{v} - 1)$$

$$J = \boldsymbol{w}^T E\{\boldsymbol{X}\boldsymbol{Y}^T\}\boldsymbol{v} - \lambda_1(\boldsymbol{w}^T\boldsymbol{w} - 1) - \lambda_2(\boldsymbol{v}^T\boldsymbol{v} - 1)$$

$$J = \boldsymbol{w}^T\boldsymbol{S}_{XY}\boldsymbol{v} - \lambda_1(\boldsymbol{w}^T\boldsymbol{w} - 1) - \lambda_2(\boldsymbol{v}^T\boldsymbol{v} - 1)$$

$$\frac{\partial J}{\partial \boldsymbol{w}} = \boldsymbol{S}_{XY}\boldsymbol{v} - 2\lambda_1\boldsymbol{w} = \boldsymbol{0}$$

$$\frac{\partial J}{\partial \boldsymbol{v}} = \boldsymbol{S}_{YX}\boldsymbol{w} - 2\lambda_2\boldsymbol{v} = \boldsymbol{0}$$

$$\boldsymbol{S}_{XY}\boldsymbol{S}_{YX}\boldsymbol{w} = 4\lambda_1\lambda_2\boldsymbol{w}$$

$$\boldsymbol{S}_{YX}\boldsymbol{S}_{XY}\boldsymbol{v} = 4\lambda_1\lambda_2\boldsymbol{v}$$

- So, we now have the direction vectors ***w*** and ***v*** in both spaces!
- That also means that we have the random variables T and U!
- Whilst ***X*** predicts ***Y***, PLS utilizes *T* – instead of ***X*** – to predict ***Y***!

Partial Least Squares – Introduction

- With $T = \boldsymbol{X}^T \boldsymbol{w}$, we get:

  $\boldsymbol{F} = \boldsymbol{X} - \boldsymbol{T}\boldsymbol{p}$ and $\boldsymbol{E} = \boldsymbol{Y} - \boldsymbol{T}\boldsymbol{q}$ – these being the residual vectors for $\boldsymbol{X}$ and $\boldsymbol{Y}$, respectively.

- The parameter vectors $\boldsymbol{p}$ and $\boldsymbol{q}$ can be obtained by solving two least squares regression problems – minimizing the length of the residual vectors:

  $\boldsymbol{p} = \frac{E\{\boldsymbol{X}T\}}{E\{T^2\}}$ and $\boldsymbol{q} = \frac{E\{\boldsymbol{Y}T\}}{E\{T^2\}}$

- After that, the PLS algorithm can be repeated using the residual vectors $\boldsymbol{F}$ and $\boldsymbol{E}$ instead of the original random vectors $\boldsymbol{X}$ and $\boldsymbol{Y}$.

- This gives rise to the following iterative algorithm, which is referred to as the standard PLS algorithm and detailed on the next slide. This algorithm was first published by Herman Wold in 1966.

Partial Least Squares – Algorithm

- Let **X** and **Y** be two data matrices that store a total of n data points drawn from the random vectors ***X*** and ***Y***, respectively:

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1N} \\ x_{21} & x_{22} & \cdots & x_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nN} \end{bmatrix} \quad \mathbf{Y} = \begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1N} \\ y_{21} & y_{22} & \cdots & y_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ y_{n1} & y_{n2} & \cdots & y_{nN} \end{bmatrix}$$

- The first step is to normalize the data matrices, *i.e*. the observations in each column are mean centered and scaled to have a unit variance:

$$\text{Sample mean vectors} : \bar{\mathbf{x}} = \frac{1}{n}\mathbf{X}^T\mathbf{1}$$

$$: \bar{\mathbf{y}} = \frac{1}{n}\mathbf{Y}^T\mathbf{1}$$

$$\text{Sample variance vectors} : \boldsymbol{\sigma}_X^2 = \frac{1}{n-1}\text{diag}\left(\left[\mathbf{X} - \mathbf{1}\bar{\mathbf{x}}^T\right]^T\left[\mathbf{X} - \mathbf{1}\bar{\mathbf{x}}^T\right]\right)$$

$$: \boldsymbol{\sigma}_Y^2 = \frac{1}{n-1}\text{diag}\left(\left[\mathbf{Y} - \mathbf{1}\bar{\mathbf{x}}^T\right]^T\left[\mathbf{Y} - \mathbf{1}\bar{\mathbf{x}}^T\right]\right)$$

$$\text{Normalizing both matrices} : \mathbf{X}_0 = \left[\mathbf{X} - \mathbf{1}\bar{\mathbf{x}}^T\right]\left[\text{diag}\left(\boldsymbol{\sigma}_X^2\right)\right]^{-1/2} \quad \mathbf{Y}_0 = \left[\mathbf{Y} - \mathbf{1}\bar{\mathbf{y}}^T\right]\left[\text{diag}\left(\boldsymbol{\sigma}_Y^2\right)\right]^{-1/2}$$

Partial Least Squares – Algorithm

- Next, define the covariance and cross-covariance matrices:

$$\text{Sample covariance matrix} : \mathbf{S}_{XX}^{(0)} = \frac{1}{n-1}\mathbf{X}_0^T\mathbf{X}_0, \quad \mathbf{M}_{XX}^{(0)} = (n-1)\mathbf{S}_{XX}^{(0)}$$

$$\text{Sample cross-covariance matrix} : \mathbf{S}_{XY}^{(0)} = \frac{1}{n-1}\mathbf{X}_0^T\mathbf{Y}_0, \quad \mathbf{M}_{XY}^{(0)} = (n-1)\mathbf{S}_{XY}^{(0)}$$

- Setup the PLS iteration and determining the regression vectors

for $i = 1:m$

$\varepsilon = 100;$

$\mathbf{w}_0 = \mathbf{M}_{XX}^{(i-1)}(:,1) / \text{norm}\left(\mathbf{M}_{XX}^{(i-1)}(:,1)\right);$

while $\varepsilon < 1e\text{-}10$

$\mathbf{v} = \mathbf{M}_{YX}^{(i-1)}\mathbf{w}_0;$

$\mathbf{v} = \mathbf{v}/\text{norm}(\mathbf{v});$

$\mathbf{w} = \mathbf{M}_{XY}^{(i-1)}\mathbf{v};$

$\mathbf{w}_u = \mathbf{w}/\text{norm}(\mathbf{w});$

$\varepsilon = \text{norm}(\mathbf{w}_u - \mathbf{w}_0);$

$\mathbf{w}_0 = \mathbf{w}_u;$

end

$\mathbf{w}_i = \mathbf{w}_0;$

$\mathbf{p}_i = \mathbf{M}_{XX}^{(i-1)}\mathbf{w}_i / \left(\mathbf{w}_i^T\mathbf{M}_{XX}^{(i-1)}\mathbf{w}_i\right);$

$\mathbf{q}_i = \mathbf{M}_{YX}^{(i-1)}\mathbf{w} / \left(\mathbf{w}_i^T\mathbf{M}_{XX}^{(i-1)}\mathbf{w}_i\right);$

$\mathbf{X}_i = \mathbf{X}_{i-1} - \mathbf{X}_{i-1}\mathbf{w}_i\mathbf{p}_i^T;$

$\mathbf{Y}_i = \mathbf{Y}_{i-1} - \mathbf{X}^{(i-1)}\mathbf{w}_i\mathbf{q}_i^T;$

$\mathbf{M}_{XX}^{(i)} = \mathbf{X}_i^T\mathbf{X}_i;$

$\mathbf{M}_{XY}^{(i)} = \mathbf{X}_i^T\mathbf{Y}_i;$

$\mathbf{P}(:,\text{i}) = \mathbf{p}_\text{i};$

$\mathbf{Q}(:,\text{i}) = \mathbf{q}_\text{i};$

$\mathbf{W}(:,\text{i}) = \mathbf{w}_\text{i};$

end

Kernel Partial Least Squares – Algorithm

- The regression matrix can now be estimated as follows:

$$\mathbf{B} = \mathbf{Q}\left[\mathbf{W}^T\mathbf{P}\right]^{-1}\mathbf{W}^T$$

- To establish a nonlinear extension of the standard PLS algorithm, let's look at the standard algorithm again:

$$\lambda\mathbf{v} = \mathbf{Y}_0^T \quad \underbrace{\mathbf{X}_0\mathbf{X}_0^T}_{\text{This is a Gram matrix}} \quad \mathbf{Y}_0\mathbf{v}$$

- We can "kernelize" the above Gram matrix by using the following nonlinear transformation involving the random vectors ***X***, ***Y*** and ***E***:

$$\boldsymbol{G} = \boldsymbol{\psi}\left(\boldsymbol{X}\right) \quad \boldsymbol{Y} = \mathbf{B}\boldsymbol{G} + \boldsymbol{E}$$

- Based on the data matrix **X**, we get:

$$\mathbf{G} = \boldsymbol{\psi}\left(\mathbf{X}_0\right) \quad \mathbf{Y}_0 = \mathbf{G}_0\mathbf{B}^T + \mathbf{E}$$

Kernel Partial Least Squares – Algorithm

- Using the Gram matrix based on $\boldsymbol{\psi}(\mathbf{X}_0)$, we can compute the projection vector **v** as follows:

$$\lambda\mathbf{v} = \mathbf{Y}_0^T \underbrace{\left[\mathbf{I} - \tfrac{1}{n}\mathbf{1}\mathbf{1}^T\right]\mathbf{G}\mathbf{G}^T\left[\mathbf{I} - \tfrac{1}{n}\mathbf{1}\mathbf{1}^T\right]}_{\text{This is } \boldsymbol{\Phi}_X(\mathbf{X}_0,\mathbf{X}_0), \text{ the Gram matrix for } \boldsymbol{\psi}(\mathbf{X}_0)} \mathbf{Y}_0\mathbf{v}$$

- Once we have **v**, it is easy to compute the vector **u**:

$$\mathbf{u} = \mathbf{Y}_0\mathbf{v}$$

- The next step is to compute the vector **t**. For linear PLS, we can derive the following relationship:

$$\mathbf{t} = \mathbf{X}_0\mathbf{w}; \quad \mathbf{w} \propto \mathbf{M}_{XY}\mathbf{v} = \mathbf{X}_0^T \underbrace{\mathbf{Y}_0\mathbf{v}}_{\mathbf{u}}$$

$$\mathbf{t} \propto \mathbf{X}_0\mathbf{X}_0^T\mathbf{u}$$

Kernel Partial Least Squares – Algorithm

- Instead of the linear Gram matrix $\mathbf{X}_0\mathbf{X}_0^{\mathbf{T}}$, we can also use the nonlinear Gram matrix $\boldsymbol{\Phi}_X(\mathbf{X}_0, \mathbf{X}_0)$, which gives rise to:

$$\mathbf{t} \propto \boldsymbol{\Phi}\left(\mathbf{X}_0, \mathbf{X}_0\right)\mathbf{u}$$

- To address the scaling problem, as we cannot compute the projection vector w, we can scale the vector t to unit length:

$$\mathbf{t} = \mathbf{t}/\mathrm{norm}\left(\mathbf{t}\right)$$

- Now, we can deflate the Gram matrix, $\mathbf{G}_0\mathbf{G}_0^T = \boldsymbol{\Phi}_X(\mathbf{X}_0, \mathbf{X}_0)$:

$$\mathbf{G}_i = \mathbf{G}_{i-1} - \mathbf{t}_{i-1}\mathbf{p}_{i-1}^T = \mathbf{G}_{i-1} - \mathbf{t}_{i-1}\frac{\mathbf{t}_{i-1}^T\mathbf{G}_{i-1}}{\underbrace{\mathbf{t}_{i-1}^T\mathbf{t}_{i-1}}_{1}} = \left[\mathbf{I} - \mathbf{t}_{i-1}\mathbf{t}_{i-1}^T\right]\mathbf{G}_{i-1}$$

$$\mathbf{G}_i\mathbf{G}_i^T = \left[\mathbf{I} - \mathbf{t}_{i-1}\mathbf{t}_{i-1}^T\right]\mathbf{G}_{i-1}\mathbf{G}_{i-1}^T\left[\mathbf{I} - \mathbf{t}_{i-1}\mathbf{t}_{i-1}^T\right]$$

Kernel Partial Least Squares – Algorithm

and the response matrix $\mathbf{Y}_0$:

$$\mathbf{Y}_i = \mathbf{Y}_{i-1} - \mathbf{t}_{i-1}\mathbf{q}_{i-1}^T = \mathbf{Y}_{i-1} - \mathbf{t}_{i-1}\frac{\mathbf{t}_{i-1}^T\mathbf{Y}_{i-1}}{\underbrace{\mathbf{t}_{i-1}^T\mathbf{t}_{i-1}}_{1}} = \left[\mathbf{I} - \mathbf{t}_{i-1}\mathbf{t}_{i-1}^T\right]\mathbf{Y}_{i-1}$$

- The last step is to compute the regression matrix. Again, let's look again at the linear PLS algorithm first:

$$\mathbf{B}^T = \mathbf{W}\left[\mathbf{P}^T\mathbf{W}\right]^{-1}\mathbf{Q}^T \quad \mathbf{Q} = \mathbf{Y}_0^T\mathbf{T}\left[\mathbf{T}^T\mathbf{T}\right]^{-1} \quad \mathbf{P} = \mathbf{X}_0^T\mathbf{T}\left[\mathbf{T}^T\mathbf{T}\right]^{-1} \quad \mathbf{W} \propto \mathbf{X}_0^T\mathbf{U}$$

$$\mathbf{B}^T = \mathbf{X}_0^T\mathbf{U}\left[\left[\mathbf{T}^T\mathbf{T}\right]^{-1}\mathbf{T}^T\mathbf{X}_0\mathbf{X}_0^T\mathbf{U}\right]^{-1}\left[\mathbf{T}^T\mathbf{T}\right]^{-1}\mathbf{T}^T\mathbf{Y}_0$$

$$\mathbf{B}^T = \mathbf{X}_0^T\mathbf{U}\left[\mathbf{T}^T\mathbf{X}_0\mathbf{X}_0^T\mathbf{U}\right]^{-1}\mathbf{T}^T\mathbf{Y}_0$$

- Using this regression matrix for predicting a new observation yields:

$$\hat{\mathbf{y}}_0^T = \mathbf{x}_0^T\mathbf{B}^T = \mathbf{x}_0^T\mathbf{X}_0^T\mathbf{U}\left[\mathbf{T}^T\mathbf{X}_0\mathbf{X}_0^T\mathbf{U}\right]^{-1}\mathbf{T}^T\mathbf{Y}_0$$

Kernel Partial Least Squares – Algorithm

- Finally, replacing the linear Gram matrix and vector by there nonlinear counterparts gives rise to:

$$\hat{\mathbf{y}} = \mathbf{B}\overline{\boldsymbol{\psi}}(\mathbf{x}_0) = \mathbf{Y}^T\mathbf{T}\left[\mathbf{U}^T\overline{\boldsymbol{\psi}}(\mathbf{X}_0)\overline{\boldsymbol{\psi}}^T(\mathbf{X}_0)\mathbf{T}\right]^{-1}\mathbf{U}^T\overline{\boldsymbol{\psi}}(\mathbf{X}_0)\overline{\boldsymbol{\psi}}(\mathbf{x}_0)$$

- Let's put this all together and define the KPLS algorithm.
- Besides the construction of the nonlinear transformation, and with it its Gram matrix, the rest of the algorithm is related to the linear PLS algorithm.
- Compared to artificial neural networks, which have many network weights, the "only" parameter that needs to be specified is the kernel parameter. The remaining parameter are obtained by a linear regression problem an solved using the robust PLS algorithm!

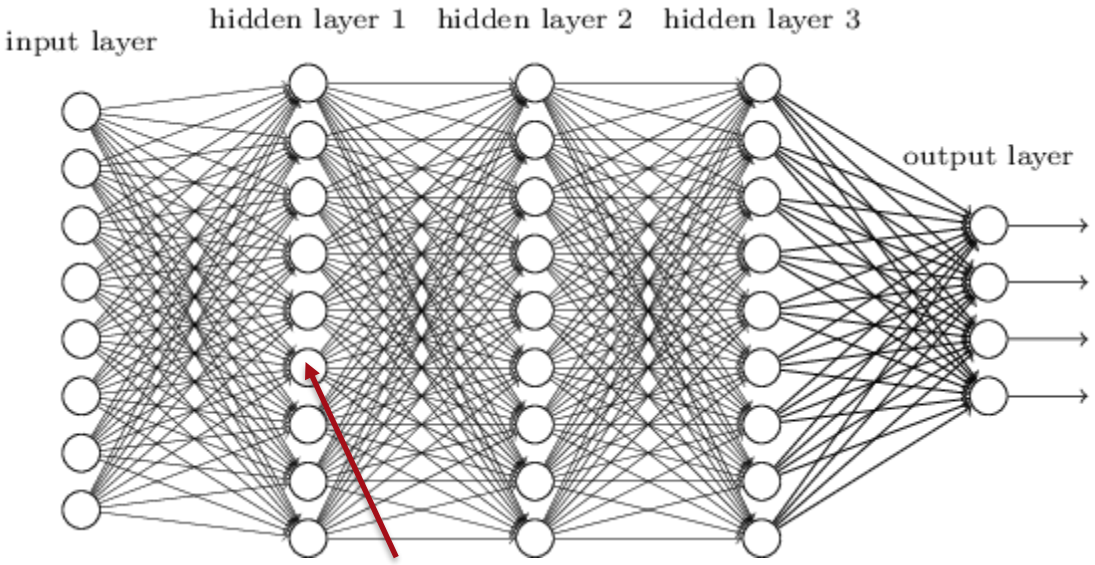
How could chemometric techniques assist in dealing with very large network architechtures?

- Disclaimer: KPLS is not a substitute to deep learning architectures!
- KPLS does run into problems if the number of data points increase, say beyond 10,000 (remember the size of the Gram matrix is equal to the number of data points squared)
- A KPLS model has the potential to outperforms competitive artificial neural network models when the number of variables x or y are larger and/or the number of data points is small.

- To see how KPCA and KPLS could be useful tools, let's examine the structure of large network topologies on the next slide in more detail

How could chemometric techniques assist in dealing with very large network architechtures?

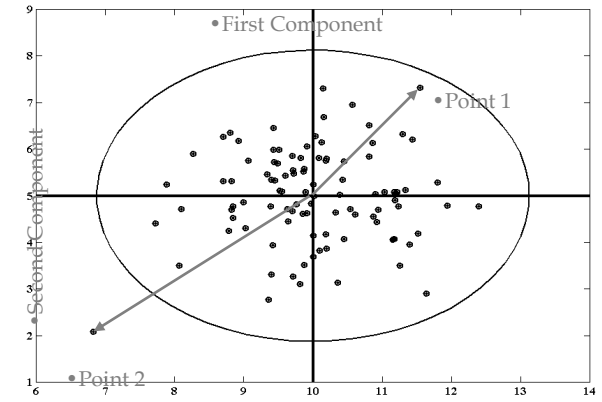
- Starting from a “small” (trained) network:

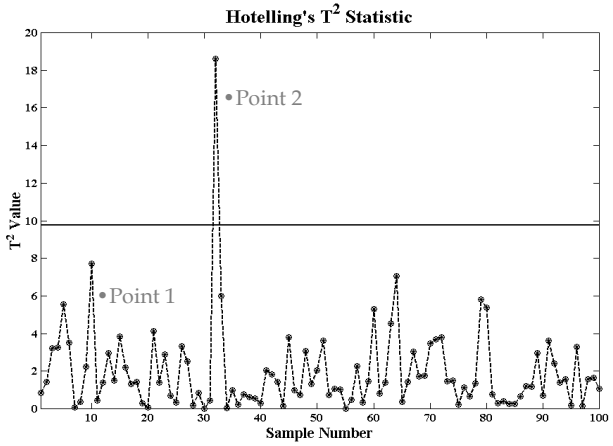


1. how do we know that this neuron is important or could be discarded if it contributes negligibly to the accuracy of the network prediction – *e.g.* for specific tasks (set of lung images)?
2. secondly, how can we statistically examine significant differences in the individual layers/layer combinations if we have two sets of images (one set that is labeled normal, whilst the other set is labeled as containing anomalies)?

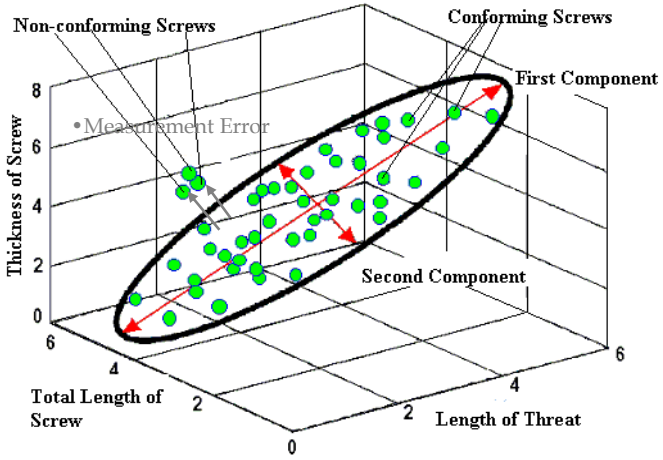
Abnormality Detection (Basis Multivariate Approach)

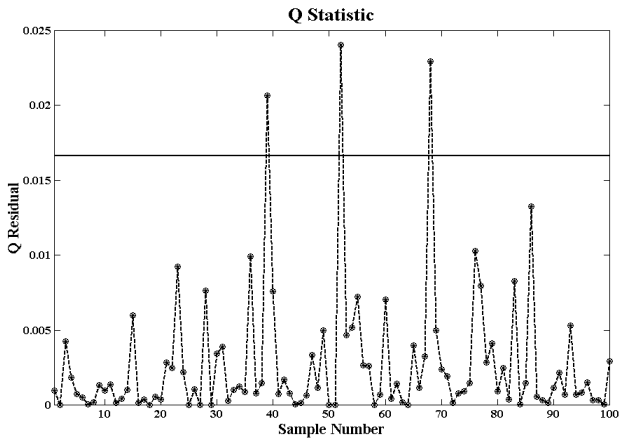
- Hotelling's T^2 Statistic





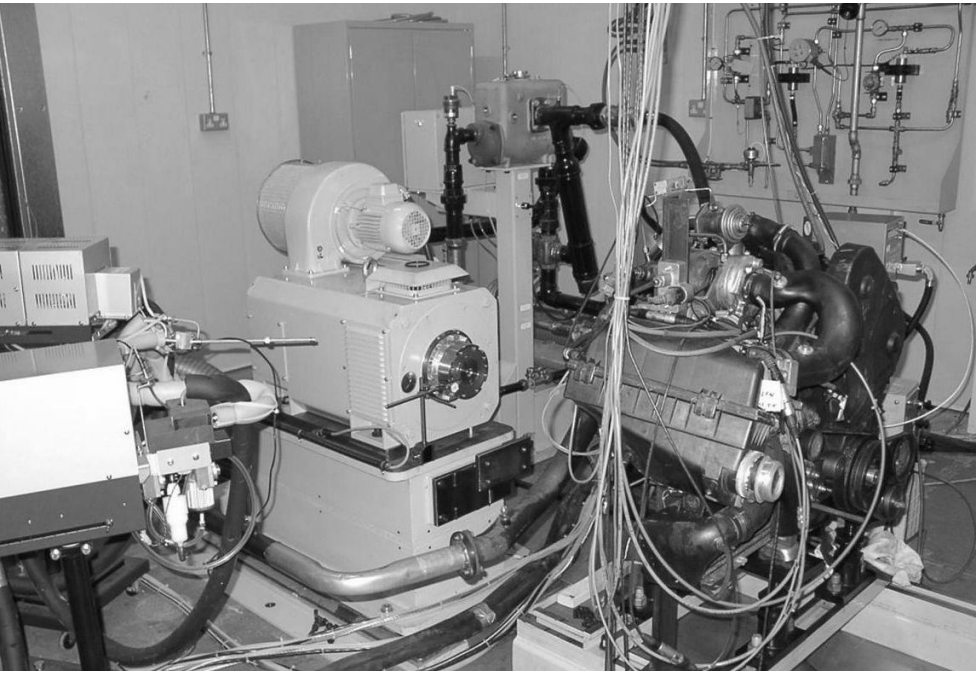
- Q Statistic

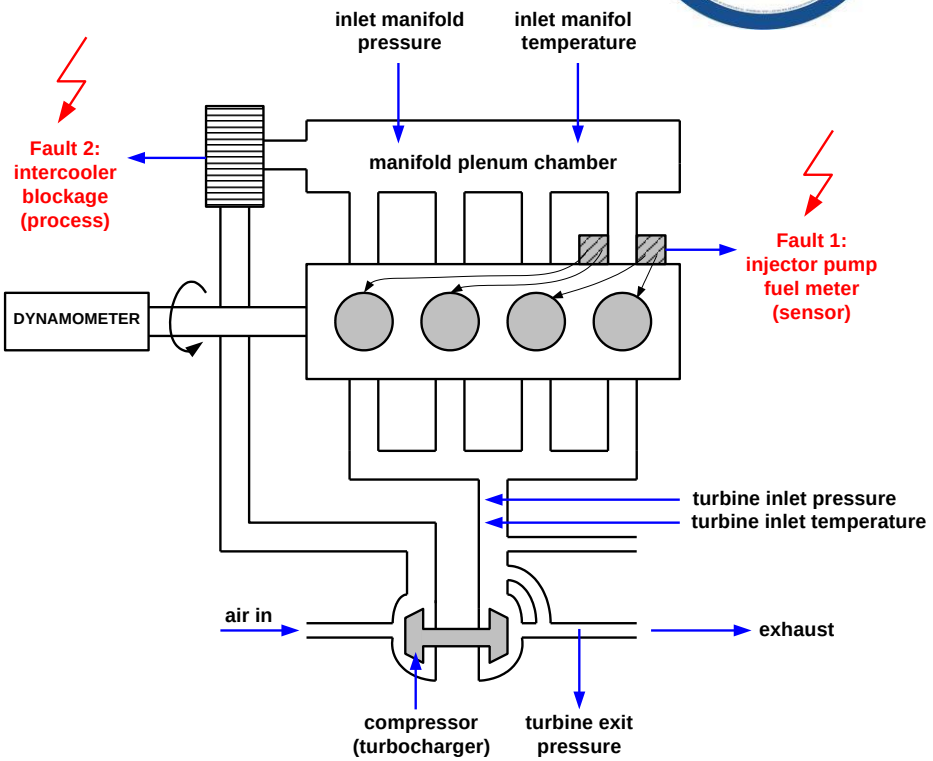




Application to Fault Diagnosis (Internal Combustion Engine)

- Analysis of data from a Volkswagen 1.9L TDI diesel engine.
- Various fault conditions were recorded and diagnosed.





Application to Fault Diagnosis (Internal Combustion Engine)

Variables analysed

No	Engine Variable	Unit	Note
1	Fuel Flow	kg/h	output
2	Air Flow	kg/h	
3	Inlet Manifold Pressure	Bar	
4	Inlet Manifold Temperature	°C	
5	Turbine Inlet Pressure	Bar	
6	Turbine Inlet Temperature	°C	

RPM	1500	2500	3500	4500
Pedal Position	30%	49%	57%	62%
	40%	59%	64%	65%
	54%	74%	74%	76%
	62%	78%	80%	83%
	100%	100%	100%	100%

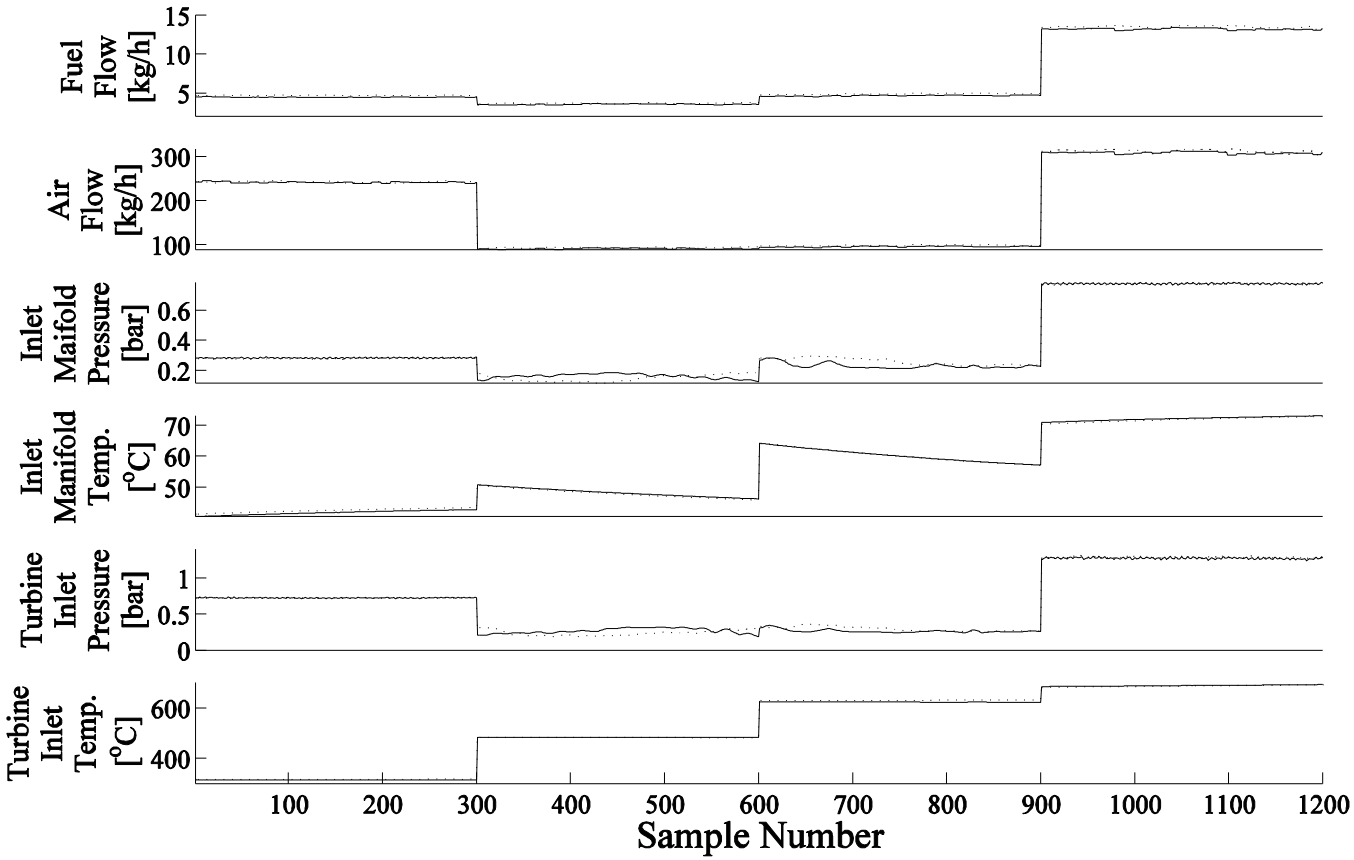
Modelling results

Principal Component	Variance Captured (%)	Variance Total (%)
1	79.5998	79.5998
2	16.4492	96.0490
3	2.4169	98.4659
4	1.0745	99.5404
5	0.4010	99.9414
6	0.0586	100.000

Number of Bottleneck Nodes	Variance Captured (%)	Note
1	97.8160	Important variation
2	99.4212	
3	99.8336	
4	99.8725	Negligible
5	99.9401	
6	99.9414	

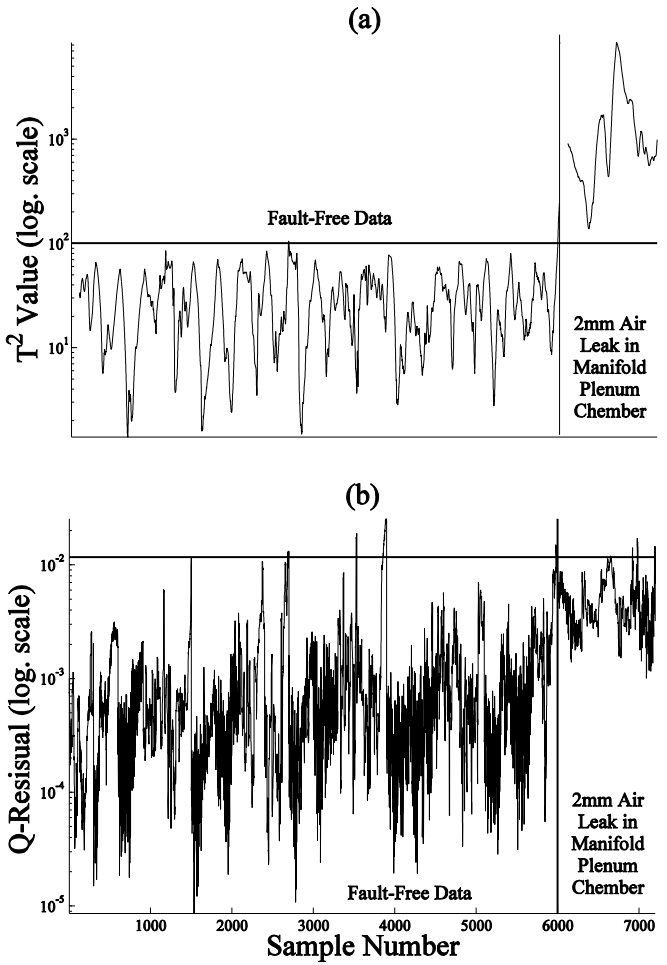
Application to Fault Diagnosis (Internal Combustion Engine)

Air leak of 2mm in the manifold plenum chamber



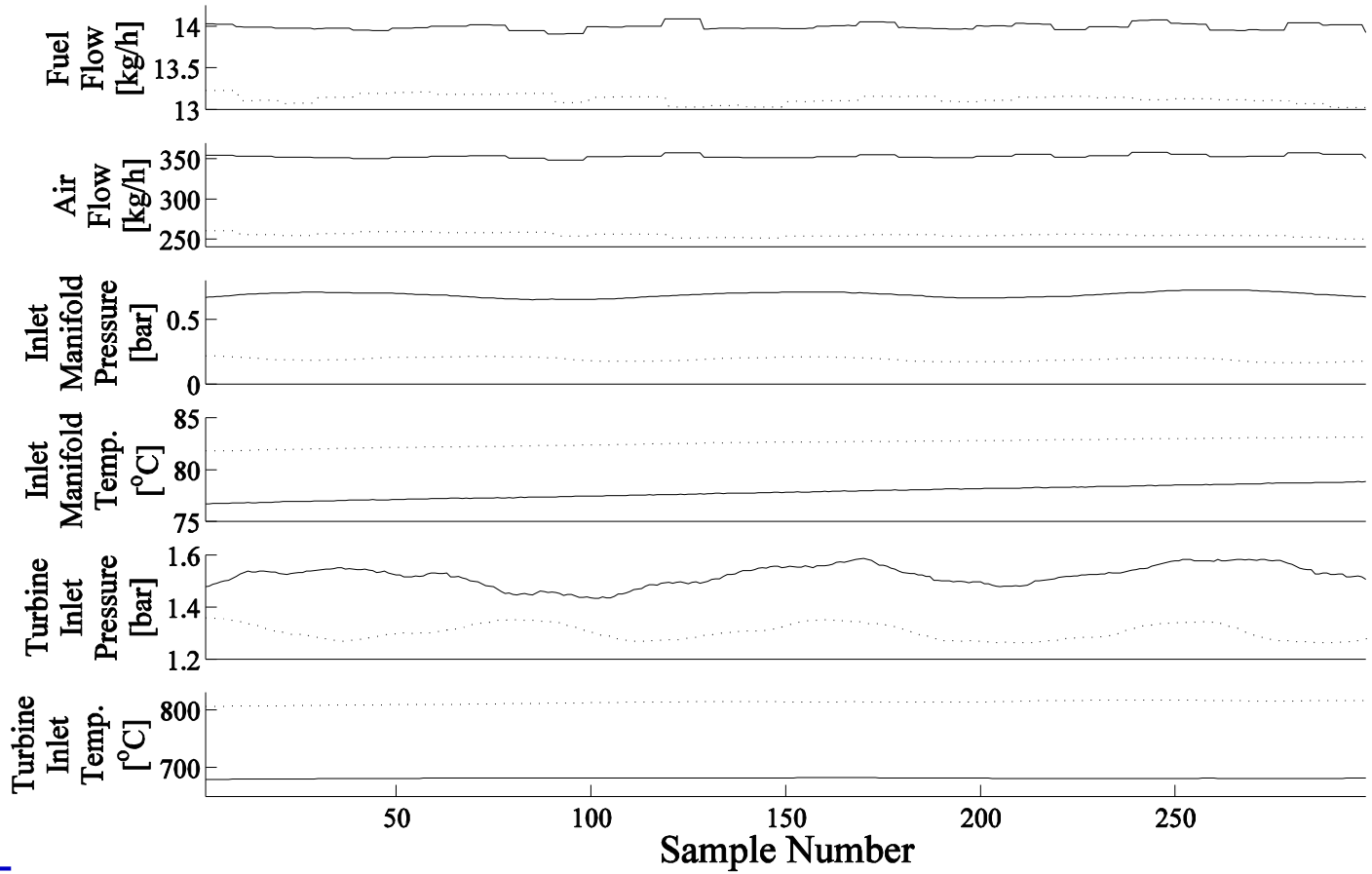
Application to Fault Diagnosis (Internal Combustion Engine)

- An incipient hole in the air intake system could be successfully detected.
- However, a detailed diagnosis as to which recorded engine variable is affected by this event could not be obtained.
- Traditional techniques fail to detect or diagnose this event.
- Model-based fault detection and diagnosis is expensive, whilst data-driven techniques are a viable alternative that are cost-effective.



Application to Fault Diagnosis (Internal Combustion Engine)

Air leak of 6mm in the manifold plenum chamber



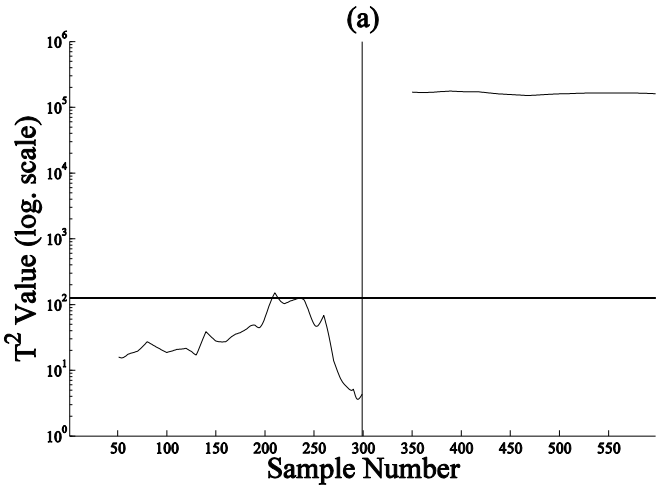
Application to Fault Diagnosis (Internal Combustion Engine)

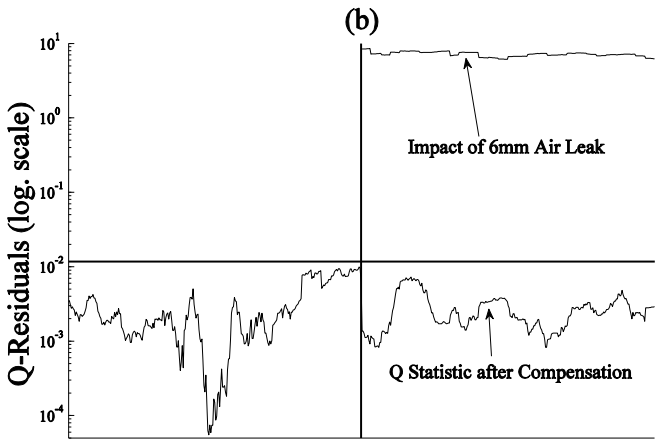
(i)

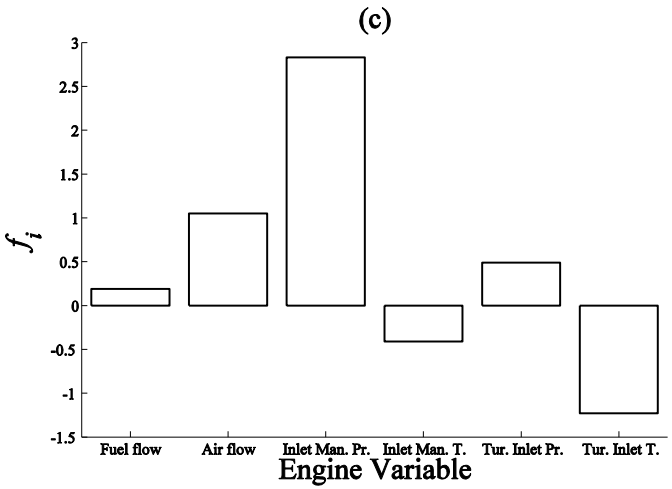
The fault could clearly be detected;

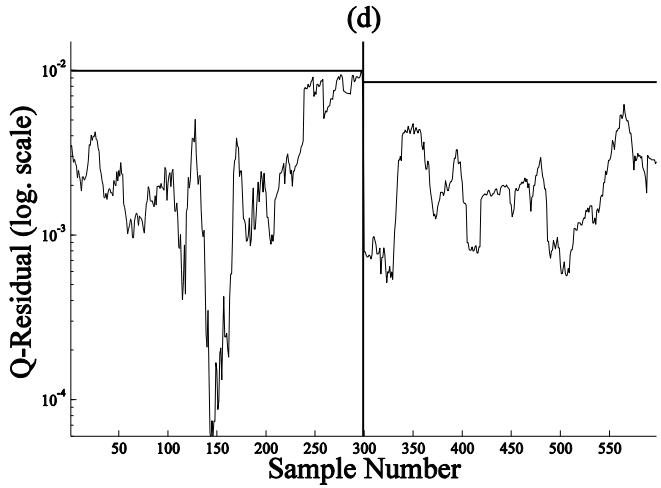
(ii)

The diagnosis provides the engine management system with sufficient information to trace this event to an air leak



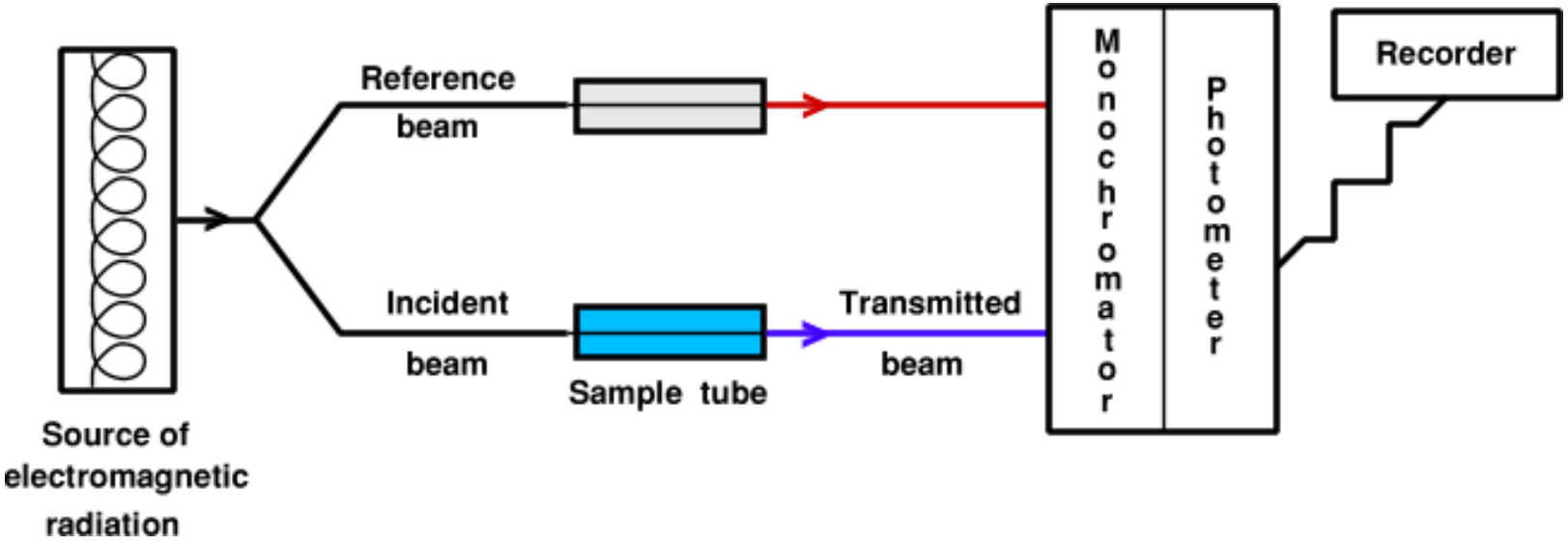




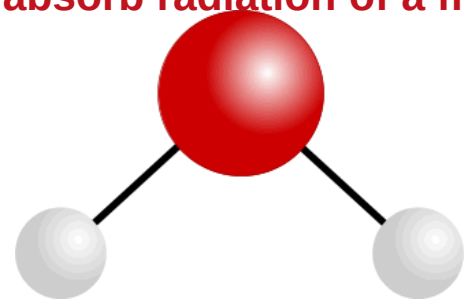


Application to Chemistry (RAMAN Spectroscopy)

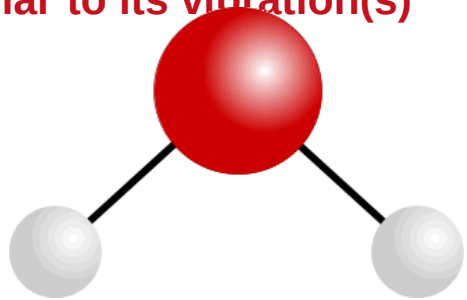
Variable Selection



- A bond will absorb radiation of a frequency similar to its vibration(s)



- normal vibration



- vibration having absorbed energy

Application to Chemistry (RAMAN Spectroscopy)

Variable Selection