2.160 IDENTIFICATION, ESTIMATION, AND LEARNING

LECTURE NOTES NO. 4 (PART 2)

4.3 The Core Algorithm of Multi-Input, Multi-Output Partial Least Squares Regression

Up to this point we have been concerned with problems for predicting only one output variable *y*. Now this section extends it to multiple output problems: prediction of outputs $y_1, \cdots, y_\ell$. See Figure 6 below. There are two approaches to this class of problems. One is to treat each output separately; apply the single-output algorithm repeatedly to individual outputs. The other is to treat the entire outputs as a vectorial quantity. Since the multiple output variables may be correlated, we could obtain a better prediction if we consider the multiple variables together. For example, congestion of cafeteria A at noon and that of cafeteria B on the same campus of MIT must be correlated. (In a snowy day people do not want to go outside.) It makes sense to predict congestion levels of both cafeterias together, rather than doing so independently. We now discuss the full-scale algorithm of multi-input, multi-output (MIMO) partial least squares regression.

In doing so, we introduce a bit more formal description of the problem. We represent input and output variables as random variables and treat sample data as instantiations coming out of the random variables. In other words, you draw samples from the random variables.

The input and output random variables can be characterized with covariance and cross-covariance matrices. First, the input variables have covariance C_{XX} defined as:

$$C_{XX} = E[xx^T] = \begin{pmatrix} E[x_1x_1] & \cdots & E[x_1x_m] \\ \vdots & \ddots & \vdots \\ E[x_mx_1] & \cdots & E[x_mx_m] \end{pmatrix} \in \Re^{m\times m} \tag{31}$$

We are interested in the situation where $m \gg 1$ and many input variables are correlated to each other. This means that many off-diagonal elements of the covariance matrix are non-zero. Similarly, the correlation among the output variables, too, can be characterized with covariance $C_{YY} = E[yy^T] \in \Re^{\ell\times\ell}$. The covariance matrices are symmetry by definition. Finally, the input and output variables may be correlated, as characterized with cross-covariance $C_{YX} = E[yx^T] \in \Re^{\ell\times m}$. Note that by definition cross variance matrices satisfy: $C_{XY} = C_{YX}^T$.

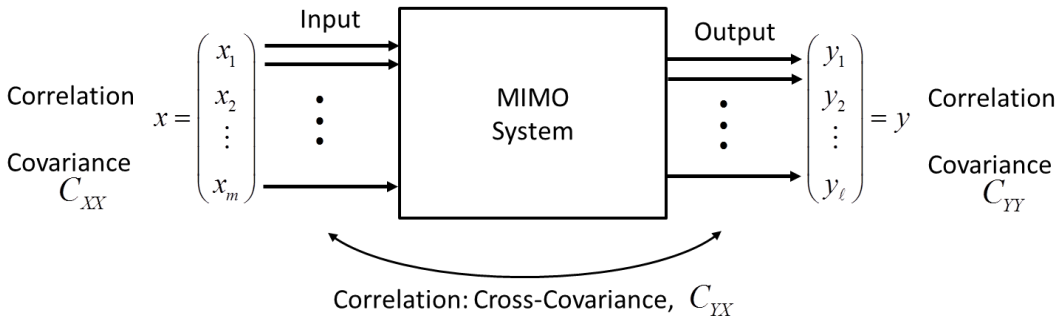

Figure 6 Covariance and cross-covariance of MIMO system

The MIMO PLSR algorithm consists of several steps of procedure.

Maximal Correlation

Similar to the single-output PLSR, MIMO-PLSR, too, extracts latent variables that are significant for predicting output *y* in response to input *x*. Unlike the single output PLSR, however, we have to consider another unit vector $w \in \Re^{\ell \times 1}$ to quantify the strength of signal in the ℓ dimensional output vector space. The scores in the input and output spaces are given by

$$z = v^T x, \quad r = w^T y \tag{32}$$

where *z* is input score and *r* is output score in the direction of unit vectors v and w, respectively. See Figure 7.

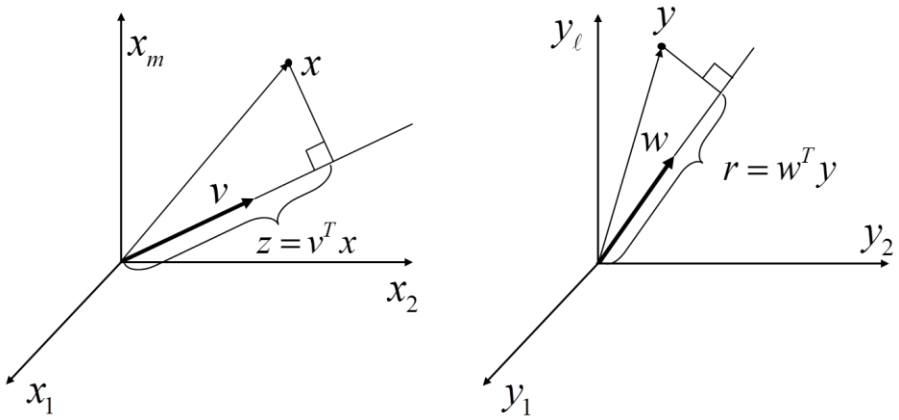

Figure 7 Unit vectors pointing in the directions of maximum correlation between input and output variables. Input and output scores, *z* and *r*, are respectively the projections of the input and output vectors onto the individual unit vectors.

Out first task is to find the directions of a pair of unit vectors that maximize the expected correlation between *z* and *r*.

$$\max_{v,w} E[zr] = \max_{v,w} E[v^T x y^T w] = \max_{v,w} v^T E[xy^T] w$$
$$\text{subject to } |v| = 1, \quad |w| = 1. \tag{33}$$

where the matrix $E[xy^T] \in \Re^{m \times \ell}$ is the covariance of random variables *x* and *y*: $E[xy^T] = C_{XY}$. Using Lagrange multipliers,

$$(v^o, w^o) = \arg\max_{v,w} \left[v^T C_{XY} w - \frac{1}{2}\lambda_v (v^T v - 1) - \frac{1}{2}\lambda_w (w^T w - 1) \right] \tag{34}$$

The necessary conditions for vectors v and w to maximize the correlation are:

$$\frac{\partial}{\partial v} = 0: \quad C_{XY} w - \lambda_v v = 0 \tag{35}$$
$$\frac{\partial}{\partial w} = 0: \quad C_{XY}{}^T v - \lambda_w w = 0 \tag{36}$$

From (35),

$$v = \frac{1}{\lambda_v} C_{XY} w$$

Substituting this into (36) yields

$$C_{YX} C_{XY} w = \lambda_v \lambda_w w \tag{37}$$

Note that $C_{XY}{}^T = C_{YX}$. Similarly, from (36)

$$w = \frac{1}{\lambda_w} C_{YX} v \tag{38}$$

Substituting this into (35) yields

$$C_{XY} C_{YX} v = \lambda_v \lambda_w v \tag{39}$$

The results (37) and (39) imply that the optimal unit vector in the input vector space v^o is the eigenvector of matrix $C_{XY} C_{YX}$, while that in the output vector space w^o is the eigenvector of matrix $C_{YX} C_{XY}$. Note that both matrices $C_{XY} C_{YX}$ and $C_{YX} C_{XY}$ are real and symmetric; All the eigenvalues and eigenvectors are real and physically meaningful.

While we used two eigenvalues in the above formulation, we can show that the two eigenvalues are the same. Pre-multiplying v^T to (35) yields

$$v^T C_{XY} w = \lambda_v v^T v = \lambda_v \tag{40}$$

Pre-multiplying w^T to (36) yields

$$w^T C_{XY}{}^T v = \lambda_w w^T w = \lambda_w \tag{41}$$

Therefore, $\lambda_v = \lambda_w$, and we can write the eigenvalues as λ^2, where the subscripts are dropped. The eigenvectors v^o and w^o associated with the largest eigenvalue of matrices $C_{XY} C_{YX}$ or $C_{YX} C_{XY}$ provide the largest correlation $\max E[zr]$.

These results can be summarized in the following theorem.

[Theorem] The unit vectors v^o and w^o are, respectively, the left and right singular vectors associated with the largest singular value in the Singular-Value Decomposition of the cross-covariance matrix C_{XY}.

Using Singular-Value Decomposition, we can decompose the covariance matrix into:

$$C_{XY} = VDW^T \tag{42}$$

where the components of the diagonal matrix *D* are the square roots of the eigenvalues of matrix $C_{XY} C_{YX}$ or matrix $C_{YX} C_{XY}$, which are non-negative. The column vectors of matrix *V* are eigenvectors of matrix $C_{XY} C_{YX}$, while those of matrix *W* are eigenvectors of matrix $C_{YX} C_{XY}$. Placing the largest singular value at the first component of the diagonal matrix *D*, we can find the

unit vectors v^o and w^o that maximize the correlation $E[zr]$ at the first columns of matrices *V* and *W*, respectively.

Optimal prediction with one latent variable

Our first objective is to predict output *y* in response to input *x* using the single set of latent variables. We have picked the direction v^o (we now drop the superscript for brevity, *v*) that is most significantly correlated with the output. We now predict *y* from $z = v^T x$ as

$$\hat{y} = zq = qv^T x \tag{43}$$

where *q* is a ℓ dimensional vector $q \in \Re^{\ell \times 1}$ to be determined so that the expected squared prediction error may be minimum.

$$q^o = \arg\min_q E[|y - \hat{y}|^2] \tag{44}$$

Using $z = v^T x$, we can expand the squared prediction error to

$$\begin{aligned} E\left[(y - x^T vq)^T (y - x^T vq)\right] &= E\left[y^T y + (x^T vq)^T x^T vq - y^T x^T vq - (x^T vq)^T y\right] \\ &= E\left[y^T y + q^T q(v^T x)(x^T v) - 2q^T yx^T v\right] = E[y^T y] + q^T qv^T E[xx^T]v - 2q^T E[yx^T]v \end{aligned} \tag{45}$$

Differentiating the above with respect to vector *q* yields

$$\frac{d}{dq} = 0: \quad 2qv^T E[xx^T]v - 2E[yx^T]v = 0 \tag{46}$$

Replacing $E[xx^T]$ by C_{XX} and $E[yx^T]$ by C_{YX}, we find that the optimal vector *q* is given by

$$q^o = \frac{E[yx^T]v}{v^T E[xx^T]v} = \frac{C_{YX}v}{v^T C_{XX}v} \tag{47}$$

Therefore,

$$\hat{y} = zq^o = \frac{C_{YX}vv^T}{v^T C_{XX}v} x \tag{48}$$

Deflation

The Singular-Value Decomposition provides a set of orthogonal bases in both input and output vector spaces. These directions, although orthogonal to each other, are not necessarily desired ones that can most effectively predict the output from an input with a fewer terms. In PLSR we employ a deflation procedure to attain not only fast convergence but also various features, including orthogonality of both unit vectors and score vectors, as discussed in the following section.

Similar to the single-output PLSR, we eliminate the information used for the first set of latent variables from the original data. Let $x' \in \Re^{m \times 1}$ be a deflated input vector:

$$x' = x - zp \tag{49}$$

where p is a vector in the input vector space $p \in \Re^{m\times 1}$ to be optimized so that the expected input residue may be smallest.

$$p^o = \arg\min_p E\left[\left|x'\right|^2\right] = \arg\min_p E\left[(x-zp)^T(x-zp)\right] \tag{50}$$

Taking derivative and setting it to zero, we can obtain the optimal p^o as

$$p^o = \frac{C_{XX}v}{v^T C_{XX} v} \tag{51}$$

Output vector y is deflated by subtracting the predicted output based on the first set of latent variables:

$$y' = y - \hat{y} = y - zq \tag{52}$$

To compute the second set of latent variable we need the covariance matrices $C'_{X'X'} = E[x'(x')^T]$ and $C'_{Y'X'} = E[y'(x')^T]$.

$$\begin{aligned} C'_{X'X'} &= E[(x-(v^T x)p)(x-(v^T x)p)^T] = E[(I-pv^T)xx^T(I-pv^T)^T] \\ &= (I-pv^T)E[xx^T](I-pv^T)^T = (I-pv^T)C_{XX}(I-pv^T)^T \end{aligned} \tag{53}$$

This can be further simplified to

$$\begin{aligned} C'_{X'X'} &= (I-pv^T)C_{XX}(I-pv^T)^T \\ &= C_{XX} - pv^T C_{XX} - C_{XX}vp^T + pv^T C_{XX}vp^T = (I-pv^T)C_{XX} \end{aligned} \tag{54}$$

since from (51), $pv^T C_{XX} v = C_{XX} v$. Moreover, $C'_{X'X'} = C_{XX}(I - vp^T)$, since covariance matrices are symmetry. Similarly, the cross covariance $C'_{Y'X'} = E[y'(x')^T]$ is given by

$$C'_{Y'X'} = C_{YX}[I - vp^T] \tag{55}$$

The second set of latent variables can be obtained from the above covariant and cross-covariance matrices in the same way as the first ones.

Summary of the algorithm

The above procedure can be repeated multiple times. Let k be the iteration number, $1 \le k \le m^* < m$, $v(k), w(k)$ be the optimal unit vectors in the input and output spaces at the k-th iteration, respectively, $z(k)$ and $r(k)$ be input and output scores, and $p(k) \in \Re^{m\times 1}$ and $q(k) \in \Re^{\ell\times 1}$ be the loading vectors in the input and output spaces, respectively. Furthermore, let $C_{XX}(k)$ be the covariance matrix of the k-th input residue $x(k)$, and $C_{YX}(k)$ be the cross-

covariance matrix between the *k*-th output residue $y(k)$ and the *k*-th input residue $x(k)$. The initial covariance and cross-covariance matrices are the ones used for the first set of latent variables:

$$C_{XX}(1) = E[xx^T], \quad C_{YX}(1) = E[yx^T] \tag{56}$$

Setting $k = 1$, we can start the iterative computation of latent variables. The covariance and cross-covariance matrices are updated according to the recursive formula:

$$\begin{aligned} C_{XX}(k+1) &= C_{XX}(k)\left[I - v(k)\, p(k)^T \right] \\ C_{YX}(k+1) &= C_{YX}(k)\left[I - v(k) q(k)^T \right] \end{aligned} \tag{57}$$

In summary, PLS extracts covariance information from the input and output variables, *x* and *y*, through the computation of score $z(k)$. The contribution of each score variable is maximized by the determination of loading vectors $p(k)$ and $q(k)$. After repeating the procedure m^* times and finding m^* sets of latent variables, we can approximately represent the input and output variables in the new coordinate systems spanned by $p(1), \cdots, p(m^*)$ and $q(1), \cdots, q(m^*)$, respectively:

$$x = \sum_{k=1}^{m^*} z(k)\, p(k) + x(m^*) \tag{58}$$

$$y = \sum_{k=1}^{m^*} z(k) q(k) + y(m^*) \tag{59}$$

We will examine the orthogonality of these basis vectors in the following section.

Unlike the PCR algorithm, the sets of unit vectors and loading vectors can only be determined sequentially. When to stop the iteration is an important issue, as will be discussed in the following sections. It depends on applications and the noise level of the data.

A Numerical Example

To demonstrate how the PLSR algorithm works, a simple, low-dimensional numerical example is discussed next. Consider a 3-dimensional input vector *x* and a 2-dimensional output vector *y*: $m = 3, \ell = 2$. The input vector has the following covariance:

$$C_{XX} = \begin{pmatrix} 1 & 0.8 & 0.9 \\ 0.8 & 1 & 0.5 \\ 0.9 & 0.5 & 1 \end{pmatrix} \tag{60}$$

Note that the off-diagonal terms are non-zero, indicating some correlation among the input variables. Let us assume that the output is related to the input as

$$y = Bx + g \tag{61}$$

where *g* is a zero mean Gaussian measurement noise vector, $g \sim N(0, 0.05I)$, and *B* is the following coefficient matrix:

$$B = \begin{pmatrix} 0.341 & 0.534 & 0.727 \\ 0.309 & 0.836 & 0.568 \end{pmatrix} \tag{62}$$

The cross-covariance matrix is then given by

$$C_{YX} = \begin{pmatrix} 1.42 & 1.17 & 1.30 \\ 1.49 & 1.37 & 1.27 \end{pmatrix} \tag{63}$$

Figure 8 shows simulated 100 samples of input and output variables. They are mean-centered.

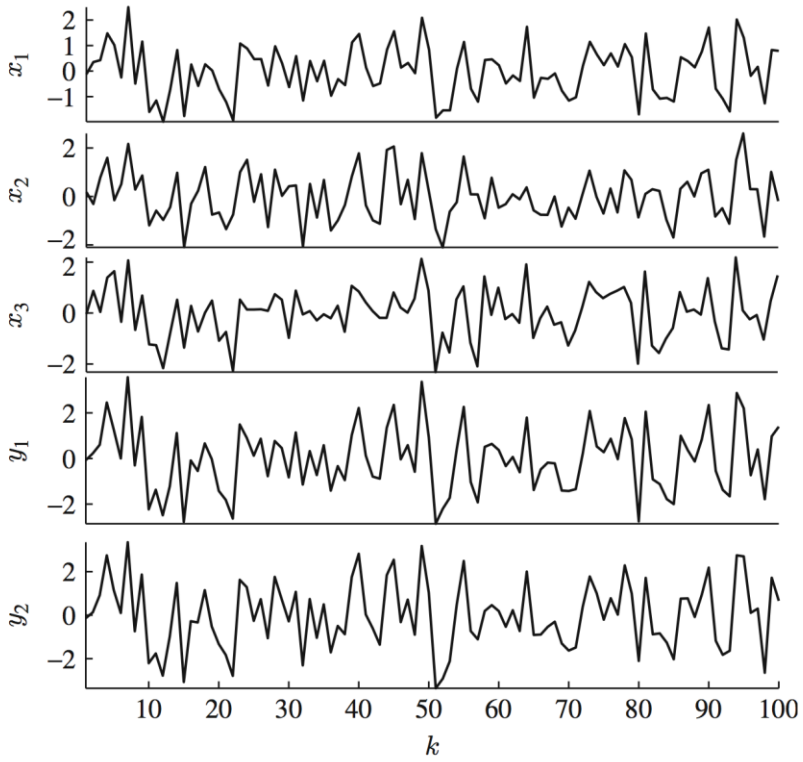

Figure 8 Sample data

The PLSR algorithm is then applied to these data. Figure 9 illustrates the iterative process of computing latent variables.

The top three panels show the data in the input space, while the bottom three panels are the ones in the output space. The left two panels are the plots of the 100 samples, the original data before deflation. Notice that the input data samples are mostly within a plane, while the output samples are strongly correlated along a diagonal line. The first round PLS analysis was performed, and the most correlated directions, $v(1), w(1)$, were determined based on the Singular-Value Decomposition of the cross-covariance matrix in (42). The directions are indicated in the left two panels with straight lines. The output loading vector $q(1)$ was determined so that the prediction error based on the first latent variable may be minimized. See (44) and (47). The input loading vector $p(1)$ was determined in such a way that it can minimize the input residue after

deflating the first set of latent variables from the original data. See (50) and (51). These vectors are:

$$v(1)=\begin{pmatrix}0.627\\0.5515\\0.550\end{pmatrix},\quad w(1)=\begin{pmatrix}0.679\\0.734\end{pmatrix},\quad p(1)=\begin{pmatrix}0.631\\0.536\\0.561\end{pmatrix},\quad q(1)=\begin{pmatrix}0.915\\0.989\end{pmatrix} \tag{64}$$

Deflating the original sample data in the input and output spaces, the residue vectors $x(2), y(2)$ were computed. As plotted in the two middle panels, the deflated samples were squashed in the direction of loading vector $p(1)$ for the input samples and in that of loading vector $q(1)$ for the output samples in accordance with (49) and (52). From this set of degenerate input and output samples, the second round latent variables were extracted. The most correlated directions, $v(2), w(2)$, were determined as indicated in the middle panels. Finally, the extraction of the third round latent variables was attempted for deflated samples $x(3), y(3)$. It turns out that the input data were completed exploited, while no meaningful information was extracted from the output data. The circle shows the strength of measurement noise.

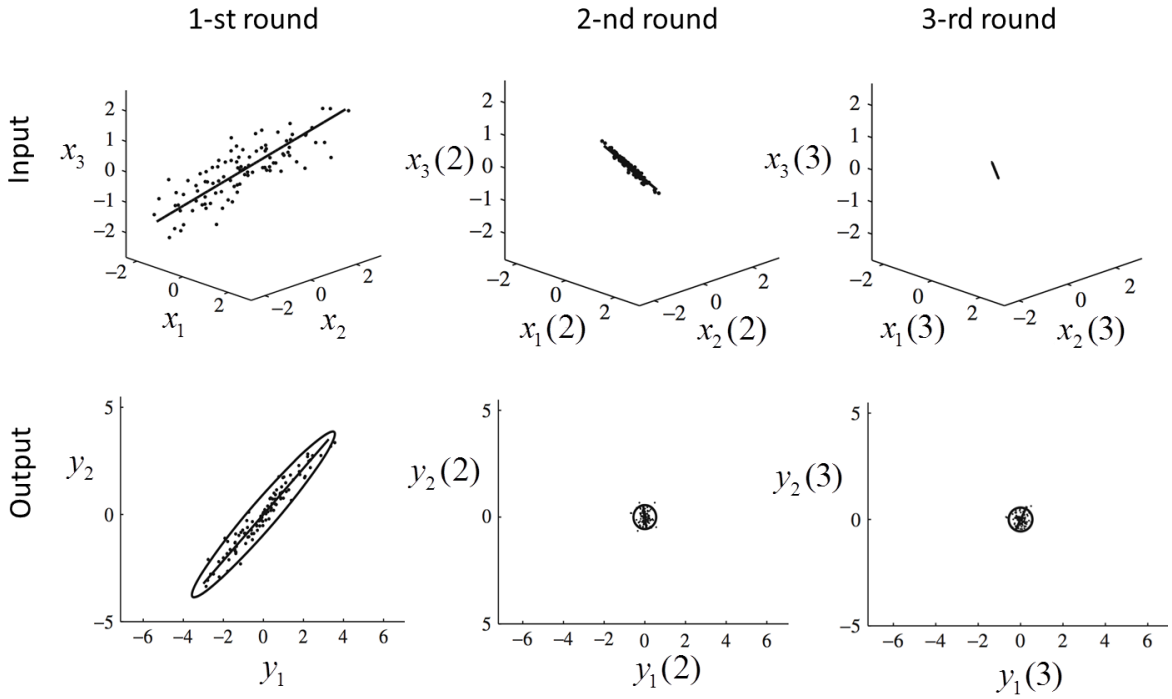

Figure 9 Numerical data of the first, second, and third sets of latent variables

4.4 Properties of MIMO Partial Least Squares Regression

The Partial Least Squares Regression algorithm we have developed for multi-input, multi-output systems has many important properties. In this section we examine some of the important properties that help us better understand the algorithm.

As shown in the numerical example, we use sample data in practical applications. Let us first introduce variables associated with sample data. As before, we write the entire input data in a *m* x *N* matrix and the entire output data in a $\ell \times N$ matrix:

$$X = \left[x^1, \cdots, x^N \right] \in \Re^{m \times N} \quad \text{and} \quad Y = \left[y^1, \cdots, y^N \right] \in \Re^{\ell \times N} \tag{65}$$

The covariance and cross-covariance matrices are estimated from these data:

$$\begin{aligned} C_{XX} &= E[xx^T] \approx \frac{1}{N} \sum_{i=1}^{N} x^i (x^i)^T = \frac{1}{N} XX^T \\ C_{XY} &= E[xy^T] \approx \frac{1}{N} \sum_{i=1}^{N} x^i (y^i)^T = \frac{1}{N} XY^T \end{aligned} \tag{66}$$

Input scores for the individual input samples can be collectively represented as:

$$Z_v = \begin{pmatrix} z^1 \\ \vdots \\ z^N \end{pmatrix} = \begin{pmatrix} (x^1)^T v \\ \vdots \\ (x^N)^T v \end{pmatrix} = X^T v \tag{67}$$

The sample data are deflated as a set of latent variables are extracted recursively. Applying the input deflation rule (49) to all the samples, we can collectively write the deflated input samples at the (*k*+1)-st iteration as:

$$X(k+1) = X(k) - p(k) Z_v^T(k) \tag{68}$$

where $p(k)$ is the input loading vector at the *k*-th iteration. From (51) and (66),

$$p(k) = \frac{C_{XX}(k) v(k)}{v^T(k) C_{XX}(k) v(k)} \approx \frac{X(k) X^T(k) v(k)}{v^T(k) X(k) X^T(k) v(k)} \tag{69}$$

Since $Z_v(k) = X^T(k) v(k)$,

$$p(k) \approx \frac{X(k) Z_v(k)}{\left| Z_v(k) \right|^2} \tag{70}$$

Similarly, the output samples are deflated as:

$$Y(k+1) = Y(k) - q(k)Z_v^T(k) \tag{71}$$

where $q(k)$ is the output loading vector at the *k*-th iteration.

$$q(k) \approx \frac{Y(k)Z_v(k)}{|Z_v(k)|^2} \tag{72}$$

Using these expressions, now we can obtain the following properties.

Orthogonality of input score vectors

Two input score vectors $Z_v(i)$ and $Z_v(j)$ are orthogonal to each other for all $i \neq j$.

Proof

From (67) and (68),

$$X(k+1) = X(k) - p(k)v^T(k)X(k) = \left[I - p(k)v^T(k)\right]X(k) \tag{73}$$

Substituting (70) into (68) also yields

$$X(k+1) = X(k)\left[I - \frac{Z_v(k)Z_v^T(k)}{|Z_v(k)|^2}\right] \tag{74}$$

In proving the orthogonality we can assume $i < j$ without loss of generality. Repeatedly applying (73) we can write

$$\begin{aligned} X(j) &= \left[I - p(j-1)v^T(j-1)\right]X(j-1) \\ &= \left[I - p(j-1)v^T(j-1)\right]\cdots\left[I - p(i+1)v^T(i+1)\right]X(i+1) \\ &= \left[I - p(j-1)v^T(j-1)\right]\cdots\left[I - p(i+1)v^T(i+1)\right]X(i)\left[I - \frac{Z_v(i)Z_v^T(i)}{|Z_v(i)|^2}\right] \end{aligned} \tag{75}$$

where (74) was used at the last line of the above equation. In order to show the orthogonality, we compute the inner product between $Z_v(i)$ and $Z_v(j)$ using (75).

$$\begin{aligned} Z_v^T(j)Z_v(i) &= v^T(j)X(j)Z_v(i) \\ &= v^T(j)\left[I - p(j-1)v^T(j-1)\right]\cdots\left[I - p(i+1)v^T(i+1)\right]X(i)\left[I - \frac{Z_v(i)Z_v^T(i)}{|Z_v(i)|^2}\right]Z_v(i) \\ &= v^T(j)\left[I - p(j-1)v^T(j-1)\right]\cdots\left[I - p(i+1)v^T(i+1)\right]X(i)\left[Z_v(i) - \frac{Z_v(i)Z_v^T(i)Z_v(i)}{|Z_v(i)|^2}\right] \\ &= v^T(j)\left[I - p(j-1)v^T(j-1)\right]\cdots\left[I - p(i+1)v^T(i+1)\right]X(i)\left[Z_v(i) - Z_v(i)\right] = 0 \end{aligned} \tag{76}$$

where $Z_v^T(i)Z_v(i) = |Z_v(i)|^2$ was used at the last line. Therefore, all the input score vectors are orthogonal.

Orthogonality of input unit vectors

Two input unit vectors $v(i)$ and $v(j)$ are orthogonal to each other for all $i \neq j$.

Proof

Recall (39): $C_{XY}C_{YX}v = \lambda^2 v$. Using (66) we can write

$$v(i) = \frac{1}{\lambda^2} C_{XY}(i)C_{YX}(i)v(i) \propto X(i)Y^T(i)Y(i)X^T(i)v(i) \tag{77}$$

In order to show the orthogonality between $v(i)$ and $v(j)$ we compute the inner product:

$$v^T(j)v(i) \propto \underbrace{v^T(j)X(i)}_{\|\ A} Y^T(i)Y(i)X^T(i)v(i) \tag{78}$$

Without loss of generality we assume $i > j$. Using (74), the above term A can be expanded to

$$\begin{aligned} A &= v^T(j)X(i-1)\left[I - \frac{Z_v(i-1)Z_v^T(i-1)}{|Z_v(i-1)|^2}\right] \\ &= \underbrace{v^T(j)X(j)}_{Z_v^T(j)}\left[I - \frac{Z_v(j)Z_v^T(j)}{|Z_v(j)|^2}\right]\cdots\left[I - \frac{Z_v(i-1)Z_v^T(i-1)}{|Z_v(i-1)|^2}\right] \\ &= \left[Z_v^T(j) - \frac{Z_v^T(j)Z_v(j)Z_v^T(j)}{|Z_v(j)|^2}\right]\cdots\left[I - \frac{Z_v(i-1)Z_v^T(i-1)}{|Z_v(i-1)|^2}\right] = 0 \end{aligned} \tag{79}$$

Therefore, input unit vectors $v(i)$ and $v(j)$ are orthogonal for all $i \neq j$ but not greater than m^*, the number of repetitions.

The orthonormal unit vectors $v(1), \cdots, v(m^*)$ form a m^*-dimensional subspace. It should be noted, however, the score $z(k)$ cannot be determined directly from a given sample x.

$$z(k) = x^T(k)v(k) \neq x^T v(k) \tag{80}$$

except for $k = 1$. The k-th score variable must be the inner product between the k-th unit vector and the k-th deflated input, not the original input. To further examine this let us obtain the first few score variables.

$$\begin{aligned} &z(2) = v^T(2)x(2) = v^T(2)[x - z(1)p(1)] = v^T(2)[I - p(1)v^T(1)]x \\ &z(3) = v^T(3)x(3) = v^T(3)[x(2) - z(2)p(2)] = v^T(3)[I - p(2)v^T(2)][I - p(1)v^T(1)]x \\ &\cdots\cdots \end{aligned} \tag{81}$$

Thus, sequential computations are required. These computations can be streamlined with use of the following new variables.

u-Vectors

If the following vectors are computed from $v(k)$ and $p(k), k = 1 \sim m^*$:

$$u(1) = v(1)$$
$$u(j) = v(j) - \sum_{i=1}^{j-1} p^T(i)v(j)u(i), \quad 2 \le j \le m^* \tag{82}$$

then the score variables can be computed directly from input *x*.

Exercise Prove that the **u**-vectors allow for directly computing score variables from input *x*:

$$z(k) = u^T(k) \cdot x, \quad k = 1, \cdots, m^* \tag{83}$$

Therefore, output y can be predicted from input x using the u-vectors:

$$\hat{y} = \sum_{k=1}^{m^*} z(k)q(k) = \left[\sum_{k=1}^{m^*} q(k)u^T(k) \right] \cdot x \tag{84}$$

4.5 Stopping Rules

One of the open questions in using PLSR as well as PCR is how to determine the number of the sets of latent variables, m^*. In PLSR, it is the question when to stop the iterative computation for extracting latent variables. This is of fundamental importance. If m^* is too small, then the prediction error may be significantly large. If it is too large, higher order latent variables can only capture noise in the data rather than meaningful signals. There must be some trade-off to make.

In the literature various Stopping Rules have been reported to address this issue. These can be classified into i) Variance and eigenvalue based criteria, ii) Cross-validation based criteria, and iii) Information-theoretic criteria[1]. In the following, we will briefly discuss the first two.

Variance of Residues

As latent variables are extracted, the magnitude of residual data decreases. The magnitude may be quantified with variance, compared to their original variance. The *k*-th input residue, for example, is quantified with the quotient given by:

$$\kappa_X(k) = \frac{E[|x(k)|^2]}{E[|x|^2]} \tag{85}$$

This variance quotient can be derived from covariance matrices using Trace. (If you are not familiar with Trace, see the appendix at the end of the chapter.)

$$\text{Trace } C_{XX} = \text{Trace } E[xx^T] = E[\text{Trace } (xx^T)] = E[\text{Trace } (x^T x)] = E[|x|^2] \tag{86}$$

[1] At the very end of the term we will discuss Akaike's Information Criterion and related topics, in which the trade-offs are made based on information measure.

Covariance matrix C_{XX} can be diagonalized using an orthonormal matrix U: $C_{XX} = UDU^T$, where diagonal matrix D consists of eigenvalues of C_{XX}. Taking Trace, we obtain:

$$\text{Trace}\, C_{XX} = \text{Trace}\, U(DU^T) = \text{Trace}\, D \underbrace{U^T U}_{\text{Identity Matrix}} = \text{Trace}\, D = \sum_i \lambda_i[C_{XX}] \qquad (87)$$

where $\lambda_i[C_{XX}]$'s are the eigenvalues of C_{XX}. Using these in (85) yields

$$\kappa_X(k) = \frac{\text{Trace}\, C_{X(k)X(k)}}{\text{Trace}\, C_{XX}} = \frac{\sum_i \lambda_i[C_{X(k)X(k)}]}{\sum_i \lambda_i[C_{XX}]} \qquad (88)$$

Similarly for the output,

$$\kappa_Y(k) = \frac{E[|y(k)|^2]}{E[|y|^2]} = \frac{\text{Trace} C_{Y(k)Y(k)}}{\text{Trace} C_{YY}} = \frac{\sum_i \lambda_i[C_{Y(k)Y(k)}]}{\sum_i \lambda_i[C_{YY}]} \qquad (89)$$

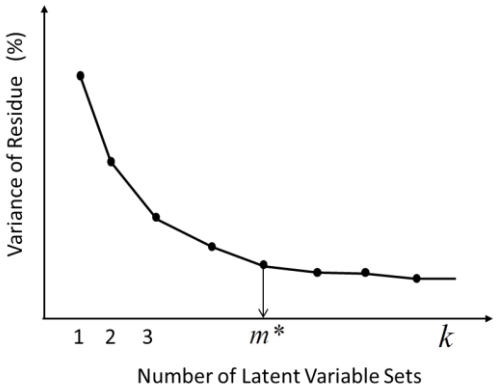

Figure 10 Typical plot of residue variance

Plotting $\kappa_X(k)$ and $\kappa_Y(k)$ against iteration number *k*, we can find how effectively each set of latent variables can extract useful information from the data. Typically, they decrease rapidly for the first few iterations, and then gently approach a constant level. It has been recommended to truncate around the elbow of the curve to pick an appropriate iteration number m^*. See Figure 10. Plots are usually in percentage scale. The output data residue often approaches the level determined by observation noise.

Cross-Validation

Another class of stopping rules that have been widely is based on statistical Cross-Validation. In general, cross validation uses the following procedure:

1. Remove a group of samples from the original dataset;
2. Construct a predictor, e.g. PLSR and PCR, using the remaining samples;
3. Apply the predictor to the removed samples and evaluate the prediction error;
4. Remove a different group of samples and repeat the process; and
5. Take average of the prediction errors obtained from all the validation tests.

See Figure 11 below.

If the number of latent variables is small, the validation accuracy may not be satisfactory. On the other hand, if too many latent variables are used, an over-fitting

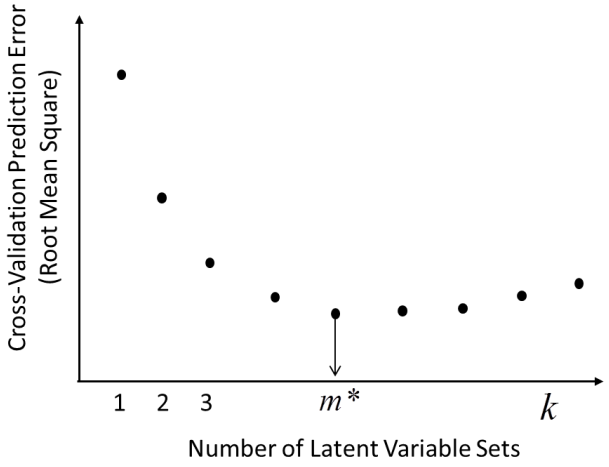

Figure 12 Typical plot of Cross Validation test

situation may occur, resulting in erratic prediction performance highly depending on which specific group of samples is removed. Plotting the validation performance against iteration number k, we can find an optimal point to stop iteration. Typically, the validation performance improves rapidly for the first few iterations, as indicated by a sharply declining curve, followed by a plateau or an increase of validation prediction error, indicating over-fitting. We can find a reasonable cut off iteration number around the bottom of the plot.

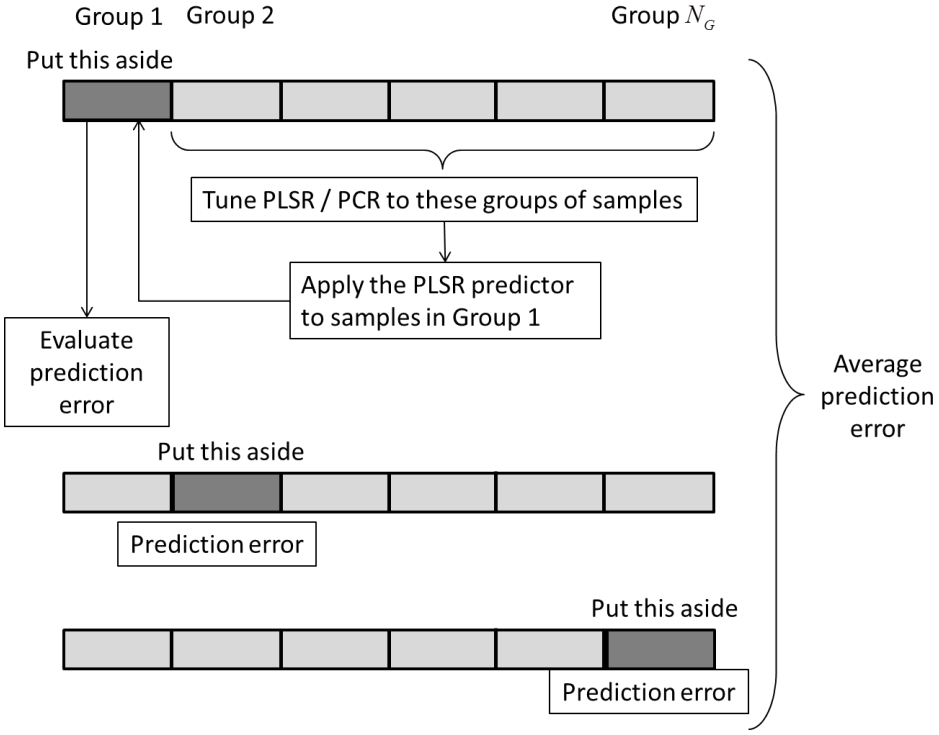

Figure 11 Cross Validation procedure

Appendix on Matrix Trace

Consider a square matrix $A = \{a_{ij}\} \in \Re^{n \times n}$. The trace of matrix A is the sum of all the diagonal elements.

$$\text{Trace } A = \sum_{i=1}^{n} a_{ii}$$

It can be shown easily through component-wise computation that the following property holds:

$$\text{Trace } BC = \text{Trace } CB$$

where matrix products *BC* and *CB* must be square matrices. Note that each does not have to be square.