A Simple Explanation of Partial Least Squares

Kee Siong Ng
keesiong.ng@gopivotal.com

Draft, April 27, 2013

1 Introduction

Partial Least Squares (PLS) is a widely used technique in chemometrics, especially in the case where the number of independent variables is significantly larger than the number of data points. There are many articles on PLS [HTF01, GK86] but the mathematical details of PLS do not always come out clearly in these treatments. This paper is an attempt to describe PLS in precise and simple mathematical terms.

2 Notation and Terminology

Definition 1. Let $\mathbf{X} = [\mathbf{x_1} \dots \mathbf{x_m}]$ be a $n \times m$ matrix. The mean-centered matrix

$$\mathbf{B} := [\mathbf{x_1} - \bar{\mathbf{x}}_\mathbf{1} \ \dots \mathbf{x_m} - \bar{\mathbf{x}}_\mathbf{m}],$$

where $\bar{\mathbf{x}}_\mathbf{i}$ is the mean value for $\mathbf{x_i}$, has zero sample mean. We will mostly work with mean-centered matrices in this document.

Definition 2. Suppose $\mathbf{X}$ is a mean-centered $n \times m$ matrix and $\mathbf{Y}$ is a mean-centered $n \times p$ matrix. The sample covariance matrix of $\mathbf{X}$ and $\mathbf{Y}$ is given by

$$cov(\mathbf{X}, \mathbf{Y}) := \frac{\mathbf{1}}{\mathbf{n-1}} \mathbf{X^T Y}.$$

The variance of $\mathbf{X}$ is given by

$$var(\mathbf{X}) := cov(\mathbf{X}, \mathbf{X}).$$

(The reason the sample covariance matrix has $n-1$ in the denominator rather than n is to correct for the fact that we are using sample mean instead of true population mean to do the centering.) $S := var(\mathbf{X})$ is symmetric. The diagonal entry $S_{j,j}$ is called the variance of x_j. The total variance of the data in $\mathbf{X}$ is given by the trace of $\mathbf{S}$: $tr(\mathbf{S}) = \sum_j S_{j,j}$. The value $S_{i,j}, i \neq j$, is called the covariance of x_i and x_j. The correlation between $\mathbf{X}$ and $\mathbf{Y}$ is defined by

$$corr(\mathbf{X}, \mathbf{Y}) := \sqrt{var(\mathbf{X})} cov(\mathbf{X}, \mathbf{Y}) \sqrt{var(\mathbf{Y})} \tag{1}$$

3 Problems with Ordinary Least Squares

To understand the motivation for using PLS in high-dimensional chemometrics data, it is important to understand how and why ordinary least squares fail in the case where we have a large number of independent variables and they are highly correlated. Readers who are already familiar with this topic can skip to the next section.

Fact 1. Given a design matrix $\mathbf{X}$ and the response vector $\mathbf{y}$, the least square estimate of the β parameter in the linear model

$$\mathbf{y} = \mathbf{X}\beta + \epsilon$$

is given by the normal equation

$$\hat{\beta} = (\mathbf{X^T X})^{-1}\mathbf{X^T y}. \tag{2}$$

Fact 2. The simplest case of linear regression yields some geometric intuition on the coefficient. Suppose we have a univariate model with no intercept:

$$\mathbf{y} = \mathbf{x}\beta + \epsilon.$$

Then the least-squares estimate $\hat{\beta}$ of β is given by

$$\hat{\beta} = \frac{\langle \mathbf{x}, \mathbf{y} \rangle}{\langle \mathbf{x}, \mathbf{x} \rangle}.$$

This is easily seen as a special case of (2). It is also easily established by differentiating

$$\sum_i (y_i - \beta x_i)^2$$

with respect to β and solving the resulting KKT equations. Geometrically, $\hat{\mathbf{y}} = \frac{\langle \mathbf{x}, \mathbf{y} \rangle}{\langle \mathbf{x}, \mathbf{x} \rangle}\mathbf{x}$ is the projection of $\mathbf{y}$ onto the vector $\mathbf{x}$.

Regression by Successive Orthogonalisation The problem with using ordinary least squares on high-dimensional data is clearly brought out in a linear regression procedure called Regression by Successive Orthogonalisation. This section is built on the material covered in [HTF01].

Definition 3. Two vectors $\mathbf{u}$ and $\mathbf{v}$ are said to be orthogonal if $\langle \mathbf{u}, \mathbf{v} \rangle = 0$; *i.e.,* the vectors are perpendicular. A set of vectors is said to be orthogonal if every pair of (non-identical) vectors from the set is orthogonal. A matrix is orthogonal if the set of its column vectors are orthogonal.

Fact 3. For an orthogonal matrix $\mathbf{X}$, we have $\mathbf{X}^{-1} = \mathbf{X}^T$.

Fact 4. Orthogonalisation of a matrix $\mathbf{X} = [\mathbf{x_1}\, \mathbf{x_2} \cdots \mathbf{x_m}]$ can be done using the Gram-Schmidt process. Writing

$$proj_{\mathbf{u}}(\mathbf{v}) := \frac{\langle \mathbf{u}, \mathbf{v} \rangle}{\langle \mathbf{u}, \mathbf{u} \rangle}\mathbf{u},$$

the procedure transforms $\mathbf{X}$ into an orthogonal matrix $\mathbf{U} = [\mathbf{u_1}\, \mathbf{u_2} \cdots \mathbf{u_m}]$ via these steps:

$$\begin{aligned}
\mathbf{u_1} &:= \mathbf{x_1} \\
\mathbf{u_2} &:= \mathbf{x_2} - proj_{\mathbf{u_1}}(\mathbf{x_2}) \\
\mathbf{u_3} &:= \mathbf{x_3} - proj_{\mathbf{u_1}}(\mathbf{x_3}) - proj_{\mathbf{u_2}}(\mathbf{x_3}) \\
&\vdots \\
\mathbf{u_m} &:= \mathbf{x_m} - \sum_{j=1}^{n-1} proj_{\mathbf{u_j}}(\mathbf{x_m})
\end{aligned}$$

The Gram-Schmidt process also gives us the **QR** factorisation of $\mathbf{X}$, where $\mathbf{Q}$ is made up of the orthogonal $\mathbf{u_i}$ vectors normalised to unit vectors as necessary, and the upper triangular $\mathbf{R}$ matrix is obtained from the $proj_{\mathbf{u_i}}(\mathbf{x_j})$ coefficients. Gram-Schmidt is known to be numerically unstable; a better procedure to do orthogonalisation and **QR** factorisation is the Householder transformation. Householder transformation is the dual of Gram-Schmidt in the following sense: Gram-Schmidt computes $\mathbf{Q}$ and gets $\mathbf{R}$ as a side product; Householder computes $\mathbf{R}$ and gets $\mathbf{Q}$ as a side product [GBGL08].

Fact 5. If the column vectors of the design matrix $\mathbf{X} = [\mathbf{x_1}\ \mathbf{x_2} \cdots \mathbf{x_m}]$ forms an orthogonal set, then it follows from (2) that

$$\hat{\beta}^{\mathbf{T}} = \left[\frac{\langle x_1, y \rangle}{\langle x_1, x_1 \rangle} \ \frac{\langle x_2, y \rangle}{\langle x_2, x_2 \rangle} \cdots \frac{\langle x_m, y \rangle}{\langle x_m, x_m \rangle} \right], \tag{3}$$

since $\mathbf{X}^T\mathbf{X} = diag(\langle x_1, x_1 \rangle, \ldots, \langle x_m, x_m \rangle)$. In other words, $\hat{\beta}$ is made up of the univariate estimates. This means when the input variables are orthogonal, they have no effect on each other's parameter estimates in the model.

Fact 6. One way to perform regression known as the Gram-Schmidt procedure for multiple regression is to first decompose the design matrix $\mathbf{X}$ into $\mathbf{X} = \mathbf{U\Gamma}$, where $\mathbf{U} = [\mathbf{u_1}\ \mathbf{u_2} \cdots \mathbf{u_m}]$ is the orthogonal matrix obtained from Gram-Schmidt, and $\mathbf{\Gamma}$ is the upper triangular matrix defined by

$$\Gamma_{l,l} = 1 \quad \text{and} \quad \Gamma_{l,j} = \frac{\langle \mathbf{u_l}, \mathbf{x_j} \rangle}{\langle \mathbf{u_l}, \mathbf{u_l} \rangle}$$

for $l < j$, and then solve the associated regression problem $\mathbf{U}\alpha = \mathbf{y}$ using (3). The following shows the relationship between α and the β in $\mathbf{X}\beta = \mathbf{y}$:

$$\mathbf{X}\beta = \mathbf{y} \implies \mathbf{U\Gamma}\beta = \mathbf{y} \implies \mathbf{\Gamma}\beta = \mathbf{U^T y} = \alpha.$$

Since $\Gamma_{m,m} = 1$, we have

$$\beta(m) = \alpha(m) = \frac{\langle \mathbf{u_m}, \mathbf{y} \rangle}{\langle \mathbf{u_m}, \mathbf{u_m} \rangle}. \tag{4}$$

Fact 7. Since any $\mathbf{x_j}$ can be shifted into the last position in the design matrix $\mathbf{X}$, Equation (4) tells us something useful: The regression coefficient $\beta(j)$ of $\mathbf{x_j}$ is the univariate estimate of regressing y on the residual of regressing $\mathbf{x_j}$ on $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_{j-1}}, \mathbf{x_{j+1}}, \ldots, \mathbf{x_n}$. Intuitively, $\beta(j)$ represents the additional contribution of $\mathbf{x_j}$ on $\mathbf{y}$, after $\mathbf{x_j}$ has been adjusted for $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_{j-1}}, \mathbf{x_{j+1}}, \ldots, \mathbf{x_n}$. From the above, we can now see how multiple linear regression can break in practice. If $\mathbf{x_n}$ is highly correlated with some of the other $\mathbf{x_k}$'s, the residual vector $\mathbf{u_n}$ will be close to zero and, from (4), the regression coefficient $\beta(m)$ will be very unstable. Indeed, this will be true for all the variables in the correlated set.

4 Principal Component Regression

Partial least squares and the closely related principal component regression technique are both designed to handle the case of a large number of correlated independent variables, which is common in chemometrics. To understand partial least squares, it helps to first get a handle on principal component regression, which we now cover.

The idea behind principal component regression is to first perform a principal component analysis (PCA) on the design matrix and then use only the first k principal components to do the regression. To understand how it works, it helps to first understand PCA.

Definition 4. A matrix $\mathbf{A}$ is said to be *orthogonally diagonalisable* if there are an orthogonal matrix $\mathbf{P}$ and a diagonal matrix $\mathbf{D}$ such that

$$\mathbf{A} = \mathbf{PDP^T} = \mathbf{PDP^{-1}}.$$

Fact 8. An $n \times n$ matrix $\mathbf{A}$ is orthogonally diagonalisable if and only if $\mathbf{A}$ is a symmetric matrix (*i.e.,* $\mathbf{A^T} = \mathbf{A}$).

Fact 9. From the Spectral Theorem for Symmetric Matrices [Lay97], we know that an $n \times n$ symmetric matrix $\mathbf{A}$ has n real eigenvalues, counting multiplicities, and that the corresponding eigenvectors are orthogonal. (Eigenvectors are not orthogonal in general.) A symmetric matrix $\mathbf{A}$ can thus be orthogonally diagonalised this way

$$\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{U}^{\mathbf{T}},$$

where $\mathbf{U}$ is made up of the eigenvectors of $\mathbf{A}$ and $\mathbf{D}$ is the diagonal matrix made up of the eigenvalues of $\mathbf{A}$.

Another result we will need relates to optimisation of quadratic forms under a certain form of constraint.

Fact 10. Let $\mathbf{A}$ be a symmetric matrix. Then the solution of the optimisation problem

$$\max_{\mathbf{x}\,:\,||\mathbf{x}||=1} \mathbf{x}^{\mathbf{T}}\mathbf{A}\mathbf{x}$$

is given by the largest eigenvalue $\lambda_{\max}$ of $\mathbf{A}$ and the $\mathbf{x}$ that realises the maximum value is the eigenvector of $\mathbf{A}$ corresponding to $\lambda_{\max}$.

Suppose $\mathbf{X}$ is an $n \times m$ matrix in mean-centered form. (In other words, we have n data points and m variables.) The goal of principal component analysis is to find an orthogonal $m \times m$ matrix $\mathbf{P}$ that determines a change of variable

$$\mathbf{T} = \mathbf{X}\mathbf{P} \tag{5}$$

such that the new variables $t_1, \ldots, t_m$ are uncorrelated and arranged in order of decreasing variance. In other words, we want the covariance matrix $cov(\mathbf{T}, \mathbf{T})$ to be diagonal and that the diagonal entries are in decreasing order. The $m \times m$ matrix $\mathbf{P}$ is called the *principal components* of $\mathbf{X}$ and the $n \times m$ matrix $\mathbf{T}$ is called the *scores*.

Fact 11. Since one can show $\mathbf{T}$ is in mean-centered form, the covariance matrix of $\mathbf{T}$ is given by

$$cov(\mathbf{T}, \mathbf{T}) = \frac{1}{n-1}\mathbf{T}^{\mathbf{T}}\mathbf{T} = \frac{1}{n-1}(\mathbf{P}^{\mathbf{T}}\mathbf{X}^{\mathbf{T}})(\mathbf{X}\mathbf{P}) = \frac{1}{n-1}\mathbf{P}^{\mathbf{T}}\mathbf{X}^{\mathbf{T}}\mathbf{X}\mathbf{P} = \mathbf{P}^{\mathbf{T}}\mathbf{S}\mathbf{P}, \tag{6}$$

where $\mathbf{S} = \frac{1}{n-1}\mathbf{X}^{\mathbf{T}}\mathbf{X}$ is the covariance matrix of $\mathbf{X}$. Since $\mathbf{S}$ is symmetric, by Fact 9, we have $\mathbf{S} = \mathbf{U}\mathbf{D}\mathbf{U}^{\mathbf{T}}$, where $\mathbf{D} = diag[\lambda_1\, \lambda_2 \ldots \lambda_m]$ are the eigenvalues such that $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_m$ and $\mathbf{U}$ is made up of the corresponding eigenvectors. Plugging this into (6) we obtain

$$cov(\mathbf{T}, \mathbf{T}) = \mathbf{P}^{\mathbf{T}}\mathbf{S}\mathbf{P} = \mathbf{P}^{\mathbf{T}}\mathbf{U}\mathbf{D}\mathbf{U}^{\mathbf{T}}\mathbf{P}. \tag{7}$$

By setting $\mathbf{P}$ to $\mathbf{U}$, the eigenvectors of the covariance matrix of $\mathbf{X}$, we see that $cov(\mathbf{T}, \mathbf{T})$ simplifies to $\mathbf{D}$, and we get the desired result. The eigenvectors are called the principal components of the data.

Often times, most of the variance in $\mathbf{X}$ can be captured by the first few principal components. Suppose we take $\mathbf{P}_{|\mathbf{k}}$ to be the first $k \leq m$ principal components. Then we can construct an approximation of $\mathbf{X}$ via

$$\mathbf{T}_{|\mathbf{k}} := \mathbf{X}\mathbf{P}_{|\mathbf{k}},$$

where $\mathbf{T}_{|\mathbf{k}}$ can be understood as a $n \times k$ compression of the $n \times m$ matrix that captures most of the variance of the data in $\mathbf{X}$. The idea behind principal component regression is to use $\mathbf{T}_{|\mathbf{k}}$, for a suitable value of k, in place of $\mathbf{X}$ as the design matrix. By construction, the columns of $\mathbf{T}_{|\mathbf{k}}$ are uncorrelated.

Fact 12. One way to compute the principal components of a matrix $\mathbf{X}$ is to perform singular value decomposition, which gives

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{P}^\mathbf{T},$$

where $\mathbf{U}$ is an $n \times n$ matrix made up of the eigenvectors of $\mathbf{X}\mathbf{X}^\mathbf{T}$, $\mathbf{P}$ is an $m \times m$ matrix made up of the eigenvectors of $\mathbf{X}^\mathbf{T}\mathbf{X}$ (*i.e.,* the principal components), and $\mathbf{\Sigma}$ is an $n \times m$ diagonal matrix made up of the square roots of the non-zero eigenvalues of both $\mathbf{X}^\mathbf{T}\mathbf{X}$ and $\mathbf{X}\mathbf{X}^\mathbf{T}$. Also, since

$$\mathbf{X} = \mathbf{T}\mathbf{P}^\mathbf{T} = \mathbf{U}\mathbf{\Sigma}\mathbf{P}^\mathbf{T},$$

we see that $\mathbf{T} = \mathbf{U}\mathbf{\Sigma}$.

Fact 13. There is another iterative method for finding the principal components and scores of a matrix $\mathbf{X}$ called the Nonlinear Iterative Partial Least Squares (NIPALS) algorithm. It is built on the observation that a principal component $\mathbf{p}$ in $\mathbf{P}$ and a vector $\mathbf{t}$ in the scores $\mathbf{T}$ satisfy:

$$\mathbf{X}^\mathbf{T}\mathbf{X}\mathbf{p} = \lambda_p \mathbf{p} \tag{8}$$

$$\mathbf{t} = \mathbf{X}\mathbf{p} \tag{9}$$

$$\mathbf{p} = \frac{1}{\lambda_p}\mathbf{X}^\mathbf{T}\mathbf{t}, \tag{10}$$

where (10) is obtained by substituting (9) into (8). The algorithm works by initially setting $\mathbf{t}$ to an arbitrary $\mathbf{x_i}$ in $\mathbf{X}$ and then iteratively applying (10) and (9) until the $\mathbf{t}$ vector stops changing. This gives us the first principal component $\mathbf{p}$ and scores vector $\mathbf{t}$. To find the subsequent components/vectors, we set

$$\mathbf{X} := \mathbf{X} - \mathbf{t}\mathbf{p}^\mathbf{T}$$

and then repeat the same steps. The NIPALS algorithm will turn out to be important in understanding the Partial Least Squares algorithm.

5 Partial Least Squares

We are now in a position to understand PLS. The idea behind Partial Least Squares is to decompose both the design matrix $\mathbf{X}$ and response matrix $\mathbf{Y}$ (we consider the general case of multiple responses here) like in principal component analysis:

$$\mathbf{X} = \mathbf{T}\mathbf{P}^\mathbf{T} \quad \mathbf{Y} = \mathbf{U}\mathbf{Q}^\mathbf{T}, \tag{11}$$

and then perform regression between $\mathbf{T}$ and $\mathbf{U}$. If we do the decompositions of $\mathbf{X}$ and $\mathbf{Y}$ independently using NIPALS, we get these two sets of update rules:

$\mathbf{t} := \mathbf{x_j}$ for some j	$\mathbf{u} := \mathbf{y_j}$ for some j
Loop	Loop
$\quad \mathbf{p} := \mathbf{X}^\mathbf{T}\mathbf{t}/\|\|\mathbf{X}^\mathbf{T}\mathbf{t}\|\|$	$\quad \mathbf{q} := \mathbf{Y}^\mathbf{T}\mathbf{u}/\|\|\mathbf{Y}^\mathbf{T}\mathbf{u}\|\|$
$\quad \mathbf{t} := \mathbf{X}\mathbf{p}$	$\quad \mathbf{u} := \mathbf{Y}\mathbf{q}$
Until $\mathbf{t}$ stop changing	Until $\mathbf{u}$ stop changing

The idea behind partial least squares is that we want the decompositions of $\mathbf{X}$ and $\mathbf{Y}$ to be done by taking information from each other into account. One intuitive way to achieve that is to swap $\mathbf{t}$ and $\mathbf{u}$ in the update rules for $\mathbf{p}$ and $\mathbf{q}$ above and then interleave the two sets of update rules

inside the loop, resulting in the following procedure:

$$
\begin{aligned}
&\mathbf{u} := \mathbf{y_j} \quad \text{for some } j \\
&\text{Loop} \\
&\quad \mathbf{p} := \mathbf{X^T u}/||\mathbf{X^T u}|| \\
&\quad \mathbf{t} := \mathbf{Xp} \\
&\quad \mathbf{q} := \mathbf{Y^T t}/||\mathbf{Y^T t}|| \\
&\quad \mathbf{u} := \mathbf{Yq} \\
&\text{Until } \mathbf{t} \text{ stop changing}
\end{aligned} \tag{12}
$$

In the case when the $\mathbf{Y}$ block has only one variable, we can set $\mathbf{q}$ to $\mathbf{1}$ and the last two steps of the loop can be omitted.

The above gives the routine for finding the first set of PLS components and loadings. For subsequent components and vectors, we set

$$
\begin{aligned}
\mathbf{X} &:= \mathbf{X} - \mathbf{tp^T} \\
\mathbf{Y} &:= \mathbf{Y} - \mathbf{uq^T}
\end{aligned}
$$

and then repeat the same steps. After l such steps, we obtain two $n \times l$ matrices $\mathbf{T}$ and $\mathbf{U}$ and matrices $\mathbf{P}$ and $\mathbf{Q}$ related by (11). To get a regression model relating $\mathbf{Y}$ and $\mathbf{X}$, we first fit β for

$$\mathbf{U} = \mathbf{T}\beta.$$

Substituting the resulting model into (11) we obtain

$$\mathbf{Y} = \mathbf{UQ^T} = \mathbf{T}\beta\mathbf{Q^T} = \mathbf{XP}\beta\mathbf{Q^T}.$$

Given any $\mathbf{x}$, we can use $\mathbf{P}, \mathbf{Q}$ and β to compute the corresponding $\mathbf{y}$ value in the obvious way.

Fact 14. Algorithm (12) is somewhat mysterious. Let's look at what the computed terms mean, starting with $\mathbf{p}$.

$$
\begin{aligned}
\mathbf{p} &= \mathbf{X^T u}/||\mathbf{X^T u}|| \\
&= \mathbf{X^T Yq}/||\mathbf{X^T Yq}|| \\
&= \mathbf{X^T YY^T t}/||\mathbf{X^T YY^T t}|| \\
&= \mathbf{X^T YY^T Xp}/||\mathbf{X^T YY^T Xp}|| \\
&= \frac{1}{\lambda}(\mathbf{Y^T X})^\mathbf{T}(\mathbf{Y^T X})\mathbf{p}
\end{aligned} \tag{13}
$$

In other words, $\mathbf{p}$ is an eigenvector of the covariance matrix of $\mathbf{Y^T X}$. In fact, it's easy to see that (13) is exactly the update rule in the Power method [GL96, §7.3.1] used for computing the largest eigenvalue-eigenvector pair for the symmetric $\mathbf{X^T YY^T X}$ matrix. By Fact 10, we now know $\mathbf{p}$ is the solution to the optimisation problem

$$
\begin{aligned}
&\arg \max_{\mathbf{p}\,:\,||\mathbf{p}||=1} \mathbf{p^T X^T YY^T Xp} \\
=&\arg \max_{\mathbf{p}\,:\,||\mathbf{p}||=1} (\mathbf{Y^T Xp})^\mathbf{T}(\mathbf{Y^T Xp}) \\
=&\arg \max_{\mathbf{p}\,:\,||\mathbf{p}||=1} (cov(\mathbf{Y}, \mathbf{Xp}))^T cov(\mathbf{Y}, \mathbf{Xp}) \\
=&\arg \max_{\mathbf{p}\,:\,||\mathbf{p}||=1} (\sqrt{var(\mathbf{Y})} corr(\mathbf{Y}, \mathbf{Xp}) \sqrt{var(\mathbf{Xp})})^T \sqrt{var(\mathbf{Y})} corr(\mathbf{Y}, \mathbf{Xp}) \sqrt{var(\mathbf{Xp})} \\
=&\arg \max_{\mathbf{p}\,:\,||\mathbf{p}||=1} var(\mathbf{Xp})(corr(\mathbf{Y}, \mathbf{Xp}))^T corr(\mathbf{Y}, \mathbf{Xp})
\end{aligned} \tag{14}
$$

We have used (1) in the above. (14) is the most common statistical interpretation of PLS given in places like [HTF01] and [FF93]. It is worth noting that in PCA, we are only picking the $\mathbf{p}$ that maximises $var(\mathbf{Xp})$.

Fact 15. Another way to look at the above is that we have

$$\begin{aligned} cov(\mathbf{Y^T XP}, \mathbf{Y^T XP}) &= \frac{1}{k-1}(\mathbf{Y^T XP})^\mathbf{T}\mathbf{Y^T XP} \\ &= \frac{1}{k-1}\mathbf{P^T X^T Y Y^T XP} \\ &= \frac{1}{k-1}\mathbf{P^T}(\mathbf{Y^T X})^\mathbf{T}(\mathbf{Y^T X})\mathbf{P} \\ &= \mathbf{P^T U D U^T P}, \end{aligned}$$

where $\frac{1}{k-1}(\mathbf{Y^T X})^\mathbf{T}(\mathbf{Y^T X})$ is the covariance matrix of $\mathbf{Y^T X}$, and $\mathbf{U}$ is its eigenvectors and $\mathbf{D} = diag(\lambda_1 \dots \lambda_m)$ are its eigenvalues such that $\lambda_1 > \lambda_2 > \cdots > \lambda_m$. By the the same reasoning as in Fact 11, we see that the $\mathbf{P}$ matrix in (11) is chosen so that

$$cov(\mathbf{Y^T XP}, \mathbf{Y^T XP}) = \mathbf{D}.$$

By the same reasoning as above, we can show that $\mathbf{q}$ is an eigenvector of $(\mathbf{X^T Y})^\mathbf{T}\mathbf{X^T Y}$ and the whole $\mathbf{Q}$ matrix in (11) is chosen so that

$$cov(\mathbf{X^T YQ}, \mathbf{X^T YQ}) = \mathbf{D'}, \tag{15}$$

where $\mathbf{D'} = diag(\lambda'_1 \dots \lambda'_k)$, $\lambda'_1 > \lambda'_2 > \cdots > \lambda'_k$, is the eigenvalues of the covariance matrix of $\mathbf{X^T Y}$.

6 Some Alternatives to PLS

Techniques like PCR and PLS are designed to deal with highly correlated independent variables. Highly correlated variables usually manifest themselves in the form of large number of high coefficient values, which are used to offset each other. Another way to solve this problem is to control the norm of the coefficient vector directly, a technique called regularisation.

Fact 16. A popular form of regularised linear regression is ridge regression, in which we change the objective function from the standard least squares formulation

$$\min_\beta (\mathbf{y} - \mathbf{X}\beta)^\mathbf{T}(\mathbf{y} - \mathbf{X}\beta)$$

to the following

$$\min_\beta (\mathbf{y} - \mathbf{X}\beta)^\mathbf{T}(\mathbf{y} - \mathbf{X}\beta) + \lambda\beta^\mathbf{T}\beta, \tag{16}$$

for a given value of λ. The solution to (16) can be shown to be

$$\hat{\beta}^{ridge} = (\mathbf{X^T X} + \lambda\mathbf{I})^{-\mathbf{1}}\mathbf{X^T y}.$$

Note the similarity to (2). The 'right' value of λ for a given problem is usually obtained through cross validation.

Fact 17. Another popular form of regularised linear regression is lasso, which solves the following problem:

$$\min_\beta (\mathbf{y} - \mathbf{X}\beta)^\mathbf{T}(\mathbf{y} - \mathbf{X}\beta) + \lambda \sum_\mathbf{i} |\beta_\mathbf{i}|.$$

Experimental and theoretical studies in [HTF01] show that PLS, PCR, and ridge regression tend to behave similarly. Ridge regression maybe preferred for its relative interpretational and computational simplicity.

References

[FF93] Ildiko E. Frank and Jerome H. Friedman. A statistical view of some chemometrics regression tools (with discussion). *Technometrics*, 35(2):109–148, 1993.

[GBGL08] Timothy Gowers, June Barrow-Green, and Imre Leader, editors. *The Princeton Companion to Mathematics*. Princeton University Press, 2008.

[GK86] Paul Geladi and Bruce R. Kowalski. Partial least squares regression: A tutorial. *Analytica Chimica Acta*, 185:1–17, 1986.

[GL96] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. John Hopkins University Press, third edition, 1996.

[HTF01] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer, 2001.

[Lay97] David Lay. *Linear Algebra And Its Applications*. Addison Wesley Longman, 2nd edition, 1997.

A R Code Samples

Here are example R code for doing different versions of linear regression. The original author is Jean-Philippe.Vert@mines.org and the code appears at http://cbio.ensmp.fr/~jvert/svn/tutorials/practical/linearregression/linearregression.R.

```
####################################
## Prepare data
####################################

# Download prostate data
con = url ("http://www-stat.stanford.edu/~tibs/ElemStatLearn/datasets/prostate.data")
prost=read.csv(con,row.names=1,sep="\t")
# Alternatively, load the file and read from local file as follows
# prost=read.csv('prostate.data.txt',row.names=1,sep="\t")

# Scale data and prepare train/test split
prost.std <- data.frame(cbind(scale(prost[,1:8]),prost$lpsa))
names(prost.std)[9] <- 'lpsa'
data.train <- prost.std[prost$train,]
data.test <- prost.std[!prost$train,]
y.test <- data.test$lpsa
n.train <- nrow(data.train)


####################################
## PCR
####################################

library(pls)

m.pcr <- pcr(lpsa ~ .,data=data.train , validation="CV")

# select number of components (by CV)
ncomp <- which.min(m.pcr$validation$adj)

# predict
y.pred.pcr <- predict(m.pcr,data.test , ncomp=ncomp)
summary((y.pred.pcr - y.test)^2)

####################################
## PLS
####################################

library(pls)

m.pls <- plsr(lpsa ~ .,data=data.train , validation="CV")

# select number of components (by CV)
ncomp <- which.min(m.pls$validation$adj)

# predict
y.pred.pls <- predict(m.pls,data.test , ncomp=ncomp)
summary((y.pred.pls - y.test)^2)
```

```
####################################
## Ridge regression
####################################

library(MASS)
m.ridge <- lm.ridge(lpsa ~ .,data=data.train, lambda = seq(0,20,0.1))
plot(m.ridge)

# select parameter by minimum GCV
plot(m.ridge$GCV)

# Predict is not implemented so we need to do it ourselves
y.pred.ridge = scale(data.test[,1:8],center = F, scale = m.ridge$scales)
                %*% m.ridge$coef[,which.min(m.ridge$GCV)] + m.ridge$ym
summary((y.pred.ridge - y.test)^2)

####################################
## Lasso
####################################

library(lars)
m.lasso <- lars(as.matrix(data.train[,1:8]),data.train$lpsa)
plot(m.lasso)

# Cross-validation
r <- cv.lars(as.matrix(data.train[,1:8]),data.train$lpsa)
# bestfraction <- r$fraction[which.min(r$cv)]
bestfraction <- r$index[which.min(r$cv)]

# Observe coefficients
coef.lasso <- predict(m.lasso,as.matrix(data.test[,1:8]),
                      s=bestfraction,type="coefficient",mode="fraction")
coef.lasso

# Prediction
y.pred.lasso <- predict(m.lasso,as.matrix(data.test[,1:8]),
                        s=bestfraction,type="fit",mode="fraction")$fit
summary((y.pred.lasso - y.test)^2)
```