Chapter 3

Multivariate Statistics

3.1 Introduction

We discuss covariance matrices, multivariate linear regression, feature selection, principal component analysis and singular value decomposition. See Chatfield's book on multivariate analysis for more details [10]. Also, a good practical introduction to the material on regression is presented by Kleinbaum et al. [32]. More details of matrix manipulations are available in Weisberg [64] and Strang has a great in-depth intro to linear algebra [58]. See also relevant material in *Numerical Recipes* [49].

3.2 Multivariate Linear Regression

For a multivariate linear data set, the dependent variable y_i is modelled as a linear combination of the input variables $\boldsymbol{x}_i$ and an error term [1]

$$y_i = \boldsymbol{x}_i \boldsymbol{w} + e_i \tag{3.1}$$

where $\boldsymbol{x}_i$ is a row vector, $\boldsymbol{w}$ is a column vector and e_i is an error. The overall goodness of fit can be assessed by the least squares cost function

$$E = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \tag{3.2}$$

where $\hat{\boldsymbol{y}} = \boldsymbol{x}_i \boldsymbol{w}$.

[1] The error term is introduced because, very often, given a particular data set it will not be possible to find an exact linear relationship between $\boldsymbol{x}_i$ and y_i for every i. We therefore cannot directly estimate the weights as $\boldsymbol{X}^{-1}\boldsymbol{y}$.

3.2.1 Estimating the weights

The least squares cost function can be written in matrix notation as

$$E = (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}) \tag{3.3}$$

where $\boldsymbol{X}$ is an N-by-p matrix whose rows are made up of different input vectors and $\boldsymbol{y}$ is a vector of targets. The weight vector that minimises this cost function can be calculated by setting the first derivative of the cost function to zero and solving the resulting equation.

By expanding the brackets and collecting terms (using the matrix identity $(\boldsymbol{AB})^T = \boldsymbol{B}^T\boldsymbol{A}^T$ we get

$$E = \boldsymbol{y}^T\boldsymbol{y} - 2\boldsymbol{w}\boldsymbol{X}^T\boldsymbol{y} - \boldsymbol{w}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} \tag{3.4}$$

The derivative with respect to $\boldsymbol{w}$ is [2]

$$\frac{\partial E}{\partial \boldsymbol{w}} = -2\boldsymbol{X}^T\boldsymbol{y} - 2\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} \tag{3.5}$$

Equating this derivative to zero gives

$$(\boldsymbol{X}^T\boldsymbol{X})\boldsymbol{w} = \boldsymbol{X}^T\boldsymbol{y} \tag{3.6}$$

which, in regression analysis, is known as the 'normal equation'. Hence,

$$\hat{\boldsymbol{w}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y} \tag{3.7}$$

This is the general solution for multivariate linear regression [3]. It is a unique minimum of the least squares error function (ie. this is the only solution).

Once the weights have been estimated we can then estimate the error or noise variance from

$$\sigma_e^2 = \frac{1}{N-1}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{3.8}$$

3.2.2 Understanding the solution

If the inputs are zero mean then the input covariance matrix multiplied by N-1 is

$$\boldsymbol{C}_x = \boldsymbol{X}^T\boldsymbol{X} \tag{3.9}$$

The weights can therefore be written as

$$\hat{\boldsymbol{w}} = \boldsymbol{C}_x^{-1}\boldsymbol{X}^T\boldsymbol{y} \tag{3.10}$$

ie. the inverse covariance matrix times the inner products of the inputs with the output (the ith weight will involve the inner product of the ith input with the output).

[2]From matrix calculus [37] we know that the derivative of $\boldsymbol{c}^T\boldsymbol{B}\boldsymbol{c}$ with respect to $\boldsymbol{c}$ is $(\boldsymbol{B}^T + \boldsymbol{B})\boldsymbol{c}$. Also we note that $\boldsymbol{X}^T\boldsymbol{X}$ is symmetric.

[3]In practice we can use the equivalent expression $\hat{\boldsymbol{w}} = \boldsymbol{X}^{+1}\boldsymbol{y}$ where $\boldsymbol{X}^{+1}$ is the pseudo-inverse [58]. This method is related to Singular Value Decomposition and is discussed later.

Single input

For a single input $\boldsymbol{C}_x^{-1} = 1/(N-1)\sigma_{x_1}^2$ and $\boldsymbol{X}^T\boldsymbol{y} = (N-1)\sigma_{x_1 y}$. Hence

$$\hat{w}_1 = \frac{\sigma_{x_1 y}}{\sigma_{x_1}^2} \tag{3.11}$$

This is *exactly* the same as the estimate for the slope in linear regression (first lecture). This is re-assuring.

Uncorrelated inputs

For two uncorrelated inputs

$$\boldsymbol{C}_x^{-1} = \begin{bmatrix} \frac{1}{(N-1)\sigma_{x_1}^2} & 0 \\ 0 & \frac{1}{(N-1)\sigma_{x_2}^2} \end{bmatrix} \tag{3.12}$$

We also have

$$\boldsymbol{X}^T\boldsymbol{y} = \begin{bmatrix} (N-1)\sigma_{x_1,y} \\ (N-1)\sigma_{x_2,y} \end{bmatrix} \tag{3.13}$$

The two weights are therefore

$$\begin{aligned} \hat{w}_1 &= \frac{\sigma_{x_1 y}}{\sigma_{x_1}^2} \\ \hat{w}_2 &= \frac{\sigma_{x_2 y}}{\sigma_{x_2}^2} \end{aligned} \tag{3.14}$$

Again, these solutions are the same as for the univariate linear regression case.

General case

If the inputs are correlated then a coupling is introduced in the estimates of the weights; weight 1 becomes a function of $\sigma_{x_2 y}$ as well as $\sigma_{x_1 y}$

$$\hat{\boldsymbol{w}} = \begin{bmatrix} \sigma_{x_1}^2 & \sigma_{x_1 x_2} \\ \sigma_{x_1 x_2} & \sigma_{x_2}^2 \end{bmatrix}^{-1} \begin{bmatrix} \sigma_{x_1,y} \\ \sigma_{x_2,y} \end{bmatrix} \tag{3.15}$$

3.2.3 Feature selection

Some of the inputs in a linear regression model may be very useful in predicting the output. Others, not so. So how do we find which inputs or *features* are useful ? This problem is known as feature selection.

The problem is tackled by looking at the coefficients of each input (ie. the weights) and seeing if they are significantly non-zero. The procedure is identical to that described for univariate linear regression.

The only added difficulty is that we have more inputs and more weights, but the procedure is basically the same. Firstly, we have to estimate the variance on each weight. This is done in the next section. We then compare each weight to zero using a t-test.

The weight covariance matrix

Different instantiations of target noise will generate different estimated weight vectors according to equation 3.7. For the case of Gaussian noise we do not actually have to compute the weights on many instantiations of the target noise and then compute the sample covariance [4]; the corresponding weight covariance matrix is given by the equation

$$\boldsymbol{\Sigma} = Var((\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}) \tag{3.16}$$

In the appendix we show that this can be evaluated as

$$\boldsymbol{\Sigma} = \sigma_e^2(\boldsymbol{X}^T\boldsymbol{X})^{-1} \tag{3.17}$$

The correlation in the inputs introduces a correlation in the weights; for uncorrelated inputs the weights will be uncorrelated. The variance of the jth weight, w_j, is then given by the jth diagonal entry in the covariance matrix

$$\sigma_{w_j}^2 = \boldsymbol{\Sigma}_{jj} \tag{3.18}$$

To see if a weight is significantly non-zero we then compute $CDF_t(t)$ (the cumulative density function; see earlier lecture) where $t = w_j/\sigma_{w_j}$ and if it is above some threshold, say $p = 0.05$, the corresponding feature is removed.

Note that this procedure, which is based on a t-test, is exactly equivalent to a similar procedure based on a partial F-test (see, for example, [32] page 128).

If we do remove a weight then we must recompute all the other weights (and variances) *before* deciding whether or not the other weights are significantly non-zero. This usually proceeds in a stepwise manner where we start with a large number of features and reduce them as necessary (*stepwise backward selection*) or gradually build up the number of features (*stepwise forward selection*) [32].

Note that, if the weights were uncorrelated we could do feature selection in a single step; we would not have to recompute weight values after each weight removal. This provides one motivation for the use of orthogonal transforms in which the weights *are* uncorrelated. Such transforms include Fourier and Wavelet transforms as we shall see in later lectures.

[4] But this type of procedure is the basis of bootstrap estimates of parameter variances. See [17].

3.2.4 Example

Suppose we wish to predict a time series x_3 from two other time series x_1 and x_2. We can do this with the following regression model [5]

$$x_3 = w_0 + w_1 x_1 + w_2 x_2 \tag{3.19}$$

and the weights can be found using the previous formulae. To cope with the constant, w_0, we augment the $\boldsymbol{X}$ vector with an additional column of 1's.

We analyse data having covariance matrix $\boldsymbol{C}_1$ and mean vector $\boldsymbol{m}_1$ (see equations 2.15 and 2.14 in an earlier lecture). $N = 50$ data points were generated and are shown in Figure 3.1. The weights were then estimated from equation 3.7 as

$$\begin{aligned}\hat{\boldsymbol{w}} &= [w_1, w_2, w_0]^T \\ &= [1.7906, -0.0554, 0.6293]^T\end{aligned} \tag{3.20}$$

Note that w_1 is much bigger than w_2. The weight covariance matrix was estimated from equation B.27 as

$$\boldsymbol{\Sigma} = \begin{bmatrix} 0.0267 & 0.0041 & -0.4197 \\ 0.0041 & 0.0506 & -0.9174 \\ -0.4197 & -0.9174 & 21.2066 \end{bmatrix} \tag{3.21}$$

giving $\sigma_{w_1} = 0.1634$ and $\sigma_{w_2} = 0.2249$. The corresponding t-statistics are $t_1 = 10.96$ and $t_2 = -0.2464$ giving p-values of 10^{-15} and 0.4032. This indicates that the first weight is significantly different from zero but the second weight is not ie. x_1 is a good predictor of x_3 but x_2 is not. We can therefore remove x_2 from our regression model.

Question: But what does linear regression tell us about the data that the correlation/covariance matrix does'nt ? *Answer:* Partial correlations.

3.2.5 Partial Correlation

Remember (see eg. equation 1.36 from lecture 1), the square of the correlation coefficient between two variables x_1 and y is given by

$$r^2_{x_1 y} = \frac{\sigma^2_y - \sigma^2_e(x_1)}{\sigma^2_y} \tag{3.22}$$

where $\sigma^2_e(x_1)$ is the variance of the errors from using a linear regression model based on x_1 to predict y. Writing $\sigma^2_y = \sigma^2_e(0)$, ie. the error with no predictive variables

$$r^2_{x_1 y} = \frac{\sigma^2_e(0) - \sigma^2_e(x_1)}{\sigma^2_e(0)} \tag{3.23}$$

[5] Strictly, we can only apply this model if the samples *within* each time series are independent (see later). To make them independent we can randomize the time index thus removing any correlation between lagged samples. We therefore end up with a random variables rather than time series.

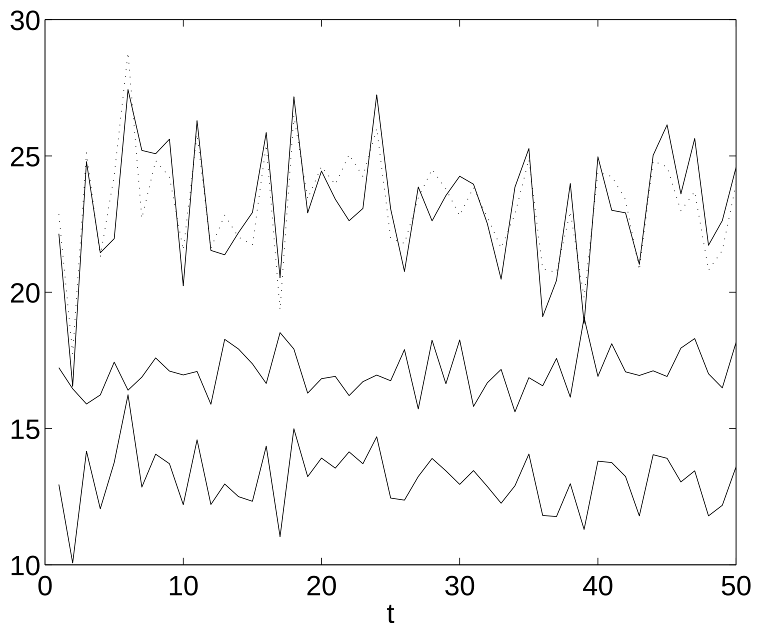

Figure 3.1: *Three time series having the correlation matrix* $\boldsymbol{C}_1$ *and mean vector* $\boldsymbol{m}_1$ *shown in the text. The dotted line shows the value of the third time series as predicted from the other two using a regression model.*

When we have a second predictive variable x_2, the square of the *partial correlation* between x_2 and y is defined as

$$r^2_{x_2y|x_1} = \frac{\sigma_e^2(x_1) - \sigma_e^2(x_1, x_2)}{\sigma_e^2(x_1)} \tag{3.24}$$

where $\sigma_e^2(x_1, x_2)$ is the variance of the errors from the regression model based on x_1 and x_2. It's the extra proportion of variance in y explained by x_2. It's different to $r^2_{x_2y}$ because x_2 may be correlated to x_1 which itself explains some of the variance in y. After *controlling* for this, the resulting proportionate reduction in variance is given by $r^2_{x_2y|x_1}$. More generally, we can define pth order partial correlations which are the correlations between two variables after controlling for p variables.

The sign of the partial correlation is given by the sign of the corresponding regression coefficient.

Relation to regression coefficients

Partial correlations are to regression coefficients what the correlation is to the slope in univariate linear regression. If the partial correlation is significantly non-zero then the corresponding regression coefficient will also be. And vice-versa.

3.3 Principal Component Analysis

Given a set of data vectors $\{\boldsymbol{x}_n\}$ we can construct a covariance matrix

$$\boldsymbol{C} = \frac{1}{N}\sum_n (\boldsymbol{x}_n - \bar{\boldsymbol{x}})(\boldsymbol{x}_n - \bar{\boldsymbol{x}})^T \tag{3.25}$$

or, if we construct a matrix $\boldsymbol{X}$ with rows equal to $\boldsymbol{x}_n - \bar{\boldsymbol{x}}$ then

$$\boldsymbol{C} = \frac{1}{N}\boldsymbol{X}^T\boldsymbol{X} \tag{3.26}$$

Because covariance matrices are real and symmetric we can apply the spectral theorem

$$\boldsymbol{C} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^T \tag{3.27}$$

If the eigenvectors (columns of $\boldsymbol{Q}$) are normalised to unit length, they constitute an orthonormal basis. If the eigenvalues are then ordered in magnitude such that $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_d$ then the decomposition is known as Principal Component Analysis (PCA). The projection of a data point $\boldsymbol{x}_n$ onto the principal components is

$$\boldsymbol{y}_n = \boldsymbol{Q}^T\boldsymbol{x}_n \tag{3.28}$$

The mean projection is

$$\bar{\boldsymbol{y}} = \boldsymbol{Q}^T\bar{\boldsymbol{x}} \tag{3.29}$$

The covariance of the projections is given by the matrix

$$\boldsymbol{C}_y = \frac{1}{N}\sum_n (\boldsymbol{y}_n - \bar{\boldsymbol{y}})(\boldsymbol{y}_n - \bar{\boldsymbol{y}})^T \tag{3.30}$$

Substituting in the previous two expressions gives

$$\begin{aligned}\boldsymbol{C}_y &= \frac{1}{N}\sum_n \boldsymbol{Q}^T(\boldsymbol{x}_n - \bar{\boldsymbol{x}})(\boldsymbol{x}_n - \bar{\boldsymbol{x}})^T\boldsymbol{Q} \\ &= \boldsymbol{Q}^T\boldsymbol{C}\boldsymbol{Q} \\ &= \boldsymbol{\Lambda}\end{aligned} \tag{3.31}$$

where $\boldsymbol{\Lambda}$ is the diagonal eigenvalue matrix with entries λ_k ($\sigma_k^2 = \lambda_k$). This shows that the variance of the kth projection is given by the kth eigenvalue. Moreover, it says that the projections are uncorrelated. PCA may therefore be viewed as a linear transform

$$\boldsymbol{y} = \boldsymbol{Q}^T\boldsymbol{x} \tag{3.32}$$

which produces uncorrelated data.

3.3.1 The Multivariate Gaussian Density

In d dimensions the general multivariate normal probability density can be written

$$p(\boldsymbol{x}) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{C}|^{1/2}}\exp\left(-\frac{1}{2}(\boldsymbol{x} - \bar{\boldsymbol{x}})^T C^{-1}(\boldsymbol{x} - \bar{\boldsymbol{x}})\right) \tag{3.33}$$

where the mean $\bar{\boldsymbol{x}}$ is a d-dimensional vector, $\boldsymbol{C}$ is a $d \times d$ covariance matrix, and $|\boldsymbol{C}|$ denotes the determinant of $\boldsymbol{C}$. Because the determinant of a matrix is the product of its eigenvalues then for covariance matrices, where the eigenvalues correspond to variances, the determinant is a single number which represents the total volume of variance. The quantity

$$M(\boldsymbol{x}) = (\boldsymbol{x} - \bar{\boldsymbol{x}})^T C^{-1} (\boldsymbol{x} - \bar{\boldsymbol{x}}) \tag{3.34}$$

which appears in the exponent is called the *Mahalanobis distance* from $\boldsymbol{x}$ to $\bar{\boldsymbol{x}}$. This is the equation for an ellipse (see earlier). The directions of the axes are given by the principal components and the lengths are given by $\sigma_i M(\boldsymbol{x})$ where σ_i is the standard deviation of the data in the ith direction (see earlier section on quadratic forms and note that $\lambda_i = \sigma_i^2$). We can therefore map a given probability $p(\boldsymbol{x})$ to a Mahalanobis distance (using equation E.9) and from that plot the ellipse axes. See the figure in the appendix.

3.3.2 Dimensionality Reduction

Given that the eigenvalues in PCA are ordered and that they correspond to the variance of the data in orthogonal directions then it would seem plausible that a reasonable data reconstruction could be obtained from just a few of the larger components and this is indeed the case.

If we retain only a subset $M < d$ of the basis vectors then a data point can be reconstructed as

$$\hat{\boldsymbol{x}}_n = \sum_{k=1}^{M} w_k^n \boldsymbol{q}_k + \sum_{k=M+1}^{d} b_k \boldsymbol{q}_k \tag{3.35}$$

where the b_k are constants (they don't depend on n) and, as we have seen, $w_k^n = \boldsymbol{q}_k^T \boldsymbol{x}_n$. If we keep only the projections w_k^n and the associated eigenvectors $\boldsymbol{q}_k$ we have reduced the dimension of our data set from d to M. Now, given that the actual data point can be written as

$$\boldsymbol{x}_n = \sum_{k=1}^{d} w_k^n \boldsymbol{q}_k \tag{3.36}$$

where the sum is over all d components (not just M) then the reconstruction error is

$$\boldsymbol{x}_n - \hat{\boldsymbol{x}}_n = \sum_{k=M+1}^{d} (w_k^n - b_k) \boldsymbol{q}_k \tag{3.37}$$

It is the cost of replacing the variable w_k^n by a constant b_k. The reconstruction error averaged over the whole data set is

$$\begin{aligned} E_M &= \frac{1}{N} \sum_{n=1}^{N} ||\boldsymbol{x}_n - \hat{\boldsymbol{x}}_n|| \\ &= \frac{1}{N} \sum_{n=1}^{N} \sum_{k=M+1}^{d} (w_k^n - b_k)^2 \end{aligned} \tag{3.38}$$

where the $\boldsymbol{q}_k$'s disappear because $\boldsymbol{q}_k^T \boldsymbol{q}_k = 1$. We can minimise E_M by setting

$$\begin{aligned} b_k &= \frac{1}{N} \sum_{n=1}^{N} w_k^n \\ &= \boldsymbol{q}_k^T \bar{\boldsymbol{x}} \end{aligned} \tag{3.39}$$

which is the mean projection in direction $\boldsymbol{q}_k$. The error is therefore

$$\begin{aligned} E_M &= \frac{1}{N} \sum_{n=1}^{N} \sum_{k=M+1}^{d} \left[\boldsymbol{q}_k^T (\boldsymbol{x}_n - \bar{\boldsymbol{x}}) \right]^2 \\ &= \frac{N}{N} \sum_{k=M+1}^{d} \boldsymbol{q}_k^T \boldsymbol{C} \boldsymbol{q}_k \\ &= \sum_{k=M+1}^{d} \lambda_k \end{aligned} \tag{3.40}$$

The reconstruction error is therefore minimised, for a given M, by throwing away the $d - M$ smallest components, as you would expect. The corresponding error is just the sum of the corresponding eigenvalues.

3.3.3 Singular Value Decomposition

The eigenvalue-eigenvector factorisation (see equation 2.85)

$$\boldsymbol{A} = \boldsymbol{Q} \boldsymbol{\Lambda} \boldsymbol{Q}^T \tag{3.41}$$

applies to square symmetric matrices only. There is an equivalent factorisation for rectangular matrices, having N rows and d columns, called Singular Value Decomposition (SVD)

$$\boldsymbol{A} = \boldsymbol{Q}_1 \boldsymbol{D} \boldsymbol{Q}_2^T \tag{3.42}$$

where $\boldsymbol{Q_1}$ is an orthonormal N-by-N matrix, $\boldsymbol{Q_2}$ is an orthonormal d-by-d matrix, $\boldsymbol{D}$ is a diagonal matrix of dimension N-by-d and the kth diagonal entry in $\boldsymbol{D}$ is known as the kth singular value, σ_k.

If we substitute the SVD of $\boldsymbol{A}$ into $\boldsymbol{A}^T \boldsymbol{A}$, after some rearranging, we get

$$\boldsymbol{A}^T \boldsymbol{A} = \boldsymbol{Q}_2 \boldsymbol{D}^T \boldsymbol{D} \boldsymbol{Q}_2^T \tag{3.43}$$

which is of the form $\boldsymbol{A} = \boldsymbol{Q} \boldsymbol{\Lambda} \boldsymbol{Q}^T$ where $\boldsymbol{Q} = \boldsymbol{Q}_2$ and $\boldsymbol{\Lambda} = \boldsymbol{D}^T \boldsymbol{D}$. This shows that the columns of $\boldsymbol{Q}_2$ contain the eigenvectors of $\boldsymbol{A}^T \boldsymbol{A}$ and that $\boldsymbol{D}$ contains the square roots of the corresponding eigenvalues. Similarly, by substituting the SVD of $\boldsymbol{A}$ into $\boldsymbol{A} \boldsymbol{A}^T$ we can show that the columns of $\boldsymbol{Q}_1$ are the eigenvectors of $\boldsymbol{A} \boldsymbol{A}^T$.

Relation to PCA

Given a data matrix $\boldsymbol{X}$ constructed as before (see PCA section), except that the matrix is scaled by a normalisation factor $\sqrt{1/N}$, then $\boldsymbol{X}^T\boldsymbol{X}$ is equivalent to the covariance matrix $\boldsymbol{C}$. If we therefore decompose $\boldsymbol{X}$ using SVD, the principal components will apear in $\boldsymbol{Q}_2$ and the square roots of the corresponding eigenvalues will appear in $\boldsymbol{D}$.

Therefore we can implement PCA in one of two ways (i) compute the covariance matrix and perform an eigendecomposition or (ii) use SVD directly on the (normalised) data matrix.

The Pseudo-Inverse

Given the SVD of a matrix

$$\boldsymbol{A} = \boldsymbol{Q}_1\boldsymbol{D}\boldsymbol{Q}_2^T \tag{3.44}$$

the *Pseudo-Inverse* of $\boldsymbol{A}$ is defined as

$$\boldsymbol{A}^+ = \boldsymbol{Q}_2\boldsymbol{D}^+\boldsymbol{Q}_1^T \tag{3.45}$$

where $\boldsymbol{D}^+$ is a d-by-N matrix with diagonal entries $1/\sigma_1, 1/\sigma_2, ..., 1/\sigma_d$. The matrix $\boldsymbol{D}^+$ can be computed as

$$\boldsymbol{D}^+ = (\boldsymbol{D}^T\boldsymbol{D})^{-1}\boldsymbol{D}^T \tag{3.46}$$

The Pseudo-Inverse is used in the solution of the multivariate linear regression problem (see equation 3.7)

$$\hat{\boldsymbol{w}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y} \tag{3.47}$$

We can substitute the SVD for $\boldsymbol{X}$ into the above expression in a series of steps to give

$$\boldsymbol{X}^T\boldsymbol{X} = \boldsymbol{Q}_2\boldsymbol{D}^T\boldsymbol{D}\boldsymbol{Q}_2^T \tag{3.48}$$

The inverse is

$$(\boldsymbol{X}^T\boldsymbol{X})^{-1} = \boldsymbol{Q}_2(\boldsymbol{D}^T\boldsymbol{D})^{-1}\boldsymbol{Q}_2^T \tag{3.49}$$

Hence

$$(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T = \boldsymbol{Q}_2(\boldsymbol{D}^T\boldsymbol{D})^{-1}\boldsymbol{D}^T\boldsymbol{Q}_1^T \tag{3.50}$$

Substituting for $\boldsymbol{D}^+$ gives

$$\begin{aligned} (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T &= \boldsymbol{Q}_2\boldsymbol{D}^+\boldsymbol{Q}_1^T \\ &= \boldsymbol{X}^+ \end{aligned} \tag{3.51}$$

Therefore, the linear regression weights can be computed by projecting the targets onto the Pseudo-Inverse of the input data matrix

$$\hat{\boldsymbol{w}} = \boldsymbol{X}^+\boldsymbol{y} \tag{3.52}$$