Lesson 4

"Quick and Dirty"

Since its introduction by C. F. Gauss in the early 1800's, the least-squares parameter matching technique has penetrated to all fields of research and practical engineering work, and it still seems to be among the only ones that are routinely used. However, there are some probloms that are not easily detected — these problems become evident only in the complex modeling tasks, where there is plenty of data that is necessarily not optimally conditioned. In this chapter, the least-squares regression method is first derived, and modifications are presented; finally, the fundamental problem (so called *multicollinearity*) plaguing this method is explained, giving motivation to search for more sophisticated methods.

4.1 Linear regression model

As presented in the previous chapter, assume that the measurement data is collected in the matrices X of dimension $k \times n$ and Y of dimension $k \times m$. It is assumed that there are (much) more measurement samples than what is the dimension of the data, that is, $k \gg n$. One would like to find the matrix F so that

$$Y = X \cdot F \tag{4.1}$$

would hold. Finding a good matrix F is the main emphasis from now on. Even though the modeling problem can be formulated in such a simple way, in a multivariate system the task is far from trivial. There are $n \cdot m$ free parameters in the model, and the optimum is searched for in this parameter space.

4.1.1 Least-squares solution

To start with, first study a model of just one output signal Y_i, so that $m = 1$. The parameter matrix reduces to a vector F_i:

$$Y_i = X \cdot F_i. \tag{4.2}$$

Solving for F_i in (4.2) means that somehow X should be inverted; however, X is not invertible, and because $k > n$, generally no exact solutions can be found. To find the best approximation, the model needs to be extended to include the modeling errors as

$$\tilde{Y}_i = X \cdot F_i + E_i, \tag{4.3}$$

where E_i is a $k \times 1$ vector containing the reconstruction error for each measurement sample k. It is only these noisy measurements $\tilde{y}$ that are assumed to be available for modeling; in what follows, the sloppy notation y will for brevity still be used to denote the noisy data. Now there are more unknowns than there are constraints, and the problem can be transformed into a form where optimization is being carried out. It needs to be recognized that the formulation in (4.3) is just a model representing the coupling of uncertainty in the system. The variables $e_i(\kappa)$ do not represent any real noise signals in the system, they only stand for the match between the model and the data. In this sense, minimizing this uncertainty is a justified objective.

The errors can be solved from (4.3) as $E_i = Y_i - XF_i$; these errors should be somehow simultaneously minimized. It turns out that the easiest way to proceed — and also theoretically well motivated, as shown in the previous chapter — is to *minimize the sum of error squares.* The sum of the squared errors can be expressed as

$$\begin{aligned} E_i^T E_i &= (Y_i - XF_i)^T (Y_i - XF_i) \\ &= Y_i^T Y_i - Y_i^T X F_i - F_i^T X^T Y_i + F_i^T X^T X F_i. \end{aligned} \tag{4.4}$$

This (scalar) can be differentiated with respect to the parameter vector F_i:

$$\frac{d\left(E_i^T E_i\right)}{d\,F_i} = \mathbf{0} - X^T Y_i - X^T Y_i + 2X^T X F_i. \tag{4.5}$$

Because

$$\frac{d^2\left(E_i^T E_i\right)}{d\,F_i^2} = 2X^T X > 0, \tag{4.6}$$

this extremum is minimum, and because the extremum of a quadratic function is unique, setting the derivative to zero (vector) gives the unique optimum parameters:

$$-2X^T Y_i + 2X^T X F_i = \mathbf{0}, \tag{4.7}$$

resulting in

$$F_i = \left(X^T X\right)^{-1} X^T Y_i. \tag{4.8}$$

The estimate for y_i is found as

$$\hat{y}_{\text{est},i} = F_i^T x_{\text{est}} = Y_i^T X \left(X^T X\right)^{-1} x_{\text{est}}. \tag{4.9}$$

This result can be intuitively interpreted also in terms of correlation matrices: First, covariance structure in x is eliminated (multiplication by $\left(\frac{1}{k}X^TX\right)^{-1}$), and after that the "whitened" data is mapped onto y utilizing the cross-correlation structure (as revealed by $\frac{1}{k}X^TY$).

4.1.2 Piece of analysis

In what follows, some theoretical analyses that can be used to evaluate the above least squares model are presented. More analysis can be found in various sources, for example, in [35].

Model consistency

Because the model construction was an optimization process based on stochastic data, the model parameters cannot be assumed to be quite accurate. Indeed, for the parameter estimates one can write

$$\begin{aligned} \hat{F}_i &= \left(X^TX\right)^{-1}X^TY_i \\ &= \left(X^TX\right)^{-1}X^T\left(XF_i+E_i\right) \\ &= F_i+\left(X^TX\right)^{-1}X^T\cdot E_i. \end{aligned} \tag{4.10}$$

Here, $\hat{F}_i$ are the estimates, whereas F_i is assumed to contain the "true" noiseless parameter values. From this one can write the expression for parameter errors:

$$\tilde{F}_i = F_i-\hat{F}_i = \left(X^TX\right)^{-1}X^T\cdot E_i. \tag{4.11}$$

The expected parameter error is zero, *assuming* that X and E_i do not correlate (this issue is studied later):

$$\mathrm{E}\{\tilde{F}_i\} = \left(X^TX\right)^{-1}X^T\,\mathrm{E}\{E_i\} = 0. \tag{4.12}$$

If this uncorrelatedness assumption does not hold, there will be *bias*. If X is deterministic and E has zero mean, as was assumed, there will be no problem; however, these assumptions cannot always be fulfilled (see Section 4.2.1).

Parameter sensitivity

The reliability of the regression model (4.8) can be approximated, for example, by checking how much the parameters vary as there are stochastic variations in E_i. The parameter vector covariance matrix becomes, applying (4.11)

$$\begin{aligned} \mathrm{E}\{\tilde{F}_i\tilde{F}_i^T\} &= \mathrm{E}\left\{\left(\left(X^TX\right)^{-1}X^TE_i\right)\left(\left(X^TX\right)^{-1}X^TE_i\right)^T\right\} \\ &= \left(X^TX\right)^{-1}X^T\cdot\mathrm{E}\left\{E_iE_i^T\right\}\cdot X\left(X^TX\right)^{-1} \\ &= \left(X^TX\right)^{-1}X^T\sigma_e^2I\,X\left(X^TX\right)^{-1} \\ &= \sigma_e^2\left(X^TX\right)^{-1}. \end{aligned} \tag{4.13}$$

The noise variance σ_e^2 can be approximated as the variance of the reconstruction error $\tilde{Y}_i = Y_i - \hat{Y}_i = Y_i - X\hat{F}_i$. The parameter variance is also intimately related, not only to the noise properties determined by the error variance σ_e^2, but also to the properties of the matrix $X^T X$ — see Sec. 4.3 for more analysis.

The estimate for the parameter error can be applied, for example, when assessing the relevance of the input variables x_j. For example, assume that a least-squares model is constructed, and the corresponding diagonal element in the model parameter covariance matrix is $\mathrm{E}\{\tilde{F}_{jj}^2\} = \sigma_{jj}^2$. Now, assuming that the probability density function form for the error is known (Gaussian?), one can approximate the probability that the parameter F_{jj}, rather than being the estimated $\tilde{F}_{ii}$, actually has zero value. This would mean that there is no contribution of that variable in the model, and it could be ignored.

Without going into details, it turns out that the expression for parameter covariance (4.13) reaches the *Cramer-Rao lower bound,* meaning that for Gaussian data the least-squares model implements the best possible, or *efficient,* estimator for the parameters.

Measures of fit

To evaluate how well the regression model matches the training data, the so called R *squared* criterion can be applied: how much of the real output variance can be explained by the model. That is, one can calculate the quantity

$$R^2 = 1 - \frac{SS_{\mathrm{E}}}{SS_{\mathrm{T}}}, \tag{4.14}$$

where, for the i'th output,

- The "error sum of squares" is defined as

$$SS_{\mathrm{E}} = (Y_i - \hat{Y}_i)^T (Y_i - \hat{Y}_i) = (Y_i - XF_i)^T (Y_i - XF_i). \tag{4.15}$$

- The "total sum of squares" (for zero-mean data) is

$$SS_{\mathrm{T}} = Y_i^T Y_i. \tag{4.16}$$

So, R^2 measures how much of the total variation in the output can be explained by the model. This quantity has value 1 if all the variation in the output can be exactly predicted, and lower value otherwise.

This R^2 is a traditional measure for characterizing least-squares fitting. However, it needs to be emphasized here that it is *not* a good approach for evaluating model goodness. It simply measures data fit, not model goodness: It uses the same data for evaluation that was used for model construction. Applying this criterion for comparing model structures, the least-squares model would always outperform the other structures (to be studied later), no matter how sensitive the model is to noise!

4.1.3 Multivariate case

If there are various output signals, so that $m > 1$, the above analysis can be carried out for each of them separately. When collected together, there holds

$$\begin{cases} F_1 &= \left(X^T X\right)^{-1} X^T Y_1 \\ &\vdots \\ F_m &= \left(X^T X\right)^{-1} X^T Y_m. \end{cases} \tag{4.17}$$

It turns out that this set of formulas can simultaneously be rewritten in a compact matrix form, so that

$$F = \left(\; F_1 \; \middle| \; \cdots \; \middle| \; F_m \; \right) = \left(X^T X\right)^{-1} X^T \cdot \left(\; Y_1 \; \middle| \; \cdots \; \middle| \; Y_m \; \right). \tag{4.18}$$

This means that the *multilinear regression (MLR)* model from X to estimated Y can be written as

$$F_{\text{MLR}} = \left(X^T X\right)^{-1} X^T Y. \tag{4.19}$$

The MLR solution to modeling relationships between variables is exact and optimal in the sense of the least squares criterion, implementing the *pseudoinverse* of the matrix X. However, in Sec. 4.3 it will be shown that one has to be careful when using this regression method: In practical applications and in nonideal environments this MLR approach *may collapse altogether.* The problem is that trying to explain noisy data too exactly may make the model sensitive to individual noise realizations. In any case, in later chapters, the above MLR model is used as the basic engine to reach mappings between variables; the deficiencies of the basic approach are taken care of separately.

The basic MLR solution can be extended and modified in many ways. For example, assuming that not all samples are assumed to be equally informative, one can define the weighted cost criterion for output i as

$$J_i = \sum_{\kappa=1}^{k} w(\kappa) \cdot e_i^2(\kappa) = E_i^T W E_i, \tag{4.20}$$

where the $k \times k$ matrix W contains the weighting factors on the diagonal. then the solution (as expanded to multiple outputs) as

$$F = \left(X^T W X\right)^{-1} X^T W Y. \tag{4.21}$$

The parallel structure of the multivariate problems can be utilized also more generally for extending formulas to multivariate cases. For example, the exponentially weighted (so that $w(\kappa) = \lambda^{k-\kappa}$ with the forgetting factor $0 \ll \lambda ¡ 1$) recursive least-squares algorithm [**?**] corresponding to (4.21) can be extended to multiple outputs, so that $m > 1$, as

$$\begin{array}{rcl} F(k) &=& F(k-1) + \left(R(k)\right)^{-1} x(k) \left(y(k) - F^T(k) x(k)\right)^T \\ R(k) &=& \lambda R(k-1) + x(k) x^T(k). \end{array} \tag{4.22}$$

4.2 "Colored noise"

The above MLR formula will be the standard approach to implementing the mapping between two sets of variables in later chapters. As was observed, it is optimal and efficient — but only if the mapping problem is appropriately conditioned. There are two basic problems that will be discussed during the rest of this chapter. Both of the problems become acute in multivariate cases where the quality of the high-dimensional data cannot be assured.

From the practical point of view, the first problem is caused by the *incompatible model structure* assumption; this issue is studied in this section. In Section 4.3, it is *the robustness problem* that is studied.

4.2.1 Error in variables

In the beginning, it was assumed that the nature of the variables is heterogeneous: It was assumed that Y only is stochastic, noise E being added to it, and X was assumed to be deterministic. To understand the problem of deterministic vs. stochastic variables, study an example.

Now, study a familiar-looking case: Assume that system dynamics is to be modeled:

$$y(k) = F^T \cdot x(k) + e(k), \tag{4.23}$$

where

$$x(k) = \begin{pmatrix} \tilde{y}(k-1) \\ \vdots \\ \tilde{y}(k-n) \end{pmatrix} = \begin{pmatrix} y(k-1) + e(k-1) \\ \vdots \\ y(k-n) + e(k-n) \end{pmatrix}. \tag{4.24}$$

When the input x is defined so that the former outputs are "recirculated" into input, one is identifying the *auto-regressive* (AR) model structure. It is clear that the assumption of x being deterministic collapses; what is more, X becomes correlated with E, so that the model in (4.11) becomes biased.

Clearly, it has to be assumed that X measurements can also contain uncertainty. The model matching problem becomes very different if both X and Y blocks are regarded as equally stochastic data values and errors in all variables should be taken into account (see Fig. 4.1). This assumption results in the so called Error In Variables (EIV) model.

There exists a wide variety of different ways to implement the data homogeneity in the models. As an example, below, one approach is presented for circumventing the problems of correlated noise. A more concise treatment is carried out in the next chapter, where the problem is attacked from a fresh point of view[1].

[1] In chapter 11, a method called Total Least Squares is presented that also addresses Errors in Variables

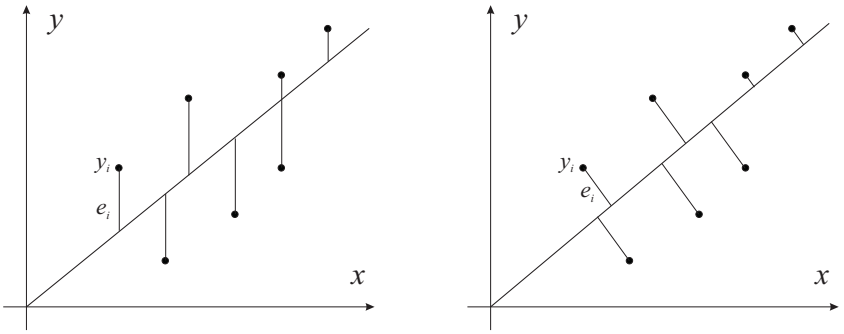

Figure 4.1: Normal least-squares matching principle, on the left, assuming that only the y variables contain noise, and the *total least squares* principle, assuming that errors in all variables are equally meaningful, on the right. For visualization purposes only one x and y variable is employed

4.2.2 Instrumental variables

When constructring linear regression models, after all, it is all about inverting the mapping: Starting from $XF = Y$ solve the mapping matrix F. The challenge is caused by the uninvertibility of the matrix X. However, if the original model holds, there must also hold $\mathcal{X}^T Y = \mathcal{X}^T XF$, where $\mathcal{X}$ is some $k \times n$ matrix. Now, assuming that the matrix $\mathcal{X}^T X$ is invertible, one can solve

$$F = \left(\mathcal{X}^T X\right)^{-1} \mathcal{X}^T Y. \tag{4.25}$$

As in (4.10), one can find the correspondence between the noise and the parameter matrix

$$\hat{F} = F + \left(\mathcal{X}^T X\right)^{-1} \mathcal{X}^T E, \tag{4.26}$$

and, further, for the parameter error one has

$$\tilde{F} = \left(\mathcal{X}^T X\right)^{-1} \mathcal{X}^T E. \tag{4.27}$$

It is interesting here that it is no more the correlation between X and E that determines the model bias: To minimize the model error, there should be high correlation between $\mathcal{X}$ and X, and low correlation between $\mathcal{X}$ and E. Naturally, the first objective is reached for $\mathcal{X} = X$, resulting in the nominal MLR, but when the other objective is also emphasized, non-trivial alternatives can be proposed. The variables in § are called *instrumental variables.*

How to reach good properties for the instruments, is dependent of the situation. For example, if in $y(\kappa) = f^T x(\kappa)$ the $x(\kappa)$ data vector consists of the past values of (scalar) $y(\kappa)$, as shown in (4.24), meaning that AR modeling of a dynamic system is being carried out, different choices have been studied a lot. A good choice for instruments in such case would be to use the correct (noiseless) values of $y(\kappa)$ as collected in §(κ): This would be the optimal choice, and, indeed,

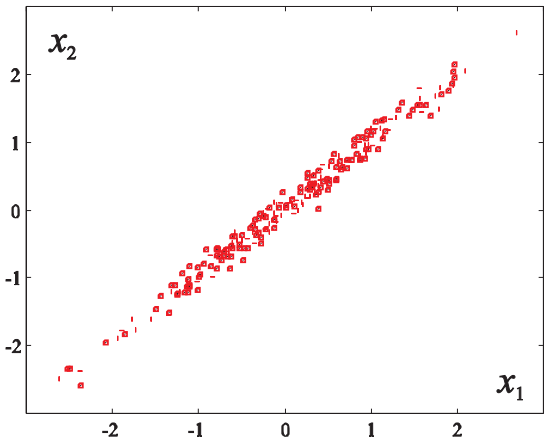

Figure 4.2: Collinearity visualized in two dimensions

this can be approximately implemented. When using the model, in the reconstructed values of y the noise realization has (hopefully) been abstracted away, and these estimates can be used as instruments: Select $\S(\kappa) = \hat{x}(\kappa)$, where $\hat{x}(\kappa)$ vector consists of the past values of $\hat{y}(\kappa)$. When the cycle of first determining a preliminary model and thereafter refining the instruments is repeated, the model parameters finally converge to unbiased values.

4.3 Collinearity

In the previous section, the problem of data heterogeneity was discussed. The deficiencies of MLR become even more painstaking when dimensional complexity is faced.

The MLR regression model is optimal[2]. In simple cases it is difficult to see why *optimality* is in contrast with *usability.* Today, when the problems to be modeled involve large amounts of poor data, the problems of MLR have become evident. The main problem plaguing MLR is caused by *(multi)collinearity.* What this means can best be explained using an example.

4.3.1 Example: When variables are redundant

Assume that one can observe two variables x_1 and x_2, so that $x = (\ x_1 \quad x_2\ )^T$. Further, assume that these variables are not strictly independent; they can be written as $x_1(\kappa) = \xi(\kappa) + \epsilon_1(\kappa)$ and $x_2(\kappa) = \xi(\kappa) + \epsilon_2(\kappa)$, where the sequences $\epsilon_1(\kappa)$ and $\epsilon_2(\kappa)$ are mutually uncorrelated, both having the same variance σ^2. This can be interpreted so that we have two noisy measurements of the same underlying variable ξ, and together these measurements should give a more reliable estimate for it.

[2]of course, only in the least-squares sense; but, because of the mathematical benefits, the same criterion will be applied later, too

Let us check what this collinearity of x_1 and x_2 means in practice. First, calculate the matrix X^TX that has an essential role in the regression formula

$$\begin{aligned} X^TX &= \begin{pmatrix} \sum_\kappa x_1^2(\kappa) & \sum_\kappa x_1(\kappa)x_2(\kappa) \\ \sum_\kappa x_1(\kappa)x_2(\kappa) & \sum_\kappa x_2^2(\kappa) \end{pmatrix} \\ &\approx k \cdot \begin{pmatrix} \mathrm{E}\{\xi^2\} + \sigma^2 & \mathrm{E}\{\xi^2\} \\ \mathrm{E}\{\xi^2\} & \mathrm{E}\{\xi^2\} + \sigma^2 \end{pmatrix}. \end{aligned} \tag{4.28}$$

To understand the properties of the regression formula, let us study the eigenvalues of the above matrix. It turns out that the solutions to the eigenvalue equation

$$\det\left\{\lambda \cdot I_2 - k \cdot \begin{pmatrix} \mathrm{E}\{\xi^2(\kappa)\} + \sigma^2 & \mathrm{E}\{\xi^2(\kappa)\} \\ \mathrm{E}\{\xi^2(\kappa)\} & \mathrm{E}\{\xi^2(\kappa)\} + \sigma^2 \end{pmatrix}\right\} = 0 \tag{4.29}$$

are

$$\begin{cases} \lambda_1 = 2k \cdot \mathrm{E}\{\xi^2(\kappa)\} + k\sigma^2, & \text{and} \\ \lambda_2 = k\sigma^2. \end{cases} \tag{4.30}$$

The theory of matrices reveals that the *condition number* of a matrix determines its numerical properties — that is, the ratio between its largest and smallest eigenvalue dictates how vulnerable the formulas containing it are to unmodeled noise. As the condition number grows towards infinity the matrix becomes gradually uninvertible. In this case, the matrix X^TX has the condition number

$$\mathrm{cond}\{X^TX\} = 1 + 2 \cdot \frac{\mathrm{E}\{\xi^2(\kappa)\}}{\sigma^2}, \tag{4.31}$$

telling us that the smaller the difference between the variables x_1 and x_2 is (σ^2 being small), the higher the sensitivity of the regression formula becomes.

The above result reveals that when using regression analysis, one has to be careful: It is the matrix X^TX that has to be inverted, and problems with invertibility are reflected in the model behavior. There only need to exist two linearly dependent measurements among the variables in x, and the problem instantly becomes ill-conditioned. In practice, it may be extremely difficult to avoid this kind of "almost" collinear variables — as an example, take a system that has to be modeled using partial differential equation (PDE) model (say, a rod that is being heated). PDE models are often called "infinite-dimensional"; that is, one needs very high number (in principle, infinitely many) measurements to uniquely determine the process state. It is not a surprise that temperature readings along the rod do not change rapidly, or nearby measurements deliver almost identical values, variables becoming linearly dependent; a regression model trying to utilize all the available information becomes badly behaving. When aiming towards accuracy, the model robustness is ruined!

To see an example of what collinear data looks like in a two-dimensional space, see Fig. 4.2: the data points in the figures are created using the above model, where $\mathrm{E}\{\xi^2(\kappa)\} = 1.0$ and $\sigma^2 = 0.01$, the sequences being normally distributed random processes. The data points seem to be located along a line; they do not really seem to "fill" the whole plane. Intuitively, this is the key to understanding the ideas of further analyses in later chapters.

The TLS approach by no means solves the above collinearity problem — on the contrary, even more severe problems emerge. Note that the last principal component essentially spans the null space of the covariance matrix, that is, if there is linear dependency among the variables, this dependency dominates in f'. Assuming that the linear dependency is between, say, input variables x_i and x_j, the parameters f'_i and f'_j have high values, all other coefficients being near zero. Now, if (11.21) is applied, the parameter f'_y (having negligible numerical value) in the denominator makes the model badly conditioned. The main problem with TLS is that while solving a minor problem (error in variables), it may introduce more pathological problems in the model.

4.3.2 Patch fixes

Because of the practical problems caused by collinearity, various ways to overcome the problems have been proposed. In what follows, two of such propositions are briefly presented — more sophisticated analyses are concentrated on in next chapters.

Orthogonal least squares

Because the basic source of problems in linear regression is related to inversion of the matrix $X^T X$, one can try to avoid the problem by enhancing the numerical properties of this matrix. Intuitively, it is clear that if the input variables were mutually orthogonal, so that $X^T X = I$, the numerical properties would be nice. Indeed, one can construct new variables Z so that this orthogonality holds using the so called *Gram-Schmidt procedure:* Corresponding to all indices $1 \leq i \leq n$, define Z_i by

$$Z'_i = X_i - \sum_{j=1}^{i-1} X_i^T Z_j \cdot Z_j, \tag{4.32}$$

and normalize it,

$$Z_i = Z'_i / \sqrt{Z_i'^T Z'_i}, \tag{4.33}$$

starting from $Z_1 = X_1 / \sqrt{X_1^T X_1}$. These data manipulation operations can be presented in a matrix form

$$Z = X \cdot M, \tag{4.34}$$

where M is an upper-triangular matrix[3]. It is easy to see that there holds

$$Z_i^T Z_j = \begin{cases} 1, & \text{if } i = j, \text{ and} \\ 0, & \text{otherwise,} \end{cases} \tag{4.35}$$

[3]Actually, the so called *QR factorization* of X that is readily available, for example, in `Matlab`, gives the same result (note that the resulting `R` matrix is the inverse of our M. The inversions of the triangular matrix are, however, nicely conditioned)

so that $Z^T Z = I$. Using these intermediate variables one has the mapping matrix from Z to Y as

$$F = \left(Z^T Z\right)^{-1} Z^T Y = Z^T Y, \tag{4.36}$$

or returning to the original variables X, the *Orthogonal Least Squares (OLS)* formula becomes

$$F_{\text{OLS}} = M Z^T Y. \tag{4.37}$$

Of course, reformatting formulas does not solve the fundamental problems — the inversion of the matrix is implicitly included in the construction of M. However, it turns out that reorganizing the calculations still often enhances the numerical properties of the problem.

Ridge regression

Ridge Regression (RR) is another (ad hoc) method of avoiding the collinearity problem — the basic idea is to explicitly prevent the covariance matrix from becoming singular. Ridge regression belongs to a large class of *regularization* methods where the numerical properties of the data — as seen by the algorithms — are somehow enhanced. The idea here is not to minimize exclusively the squared error, but to include weighting for *parameter size* in the optimization criterion: The badly-behaving nature of models is reflected in excessive parameter values. Instead of (4.4), the criterion that is really minimized is

$$\begin{aligned} &E_i^T E_i + F_i^T Q_i F_i = \\ &\qquad Y_i^T Y_i - Y_i^T X F_i - F_i^T X^T Y_i + F_i^T X^T X F_i + F_i^T Q_i F_i, \end{aligned} \tag{4.38}$$

where Q_i is a positive definite weighting matrix. Differentiation yields

$$\frac{d\left(E_i^T E_i\right)}{dF_i} = \mathbf{0} - X^T Y_i - X^T Y_i + 2X^T X F_i + 2Q_i F_i. \tag{4.39}$$

Setting the derivative to zero again gives the optimum:

$$-2X^T Y_i + 2X^T X F_i + 2Q_i F_i = \mathbf{0}, \tag{4.40}$$

resulting in

$$F_i = \left(X^T X + Q_i\right)^{-1} X^T Y_i. \tag{4.41}$$

In the multi-output case, assuming that $Q_i = Q$ is the same for all outputs, one can compactly write

$$F_{\text{RR}} = \left(X^T X + Q\right)^{-1} X^T Y. \tag{4.42}$$

Usually there is no a priori information about the parameter values and the weighting matrix Q cannot be uniquely determined. The normal procedure is

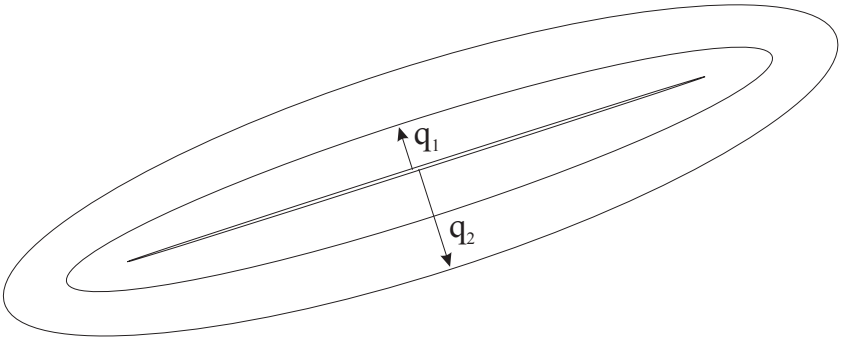

Figure 4.3: The "virtual" distribution of collinear data as seen by the ridge regression algorithm for different values of q

to let Q be diagonal; what is more, it is often chosen as $Q = q \cdot I$, where $q > 0$ is a small number. This approach efficiently prevents the matrix being inverted from becoming singular.

The key point here is that the matrix Q is added to the (unscaled) data covariance matrix. Study the eigenvalues; another way to determine the eigenvalues is to solve the determinant expression

$$\left|\left(X^T X + q \cdot I\right) - \lambda I\right| = \left|X^T X - (\lambda - q) \cdot I\right|. \tag{4.43}$$

When adding $q\,I$ to the matrix $X^T X$, its all eigenvalues are shifted up by the amount q, so that originally zero eigenvalues will have numerical value $q > 0$. The condition number also goes down. The model parameters are typically more conservative than in the nominal MLR case.

Note that the same ridge regression behavior in standard MLR is achieved also if white noise with covariance $\frac{1}{k}\,q\,I$ is added to data: If this added noise does not correlate with X — this assumption is easily fulfilled because the noise is artificial, being added in the algorithm — the noise-corrupted data covariance matrix is $\frac{1}{k}\left(X^T X + q\,I\right)$. This regularization approach is often explicitly used, for example, when training neural networks.

It seems that there are essentially two ways to enhance the invertibility of the matrix $X^T X$, and thus the MLR regression model properties:

1. Either, one can *ignore information* by leaving the "redundant" variables out. The problem here is that there are typically no variables with no information at all, even though this information can be highly redundant, and such variable elimination necessarily makes the model ignore available information.

2. Or, one can *introduce disinformation* by adding noise in the variables. This is effectively done when implementing regularization.

Just think of it: Either information is ignored, or noise is deliberately added to data just to make the model better behaving! There is an uneasy feeling of heuristics here, and something more sophisticated is clearly needed — the

modeling method should be matched with the data, not *vice versa.* Alternatives to MLR are presented in the following chapters.

Computer exercises

1. Check how the MLR sensitivity is affected when the data properties are changed; that is, try different values for the parameters k (number of samples), n (data dimension), dof_x (true degrees of freedom), and σ_x (deviation of the noise) below, and calculate the covariance matrix condition number:

```
k = 20;
n = 10;
dofx = 5;
sigmax = 0.001;
X = dataXY(k,n,NaN,dofx,NaN,sigmax);
Lambda = eig(X'*X/k);
max(Lambda)/min(Lambda)
```

2. Study how robust the different regression algorithms are. First generate data, and test the methods using cross-validation (try this several times for fresh data):

```
[X,Y] = dataXY(20,10,5,5,3,0.001,1.0);
E = regrCrossVal(X,Y,'regrMLR(X,Y)');
errorMLR = sum(sum(E.*E))/(20*5)
E = regrCrossVal(X,Y,'regrTLS(X,Y)');
errorTLS = sum(sum(E.*E))/(20*5)
E = regrCrossVal(X,Y,'regrOLS(X,Y)');
errorOLS = sum(sum(E.*E))/(20*5)
E = regrCrossVal(X,Y,'regrRR(X,Y,0.001)');   % Change this!
errorRR = sum(sum(E.*E))/(20*5)
```