Chapter 6

Linear Regression

Given a data set $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$the objective is to learn the relationship between features and the target. We usually start by hypothesizing the functional form of this relationship. For example,

$$f(\boldsymbol{x}) = w_0 + w_1 x_1 + w_2 x_2$$

where $\boldsymbol{w} = (w_0, w_1, w_2)$ is a set of parameters that need to be determined (learned) and $\boldsymbol{x} = (x_1, x_2)$. Alternatively, we may hypothesize that $f(\boldsymbol{x}) = \alpha + \beta x_1 x_2$, where $\boldsymbol{\theta} = (\alpha, \beta)$ is another set of parameters to be learned. In the former case, the target function is modeled as a *linear combination* of features and parameters; i.e.,

$$f(\boldsymbol{x}) = \sum_{j=0}^{d} w_j x_j,$$

where we extended $\boldsymbol{x}$ to $(x_0 = 1, x_1, x_2, \ldots, x_d)$. This same expression can be written in the algebraic form as $f(\mathbf{x}) = \mathbf{x}^\top \mathbf{w}$. Finding the best parameters $\boldsymbol{w}$ is then referred to as *linear regression problem*, whereas all other types of relationship between the features and the target fall into a category of *non-linear regression*. In either situation, the regression problem can be presented as a probabilistic modeling approach that reduces to parameter estimation; i.e., to an optimization problem with the goal of maximizing or minimizing some performance criterion between target values $\{y_i\}_{i=1}^n$ and predictions $\{f(\boldsymbol{x}_i)\}_{i=1}^n$. We can think of a particular optimization algorithm as the *learning* or *training algorithm*.

6.1 Maximum likelihood formulation

We shall now consider a statistical formulation of linear regression. We first lay out the assumptions behind this process and subsequently formulate the problem through maximization of the conditional likelihood function. In the following section, we will show how to solve the optimization problem and analyze the solution and its basic statistical properties.

Let us assume that the observed data set $\mathcal{D}$ is a product of a data generating process in which n data points were drawn independently and according to the same distribution $p(\boldsymbol{x})$. Let us also assume that the target variable Y has an underlying linear relationship with features $\boldsymbol{X} = (X_1, X_2, \ldots, X_d)$, modified by some error term ε that follows a zero-mean Gaussian distribution; i.e., $\varepsilon : \mathcal{N}(0, \sigma^2)$. That is, for a given input $\boldsymbol{x}$, the target y is a realization of a random variable Y defined as

$$Y = \sum_{j=0}^{d} \omega_j X_j + \varepsilon,$$

where $\boldsymbol{\omega} = (\omega_0, \omega_1, \ldots, \omega_d)$ is a set of unknown coefficients we will seek to estimate. Generally, the assumption of normality for the error term is reasonable (recall the central limit theorem!), although the independence between ε and $\boldsymbol{X}$ may not hold in practice. Using a few simple properties of expectations, we can see that Y also follows a Gaussian distribution; i.e., its conditional density is $p(y|\boldsymbol{x}, \boldsymbol{\omega}) = \mathcal{N}(\mu, \sigma^2)$, where μ is expressed in an algebraic notation as $\boldsymbol{\omega}^\top \mathbf{x}$.

In linear regression, we seek to approximate the target as $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$, where weights $\mathbf{w}$ are to be determined. We first write the conditional likelihood function for a single pair $(\boldsymbol{x}, y)$ as

$$p(y|\boldsymbol{x}, \boldsymbol{w}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\left(y - \sum_{j=0}^{d} w_j x_j\right)^2}{2\sigma^2} \right),$$

where we use the notation $\exp(a) = e^a$, to make the exponent easier to read. Observe that the only change from the conditional density function of Y is that coefficients $\boldsymbol{w}$ are used instead of $\boldsymbol{\omega}$. Incorporating the entire data set $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, we can now write the conditional likelihood function as $p(\mathbf{y}|\mathbf{X}, \mathbf{w})$, where $\mathbf{X}$ is the data matrix, and find weights as

$$\mathbf{w}_{\text{ML}} = \arg\max_{\mathbf{w}} \{p(\mathbf{y}|\mathbf{X}, \mathbf{w})\}.$$

Since the n examples are independent and identically distributed (i.i.d.), we have

$$\begin{aligned} p(\mathbf{y}|\mathbf{X}, \mathbf{w}) &= \prod_{i=1}^{n} p(y_i|\mathbf{x}_i, \mathbf{w}) \\ &= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\left(y_i - \sum_{j=0}^{d} w_j x_{ij}\right)^2}{2\sigma^2} \right). \end{aligned}$$

For the reasons of mathematical convenience, we will look at the logarithm (monotonic function) of the likelihood function and express the log-likelihood as

$$\ln(p(\mathbf{y}|\mathbf{X}, \mathbf{w})) = -\sum_{i=1}^{n} \log\left(\sqrt{2\pi\sigma^2}\right) - \frac{1}{2\sigma^2} \sum_{i=1}^{n} \left(y_i - \sum_{j=0}^{d} w_j x_{ij} \right)^2.$$

Given that the first term on the right-hand hand side is independent of $\boldsymbol{w}$, maximizing the likelihood function corresponds exactly to minimizing the sum of squared errors

$$\begin{aligned} \text{Err}(\mathbf{w}) &= \sum_{i=1}^{n} (f(\mathbf{x}_i) - y_i)^2 \qquad \triangleright\, f(\mathbf{x}_i) = \sum_{j=0}^{d} w_j x_{ij} \\ &= \sum_{i=1}^{n} e_i^2. \end{aligned}$$

Geometrically, this error is the square of the Euclidean distance between the vector of predictions $\hat{\boldsymbol{y}} = (f(\boldsymbol{x}_1), f(\boldsymbol{x}_2), \ldots, f(\boldsymbol{x}_n))$ and the vector of observed target values $\mathbf{y} = (y_1, y_2, \ldots, y_n)$. A simple example illustrating the linear regression problem is shown in Figure 6.1.

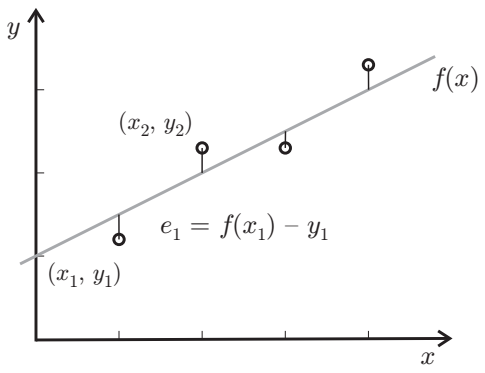

Figure 6.1: An example of a linear regression fitting on data set $\mathcal{D} = \{(1, 1.2), (2, 2.3), (3, 2.3), (4, 3.3)\}$. *The task of the optimization process is to find the best linear function* $f(x) = w_0 + w_1 x$ *so that the sum of squared errors* $e_1^2 + e_2^2 + e_3^2 + e_4^2$ *is minimized.*

To more explicitly see why the maximum likelihood solution corresponds to minimizing $\text{Err}(\mathbf{w})$, notice that maximizing the likelihood is equivalent to maximizing the log-likelihood (because log is monotonic) which is equivalent to minimizing the negative log-likelihood. Therefore, the maximum likelihood $\mathbf{w}_{\text{ML}}$ corresponds to

$$\begin{aligned}
\mathbf{w}_{\text{ML}} &= \underset{\mathbf{w}\in\mathbb{R}^d}{\text{argmin}} - \ln(p(\mathbf{y}|\mathbf{X}, \mathbf{w})) \\
&= \underset{\mathbf{w}\in\mathbb{R}^d}{\text{argmin}} \sum_{i=1}^{n} \log\left(\sqrt{2\pi\sigma^2}\right) + \frac{1}{2\sigma^2} \sum_{i=1}^{n} \left(y_i - \sum_{j=0}^{d} w_j x_{ij}\right)^2 \\
&= \underset{\mathbf{w}\in\mathbb{R}^d}{\text{argmin}} \sum_{i=1}^{n} \left(y_i - \sum_{j=0}^{d} w_j x_{ij}\right)^2 \\
&= \underset{\mathbf{w}\in\mathbb{R}^d}{\text{argmin}}\, \text{Err}(\mathbf{w})
\end{aligned}$$

In the next sections, we will discuss how to solve this optimization and the properties of the solution.

Note that we could have simply started with some expert-defined error function and solved the optimization problem. However, the statistical framework provides insights into the assumptions behind OLS regression. In particular, the assumptions include that the data $\mathcal{D}$ was drawn i.i.d.; there is an underlying linear relationship between features and the target; that the noise (error term) is zero-mean Gaussian and independent of the features; and that there is an absence of noise in the collection of features.

6.2 Ordinary Least-Squares (OLS) Regression

To minimize the sum of squared errors, we shall first re-write $\text{Err}(\mathbf{w})$ as

$$\begin{aligned}\text{Err}(\mathbf{w}) &= \sum_{i=1}^{n} \left(f(\mathbf{x}_i) - y_i\right)^2 \\ &= \sum_{i=1}^{n} \left(\sum_{j=0}^{d} w_j x_{ij} - y_i\right)^2,\end{aligned}$$

where, again, we expanded each data point $\mathbf{x}_i$ by $x_{i0} = 1$ to simplify the expression.

We now calculate the gradient $\nabla\text{Err}(\mathbf{w})$. Finding weights for which $\nabla\text{Err}(\mathbf{w}) = \mathbf{0}$ will result in a stationary point. To ensure that this stationary point is a global minimum, we need a bit more information. We can look at the second derivative; this requires understanding of Hessian, so we include this later in the notes in Example 8. But, fortunately, it is even simpler here, since we know that this objective is convex in $\mathbf{w}$; therefore, any stationary point will be a global minimum.

Now, we set the partial derivatives to 0 and solve the equations for each weight w_j

$$\begin{aligned}\frac{\partial \text{Err}}{\partial w_0} &= 2\sum_{i=1}^{n} \left(\sum_{j=0}^{d} w_j x_{ij} - y_i\right) x_{i0} = 0 \\ \frac{\partial \text{Err}}{\partial w_1} &= 2\sum_{i=1}^{n} \left(\sum_{j=0}^{d} w_j x_{ij} - y_i\right) x_{i1} = 0 \\ &\vdots \\ \frac{\partial \text{Err}}{\partial w_d} &= 2\sum_{i=1}^{n} \left(\sum_{j=0}^{d} w_j x_{ij} - y_i\right) x_{ik} = 0\end{aligned}$$

This results in a system of $d+1$ linear equations with $d+1$ unknowns that can be routinely solved (e.g., by using Gaussian elimination).

While this formulation is useful, it does not allow us to obtain a closed-form solution for $\mathbf{w}$ or discuss the existence and multiplicity of solutions. To address the first point we will exercise some matrix calculus, while the remaining points will be discussed later. We will first write the sum of square errors using the matrix notation as

$$\begin{aligned}\text{Err}(\mathbf{w}) &= (\mathbf{X}\mathbf{w} - \mathbf{y})^\top (\mathbf{X}\mathbf{w} - \mathbf{y}) \\ &= \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2,\end{aligned}$$

where $\|\mathbf{v}\|_2 = \sqrt{\mathbf{v}^\top \mathbf{v}} = \sqrt{v_1^2 + v_2^2 + \ldots v_n^2}$ is the length of vector $\mathbf{v}$; it is also called the ℓ_2 norm. We can now formalize the *ordinary least-squares (OLS) linear regression problem* as

$$\mathbf{w}_{\text{ML}} = \arg\min_{\mathbf{w}} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2.$$

We proceed by finding $\nabla \text{Err}(\mathbf{w})$. The gradient function $\nabla \text{Err}(\mathbf{w})$ is a derivative of a scalar with respect to a vector. However, the intermediate steps of calculating the gradient require derivatives of vectors with respect to vectors (some of the rules of such derivatives are shown in Table A.1). Application of the rules from Table A.1 results in

$$\nabla \text{Err}(\mathbf{w}) = 2\mathbf{X}^\top \mathbf{X}\mathbf{w} - 2\mathbf{X}^\top \mathbf{y}$$

and, therefore, from $\nabla \text{Err}(\mathbf{w}) = \mathbf{0}$ we find that

$$\mathbf{w}_{\text{ML}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}. \tag{6.1}$$

We can now express the predicted target values as

$$\begin{aligned} \hat{\mathbf{y}} &= \mathbf{X}\mathbf{w}_{\text{ML}} \\ &= \mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}. \end{aligned}$$

The matrix $\mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top$ is called the *projection matrix*; we will see later that it projects $\mathbf{y}$ to the column space of $\mathbf{X}$.

Example 14: Consider again data set $\mathcal{D} = \{(1, 1.2), (2, 2.3), (3, 2.3), (4, 3.3)\}$ from Figure 6.1. We want to find the optimal coefficients of the least-squares fit for $f(x) = w_0 + w_1 x$ and then calculate the sum of squared errors on $\mathcal{D}$ after the fit.

The OLS fitting can now be performed using

$$\mathbf{x} = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \end{bmatrix}, \ \mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}, \ \mathbf{y} = \begin{bmatrix} 1.2 \\ 2.3 \\ 2.3 \\ 3.3 \end{bmatrix},$$

where a column of ones was added to $\mathbf{x}$ to allow for a non-zero intercept ($y = w_0$ when $x = 0$). Substituting $\mathbf{x}$ and $\mathbf{y}$ into Eq. (6.1) results in $\mathbf{w} = (0.7, 0.63)$ and the sum of square errors is $\text{Err}(\mathbf{w}) = 0.223$. □

As seen in the example above, it is a standard practice to add a column of ones to the data matrix $\mathbf{x}$ in order to ensure that the fitted line, or generally a hyperplane, does not have to pass through the origin of the coordinate system. This effect, however, can be achieved in other ways. Consider the first component of the gradient vector

$$\frac{\partial \text{Err}}{\partial w_0} = 2 \sum_{i=1}^{n} \left(\sum_{j=0}^{d} w_j x_{ij} - y_i \right) x_{i0} = 0$$

where, because $x_{i0} = 1$ by definition, we obtain that

$$0 = \sum_{i=1}^{n} \left(\sum_{j=0}^{d} w_j x_{ij} - y_i \right) = \sum_{i=1}^{n} \left(w_0 + \sum_{j=1}^{d} w_j x_{ij} - y_i \right)$$

giving

$$\sum_{i=1}^{n} w_0 = \sum_{i=1}^{n} y_i - \sum_{j=1}^{d} w_j \sum_{i=1}^{n} x_{ij}.$$

When all features (columns of $\mathbf{X}$) are normalized to have zero mean, i.e. when $\sum_{i=1}^{n} x_{ij} = 0$ for any column j, it follows that

$$w_0 = \frac{1}{n} \sum_{i=1}^{n} y_i.$$

We see now that if the target variable is normalized to the zero mean as well, it follows that $w_0 = 0$ and that the column of ones is not needed.

6.2.1 Weighted error function

In some applications it is useful to consider minimizing the weighted error function

$$\mathrm{Err}(\mathbf{w}) = \sum_{i=1}^{n} c_i \left(\sum_{j=0}^{d} w_j x_{ij} - y_i \right)^2,$$

where $c_i > 0$ is a cost for data point i. Expressing this in a matrix form, the goal is to minimize $(\mathbf{X}\mathbf{w} - \mathbf{y})^\top \mathbf{C} (\mathbf{X}\mathbf{w} - \mathbf{y})$, where $\mathbf{C} = \mathrm{diag}\,(c_1, c_2, \ldots, c_n)$. Using a similar approach as above, it can be shown that the weighted least-squares solution $\mathbf{w}_{\mathrm{C}}$ can be expressed as

$$\mathbf{w}_{\mathrm{C}} = \left(\mathbf{X}^\top \mathbf{C} \mathbf{X} \right)^{-1} \mathbf{X}^\top \mathbf{C} \mathbf{y}.$$

In addition, it can be derived that

$$\mathbf{w}_{\mathrm{C}} = \mathbf{w}_{\mathrm{ML}} + \left(\mathbf{X}^\top \mathbf{C} \mathbf{X} \right)^{-1} \mathbf{X}^\top (\mathbf{I} - \mathbf{C}) (\mathbf{X}\mathbf{w}_{\mathrm{ML}} - \mathbf{y}),$$

where $\mathbf{w}_{\mathrm{ML}}$ is provided by Eq. (6.1). We can see that the solutions are identical when $\mathbf{C} = \mathbf{I}$, but also when $\mathbf{X}\mathbf{w}_{\mathrm{ML}} = \mathbf{y}$.

6.2.2 Predicting multiple outputs simultaneously

The extension to multiple outputs is straightforward, where now the target is an m-dimensional vector, $\mathbf{y} \in \mathbb{R}^m$, rather than a scalar, giving target matrix $\mathbf{Y} \in \mathbb{R}^{n \times m}$. Correspondingly, the weights $\mathbf{W} \in \mathbb{R}^{d \times m}$ to give $\mathbf{W}^\top \mathbf{x} \in \mathbb{R}^m$, with error

$$\begin{aligned} \mathrm{Err}(\mathbf{W}) &= \|\mathbf{X}\mathbf{W} - \mathbf{Y}\|_F^2 = \sum_{i=1}^{n} \|\mathbf{X}_{i,:}\mathbf{W} - \mathbf{Y}_{i,:}\|_2^2 && \triangleright \text{ Frobenius norm} \\ &= \mathrm{trace}\left((\mathbf{X}\mathbf{W} - \mathbf{Y})^\top (\mathbf{X}\mathbf{W} - \mathbf{Y}) \right) \end{aligned}$$

and solution

$$\mathbf{W}_{\mathrm{ML}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{Y}.$$

Exercise: Derive this solution, by taking partial derivatives or, preferably, by using gradient rules for matrix variables. A good resource for matrix gradients is the matrix cookbook [15].

6.3 An Algebraic Perspective

Another powerful tool for analyzing and understanding linear regression comes from linear and applied linear algebra. In this section we take a detour to address fundamentals of linear algebra and then apply these concepts to deepen our understanding of regression. In linear algebra, we are frequently interested in solving the following set of equations, given below in a matrix form

$$\mathbf{A}\mathbf{x} = \mathbf{b}. \tag{6.2}$$

Here, $\mathbf{A}$ is an $m \times n$ matrix, $\mathbf{b}$ is an $m \times 1$ vector, and $\mathbf{x}$ is an $n \times 1$ vector that is to be found. All elements of $\mathbf{A}$, $\mathbf{x}$, and $\mathbf{b}$ are considered to be real numbers. We shall start with a simple scenario and assume $\mathbf{A}$ is a square, 2×2 matrix. This set of equations can be expressed as

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 &= b_1 \\ a_{21}x_1 + a_{22}x_2 &= b_2 \end{aligned}$$

For example, we may be interested in solving

$$\begin{aligned} x_1 + 2x_2 &= 3 \\ x_1 + 3x_2 &= 5 \end{aligned}$$

This is a convenient formulation when we want to solve the system, e.g. by Gaussian elimination. However, it is not a suitable formulation to understand the question of the existence of solutions. In order for us to do this, we briefly review the basic concepts in linear algebra.

6.3.1 The four fundamental subspaces

The objective of this section it to briefly review the *four fundamental subspaces* in linear algebra (column space, row space, nullspace, left nullspace) and their mutual relationship. We shall start with our example from above and write the system of linear equations as

$$\begin{bmatrix} 1 \\ 1 \end{bmatrix} x_1 + \begin{bmatrix} 2 \\ 3 \end{bmatrix} x_2 = \begin{bmatrix} 3 \\ 5 \end{bmatrix}.$$

We can see now that by solving $\mathbf{A}\mathbf{x} = \mathbf{b}$ we are looking for the right amounts of vectors $(1, 1)$ and $(2, 3)$ so that their linear combination produces $(3, 5)$; these amounts are $x_1 = -1$ and $x_2 = 2$. Let us define $\mathbf{a}_1 = (1, 1)$ and $\mathbf{a}_2 = (2, 3)$ to be the column vectors of $\mathbf{A}$; i.e. $\mathbf{A} = [\mathbf{a}_1\ \mathbf{a}_2]$. Thus, $\mathbf{A}\mathbf{x} = \mathbf{b}$ will be solvable whenever $\mathbf{b}$ can be expressed as a linear combination of the column vectors $\mathbf{a}_1$ and $\mathbf{a}_2$.

All linear combinations of the columns of matrix $\mathbf{A}$ constitute the *column space* of $\mathbf{A}$, $C(\mathbf{A})$, with vectors $\mathbf{a}_1 \ldots \mathbf{a}_n$ being a basis of this space. Both $\mathbf{b}$ and $C(\mathbf{A})$ lie in the m-dimensional space $\mathbb{R}^m$. Therefore, what $\mathbf{A}\mathbf{x} = \mathbf{b}$ is saying is that $\mathbf{b}$ must lie in the column space of $\mathbf{A}$ for the equation to have solutions. In the example above, if columns of $\mathbf{A}$ are linearly independent[1], the solution is unique; i.e., there exists only one linear combination

[1] As a reminder, two vectors are independent if their linear combination is zero only when both x_1 and x_2 are zero.

of the column vectors that will give $\mathbf{b}$. Otherwise, because $\mathbf{A}$ is a square matrix, the system has no solutions. An example of such a situation is

$$\begin{bmatrix} 1 & 2 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 3 \\ 5 \end{bmatrix},$$

where $\mathbf{a}_1 = (1, 1)$ and $\mathbf{a}_2 = (2, 2)$. Here, $\mathbf{a}_1$ and $\mathbf{a}_2$ are (linearly) dependent because $2\mathbf{a}_1 - \mathbf{a}_2 = 0$. There is a deep connection between the spaces generated by a set of vectors and the properties of the matrix $\mathbf{A}$. For now, using the example above, it suffices to say that if $\mathbf{a}_1$ and $\mathbf{a}_2$ are independent the matrix $\mathbf{A}$ is non-singular (singularity can be discussed only for square matrices), that is of full rank.

In an equivalent manner to the column space, all linear combinations of the rows of $\mathbf{A}$ constitute the *row space*, denoted by $C(\mathbf{A}^\top)$, where both $\mathbf{x}$ and $C(\mathbf{A}^\top)$ are in $\mathbb{R}^n$. All solutions to $\mathbf{Ax} = \mathbf{0}$ constitute the *nullspace* of the matrix, $N(\mathbf{A})$, while all solutions of $\mathbf{A}^\top\mathbf{y} = \mathbf{0}$ constitute the so-called *left nullspace* of $\mathbf{A}$, $N(\mathbf{A}^\top)$. Clearly, $C(\mathbf{A})$ and $N(\mathbf{A}^\top)$ are embedded in $\mathbb{R}^m$, whereas $C(\mathbf{A}^\top)$ and $N(\mathbf{A})$ are in $\mathbb{R}^n$. However, the pairs of subspaces are orthogonal (vectors $\mathbf{u}$ and $\mathbf{v}$ are orthogonal if $\mathbf{u}^\top\mathbf{v} = 0$); that is, any vector in $C(\mathbf{A})$ is orthogonal to all vectors from $N(\mathbf{A}^\top)$ and any vector in $C(\mathbf{A}^\top)$ is orthogonal to all vectors from $N(\mathbf{A})$. This is easy to see: if $\mathbf{x} \in N(\mathbf{A})$, then by definition $\mathbf{Ax} = \mathbf{0}$, and thus each row of $\mathbf{A}$ is orthogonal to $\mathbf{x}$. If each row is orthogonal to $\mathbf{x}$, then so are all linear combinations of rows.

Orthogonality is a key property of the four subspaces, as it provides useful decomposition of vectors $\mathbf{x}$ and $\mathbf{b}$ from Eq. (6.2) with respect to $\mathbf{A}$ (we will exploit this in the next Section). For example, any $\mathbf{x} \in \mathbb{R}^n$ can be decomposed as

$$\mathbf{x} = \mathbf{x}_r + \mathbf{x}_n,$$

where $\mathbf{x}_r \in C(\mathbf{A}^\top)$ and $\mathbf{x}_n \in N(\mathbf{A})$, such that $\|\mathbf{x}\|_2^2 = \|\mathbf{x}_r\|_2^2 + \|\mathbf{x}_n\|_2^2$. Similarly, every $\mathbf{b} \in \mathbb{R}^m$ can be decomposed as

$$\mathbf{b} = \mathbf{b}_c + \mathbf{b}_l,$$

where $\mathbf{b}_c \in C(\mathbf{A})$, $\mathbf{b}_l \in N(\mathbf{A}^\top)$, and $\|\mathbf{b}\|_2^2 = \|\mathbf{b}_c\|_2^2 + \|\mathbf{b}_l\|_2^2$.

We mentioned above that the properties of fundamental spaces are tightly connected with the properties of matrix $\mathbf{A}$. So, to conclude this section, let us briefly discuss the *rank* of a matrix and its relationship with the dimensions of the fundamental subspaces. The *basis* of the space is the smallest set of vectors that span the space (this set of vectors is not unique). The size of the basis is also called the dimension of the space. In the example at the beginning of this subsection, we had a two dimensional column space with basis vectors $\mathbf{a}_1 = (1, 1)$ and $\mathbf{a}_2 = (2, 3)$. On the other hand, for $\mathbf{a}_1 = (1, 1)$ and $\mathbf{a}_2 = (2, 2)$ we had a one dimensional column space; i.e., a line, fully determined by any of the basis vectors. Unsurprisingly, the dimension of the space spanned by column vectors equals the *rank* of matrix $\mathbf{A}$. One of the fundamental results in linear algebra is that the rank of $\mathbf{A}$ is identical to the dimension of $C(\mathbf{A})$, which in turn is identical to the dimension of $C(\mathbf{A}^\top)$.

6.3.2 Minimizing $\|\mathbf{Ax} - \mathbf{b}\|_2^2$

Let us now look again at the solutions to $\mathbf{Ax} = \mathbf{b}$. In general, there are three different outcomes:

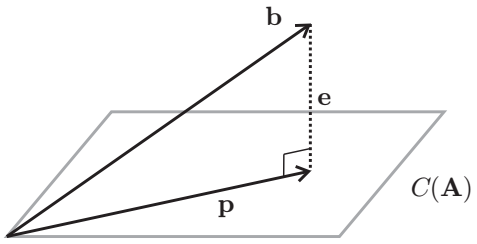

Figure 6.2: Illustration of the projection of vector $\mathbf{b}$ *to the column space of matrix* $\mathbf{A}$*. Vectors* $\mathbf{p}$ *(*$\mathbf{b}_c$*) and* $\mathbf{e}$ *(*$\mathbf{b}_l$*) represent the projection point and the error, respectively.*

1. there are no solutions to the system
2. there is a unique solution to the system, and
3. there are infinitely many solutions.

These outcomes depend on the relationship between the rank (r) of $\mathbf{A}$ and dimensions m and n. We already know that when $r = m = n$ (square, invertible, full rank matrix $\mathbf{A}$) there is a unique solution to the system, but let us investigate other situations. Generally, when $r = n < m$ (full column rank), the system has either one solution or no solutions, as we will see momentarily. When $r = m < n$ (full row rank), the system has infinitely many solutions. Finally, in cases when $r < m$ and $r < n$, there are either no solutions or there are infinitely many solutions. Because $\mathbf{Ax} = \mathbf{b}$ may not be solvable, we generalize solving $\mathbf{Ax} = \mathbf{b}$ to minimizing $\|\mathbf{Ax} - \mathbf{b}\|_2$. In such a way, all situations can be considered in a unified framework.

Let us consider the following example

$$\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 1 & 3 \\ 1 & 4 \end{bmatrix}, \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix},$$

which illustrates an instance where we are unlikely to have a solution to $\mathbf{Ax} = \mathbf{b}$, unless there is some constraint on b_1, b_2, and b_3; here, the constraint is $b_3 = 2b_2 - b_1$. In this situation, $C(\mathbf{A})$ is a 2D plane in $\mathbb{R}^3$ spanned by the column vectors $\mathbf{a}_1 = (1, 1, 1)$ and $\mathbf{a}_2 = (2, 3, 4)$. If the constraint on the elements of $\mathbf{b}$ is not satisfied, our goal is to try to find a point in $C(\mathbf{A})$ that is closest to $\mathbf{b}$**.** This happens to be the point where $\mathbf{b}$ is projected to $C(\mathbf{A})$, as shown in Figure 6.2. We will refer to the projection of $\mathbf{b}$ to $C(\mathbf{A})$ as $\mathbf{p}$. Now, using the standard algebraic notation, we have the following equations

$$\mathbf{b} = \mathbf{p} + \mathbf{e}$$
$$\mathbf{p} = \mathbf{Ax}$$

Since $\mathbf{p}$ and $\mathbf{e}$ are orthogonal, we know that $\mathbf{p}^\top \mathbf{e} = 0$. Let us now solve for $\mathbf{x}$

$$(\mathbf{Ax})^\top (\mathbf{b} - \mathbf{Ax}) = 0$$
$$\mathbf{x}^\top \mathbf{A}^\top \mathbf{b} - \mathbf{x}^\top \mathbf{A}^\top \mathbf{Ax} = 0$$
$$\mathbf{x}^\top \left(\mathbf{A}^\top \mathbf{b} - \mathbf{A}^\top \mathbf{Ax} \right) = 0$$

and thus

$$\mathbf{x}^* = \left(\mathbf{A}^\top \mathbf{A}\right)^{-1} \mathbf{A}^\top \mathbf{b}.$$

This is exactly the same solution as one that minimized the sum of squared errors and maximized the likelihood. The matrix

$$\mathbf{A}^\dagger = \left(\mathbf{A}^\top \mathbf{A}\right)^{-1} \mathbf{A}^\top$$

is called the Moore-Penrose pseudo-inverse or simply a pseudo-inverse. This is an important matrix because it always exists and is unique, even in situations when the inverse of $\mathbf{A}^\top \mathbf{A}$ does not exist. This happens when $\mathbf{A}$ has dependent columns (technically, $\mathbf{A}$ and $\mathbf{A}^\top \mathbf{A}$ will have the same nullspace that contains more than just the origin of the coordinate system; thus the rank of $\mathbf{A}^\top \mathbf{A}$ is less than n). Let us for a moment look at the projection vector $\mathbf{p}$. We have

$$\begin{aligned} \mathbf{p} &= \mathbf{A}\mathbf{x} \\ &= \mathbf{A}\left(\mathbf{A}^\top \mathbf{A}\right)^{-1} \mathbf{A}^\top \mathbf{b}, \end{aligned}$$

where $\mathbf{A}\left(\mathbf{A}^\top \mathbf{A}\right)^{-1} \mathbf{A}^\top$ is the matrix that projects $\mathbf{b}$ to the column space of $\mathbf{A}$.

While we arrived at the same result as in previous sections, the tools of linear algebra allow us to discuss OLS regression at a deeper level. Let us examine for a moment the existence and multiplicity of solutions to

$$\arg\min_{\mathbf{x}} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2 . \tag{6.3}$$

Clearly, the solution to this problem always exists. However, we shall now see that the solution to this problem is generally not unique and that it depends on the rank of $\mathbf{A}$. Consider $\mathbf{x}$ to be one solution to Eq. (6.3). Recall that $\mathbf{x} = \mathbf{x}_r + \mathbf{x}_n$ and that it is multiplied by $\mathbf{A}$; thus, any vector $\mathbf{x} = \mathbf{x}_r + \alpha \mathbf{x}_n$, where $\alpha \in \mathbb{R}$, is also a solution. Observe that $\mathbf{x}_r$ is common to all such solutions; if you cannot see it, assume there exists another vector from the row space and show that it is not possible. If the columns of $\mathbf{A}$ are independent, the solution is unique because the nullspace contains only the origin. Otherwise, there are infinitely many solutions. In such cases, what exactly is the solution found by projecting $\mathbf{b}$ to $C(\mathbf{A})$? Let us look at it:

$$\begin{aligned} \mathbf{x}^* &= \mathbf{A}^\dagger \mathbf{b} \\ &= \left(\mathbf{A}^\top \mathbf{A}\right)^{-1} \mathbf{A}^\top (\mathbf{p} + \mathbf{e}) \\ &= \left(\mathbf{A}^\top \mathbf{A}\right)^{-1} \mathbf{A}^\top \mathbf{p} \\ &= \mathbf{x}_r, \end{aligned}$$

as $\mathbf{p} = \mathbf{A}\mathbf{x}_r$. Given that $\mathbf{x}_r$ is unique, the solution found by the least squares optimization is the one that simultaneously minimizes $\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2$ and $\|\mathbf{x}\|_2$ (observe that $\|\mathbf{x}\|_2$ is minimized because the solution ignores any component from the nullspace). Thus, the OLS regression problem is sometimes referred to as the minimum-norm least-squares problem.

Let us now consider situations where $\mathbf{Ax} = \mathbf{b}$ has infinitely many solutions; i.e., when $\mathbf{b} \in C(\mathbf{A})$. This usually arises when $r \leq m < n$. Here, because $\mathbf{b}$ is already in the column space of $\mathbf{A}$, the only question is what particular solution $\mathbf{x}$ will be found by the minimization procedure. As we have seen above, the outcome of the minimization process is the solution with the minimum L_2 norm $\|\mathbf{x}\|_2$.

6.4 Linear regression for non-linear problems

At first, it might seem that the applicability of linear regression to real-life problems is greatly limited. After all, it is not clear whether it is realistic (most of the time) to assume that the target variable is a linear combination of features. Fortunately, the applicability of linear regression is broader because we can use it to obtain non-linear functions. The main idea is to apply a non-linear transformation to the data matrix $\mathbf{X}$ prior to the fitting step, which then enables a non-linear fit. Obtaining such a useful feature representation is a central problem in machine learning; we will discuss this in detail in Chapter 9. Here, we will first examine a simpler expanded representation that enables non-linear learning: polynomial curve fitting.

6.4.1 Polynomial curve fitting

We start with one-dimensional data. In OLS regression, we would look for the fit in the following form

$$f(x) = w_0 + w_1 x,$$

where x is the data point and $\mathbf{w} = (w_0, w_1)$ is the weight vector. To achieve a polynomial fit of degree p, we will modify the previous expression into

$$f(x) = \sum_{j=0}^{p} w_j x^j,$$

where p is the degree of the polynomial. We will rewrite this expression using a set of basis functions as

$$\begin{aligned} f(x) &= \sum_{j=0}^{p} w_j \phi_j(x) \\ &= \mathbf{w}^\top \boldsymbol{\phi}, \end{aligned}$$

where $\phi_j(x) = x^j$ and $\boldsymbol{\phi} = (\phi_0(x), \phi_1(x), \ldots, \phi_p(x))$. Applying this transformation to every data point in $\mathbf{x}$ results in a new data matrix $\boldsymbol{\Phi}$, as shown in Figure 6.3.

Following the discussion from Section 6.2, the optimal set of weights is calculated as

$$\mathbf{w}_{\text{ML}} = \left(\boldsymbol{\Phi}^\top \boldsymbol{\Phi}\right)^{-1} \boldsymbol{\Phi}^\top \mathbf{y}.$$

Example 15: In Figure 6.1 we presented an example of a data set with four data points. What we did not mention was that, given a set $\{x_1, x_2, x_3, x_4\}$, the targets were generated

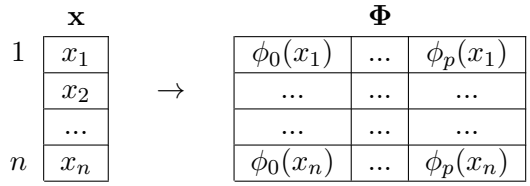

Figure 6.3: Transformation of an $n \times 1$ data matrix $\mathbf{x}$ into an $n \times (p+1)$ matrix $\mathbf{\Phi}$ using a set of basis functions ϕ_j, $j = 0, 1, \ldots, p$.

by using function $1 + \frac{x}{2}$ and then adding a measurement error $\mathbf{e} = (-0.3, 0.3, -0.2, 0.3)$. It turned out that the optimal coefficients $\mathbf{w}_{\text{ML}} = (0.7, 0.63)$ were close to the true coefficients $\boldsymbol{\omega} = (1, 0.5)$, even though the error terms were relatively significant. We will now attempt to estimate the coefficients of a polynomial fit with degrees $p = 2$ and $p = 3$. We will also calculate the sum of squared errors on $\mathcal{D}$ after the fit as well as on a large discrete set of values $x \in \{0, 0.1, 0.2, \ldots, 10\}$ where the target values will be generated using the true function $1 + \frac{x}{2}$.

Using a polynomial fit with degrees $p = 2$ and $p = 3$ results in $\mathbf{w}_2 = (0.575, 0.755, -0.025)$ and $\mathbf{w}_3 = (-3.1, 6.6, -2.65, 0.35)$, respectively. The sum of squared errors on $\mathcal{D}$ equals $\text{Err}(\mathbf{w}_2) = 0.221$ and $\text{Err}(\mathbf{w}_3) \approx 0$. Thus, the best fit is achieved with the cubic polynomial. However, the sum of squared errors on the outside data set reveal a poor generalization ability of the cubic model because we obtain $\text{Err}(\mathbf{w}) = 26.9$, $\text{Err}(\mathbf{w}_2) = 3.9$, and $\text{Err}(\mathbf{w}_3) = 22018.5$. This effect is called *overfitting*. Broadly speaking, overfitting is indicated by a significant difference in fit between the data set on which the model was trained and the outside data set on which the model is expected to be applied (Figure 6.4). In this case, the overfitting occurred because the complexity of the model was increased considerably, whereas the size of the data set remained small.

One signature of overfitting is an increase in the magnitude of the coefficients. For example, while the absolute values of all coefficients in $\mathbf{w}$ and $\mathbf{w}_2$ were less than one, the values of the coefficients in $\mathbf{w}_3$ became significantly larger with alternating signs (suggesting overcompensation). We will discuss *regularization* in Section 6.5.2 as an approach to prevent this effect. □

Polynomial curve fitting is only one way of non-linear fitting because the choice of basis functions need not be limited to powers of x. Among others, non-linear basis functions that are commonly used are the sigmoid function

$$\phi_j(x) = \frac{1}{1 + e^{-\frac{x - \mu_j}{s_j}}}$$

or a Gaussian-style exponential function

$$\phi_j(x) = e^{-\frac{(x-\mu_j)^2}{2\sigma_j^2}},$$

where μ_j, s_j, and σ_j are constants to be determined. However, this approach works only for a one-dimensional input $\mathbf{x}$. For higher dimensions, this approach can be generalized using *radial basis functions*; see Section 9.1 for more details.

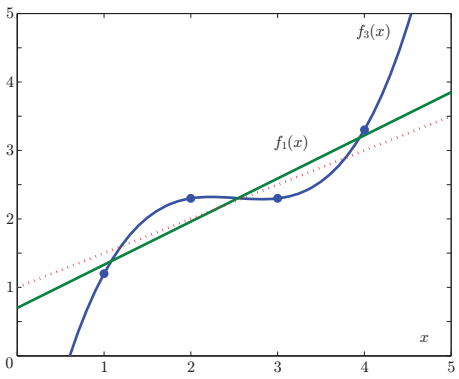

Figure 6.4: Example of a linear vs. polynomial fit on a data set shown in Figure 6.1. The linear fit, $f_1(x)$, is shown as a solid green line, whereas the cubic polynomial fit, $f_3(x)$, is shown as a solid blue line. The dotted red line indicates the target linear concept.

6.5 Stability and the bias-variance trade-off

The OLS solution can be unstable. In this section, we show why this is the case, and discuss how regularization can be used to mitigate this problem. We will then discuss a foundational concept in machine learning: the bias-variance trade-off.

6.5.1 Sensitivity of the OLS solution

The OLS solution is unstable if $\mathbf{X}^\top\mathbf{X}$ is not invertible. This can occur for two main reasons: linearly dependent features and small datasets. Data sets often include large numbers of features, which are sometimes identical, similar, or nearly linearly dependent. If the dataset is small, it is feasible that some features are the same across samples, again resulting in low-rank $\mathbf{X}$. When $\mathbf{X}^\top\mathbf{X}$ is not invertible—or ill-conditioned—the OLS solution is highly sensitive to small perturbations in $\mathbf{y}$ and $\mathbf{X}$.

To see why, we will look at the *singular value decomposition* of $\mathbf{X}$. As with the previous linear algebra constructs, it allows us to easily examine properties of $\mathbf{X}$. Let's consider the common case, where $n > d$: the number of samples is greater than the input dimension. The singular value decomposition of $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$ for orthonormal matrices[2] $\mathbf{U} \in \mathbb{R}^{n\times n}, \mathbf{V} \in \mathbb{R}^{d\times d}$ and non-negative (rectangular) diagonal matrix $\mathbf{\Sigma} \in \mathbb{R}^{n\times d}$. The diagonal entries in $\mathbf{\Sigma}$ are the singular values, which we typically order in descending order $\sigma_1, \sigma_2, \ldots, \sigma_d$, giving

$$\mathbf{\Sigma} = \begin{bmatrix} \sigma_1 & 0 & 0 & \ldots & 0 \\ 0 & \sigma_2 & 0 & \ldots & 0 \\ & & \vdots & & \\ 0 & 0 & \ldots & 0 & \sigma_d \\ 0 & 0 & \ldots & 0 & 0 \\ & & \vdots & (n-d) \text{ rows} & \text{of zeros} \\ 0 & 0 & \ldots & 0 & 0 \end{bmatrix} = \begin{bmatrix} \mathbf{\Sigma}_d \\ \mathbf{0} \end{bmatrix} \qquad \text{where } \mathbf{\Sigma}_d = \begin{bmatrix} \sigma_1 & 0 & \ldots & 0 \\ 0 & \sigma_2 & \ldots & 0 \\ & & \vdots & \\ 0 & 0 & \ldots & \sigma_d \end{bmatrix}.$$

[2] An orthonormal matrix $\mathbf{U}$ is square matrix that satisfies $\mathbf{U}^\top\mathbf{U} = \mathbf{I}$ and $\mathbf{U}\mathbf{U}^\top = \mathbf{I}$

Any matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ can be decomposed into its singular value decomposition, because any linear transformation can be decomposed into a rotation (multiplication by $\mathbf{V}^\top$), followed by a scaling (multiplication by $\mathbf{\Sigma}$), followed again by a rotation (multiplication by $\mathbf{U}$).

This decomposition simplifies analysis of the properties of a matrix. For example, the number of non-zero singular values constitutes the rank of $\mathbf{X}$. To see why, assume $\sigma_d = 0$, and $\sigma_{d-1} > 0$, meaning $\mathbf{X}$ has rank $d-1$. Take any vector $\mathbf{w} \in \mathbb{R}^d$, and consider $\mathbf{X}\mathbf{w}$. We can write this product as $\mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top\mathbf{w} = \mathbf{U}\mathbf{\Sigma}\tilde{\mathbf{w}}$ for $\tilde{\mathbf{w}} = \mathbf{V}^\top\mathbf{w}$. The product $\mathbf{\Sigma}\tilde{\mathbf{w}}$ sets the last dimension of $\tilde{\mathbf{w}}$ to zero, effectively removing that dimension and so projecting $\tilde{\mathbf{w}}$ into a lower-dimensional $(d-1)$ space. Then it rotates that projected vector afterwards, using $\mathbf{U}$, but cannot undo that projection into a lower-dimensional space. Therefore, $\mathbf{X}\mathbf{w}$ can only product $\hat{\mathbf{y}} = \mathbf{X}\mathbf{w}$ that lie in a $d-1$-dimensional plane, rotated in $\mathbb{R}^{d-1}$. This decomposition, then, can help us understand the space of possible predictions for linear regression $\mathbf{X}\mathbf{w}$.

Now we can discuss the least-squares solution, in terms of the singular value decomposition of $\mathbf{X}$. Notice that

$$\mathbf{X}^\top\mathbf{X} = \mathbf{V}\mathbf{\Sigma}^\top\mathbf{U}^\top\mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top = \mathbf{V}\mathbf{\Sigma}_d^2\mathbf{V}^\top$$

because $\mathbf{U}$ is orthonormal and so $\mathbf{U}^\top\mathbf{U} = \mathbf{I}$ the identity matrix ($\mathbf{I}$ is a diagonal matrix with ones on the diagonal). The inverse of $\mathbf{X}^\top\mathbf{X}$ exists if $\mathbf{X}$ is full rank, i.e., $\mathbf{\Sigma}_d$ has no zeros on the diagonal, because $(\mathbf{X}^\top\mathbf{X})^{-1} = \mathbf{V}\mathbf{\Sigma}_d^{-2}\mathbf{V}^\top$. The resulting solution for $\mathbf{w}$ looks like[3]

$$\mathbf{w} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y} = \mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{U}^\top\mathbf{y} = \sum_{j=1}^{d} \frac{\mathbf{u}_j^\top\mathbf{y}}{\sigma_j}\mathbf{v}_j \tag{6.4}$$

where $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_n] \in \mathbb{R}^{n \times n}$ is the orthonormal matrix composed of the left singular vectors, $\mathbf{U}_d = [\mathbf{u}_1, \ldots, \mathbf{u}_d] \in \mathbb{R}^{n \times d}$ is the first d left singular vectors, and $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_d] \in \mathbb{R}^{d \times d}$ is the orthonormal matrix composed of the right singular vectors.

The solution in Equation (6.5) makes it clear why the linear regression solution can be sensitive to perturbations. For small singular values, σ_j^{-1} is large and amplifies any changes in $\mathbf{y}$. For example, for slightly different noise component ϵ_i for the ith sample, the solution vector $\mathbf{w}$ could be very different. A common strategy to deal with this instability is to drop or truncate small singular values. This is a form of regularization, which we discuss in the next section.

Remark: In the general case, where $\mathbf{X}$ is not full rank, we can still obtain a least-squares solution to $\mathbf{X}^\top\mathbf{X}\mathbf{w} = \mathbf{X}^\top\mathbf{y}$. Now, there are potentially infinitely many solutions. The common choice is to select the minimum variance solution, which corresponds to dropping the components (singular vectors) for the zero singular values:

$$\mathbf{w} = \sum_{j=1}^{\text{rank of } \mathbf{X}} \frac{\mathbf{u}_j^\top\mathbf{y}}{\sigma_j}\mathbf{v}_j. \tag{6.5}$$

Example 16: [Nearly linear dependent] Let's look at a simple example of why $\mathbf{X} \in \mathbb{R}^{n \times d}$ might have small singular values. First, assume $d = 2$ and $x_2 = x_1$, i.e., that the second features is a copy of the first and simply redundant. Then $\mathbf{X} = \mathbf{U}_2\mathbf{\Sigma}_2\mathbf{V}^\top$ is the thin SVD

[3]The last step in the below equation, writing the matrix product as a sum, is not immediately obvious. As an exercise, see if you can derive this last equality.

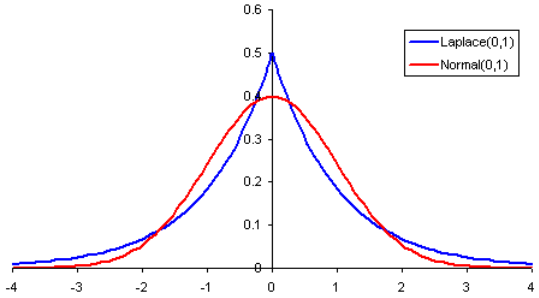

Figure 6.5: A comparison between Gaussian and Laplace priors. Both prefers values to be near zero, but the Laplace prior more strongly prefers the values to equal zero.

of $\mathbf{X}$, where $\mathbf{U}_2$ only has the first two columns of the full SVD. We can write this thin SVD because $\mathbf{X} = \mathbf{U}_2\mathbf{\Sigma}_2\mathbf{V}^\top = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$, where the zero singular values zero out the remaining columns of $\mathbf{U}$.

The SVD of just the first column $\mathbf{x}_1 \in \mathbb{R}^{n\times 1}$ is straightforward: $\mathbf{x}_1 = \mathbf{u}_1\sigma_1 v_1$, where $\mathbf{u}_1 = \mathbf{x}_1/\|\mathbf{x}_1\|$, $\sigma_1 = \|\mathbf{x}_1\|$ and $v_1 = 1$. The SVD of $\mathbf{X} = [\mathbf{x}_1 \quad \mathbf{x}_2]$ is therefore, for any n-dimensional unit vector $\mathbf{u}_2$ that is orthogonal to $\mathbf{u}_1$, and right singular vectors $\mathbf{v}_1, \mathbf{v}_2 \in \mathbb{R}^2$,

$$\mathbf{X} = [\mathbf{u}_1 \quad \mathbf{u}_2]\mathbf{\Sigma}[\mathbf{v}_1 \quad \mathbf{v}_2]^\top = [\mathbf{u}_1 \quad \mathbf{u}_2]\begin{bmatrix} 2\sigma_1 & 0 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} 0.5 & 0.5 \\ -0.5 & 0.5 \end{bmatrix} = \mathbf{u}_1\sigma_1[1.0 \quad 1.0]$$

where we extended v_1 to two-dimensions (since $d = 2$), and defined $\mathbf{v}_2$ to be orthogonal to that vector, and had to rescale σ_1 to maintain unit singular vectors. So because $\mathbf{x}_2$ is dependent on $\mathbf{x}_1$, the rank does not increase when we add it as a column and the singular value $\sigma_2 = 0$.

If instead $\mathbf{x}_2 = \mathbf{x}_1 + \epsilon$ for a small noise vector $\epsilon \in \mathbb{R}^n$, then instead we would find that σ_2 would no longer be zero, but would be very close to zero, because $\mathbf{u}_1$ and the first singular value σ_1 would largely be able to recreate $\mathbf{x}_2$. □

6.5.2 Regularization

So far, we have discussed linear regression in terms of maximum likelihood. But, as before, we can also propose a MAP objective. Instead of specifying no prior over $\mathbf{w}$, we can select a prior to help *regularize* overfitting to the observed data. We will discuss two common priors (regularizers): the Gaussian prior (ℓ_2 norm) and the Laplace prior (ℓ_1 norm), shown in Figure 6.5.

Taking the log of the zero-mean Gaussian prior, $\mathcal{N}(\mathbf{0}, \lambda^{-1}\mathbf{I})$, we get

$$-\ln p(\mathbf{w}) = \ln(2\pi|\lambda^{-1}\mathbf{I}|) + \frac{\mathbf{w}^\top\mathbf{w}}{2\lambda^{-1}} = \ln(2\pi) - d\ln(\lambda) + \frac{\lambda}{2}\mathbf{w}^\top\mathbf{w}.$$

because $|\lambda^{-1}\mathbf{I}| = \lambda^{-d}$, where $|\mathbf{A}|$ is the determinant of the matrix $\mathbf{A}$. As before, we can drop the first constant which does not affect the selection of $\mathbf{w}$.

Now we can combine the negative log-likelihood and the negative log prior. Then ignoring constants, we can add up the negative log-likelihood and negative log to the prior

to get

$$\begin{aligned}\operatorname*{argmin}_{\mathbf{w}} -\ln(p(\mathbf{y}|\mathbf{X},\mathbf{w})) - \ln p(\mathbf{w}) &= \operatorname*{argmin}_{\mathbf{w}} \frac{1}{2\sigma^2}\sum_{i=1}^{n}\left(y_i - \sum_{j=0}^{d} w_j x_{ij}\right)^2 + \frac{\lambda}{2}\mathbf{w}^\top\mathbf{w} \\ &= \operatorname*{argmin}_{\mathbf{w}} \sum_{i=1}^{n}\left(y_i - \sum_{j=0}^{d} w_j x_{ij}\right)^2 + \frac{\lambda\sigma^2}{2}\mathbf{w}^\top\mathbf{w}.\end{aligned}$$

Therefore if we assume that the weights have a zero-mean Gaussian prior $\mathcal{N}(\mathbf{0}, \lambda^{-1}\sigma^2\mathbf{I})$, then we get the following ridge regression problem:

$$c(\mathbf{w}) = (\mathbf{X}\mathbf{w} - \mathbf{y})^\top(\mathbf{X}\mathbf{w} - \mathbf{y}) + \lambda\mathbf{w}^\top\mathbf{w} \qquad \triangleright\, \|\mathbf{w}\|_2^2 = \mathbf{w}^\top\mathbf{w}$$

where λ is a user-selected parameter that is called the *regularization parameter*. The idea is to penalize weight coefficients that are too large; the larger the λ, the more large weights are penalized. Correspondingly, larger λ corresponds to a smaller covariance in the prior, pushing the weights to stay near zero. The MAP estimate, therefore, has to balance between this prior on the weights, and fitting the observed data.

If we solve this equation in a similar manner as before, we obtain

$$\mathbf{w}_{\text{MAP}} = (\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{y}.$$

This has the nice effect of shifting the squared singular values in Σ_d^2 by λ, removing stability issues with dividing by small singular values, as long as λ is itself large enough.

If we choose a Laplace distribution, we get an ℓ_1 penalized objective

$$c(\mathbf{w}) = (\mathbf{X}\mathbf{w} - \mathbf{y})^\top(\mathbf{X}\mathbf{w} - \mathbf{y}) + \lambda\|\mathbf{w}\|_1$$

which is often called the Lasso. This objective can be obtained similarly to the ℓ_2 regularized objective, but instead using a Laplace distribution with parameter λ for the prior. As with the ℓ_2 regularizer for ridge regression, this regularizer penalizes large values in $\mathbf{w}$. However, it also produces more sparse solutions, where entries in $\mathbf{w}$ are zero. This preference can be seen in Figure 6.5, where the Laplace distribution is more concentrated around zero. In practice, however, this preference is even stronger than implied by the distribution, due to how the spherical least-squares loss and the ℓ_1 regularizer interact.

Forcing entries in $\mathbf{w}$ to zero has the effect of feature selection, because zeroing entries in $\mathbf{w}$ is equivalent to removing the corresponding feature. Consider the dot product each time a prediction is made,

$$\mathbf{x}^\top\mathbf{w} = \sum_{j=0}^{d} x_j w_j = \sum_{j:w_j\neq 0} x_j w_j.$$

This is equivalent to simply dropping entries in $\mathbf{x}$ and $\mathbf{w}$ where $w_j = 0$.

For the Lasso, we no longer have a closed-form solution. We do not have a closed form solution, because we cannot solve for $\mathbf{w}$ in closed-form that provides a stationary point. Instead, we use gradient descent to compute a solution to $\mathbf{w}$. The ℓ_1 regularizer, however, is non-differentiable at 0. Understanding how to optimize this objective requires a bit more optimization background, so we provide this algorithm in the next chapter, in Algorithm 4.

6.5.3 Expectation and variance for the regularized solution

A natural question to ask is how this regularization parameter can be selected, and the impact on the final solution vector. The selection of this regularization parameter leads to a bias-variance trade-off. To understand this trade-off, we need to understand what it means for the solution to be biased, and how to characterize the variance of the solution, across possible datasets.

Let us begin with understanding the bias and variance of the non-regularized solution, presuming that the distributional assumptions behind linear regression are true. This means that there exists a true parameter $\boldsymbol{\omega}$ such that for each of the data points $Y_i = \sum_{j=0}^{d} \omega_j X_{ij} + \varepsilon_i$, where the ε_j are i.i.d. random variables drawn according to $\mathcal{N}(0, \sigma^2)$. We can characterize the solution vector (estimator) $\mathbf{w}_{\text{ML}}$ as a random variable, where the randomness is across possible datasets that could have been observed. In this sense, we are considering the dataset $\mathcal{D}$ to be a random variable, and the solution $\mathbf{w}_{\text{ML}}(\mathcal{D})$ from that dataset as a function of this random variable.

Let us now look at the expected value (with respect to training data set $\mathcal{D}$) for the weight vector $\mathbf{w}_{\text{ML}}$, with $\boldsymbol{\varepsilon} = (\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n)$:

$$\begin{aligned}
\mathbb{E}[\mathbf{w}_{\text{ML}}(\mathcal{D})] &= \mathbb{E}\left[\left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{X}^\top\left(\mathbf{X}\boldsymbol{\omega} + \boldsymbol{\varepsilon}\right)\right] \\
&= \mathbb{E}\left[\left(\mathbf{X}^\top\mathbf{X}\right)^{-1}(\mathbf{X}^\top\mathbf{X})\boldsymbol{\omega}\right] + \mathbb{E}\left[\left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{X}^\top\boldsymbol{\varepsilon}\right] \\
&= \mathbb{E}\left[\boldsymbol{\omega}\right] + \mathbb{E}\left[\left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{X}^\top\right]\mathbb{E}\left[\boldsymbol{\varepsilon}\right] \\
&= \boldsymbol{\omega},
\end{aligned}$$

where the third equality follows from the fact that the noise terms $\boldsymbol{\varepsilon}$ are independent of the features and the last equality because ω is a constant vector (non-random) and $\mathbb{E}[\boldsymbol{\varepsilon}] = \mathbf{0}$. An estimator whose expected value is the true value of the parameter is called an *unbiased estimator*. The covariance matrix for the optimal set of parameters can be expressed as

$$\begin{aligned}
\text{Cov}[\mathbf{w}_{\text{ML}}(\mathcal{D})] &= \mathbb{E}\left[\left(\mathbf{w}_{\text{ML}}(\mathcal{D}) - \boldsymbol{\omega}\right)\left(\mathbf{w}_{\text{ML}}(\mathcal{D}) - \boldsymbol{\omega}\right)^\top\right] \\
&= \mathbb{E}\left[\mathbf{w}_{\text{ML}}(\mathcal{D})\mathbf{w}_{\text{ML}}(\mathcal{D})^\top\right] - \boldsymbol{\omega}\boldsymbol{\omega}^\top
\end{aligned}$$

Taking[4] $\mathbf{X}^\dagger = \left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{X}^\top$, we have $\mathbf{w}_{\text{ML}}(\mathcal{D}) = \boldsymbol{\omega} + \mathbf{X}^\dagger\boldsymbol{\varepsilon}$, so

$$\begin{aligned}
\text{Cov}[\mathbf{w}_{\text{ML}}(\mathcal{D})] &= \mathbb{E}\left[\left(\boldsymbol{\omega} + \mathbf{X}^\dagger\boldsymbol{\varepsilon}\right)\left(\boldsymbol{\omega} + \mathbf{X}^\dagger\boldsymbol{\varepsilon}\right)^\top\right] - \boldsymbol{\omega}\boldsymbol{\omega}^\top \\
&= \boldsymbol{\omega}\boldsymbol{\omega}^\top + \mathbb{E}\left[\mathbf{X}^\dagger\boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^\top\mathbf{X}^{\dagger\top}\right] - \boldsymbol{\omega}\boldsymbol{\omega}^\top
\end{aligned}$$

because $\mathbb{E}\left[\mathbf{X}^\dagger\boldsymbol{\varepsilon}\boldsymbol{\omega}^\top\right] = \mathbb{E}\left[\mathbf{X}^\dagger\right]\mathbb{E}\left[\boldsymbol{\varepsilon}\right]\boldsymbol{\omega}^\top = \mathbf{0}$. Now because the noise terms are independent of the inputs, i.e., $\mathbb{E}\left[\boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^\top|\mathbf{X}\right] = \mathbb{E}\left[\boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^\top\right] = \sigma^2\mathbf{I}$, we can use the law of total probability (also

[4]This matrix is called the pseudo-inverse of $\mathbf{X}$. The idea of a pseudo-inverse generalizes the concept of inverses to non-invertible matrices, including rectangular matrices. It is a useful concept, but not one we will need to use again and so is not explained in-depth here.

called the tower rule), to get

$$\begin{aligned}\mathbb{E}\left[\mathbf{X}^\dagger \boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^\top \mathbf{X}^{\dagger\top}\right] &= \mathbb{E}\left[\mathbb{E}\left[\mathbf{X}^\dagger \boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^\top \mathbf{X}^{\dagger\top} | \mathbf{X}\right]\right] \\ &= \mathbb{E}\left[\mathbf{X}^\dagger \mathbb{E}\left[\boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^\top | \mathbf{X}\right] \mathbf{X}^{\dagger\top}\right] \\ &= \sigma^2 \mathbb{E}\left[\mathbf{X}^\dagger \mathbf{X}^{\dagger\top}\right].\end{aligned}$$

Thus, we have

$$\mathrm{Cov}[\mathbf{w}_{\mathrm{ML}}(\mathcal{D})] = \sigma^2 \mathbb{E}\left[(\mathbf{X}^\top \mathbf{X})^{-1}\right].$$

It can be shown that estimator $\mathbf{w}_{\mathrm{ML}}(\mathcal{D}) = \mathbf{X}^\dagger \mathbf{y}$ is the one with the smallest variance among all unbiased estimators (Gauss-Markov theorem).

Unfortunately, however, as discussed above, the matrix $\mathbf{X}^\top \mathbf{X} = \mathbf{V}\boldsymbol{\Sigma}\mathbf{V}^\top$ can be poorly conditioned, with some zero or near-zero singular values. Consequently, this covariance matrix can be poorly conditioned, with high magnitude co-variance values. This implies that, across datasets, the solution $\mathbf{w}_{\mathrm{ML}}(\mathcal{D})$ can vary widely. This type of behavior is suggestive of overfitting, and is not desirable. If our solution could be very different across several different random subsets of data, we cannot be confident in any one of these solutions.

The regularized solution, on the other hand, is much less likely to have high covariance, but will no longer be unbiased. Let $\mathbf{w}_{\mathrm{MAP}}(\mathcal{D})$ be the MAP estimate for the ℓ_2 regularized problem with some $\lambda > 0$. Using a similar analysis to above, the expected value of $\mathbf{w}_{\mathrm{MAP}}(\mathcal{D})$ is

$$\begin{aligned}\mathbb{E}[\mathbf{w}_{\mathrm{MAP}}(\mathcal{D})] &= \mathbb{E}\left[\left(\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}\right)^{-1} \mathbf{X}^\top (\mathbf{X}\boldsymbol{\omega} + \boldsymbol{\varepsilon})\right] \\ &= \mathbb{E}\left[\left(\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}\right)^{-1} (\mathbf{X}^\top \mathbf{X})\boldsymbol{\omega}\right] \\ &\neq \boldsymbol{\omega}.\end{aligned}$$

As $\lambda \to 0$, the MAP solution becomes closer and closer to being unbiased. The covariance is

$$\mathrm{Cov}[\mathbf{w}_{\mathrm{MAP}}(\mathcal{D})] = \sigma^2 \mathbb{E}\left[(\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1}\right].$$

This covariance is much less susceptible to ill-conditioned $\mathbf{X}^\top \mathbf{X}$, because as discussed above, the shift by λ improves the condition. Consequently, we expect $\mathbf{w}_{\mathrm{MAP}}$ to have lower variance across different datasets that could have been observed. This correspondingly implies that we are less likely to overfit to anyone dataset. Notice that as $\lambda \to \infty$, the variance decreases to zero, but the bias increases to infinity. As depicted in Figure 6.6, there is an optimal choice of λ that minimizes this bias-variance trade-off—if we could find it.

Exercise: Derive the covariance formula for $\mathbf{w}_{\mathrm{MAP}}(\mathcal{D})$.

The bias-variance trade-off comes in many forms. One such trade-off is in the selection of our function class. If we select a simple function class, the class is likely not large enough—not powerful enough—to represent the true function. This introduces some bias, but likely also has lower variance, because that simpler function class is less likely to overfit to any one dataset. If this class is too simple, we might say that our function is *underparametrized* and is *underfitting*. On the other hand, if we select a more powerful function class, that

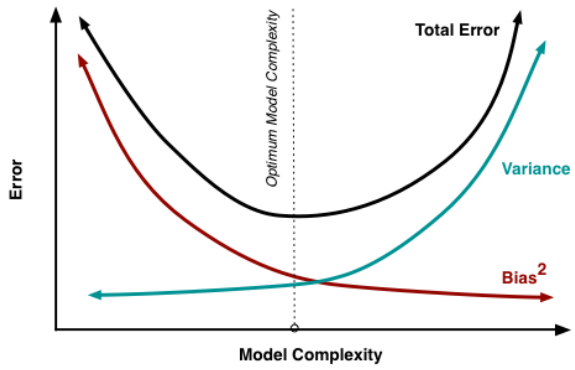

Figure 6.6: The bias-variance trade-off. Image obtained from: http://scott.fortmann-roe.com/docs/BiasVariance.html

does contain the true function, we may not have any bias but could have high variance due to the ability to find a function in your large class that overfits a given dataset. In this setting, we might say the function is over-parametrized, and though we have the ability to learn a highly accurate function, it will be difficult to actually find that function amongst this larger class. Instead, one is likely to select a model that overfits to the given data, and does not generalize to new data (i.e., performs poorly on new data).

Finding the balance between bias and variance, and between underfitting and overfitting, is a core problem in machine learning. We discuss ways to theoretically and empirically investigate this trade-off, in Chapter 10.

Remark: Above we assumed that the true model was linear, and so the only bias introduced was from the regularization. This assumed that the hypothesis space of linear functions was sufficiently powerful. In reality, when using linear regression with regularization, we are introducing bias both from selecting a simpler function class and from the regularization. If a powerful basis is used to first transform the data, to provide nonlinear functions even though the solution uses linear regression, then it is feasible that this function class is sufficiently powerful, and the bias is mostly due to regularization.