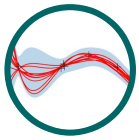
9

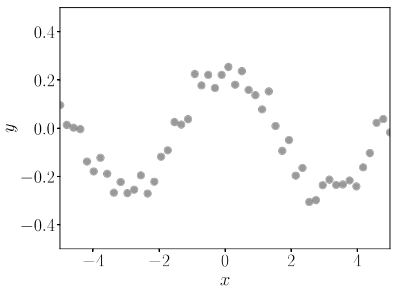
Linear Regression



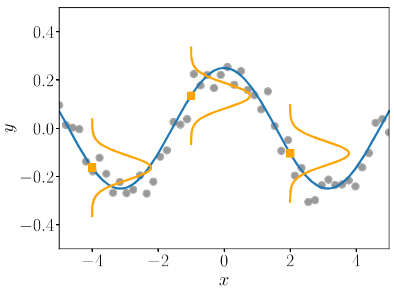
In the following, we will apply the mathematical concepts from Chapters 2, 5, 6, and 7 to solve linear regression (curve fitting) problems. In *regression*, we aim to find a function f that maps inputs $\boldsymbol{x} \in \mathbb{R}^D$ to corresponding function values $f(\boldsymbol{x}) \in \mathbb{R}$. We assume we are given a set of training inputs $\boldsymbol{x}_n$ and corresponding noisy observations $y_n = f(\boldsymbol{x}_n) + \epsilon$, where ϵ is an i.i.d. random variable that describes measurement/observation noise and potentially unmodeled processes (which we will not consider further in this chapter). Throughout this chapter, we assume zero-mean Gaussian noise. Our task is to find a function that not only models the training data, but generalizes well to predicting function values at input locations that are not part of the training data (see Chapter 8). An illustration of such a regression problem is given in Figure 9.1. A typical regression setting is given in Figure 9.1(a): For some input values x_n, we observe (noisy) function values $y_n = f(x_n) + \epsilon$. The task is to infer the function f that generated the data and generalizes well to function values at new input locations. A possible solution is given in Figure 9.1(b), where we also show three distributions centered at the function values $f(x)$ that represent the noise in the data.

regression

Regression is a fundamental problem in machine learning, and regression problems appear in a diverse range of research areas and applica-



(a) Regression problem: observed noisy function values from which we wish to infer the underlying function that generated the data.



(b) Regression solution: possible function that could have generated the data (blue) with indication of the measurement noise of the function value at the corresponding inputs (orange distributions).

Figure 9.1
(a) Dataset;
(b) possible solution to the regression problem.

LINEAR REGRESSION

Supervised task

$(\underline{x}_1, y_1)\ (\underline{x}_2 .. y_2) \ .. \ (x_N, y_N)$

$(\underline{x}_n, y_n)$, $n = 1 .. N$

Prediction function → LINEAR FUNCTION $f(x_n)$

$$y_n = \underline{\theta}^T \cdot \underline{x}_n + \varepsilon_n$$

$$\varepsilon_i \in N(0, \sigma^2)$$

$\underline{x}_i$ → features → deterministic values

y_i → observations → random values

If $\varepsilon_n \sim N(0, \sigma^2)$ ⇒

$y_n \sim N(\theta^T x_n, \sigma^2)$

tions, including time-series analysis (e.g., system identification), control and robotics (e.g., reinforcement learning, forward/inverse model learning), optimization (e.g., line searches, global optimization), and deep-learning applications (e.g., computer games, speech-to-text translation, image recognition, automatic video annotation). Regression is also a key ingredient of classification algorithms. Finding a regression function requires solving a variety of problems, including the following:

Normally, the type of noise could also be a "model choice", but we fix the noise to be Gaussian in this chapter.

- **Choice of the model (type) and the parametrization** of the regression function. Given a dataset, what function classes (e.g., polynomials) are good candidates for modeling the data, and what particular parametrization (e.g., degree of the polynomial) should we choose? Model selection, as discussed in Section 8.6, allows us to compare various models to find the simplest model that explains the training data reasonably well.
- **Finding good parameters.** Having chosen a model of the regression function, how do we find good model parameters? Here, we will need to look at different loss/objective functions (they determine what a "good" fit is) and optimization algorithms that allow us to minimize this loss.
- **Overfitting and model selection.** Overfitting is a problem when the regression function fits the training data "too well" but does not generalize to unseen test data. Overfitting typically occurs if the underlying model (or its parametrization) is overly flexible and expressive; see Section 8.6. We will look at the underlying reasons and discuss ways to mitigate the effect of overfitting in the context of linear regression.
- **Relationship between loss functions and parameter priors.** Loss functions (optimization objectives) are often motivated and induced by probabilistic models. We will look at the connection between loss functions and the underlying prior assumptions that induce these losses.
- **Uncertainty modeling.** In any practical setting, we have access to only a finite, potentially large, amount of (training) data for selecting the model class and the corresponding parameters. Given that this finite amount of training data does not cover all possible scenarios, we may want to describe the remaining parameter uncertainty to obtain a measure of confidence of the model's prediction at test time; the smaller the training set, the more important uncertainty modeling. Consistent modeling of uncertainty equips model predictions with confidence bounds.

In the following, we will be using the mathematical tools from Chapters 3, 5, 6 and 7 to solve linear regression problems. We will discuss maximum likelihood and maximum a posteriori (MAP) estimation to find optimal model parameters. Using these parameter estimates, we will have a brief look at generalization errors and overfitting. Toward the end of this chapter, we will discuss Bayesian linear regression, which allows us to reason about model parameters at a higher level, thereby removing some of the problems encountered in maximum likelihood and MAP estimation.

9.1 Problem Formulation

Because of the presence of observation noise, we will adopt a probabilistic approach and explicitly model the noise using a likelihood function. More specifically, throughout this chapter, we consider a regression problem with the likelihood function

$$p(y \mid \boldsymbol{x}) = \mathcal{N}\left(y \mid f(\boldsymbol{x}),\, \sigma^2\right). \tag{9.1}$$

Here, $\boldsymbol{x} \in \mathbb{R}^D$ are inputs and $y \in \mathbb{R}$ are noisy function values (targets). With (9.1), the functional relationship between $\boldsymbol{x}$ and y is given as

$$y = f(\boldsymbol{x}) + \epsilon\,, \tag{9.2}$$

where $\epsilon \sim \mathcal{N}\left(0,\, \sigma^2\right)$ is independent, identically distributed (i.i.d.) Gaussian measurement noise with mean 0 and variance σ^2. Our objective is to find a function that is close (similar) to the unknown function f that generated the data and that generalizes well.

In this chapter, we focus on parametric models, i.e., we choose a parametrized function and find parameters $\boldsymbol{\theta}$ that "work well" for modeling the data. For the time being, we assume that the noise variance σ^2 is known and focus on learning the model parameters $\boldsymbol{\theta}$. In linear regression, we consider the special case that the parameters $\boldsymbol{\theta}$ appear linearly in our model. An example of linear regression is given by

$$p(y \mid \boldsymbol{x}, \boldsymbol{\theta}) = \mathcal{N}\left(y \mid \boldsymbol{x}^\top \boldsymbol{\theta},\, \sigma^2\right) \tag{9.3}$$

$$\iff y = \boldsymbol{x}^\top \boldsymbol{\theta} + \epsilon\,, \quad \epsilon \sim \mathcal{N}\left(0,\, \sigma^2\right), \tag{9.4}$$

where $\boldsymbol{\theta} \in \mathbb{R}^D$ are the parameters we seek. The class of functions described by (9.4) are straight lines that pass through the origin. In (9.4), we chose a parametrization $f(\boldsymbol{x}) = \boldsymbol{x}^\top \boldsymbol{\theta}$.

A Dirac delta (delta function) is zero everywhere except at a single point, and its integral is 1. It can be considered a Gaussian in the limit of $\sigma^2 \to 0$.

The *likelihood* in (9.3) is the probability density function of y evaluated at $\boldsymbol{x}^\top \boldsymbol{\theta}$. Note that the only source of uncertainty originates from the observation noise (as $\boldsymbol{x}$ and $\boldsymbol{\theta}$ are assumed known in (9.3)). Without observation noise, the relationship between $\boldsymbol{x}$ and y would be deterministic and (9.3) would be a Dirac delta.

likelihood

Example 9.1

For $x, \theta \in \mathbb{R}$ the linear regression model in (9.4) describes straight lines (linear functions), and the parameter θ is the slope of the line. Figure 9.1(a) shows some example functions for different values of θ.

Linear regression refers to models that are linear in the parameters.

The linear regression model in (9.3)–(9.4) is not only linear in the parameters, but also linear in the inputs x. Figure 9.1(a) shows examples of such functions. We will see later that $y = \boldsymbol{\phi}^\top(\boldsymbol{x})\boldsymbol{\theta}$ for nonlinear transformations $\boldsymbol{\phi}$ is also a linear regression model because "linear regression"

The log-negative likelihood

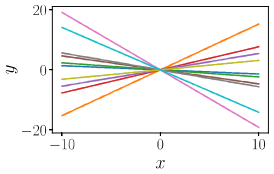
$$\mathcal{L}(\theta) = \frac{1}{2\sigma^2} \sum_{n=1}^{N} (y_n - \theta^T x_n)^2 \leftarrow$$

as seen in chapter 8.

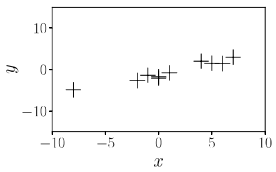
Hence LOSS FUNCTION

$$\theta_{ML} = \underset{\theta \in R^D}{\operatorname{argmin}} \mathcal{L}(\theta) \leftarrow$$

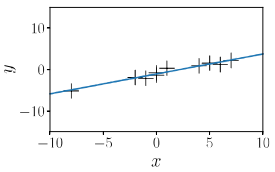
Figure 9.2 Linear regression example. (a) Example functions that fall into this category; (b) training set; (c) maximum likelihood estimate.



(a) Example functions (straight lines) that can be described using the linear model in (9.4).



(b) Training set.



(c) Maximum likelihood estimate.

refers to models that are "linear in the parameters", i.e., models that describe a function by a linear combination of input features. Here, a "feature" is a representation $\phi(\boldsymbol{x})$ of the inputs $\boldsymbol{x}$.

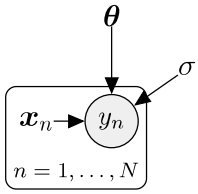
In the following, we will discuss in more detail how to find good parameters $\boldsymbol{\theta}$ and how to evaluate whether a parameter set "works well". For the time being, we assume that the noise variance σ^2 is known.

9.2 Parameter Estimation

Consider the linear regression setting (9.4) and assume we are given a *training set* $\mathcal{D} := \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$ consisting of N inputs $\boldsymbol{x}_n \in \mathbb{R}^D$ and corresponding observations/targets $y_n \in \mathbb{R}$, $n = 1, \ldots, N$. The corresponding graphical model is given in Figure 9.3. Note that y_i and y_j are conditionally independent given their respective inputs $\boldsymbol{x}_i, \boldsymbol{x}_j$ so that the likelihood factorizes according to

training set

Figure 9.3 Probabilistic graphical model for linear regression. Observed random variables are shaded, deterministic/known values are without circles.



$$p(\mathcal{Y} \mid \mathcal{X}, \boldsymbol{\theta}) = p(y_1, \ldots, y_N \mid \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N, \boldsymbol{\theta}) \tag{9.5a}$$

$$= \prod_{n=1}^{N} p(y_n \mid \boldsymbol{x}_n, \boldsymbol{\theta}) = \prod_{n=1}^{N} \mathcal{N}\left(y_n \mid \boldsymbol{x}_n^\top \boldsymbol{\theta}, \sigma^2\right), \tag{9.5b}$$

where we defined $\mathcal{X} := \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ and $\mathcal{Y} := \{y_1, \ldots, y_N\}$ as the sets of training inputs and corresponding targets, respectively. The likelihood and the factors $p(y_n \mid \boldsymbol{x}_n, \boldsymbol{\theta})$ are Gaussian due to the noise distribution; see (9.3).

In the following, we will discuss how to find optimal parameters $\boldsymbol{\theta}^* \in \mathbb{R}^D$ for the linear regression model (9.4). Once the parameters $\boldsymbol{\theta}^*$ are found, we can predict function values by using this parameter estimate in (9.4) so that at an arbitrary test input $\boldsymbol{x}_*$ the distribution of the corresponding target y_* is

$$p(y_* \mid \boldsymbol{x}_*, \boldsymbol{\theta}^*) = \mathcal{N}\left(y_* \mid \boldsymbol{x}_*^\top \boldsymbol{\theta}^*, \sigma^2\right). \tag{9.6}$$

In the following, we will have a look at parameter estimation by maximizing the likelihood, a topic that we already covered to some degree in Section 8.3.

9.2.1 Maximum Likelihood Estimation

maximum likelihood estimation

A widely used approach to finding the desired parameters $\boldsymbol{\theta}_{\text{ML}}$ is *maximum likelihood estimation*, where we find parameters $\boldsymbol{\theta}_{\text{ML}}$ that maximize the likelihood (9.5b). Intuitively, maximizing the likelihood means maximizing the predictive distribution of the training data given the model parameters. We obtain the maximum likelihood parameters as

Maximizing the likelihood means maximizing the predictive distribution of the (training) data given the parameters.

$$\boldsymbol{\theta}_{\text{ML}} = \arg\max_{\boldsymbol{\theta}} p(\mathcal{Y} \mid \mathcal{X}, \boldsymbol{\theta}) \,. \tag{9.7}$$

Remark. The likelihood $p(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta})$ is not a probability distribution in $\boldsymbol{\theta}$: It is simply a function of the parameters $\boldsymbol{\theta}$ but does not integrate to 1 (i.e., it is unnormalized), and may not even be integrable with respect to $\boldsymbol{\theta}$. However, the likelihood in (9.7) is a normalized probability distribution in $\boldsymbol{y}$. ♢

The likelihood is not a probability distribution in the parameters.

To find the desired parameters $\boldsymbol{\theta}_{\text{ML}}$ that maximize the likelihood, we typically perform gradient ascent (or gradient descent on the negative likelihood). In the case of linear regression we consider here, however, a closed-form solution exists, which makes iterative gradient descent unnecessary. In practice, instead of maximizing the likelihood directly, we apply the log-transformation to the likelihood function and minimize the negative log-likelihood.

Since the logarithm is a (strictly) monotonically increasing function, the optimum of a function f is identical to the optimum of $\log f$.

Remark (Log-Transformation). Since the likelihood (9.5b) is a product of N Gaussian distributions, the log-transformation is useful since (a) it does not suffer from numerical underflow, and (b) the differentiation rules will turn out simpler. More specifically, numerical underflow will be a problem when we multiply N probabilities, where N is the number of data points, since we cannot represent very small numbers, such as 10^{-256}. Furthermore, the log-transform will turn the product into a sum of log-probabilities such that the corresponding gradient is a sum of individual gradients, instead of a repeated application of the product rule (5.46) to compute the gradient of a product of N terms. ♢

To find the optimal parameters $\boldsymbol{\theta}_{\text{ML}}$ of our linear regression problem, we minimize the negative log-likelihood

$$-\log p(\mathcal{Y} \mid \mathcal{X}, \boldsymbol{\theta}) = -\log \prod_{n=1}^{N} p(y_n \mid \boldsymbol{x}_n, \boldsymbol{\theta}) = -\sum_{n=1}^{N} \log p(y_n \mid \boldsymbol{x}_n, \boldsymbol{\theta}) \,, \tag{9.8}$$

where we exploited that the likelihood (9.5b) factorizes over the number of data points due to our independence assumption on the training set.

In the linear regression model (9.4), the likelihood is Gaussian (due to the Gaussian additive noise term), such that we arrive at

see chapter 8 Example 8.5

$$\log p(y_n \mid \boldsymbol{x}_n, \boldsymbol{\theta}) = -\frac{1}{2\sigma^2}(y_n - \boldsymbol{x}_n^\top \boldsymbol{\theta})^2 + \text{const} \,, \tag{9.9}$$

where the constant includes all terms independent of $\boldsymbol{\theta}$. Using (9.9) in the

negative log-likelihood (9.8), we obtain (ignoring the constant terms)

$$\mathcal{L}(\boldsymbol{\theta}) := \frac{1}{2\sigma^2}\sum_{n=1}^{N}(y_n - \boldsymbol{x}_n^\top\boldsymbol{\theta})^2 \tag{9.10a}$$

$$= \frac{1}{2\sigma^2}(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\theta})^\top(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\theta}) = \frac{1}{2\sigma^2}\|\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\theta}\|^2 . \tag{9.10b}$$

The negative log-likelihood function is also called *error function*.

design matrix

The squared error is often used as a measure of distance. Recall from Section 3.1 that $\|\boldsymbol{x}\|^2 = \boldsymbol{x}^\top\boldsymbol{x}$ if we choose the dot product as the inner product.

where we define the *design matrix* $\boldsymbol{X} := [\boldsymbol{x}_1, \dots, \boldsymbol{x}_N]^\top \in \mathbb{R}^{N\times D}$ as the collection of training inputs and $\boldsymbol{y} := [y_1, \dots, y_N]^\top \in \mathbb{R}^N$ as a vector that collects all training targets. Note that the nth row in the design matrix $\boldsymbol{X}$ corresponds to the training input $\boldsymbol{x}_n$. In (9.10b), we used the fact that the sum of squared errors between the observations y_n and the corresponding model prediction $\boldsymbol{x}_n^\top\boldsymbol{\theta}$ equals the squared distance between $\boldsymbol{y}$ and $\boldsymbol{X}\boldsymbol{\theta}$.

With (9.10b), we have now a concrete form of the negative log-likelihood function we need to optimize. We immediately see that (9.10b) is quadratic in $\boldsymbol{\theta}$. This means that we can find a unique global solution $\boldsymbol{\theta}_{\text{ML}}$ for minimizing the negative log-likelihood $\mathcal{L}$. We can find the global optimum by computing the gradient of $\mathcal{L}$, setting it to $\boldsymbol{0}$ and solving for $\boldsymbol{\theta}$.

Using the results from Chapter 5, we compute the gradient of $\mathcal{L}$ with respect to the parameters as

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{\theta}} = \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}}\left(\frac{1}{2\sigma^2}(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\theta})^\top(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\theta})\right) \tag{9.11a}$$

$$= \frac{1}{2\sigma^2}\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}}\left(\boldsymbol{y}^\top\boldsymbol{y} - 2\boldsymbol{y}^\top\boldsymbol{X}\boldsymbol{\theta} + \boldsymbol{\theta}^\top\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{\theta}\right) \tag{9.11b}$$

$$= \frac{1}{\sigma^2}(-\boldsymbol{y}^\top\boldsymbol{X} + \boldsymbol{\theta}^\top\boldsymbol{X}^\top\boldsymbol{X}) \in \mathbb{R}^{1\times D} . \tag{9.11c}$$

The maximum likelihood estimator $\boldsymbol{\theta}_{\text{ML}}$ solves $\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{\theta}} = \boldsymbol{0}^\top$ (necessary optimality condition) and we obtain

Ignoring the possibility of duplicate data points, $\mathrm{rk}(\boldsymbol{X}) = D$ if $N \geqslant D$, i.e., we do not have more parameters than data points.

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{\theta}} = \boldsymbol{0}^\top \overset{(9.11c)}{\iff} \boldsymbol{\theta}_{\text{ML}}^\top\boldsymbol{X}^\top\boldsymbol{X} = \boldsymbol{y}^\top\boldsymbol{X} \tag{9.12a}$$

$$\iff \boldsymbol{\theta}_{\text{ML}}^\top = \boldsymbol{y}^\top\boldsymbol{X}(\boldsymbol{X}^\top\boldsymbol{X})^{-1} \tag{9.12b}$$

$$\iff \boldsymbol{\theta}_{\text{ML}} = (\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top\boldsymbol{y} . \tag{9.12c}$$

We could right-multiply the first equation by $(\boldsymbol{X}^\top\boldsymbol{X})^{-1}$ because $\boldsymbol{X}^\top\boldsymbol{X}$ is positive definite if $\mathrm{rk}(\boldsymbol{X}) = D$, where $\mathrm{rk}(\boldsymbol{X})$ denotes the rank of $\boldsymbol{X}$.

Remark. Setting the gradient to $\boldsymbol{0}^\top$ is a necessary and sufficient condition, and we obtain a global minimum since the Hessian $\nabla^2_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}) = \boldsymbol{X}^\top\boldsymbol{X} \in \mathbb{R}^{D\times D}$ is positive definite. ♢

Remark. The maximum likelihood solution in (9.12c) requires us to solve a system of linear equations of the form $\boldsymbol{A}\boldsymbol{\theta} = \boldsymbol{b}$ with $\boldsymbol{A} = (\boldsymbol{X}^\top\boldsymbol{X})$ and $\boldsymbol{b} = \boldsymbol{X}^\top\boldsymbol{y}$. ♢

$\min \ell(\theta) =$

$\min \frac{1}{2\sigma^2} \| y - X\theta \|^2 \rightarrow$ CONVEX

LEAST SQUARES PROBLEM

First order conditions $\Rightarrow$

$\nabla \ell(\theta) = 0 \iff$

$$\boxed{X^T X \theta = X^T y}$$ linear system

symmetric & positive definite if rank (X) maximum.

Example 9.2 (Fitting Lines)
Let us have a look at Figure 9.2, where we aim to fit a straight line $f(x) = \theta x$, where θ is an unknown slope, to a dataset using maximum likelihood estimation. Examples of functions in this model class (straight lines) are shown in Figure 9.1(a). For the dataset shown in Figure 9.1(b), we find the maximum likelihood estimate of the slope parameter θ using (9.12c) and obtain the maximum likelihood linear function in Figure 9.1(c).

Maximum Likelihood Estimation with Features

So far, we considered the linear regression setting described in (9.4), which allowed us to fit straight lines to data using maximum likelihood estimation. However, straight lines are not sufficiently expressive when it comes to fitting more interesting data. Fortunately, linear regression offers us a way to fit nonlinear functions within the linear regression framework: Since "linear regression" only refers to "linear in the parameters", we can perform an arbitrary nonlinear transformation $\boldsymbol{\phi}(\boldsymbol{x})$ of the inputs $\boldsymbol{x}$ and then linearly combine the components of this transformation. The corresponding linear regression model is

Linear regression refers to "linear-in-the-parameters" regression models, but the inputs can undergo any nonlinear transformation.

$$p(y \mid \boldsymbol{x}, \boldsymbol{\theta}) = \mathcal{N}(y \mid \boldsymbol{\phi}^\top(\boldsymbol{x})\boldsymbol{\theta}, \sigma^2)$$
$$\iff y = \boldsymbol{\phi}^\top(\boldsymbol{x})\boldsymbol{\theta} + \epsilon = \sum_{k=0}^{K-1} \theta_k \phi_k(\boldsymbol{x}) + \epsilon\,, \tag{9.13}$$

where $\boldsymbol{\phi} : \mathbb{R}^D \to \mathbb{R}^K$ is a (nonlinear) transformation of the inputs $\boldsymbol{x}$ and $\phi_k : \mathbb{R}^D \to \mathbb{R}$ is the kth component of the *feature vector* $\boldsymbol{\phi}$. Note that the model parameters $\boldsymbol{\theta}$ still appear only linearly.

feature vector

Example 9.3 (Polynomial Regression)
We are concerned with a regression problem $y = \boldsymbol{\phi}^\top(x)\boldsymbol{\theta} + \epsilon$, where $x \in \mathbb{R}$ and $\boldsymbol{\theta} \in \mathbb{R}^K$. A transformation that is often used in this context is

$$\boldsymbol{\phi}(x) = \begin{bmatrix} \phi_0(x) \\ \phi_1(x) \\ \vdots \\ \phi_{K-1}(x) \end{bmatrix} = \begin{bmatrix} 1 \\ x \\ x^2 \\ x^3 \\ \vdots \\ x^{K-1} \end{bmatrix} \in \mathbb{R}^K\,. \tag{9.14}$$

This means that we "lift" the original one-dimensional input space into a K-dimensional feature space consisting of all monomials x^k for $k = 0, \ldots, K-1$. With these features, we can model polynomials of degree $\leqslant K-1$ within the framework of linear regression: A polynomial of degree

$K-1$ is

$$f(x) = \sum_{k=0}^{K-1} \theta_k x^k = \boldsymbol{\phi}^\top(x)\boldsymbol{\theta}\,, \tag{9.15}$$

where $\boldsymbol{\phi}$ is defined in (9.14) and $\boldsymbol{\theta} = [\theta_0, \ldots, \theta_{K-1}]^\top \in \mathbb{R}^K$ contains the (linear) parameters θ_k.

Let us now have a look at maximum likelihood estimation of the parameters $\boldsymbol{\theta}$ in the linear regression model (9.13). We consider training inputs $\boldsymbol{x}_n \in \mathbb{R}^D$ and targets $y_n \in \mathbb{R}$, $n = 1, \ldots, N$, and define the *feature matrix* (*design matrix*) as

feature matrix
design matrix

$$\boldsymbol{\Phi} := \begin{bmatrix} \boldsymbol{\phi}^\top(\boldsymbol{x}_1) \\ \vdots \\ \boldsymbol{\phi}^\top(\boldsymbol{x}_N) \end{bmatrix} = \begin{bmatrix} \phi_0(\boldsymbol{x}_1) & \cdots & \phi_{K-1}(\boldsymbol{x}_1) \\ \phi_0(\boldsymbol{x}_2) & \cdots & \phi_{K-1}(\boldsymbol{x}_2) \\ \vdots & & \vdots \\ \phi_0(\boldsymbol{x}_N) & \cdots & \phi_{K-1}(\boldsymbol{x}_N) \end{bmatrix} \in \mathbb{R}^{N \times K}\,, \tag{9.16}$$

where $\Phi_{ij} = \phi_j(\boldsymbol{x}_i)$ and $\phi_j : \mathbb{R}^D \to \mathbb{R}$.

Example 9.4 (Feature Matrix for Second-order Polynomials)
For a second-order polynomial and N training points $x_n \in \mathbb{R}, n = 1, \ldots, N$, the feature matrix is

$$\boldsymbol{\Phi} = \begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ \vdots & \vdots & \vdots \\ 1 & x_N & x_N^2 \end{bmatrix}. \tag{9.17}$$

With the feature matrix $\boldsymbol{\Phi}$ defined in (9.16), the negative log-likelihood for the linear regression model (9.13) can be written as

$$-\log p(\mathcal{Y} \mid \mathcal{X}, \boldsymbol{\theta}) = \frac{1}{2\sigma^2}(\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\theta})^\top(\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\theta}) + \text{const}\,. \tag{9.18}$$

Comparing (9.18) with the negative log-likelihood in (9.10b) for the "feature-free" model, we immediately see we just need to replace $\boldsymbol{X}$ with $\boldsymbol{\Phi}$. Since both $\boldsymbol{X}$ and $\boldsymbol{\Phi}$ are independent of the parameters $\boldsymbol{\theta}$ that we wish to optimize, we arrive immediately at the *maximum likelihood estimate*

maximum likelihood estimate

$$\boldsymbol{\theta}_{\text{ML}} = (\boldsymbol{\Phi}^\top\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^\top\boldsymbol{y} \tag{9.19}$$

for the linear regression problem with nonlinear features defined in (9.13).

Remark. When we were working without features, we required $\boldsymbol{X}^\top\boldsymbol{X}$ to

$$\min \mathcal{L}(\theta) := \|\underline{y} - \underline{\Phi}\,\underline{\theta}\|_2^2 \iff$$

$$\nabla \mathcal{L}(\theta) = 0$$

$$\downarrow$$

$$\underline{\Phi}^T \underline{\Phi}\,\underline{\theta} - \underline{\Phi}^T \underline{y} = 0$$

$$\downarrow$$

$$\underline{\Phi}^T \underline{\Phi}\,\underline{\theta} = \underline{\Phi}^T \underline{y} \quad \longleftarrow \text{linear system}$$

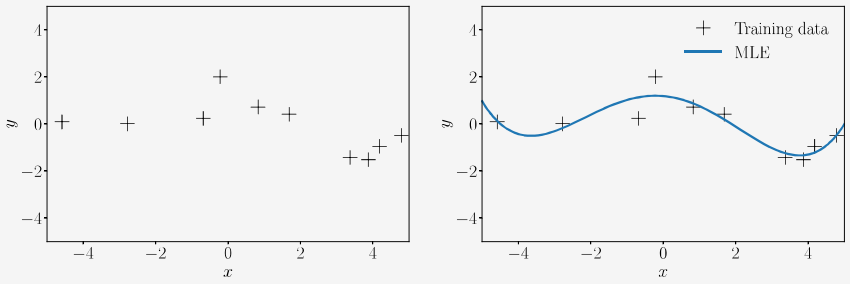
$\underline{\Phi}^T \underline{\Phi}$: symmetric & pos. def.

$$\underline{\Phi} \in \mathbb{R}^{N \times K} \qquad N > K$$

$x_i \neq x_j$.

be invertible, which is the case when the rows of $\boldsymbol{X}$ are linearly independent. In (9.19), we therefore require $\boldsymbol{\Phi}^\top \boldsymbol{\Phi} \in \mathbb{R}^{D\times D}$ to be invertible. This is the case if and only if $\mathrm{rk}(\boldsymbol{\Phi}) = D$. ♢

Example 9.5 (Maximum Likelihood Polynomial Fit)



(a) Regression dataset.

(b) Polynomial of degree 4 determined by maximum likelihood estimation.

Figure 9.4 Polynomial regression: (a) dataset consisting of (x_n, y_n) pairs, $n = 1, \ldots, 10$; (b) maximum likelihood polynomial of degree 4.

Consider the dataset in Figure 9.4(a). The dataset consists of $N = 20$ pairs (x_n, y_n), where $x_n \sim \mathcal{U}[-5, 5]$ and $y_n = -\sin(x_n/5) + \cos(x_n) + \epsilon$, where $\epsilon \sim \mathcal{N}(0, 0.2^2)$.

We fit a polynomial of degree $K = 4$ using maximum likelihood estimation, i.e., parameters $\boldsymbol{\theta}_{\mathrm{ML}}$ are given in (9.19). The maximum likelihood estimate yields function values $\boldsymbol{\phi}^\top(x_*)\boldsymbol{\theta}_{\mathrm{ML}}$ at any test location x_*. The result is shown in Figure 9.4(b).

Estimating the Noise Variance

Thus far, we assumed that the noise variance σ^2 is known. However, we can also use the principle of maximum likelihood estimation to obtain the maximum likelihood estimator σ^2_{ML} for the noise variance. To do this, we follow the standard procedure: We write down the log-likelihood, compute its derivative with respect to $\sigma^2 > 0$, set it to 0, and solve. The log-likelihood is given by

$$\log p(\mathcal{Y} \mid \mathcal{X}, \boldsymbol{\theta}, \sigma^2) = \sum_{n=1}^{N} \log \mathcal{N}\left(y_n \mid \boldsymbol{\phi}^\top(\boldsymbol{x}_n)\boldsymbol{\theta},\, \sigma^2\right) \tag{9.20a}$$

$$= \sum_{n=1}^{N} \left(-\frac{1}{2}\log(2\pi) - \frac{1}{2}\log\sigma^2 - \frac{1}{2\sigma^2}(y_n - \boldsymbol{\phi}^\top(\boldsymbol{x}_n)\boldsymbol{\theta})^2 \right) \tag{9.20b}$$

$$= -\frac{N}{2}\log\sigma^2 - \frac{1}{2\sigma^2}\underbrace{\sum_{n=1}^{N}(y_n - \boldsymbol{\phi}^\top(\boldsymbol{x}_n)\boldsymbol{\theta})^2}_{=:s} + \text{const}\,. \tag{9.20c}$$

The partial derivative of the log-likelihood with respect to σ^2 is then

$$\frac{\partial \log p(\mathcal{Y} \mid \mathcal{X}, \boldsymbol{\theta}, \sigma^2)}{\partial \sigma^2} = -\frac{N}{2\sigma^2} + \frac{1}{2\sigma^4} s = 0 \tag{9.21a}$$

$$\iff \frac{N}{2\sigma^2} = \frac{s}{2\sigma^4} \tag{9.21b}$$

so that we identify

$$\sigma^2_{\text{ML}} = \frac{s}{N} = \frac{1}{N} \sum_{n=1}^{N} (y_n - \boldsymbol{\phi}^\top(\boldsymbol{x}_n)\boldsymbol{\theta})^2 \,. \tag{9.22}$$

Therefore, the maximum likelihood estimate of the noise variance is the empirical mean of the squared distances between the noise-free function values $\boldsymbol{\phi}^\top(\boldsymbol{x}_n)\boldsymbol{\theta}$ and the corresponding noisy observations y_n at input locations $\boldsymbol{x}_n$.

9.2.2 Overfitting in Linear Regression

We just discussed how to use maximum likelihood estimation to fit linear models (e.g., polynomials) to data. We can evaluate the quality of the model by computing the error/loss incurred. One way of doing this is to compute the negative log-likelihood (9.10b), which we minimized to determine the maximum likelihood estimator. Alternatively, given that the noise parameter σ^2 is not a free model parameter, we can ignore the scaling by $1/\sigma^2$, so that we end up with a squared-error-loss function $\|\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\theta}\|^2$. Instead of using this squared loss, we often use the *root mean square error* (*RMSE*)

root mean square error
RMSE

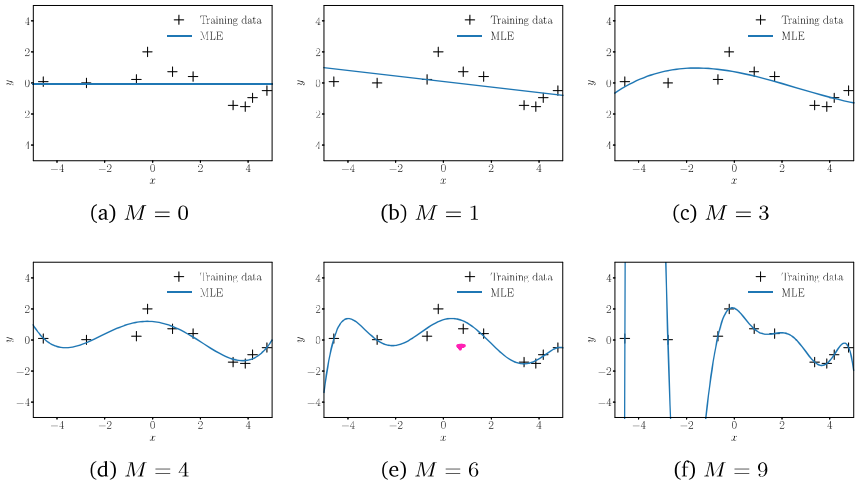
$$\text{RMSE} = \sqrt{\frac{1}{N} \|\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\theta}\|^2} = \sqrt{\frac{1}{N} \sum_{n=1}^{N} (y_n - \boldsymbol{\phi}^\top(\boldsymbol{x}_n)\boldsymbol{\theta})^2} \,, \tag{9.23}$$

which (a) allows us to compare errors of datasets with different sizes and (b) has the same scale and the same units as the observed function values y_n. For example, if we fit a model that maps post-codes ($\boldsymbol{x}$ is given in latitude, longitude) to house prices (y-values are EUR) then the RMSE is also measured in EUR, whereas the squared error is given in EUR^2. If we choose to include the factor σ^2 from the original negative log-likelihood (9.10b), then we end up with a unitless objective, i.e., in the preceding example, our objective would no longer be in EUR or EUR^2.

The RMSE is normalized.

The negative log-likelihood is unitless.

For model selection (see Section 8.6), we can use the RMSE (or the negative log-likelihood) to determine the best degree of the polynomial by finding the polynomial degree M that minimizes the objective. Given that the polynomial degree is a natural number, we can perform a brute-force search and enumerate all (reasonable) values of M. For a training set of size N it is sufficient to test $0 \leqslant M \leqslant N - 1$. For $M \leqslant N$, the maximum likelihood estimator is unique. For $M > N$, we have more parameters



(a) $M = 0$ (b) $M = 1$ (c) $M = 3$

(d) $M = 4$ (e) $M = 6$ (f) $M = 9$

Figure 9.5 Maximum likelihood fits for different polynomial degrees M.

than data points, and would need to solve an underdetermined system of linear equations ($\boldsymbol{\Phi}^\top\boldsymbol{\Phi}$ in (9.19) would also no longer be invertible) so that there are infinitely many possible maximum likelihood estimators.

Figure 9.5 shows a number of polynomial fits determined by maximum likelihood for the dataset from Figure 9.4(a) with $N = 10$ observations. We notice that polynomials of low degree (e.g., constants ($M = 0$) or linear ($M = 1$) fit the data poorly and, hence, are poor representations of the true underlying function. For degrees $M = 3, \ldots, 5$, the fits look plausible and smoothly interpolate the data. When we go to higher-degree polynomials, we notice that they fit the data better and better. In the extreme case of $M = N - 1 = 9$, the function will pass through every single data point. However, these high-degree polynomials oscillate wildly and are a poor representation of the underlying function that generated the data, such that we suffer from *overfitting*.

The case of $M = N - 1$ is extreme in the sense that otherwise the null space of the corresponding system of linear equations would be non-trivial, and we would have infinitely many optimal solutions to the linear regression problem.

overfitting

Remember that the goal is to achieve good generalization by making accurate predictions for new (unseen) data. We obtain some quantitative insight into the dependence of the generalization performance on the polynomial of degree M by considering a separate test set comprising 200 data points generated using exactly the same procedure used to generate the training set. As test inputs, we chose a linear grid of 200 points in the interval of $[-5, 5]$. For each choice of M, we evaluate the RMSE (9.23) for both the training data and the test data.

Looking now at the test error, which is a qualitive measure of the generalization properties of the corresponding polynomial, we notice that initially the test error decreases; see Figure 9.6 (orange). For fourth-order polynomials, the test error is relatively low and stays relatively constant up to degree 5. However, from degree 6 onward the test error increases significantly, and high-order polynomials have very bad generalization properties. In this particular example, this also is evident from the corresponding

Note that the noise variance $\sigma^2 > 0$.

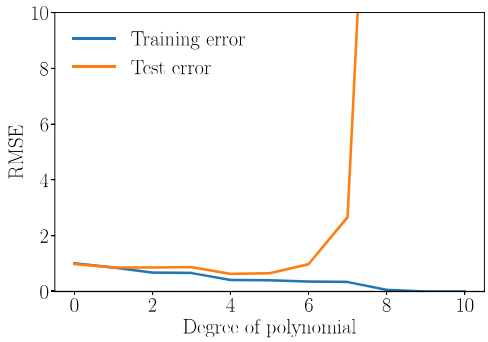


Figure 9.6 Training and test error.

training error

test error

maximum likelihood fits in Figure 9.5. Note that the *training error* (blue curve in Figure 9.6) never increases when the degree of the polynomial increases. In our example, the best generalization (the point of the smallest *test error*) is obtained for a polynomial of degree $M = 4$.

9.2.3 Maximum A Posteriori Estimation

We just saw that maximum likelihood estimation is prone to overfitting. We often observe that the magnitude of the parameter values becomes relatively large if we run into overfitting (Bishop, 2006).

To mitigate the effect of huge parameter values, we can place a prior distribution $p(\boldsymbol{\theta})$ on the parameters. The prior distribution explicitly encodes what parameter values are plausible (before having seen any data). For example, a Gaussian prior $p(\theta) = \mathcal{N}(0, 1)$ on a single parameter θ encodes that parameter values are expected lie in the interval $[-2, 2]$ (two standard deviations around the mean value). Once a dataset $\mathcal{X}$, $\mathcal{Y}$ is available, instead of maximizing the likelihood we seek parameters that maximize the posterior distribution $p(\boldsymbol{\theta} \mid \mathcal{X}, \mathcal{Y})$. This procedure is called *maximum a posteriori* (*MAP*) estimation.

maximum a posteriori

MAP

The posterior over the parameters $\boldsymbol{\theta}$, given the training data $\mathcal{X}, \mathcal{Y}$, is obtained by applying Bayes' theorem (Section 6.3) as

$$p(\boldsymbol{\theta} \mid \mathcal{X}, \mathcal{Y}) = \frac{p(\mathcal{Y} \mid \mathcal{X}, \boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{Y} \mid \mathcal{X})}. \tag{9.24}$$

Since the posterior explicitly depends on the parameter prior $p(\boldsymbol{\theta})$, the prior will have an effect on the parameter vector we find as the maximizer of the posterior. We will see this more explicitly in the following. The parameter vector $\boldsymbol{\theta}_{\text{MAP}}$ that maximizes the posterior (9.24) is the MAP estimate.

To find the MAP estimate, we follow steps that are similar in flavor to maximum likelihood estimation. We start with the log-transform and compute the log-posterior as

$$\log p(\boldsymbol{\theta} \mid \mathcal{X}, \mathcal{Y}) = \log p(\mathcal{Y} \mid \mathcal{X}, \boldsymbol{\theta}) + \log p(\boldsymbol{\theta}) + \text{const}, \tag{9.25}$$

where the constant comprises the terms that are independent of $\boldsymbol{\theta}$. We see that the log-posterior in (9.25) is the sum of the log-likelihood $p(\mathcal{Y} \mid \mathcal{X}, \boldsymbol{\theta})$ and the log-prior $\log p(\boldsymbol{\theta})$ so that the MAP estimate will be a "compromise" between the prior (our suggestion for plausible parameter values before observing data) and the data-dependent likelihood.

To find the MAP estimate $\boldsymbol{\theta}_{\text{MAP}}$, we minimize the negative log-posterior distribution with respect to $\boldsymbol{\theta}$, i.e., we solve

$$\boldsymbol{\theta}_{\text{MAP}} \in \arg\min_{\boldsymbol{\theta}} \{-\log p(\mathcal{Y} \mid \mathcal{X}, \boldsymbol{\theta}) - \log p(\boldsymbol{\theta})\} . \tag{9.26}$$

The gradient of the negative log-posterior with respect to $\boldsymbol{\theta}$ is

$$-\frac{\mathrm{d} \log p(\boldsymbol{\theta} \mid \mathcal{X}, \mathcal{Y})}{\mathrm{d}\boldsymbol{\theta}} = -\frac{\mathrm{d} \log p(\mathcal{Y} \mid \mathcal{X}, \boldsymbol{\theta})}{\mathrm{d}\boldsymbol{\theta}} - \frac{\mathrm{d} \log p(\boldsymbol{\theta})}{\mathrm{d}\boldsymbol{\theta}} , \tag{9.27}$$

where we identify the first term on the right-hand side as the gradient of the negative log-likelihood from (9.11c).

With a (conjugate) Gaussian prior $p(\boldsymbol{\theta}) = \mathcal{N}(\mathbf{0}, b^2\boldsymbol{I})$ on the parameters $\boldsymbol{\theta}$, the negative log-posterior for the linear regression setting (9.13), we obtain the negative log posterior

$$-\log p(\boldsymbol{\theta} \mid \mathcal{X}, \mathcal{Y}) = \frac{1}{2\sigma^2}(\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\theta})^\top(\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\theta}) + \frac{1}{2b^2}\boldsymbol{\theta}^\top\boldsymbol{\theta} + \text{const} . \tag{9.28}$$

Here, the first term corresponds to the contribution from the log-likelihood, and the second term originates from the log-prior. The gradient of the log-posterior with respect to the parameters $\boldsymbol{\theta}$ is then

$$-\frac{\mathrm{d} \log p(\boldsymbol{\theta} \mid \mathcal{X}, \mathcal{Y})}{\mathrm{d}\boldsymbol{\theta}} = \frac{1}{\sigma^2}(\boldsymbol{\theta}^\top\boldsymbol{\Phi}^\top\boldsymbol{\Phi} - \boldsymbol{y}^\top\boldsymbol{\Phi}) + \frac{1}{b^2}\boldsymbol{\theta}^\top . \tag{9.29}$$

We will find the MAP estimate $\boldsymbol{\theta}_{\text{MAP}}$ by setting this gradient to $\mathbf{0}^\top$ and solving for $\boldsymbol{\theta}_{\text{MAP}}$. We obtain

$$\frac{1}{\sigma^2}(\boldsymbol{\theta}^\top\boldsymbol{\Phi}^\top\boldsymbol{\Phi} - \boldsymbol{y}^\top\boldsymbol{\Phi}) + \frac{1}{b^2}\boldsymbol{\theta}^\top = \mathbf{0}^\top \tag{9.30a}$$

$$\iff \boldsymbol{\theta}^\top\left(\frac{1}{\sigma^2}\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \frac{1}{b^2}\boldsymbol{I}\right) - \frac{1}{\sigma^2}\boldsymbol{y}^\top\boldsymbol{\Phi} = \mathbf{0}^\top \tag{9.30b}$$

$$\iff \boldsymbol{\theta}^\top\left(\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \frac{\sigma^2}{b^2}\boldsymbol{I}\right) = \boldsymbol{y}^\top\boldsymbol{\Phi} \tag{9.30c}$$

$$\iff \boldsymbol{\theta}^\top = \boldsymbol{y}^\top\boldsymbol{\Phi}\left(\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \frac{\sigma^2}{b^2}\boldsymbol{I}\right)^{-1} \tag{9.30d}$$

so that the MAP estimate is (by transposing both sides of the last equality)

$$\boldsymbol{\theta}_{\text{MAP}} = \left(\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \frac{\sigma^2}{b^2}\boldsymbol{I}\right)^{-1}\boldsymbol{\Phi}^\top\boldsymbol{y} . \tag{9.31}$$

$\boldsymbol{\Phi}^\top\boldsymbol{\Phi}$ is symmetric, positive semi definite. The additional term in (9.31) is strictly positive definite so that the inverse exists.

Comparing the MAP estimate in (9.31) with the maximum likelihood estimate in (9.19), we see that the only difference between both solutions is the additional term $\frac{\sigma^2}{b^2}\boldsymbol{I}$ in the inverse matrix. This term ensures that

MAP ...

$$y_i \sim \mathcal{N}(\theta \cdot \phi(x), \sigma^2)$$

$$\theta \sim \mathcal{N}(0, b^2)$$

$$\mathcal{L}(\theta) = \underbrace{\frac{1}{2\sigma^2} \|y - \bar{\Phi}\theta\|_2^2}_{①} + \underbrace{\frac{1}{2b^2} \|\theta\|_2^2}_{②}$$

$\min_\theta \mathcal{L}(\theta)$ $\Leftrightarrow$ convex function

$$\nabla \mathcal{L}(\theta) = 0$$

$$\nabla \mathcal{L}(\theta) = \nabla ① + \nabla ②$$

$$\nabla ① = \frac{1}{\sigma^2}\left(\bar{\Phi}^T \bar{\Phi}\theta - \bar{\Phi}^T y\right)$$

$$\nabla ② = \frac{1}{b^2}\theta$$

$$\nabla \mathcal{L}(\theta) = 0$$

$$\boxed{\frac{1}{\sigma^2}\bar{\Phi}^T\bar{\Phi}\theta - \frac{1}{\sigma^2}\bar{\Phi}^T y + \frac{1}{b^2}\theta = 0}$$

to compute θ_{MAP}

linear system

$$\left(\frac{1}{\sigma^2}\bar{\Phi}^T\bar{\Phi} + \frac{1}{b^2}I\right)\theta = \frac{1}{\sigma^2}\bar{\Phi}^T y$$

matrix symmetric & positive definite

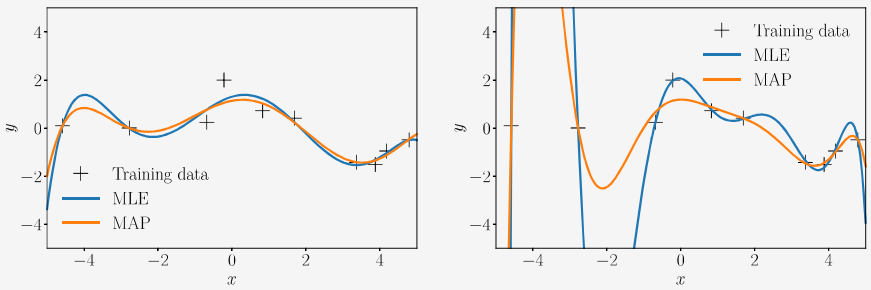
multiplying by $\frac{1}{\sigma^2}$:

$$\left(\bar{\Phi}^T\bar{\Phi} + \frac{\sigma^2}{b^2}I\right)\theta = \bar{\Phi}^T y$$

$\boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \frac{\sigma^2}{b^2}\boldsymbol{I}$ is symmetric and strictly positive definite (i.e., its inverse exists and the MAP estimate is the unique solution of a system of linear equations). Moreover, it reflects the impact of the regularizer.

Example 9.6 (MAP Estimation for Polynomial Regression)

In the polynomial regression example from Section 9.2.1, we place a Gaussian prior $p(\boldsymbol{\theta}) = \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ on the parameters $\boldsymbol{\theta}$ and determine the MAP estimates according to (9.31). In Figure 9.7, we show both the maximum likelihood and the MAP estimates for polynomials of degree 6 (left) and degree 8 (right). The prior (regularizer) does not play a significant role for the low-degree polynomial, but keeps the function relatively smooth for higher-degree polynomials. Although the MAP estimate can push the boundaries of overfitting, it is not a general solution to this problem, so we need a more principled approach to tackle overfitting.



(a) Polynomials of degree 6. (b) Polynomials of degree 8.

Figure 9.7 Polynomial regression: maximum likelihood and MAP estimates. (a) Polynomials of degree 6; (b) polynomials of degree 8.

9.2.4 MAP Estimation as Regularization

Instead of placing a prior distribution on the parameters $\boldsymbol{\theta}$, it is also possible to mitigate the effect of overfitting by penalizing the amplitude of the parameter by means of *regularization*. In *regularized least squares*, we consider the loss function

regularization
regularized least squares

$$\|\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\theta}\|^2 + \lambda \|\boldsymbol{\theta}\|_2^2 \,, \tag{9.32}$$

which we minimize with respect to $\boldsymbol{\theta}$ (see Section 8.2.3). Here, the first term is a *data-fit term* (also called *misfit term*), which is proportional to the negative log-likelihood; see (9.10b). The second term is called the *regularizer*, and the *regularization parameter* $\lambda \geqslant 0$ controls the "strictness" of the regularization.

data-fit term
misfit term
regularizer
regularization parameter

Remark. Instead of the Euclidean norm $\|\cdot\|_2$, we can choose any p-norm $\|\cdot\|_p$ in (9.32). In practice, smaller values for p lead to sparser solutions. Here, "sparse" means that many parameter values $\theta_d = 0$, which is also

useful for variable selection. For $p = 1$, the regularizer is called *LASSO* (least absolute shrinkage and selection operator) and was proposed by Tibshirani (1996). ♢

LASSO

The regularizer $\lambda \|\boldsymbol{\theta}\|_2^2$ in (9.32) can be interpreted as a negative log-Gaussian prior, which we use in MAP estimation; see (9.26). More specifically, with a Gaussian prior $p(\boldsymbol{\theta}) = \mathcal{N}(\mathbf{0}, b^2\boldsymbol{I})$, we obtain the negative log-Gaussian prior

$$-\log p(\boldsymbol{\theta}) = \frac{1}{2b^2}\|\boldsymbol{\theta}\|_2^2 + \text{const} \tag{9.33}$$

so that for $\lambda = \frac{1}{2b^2}$ the regularization term and the negative log-Gaussian prior are identical.

Given that the regularized least-squares loss function in (9.32) consists of terms that are closely related to the negative log-likelihood plus a negative log-prior, it is not surprising that, when we minimize this loss, we obtain a solution that closely resembles the MAP estimate in (9.31). More specifically, minimizing the regularized least-squares loss function yields

$$\boldsymbol{\theta}_{\text{RLS}} = (\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \lambda\boldsymbol{I})^{-1}\boldsymbol{\Phi}^\top\boldsymbol{y}\,, \tag{9.34}$$

which is identical to the MAP estimate in (9.31) for $\lambda = \frac{\sigma^2}{b^2}$, where σ^2 is the noise variance and b^2 the variance of the (isotropic) Gaussian prior $p(\boldsymbol{\theta}) = \mathcal{N}(\mathbf{0}, b^2\boldsymbol{I})$.

A point estimate is a single specific parameter value, unlike a distribution over plausible parameter settings.

So far, we have covered parameter estimation using maximum likelihood and MAP estimation where we found point estimates $\boldsymbol{\theta}^*$ that optimize an objective function (likelihood or posterior). We saw that both maximum likelihood and MAP estimation can lead to overfitting. In the next section, we will discuss Bayesian linear regression, where we use Bayesian inference (Section 8.4) to find a posterior distribution over the unknown parameters, which we subsequently use to make predictions. More specifically, for predictions we will average over all plausible sets of parameters instead of focusing on a point estimate.

STOP HERE

9.3 Bayesian Linear Regression

Previously, we looked at linear regression models where we estimated the model parameters $\boldsymbol{\theta}$, e.g., by means of maximum likelihood or MAP estimation. We discovered that MLE can lead to severe overfitting, in particular, in the small-data regime. MAP addresses this issue by placing a prior on the parameters that plays the role of a regularizer.

Bayesian linear regression

Bayesian linear regression pushes the idea of the parameter prior a step further and does not even attempt to compute a point estimate of the parameters, but instead the full posterior distribution over the parameters is taken into account when making predictions. This means we do not fit any parameters, but we compute a mean over all plausible parameters settings (according to the posterior).