Chapter 3
Uncertainty, Information, and Learning Mechanisms

The real-world is prone to uncertainty. We experience uncertainty in acquiring data, in interacting with the environment through an actuator, in representing information on a finite-precision machine, and in designing an unknown solution to a problem. The chapter formalizes and deals at first with the concept of uncertainty and the way it propagates through a computational flow. Afterwards, the basics of statistical learning are provided. It is shown how different sources of uncertainty, that depend on the chosen model family, the number of available data, and their quality and, ultimately, the complexity of the problem, are introduced when learning from data.

3.1 Uncertainty and Perturbations

3.1.1 From Errors to Perturbations

We have uncertainty whenever we have an approximated entity which, to some extent, estimates the ideal—possibly unknown—one. Such a situation can be formalized by introducing the ideal uncertainty-free entity and the real uncertainty-affected one and evaluating the error, i.e., the discrepancy between the two according to a suitable figure of merit. Since the error is strictly dependent on a specific pointwise instance, e.g., a representation error for a given value, a model error for a specific input, or a sensor error in correspondence of a particular data acquisition, we abstract the pointwise error with the concept of perturbation, a variable defined in a suitable domain with the pointwise error representing a particular realization of it.

In the following, a generic perturbation δA intervenes on the computation by modifying the status assumed by an entity from its nominal configuration A, whose domain and cardinality depends on the specific case, to a perturbed one A_p. The effect induced by the perturbation can be evaluated through a suitable figure of merit $\|A, A_p\|$ measuring the discrepancy between the two states. For instance, if we are looking at the output of a real sensor providing the constant scalar value $a \in \mathbb{R}$, then the discrepancy between the ideal nominal value and the perturbed one can be expressed as the punctual error $\|A, A_p\| = e = |a_p - a|$. Should we read another sensor instance, the pointwise error would assume a different value. In this case, the mechanism inducing uncertainty can be modeled with the signal plus noise model $a_p = a + \delta_a$ and $\|A, A_p\| = |a_p - a| = |\delta_a| = |e|$. It is evident from this example that δ_a can be described in many cases as a random variable with its probability density function fully characterizing the way uncertainty disrupt the information.

3.1.2 Perturbations

In Sect. 3.1.1 we have intuitively introduced the concept of perturbation as a random variable. More formally, the perturbation δA can be defined as a result of the perturbation operator applied to a structured variable A.

Given a generic variable $\psi \in \Psi \subset \mathbb{R}^d$, perturbation $\delta\psi$ moves ψ into a perturbed state ψ_p according to some perturbation model. Not rarely, we can model $\delta\psi$ as a multivariate random variable drawn from a perturbation probability density function $f_\psi(M, C_{\delta_\psi})$ characterized by mean M and covariance matrix C_{δ_ψ}. Ψ can either be discrete or continuous, the latter being the most common situation in signal/image processing.

Definition: Continuous Perturbations

We say that a perturbation $\delta\psi$ is continuous if $\Pr(\delta\psi = \delta\bar{\psi}) = 0, \forall\psi \in \Psi$. The definition tells us that the probability to sample a continuous perturbation space and get exactly a given perturbation is an event whose probability is null.

Definition: Acute Perturbations

We say that the square matrix A_p obtained by perturbing matrix A is acute (and the associate perturbation δA is said to be acute) if and only if

$$\lim_{A_p \to A} \operatorname{rank}(A_p) = \operatorname{rank}(A).$$

In other words, an acute perturbation does not change the rank of a matrix [51]. If perturbation δA is induced by perturbation $\delta\psi$, i.e., $\delta A = \delta A(\delta\psi)$, we also say that $\delta\psi$ is acute.

We will use the definitions above along the book and, in particular, in Chap. 5.

3.2 Perturbations at the Data Representation Level

Numerical data acquired by sensors and digitalized through an ADC are represented as a sequence of bits coded according to a given transformation which depends on the numerical information we need to represent. In the following sections, we will introduce the main transformations used in numerical representations as well as the types and characterization of uncertainty introduced when representing data in a digital format.

3.2.1 Natural Numbers $\mathbb{N}$: Binary Natural

Assume we are willing to spend n bits to represent a finite value $a \in \mathbb{N}$. It immediately comes out that we can represent only numbers belonging to a subset $\mathbb{N}(n) \subset \mathbb{N}$ given the finiteness of n. Since n bits provide 2^n independent codewords, the subset $\mathbb{N}(n)$ contains 2^n instances, e.g., the first 2^n symbols $\mathbb{N}(n) = 0, 1, \ldots, 2^n - 1$. For instance, if we have $n = 8$ we can represent the first 256 natural numbers, starting from 0 (or any other 256 numbers depending on the number of information-codeword association). The representation of values in $\mathbb{N}(n)$ follows the binary natural code representation [206], which can be easily derived once we comment that the numeric representation we are looking at is positional and weighted.

In this section we assume that the information associated with the natural number is not affected by uncertainty (the numeric instance is noise-free) and that the unique source of uncertainty is introduced by finite precision operators, such as truncation or rounding, used in order to reduce the number of bits associated with the information from n to its most significant $q \leq n$ bits.

3.2.1.1 Projection to a Subspace

Define the space of a representation as the space spanned by the vector containing as components the bits/digits considered to represent a value. If n are the bits, then $\mathbb{N}(n) = \{0, 1, 2, \ldots, 2^n - 1\}$ is the set of points in the space, with each point referenced by the generic vector in the form $[a_{n-1}, \ldots, a_1, a_0]$.

As such, an interesting projection to a lower dimensional space can be achieved by simply setting to zero the least significant $n - q$ bits of the n bits codeword associated with a (the least significant q bits are set to zero leading to value $a(q)$). The projection introduces an absolute error whose value is

$$e(q) = a - a(q) < 2^q.$$

The pointwise error is a function of the particular number instance and its stochastic characterization depends on the particular nature of the envisaged application, i.e., on the probability density function of the process generating value a. However, it is common to assume a uniform distribution for a. The projection operator introduces an absolute error that can be modeled as a uniformly distributed random variable defined in interval $[0, 2^q)$. The expected value of the error is $\frac{2^q-1}{2}$ and its variance is bounded by $\frac{2^{2(q-1)}}{3}$ [208].

3.2.1.2 Truncation

Truncation operates as a chopping operator that removes the least significant q bits from a n bits codeword. However, if truncation would simply mean chopping bits from a number, then it would not make any sense. For instance, if we consider the decimal value 123, truncation of the least significant digit would generate the number 12, per se a number not even related to the original one in terms of the absolute information content. The error would not even make sense in relative terms: for the case above, the relative error would be $\frac{123-12}{123}$. However, truncation is a key operator in embedded systems, for the reasons we will now explain.

The notation associated with the binary natural codeword is positional and weighted: value 1 in a_0 has a different meaning of a 1 in a_{n-1} (positional notation). In correspondence to bit a_i, we have a weight quantifying the information contribution carried by the bit, which is 2^i (weighted notation). What does make sense is to apply in turn the two steps

- Projection of the codeword in the subspace of dimension $n - q$;
- Apply the truncation operator so as to remove the q rightmost bits.

The final result of the transformation is that the number is now defined in an $n-q$ dimensional space. We save q bits in representing the information at the cost of an introduced source of uncertainty in the data representation (and a loss in information if the original number was uncertainty-free). Consider, for instance, decimal numbers 1234 and 2545, defined in an $n = 4$ dimensional space. We wish to reduce the space to $n - q = 2$ digits. By applying the projection to a subspace transformation, we get numbers 1200 and 2500 and, after truncation, the numbers become 12 and 25. In other words, we are keeping the most relevant part of the information content by operating into a two-dimensional subspace which somehow keeps the distance between the numbers at the net of the truncated information. The number can then be compared with other numbers defined in the same subspace.

The relative distance between the two codewords is mostly kept, although an error is introduced. In fact, by inspecting the numbers after the transformation, it is clear that they can be intended as generated with q right shifts, with the consequence that each instance of the reduced space should be weighted 10^2 to move back to the original one. The binary number $[a_{n-1}, a_{n-2} \dots , a_q, \dots , a_1, a_0]$ becomes $[a_{n-1}, a_{n-2} \dots , a_q]$ after the transformation. Each instance of the reduced space can be brought back to the original space dimension by multiplying it by value 2^q. The introduced absolute error in the original space is $e(q) = a - 2^q a(q) < 2^q$, hence inducing an error uniformly distributed in the interval $[0, 2^q)$.

3.2.1.3 Rounding

Rounding of a positive number truncates the q least significant bits and adds 1 to the unchopped part if and only if the most significant bit of the truncated segment is 1. Otherwise, the rounded value is the one defined over $n - q$ bits. In a binary natural representation, rounding provides a biased uniform error following the comments made for the projection to a subspace and the truncation operator. The advantage of rounding is that the variance $\frac{2^{2(q-2)}}{3}$ of $e(q)$ is half the truncation one.

3.2.2 Integer Numbers $\mathbb{Z}$: 2's Complement

3.2.3 2cp Notation

We are now interested in representing a value $a \in \mathbb{Z}(n) \subset \mathbb{Z}$ over n bits. A straight representation for the generic number a would be the sign and modulus notation. Such a notation is based on the fact that, although formally inaccurate, $Z = -\mathbb{N} \cup \mathbb{N}$. A generic number can then be represented with its sign (requesting a bit) and its modulus (which, being a natural number, can be represented with the binary natural representation). The sign and modulus representation is redundant, in the sense that it uses two codewords to represent the zero (-0 and $+0$) and requires differentiated hardware architectures to carry out additions and subtractions. A different approach, which is used in most embedded systems, is to use a two's complement notation (2cp) that solves both problems.

Given n bits, we have a total of 2^n available codewords, and we decide to assign half of them to represent negative numbers, and the remaining half to code positive numbers (zero included). That said, subset $\mathbb{Z}(n)$ becomes

$$\mathbb{Z}(n) = -2^{n-1}, \ldots, 0, \ldots, 2^{n-1} - 1.$$

The 2cp representation for number $a \in \mathbb{Z}(n)$ is defined as

$$a_{2cp} = \begin{cases} a_{b,n} & \text{for } a \geq 0 \\ (2^n - |a|)_{b,n} & \text{for } a < 0 \end{cases}$$

where subscript b, n stands for a binary natural representation on n bits. The transformation has remarkable properties that make the 2cp notation the one most used in embedded systems. Other expressions that can be derived from the above transformation are more immediate to generate 2cp codewords. One of these is particularly interesting since it exploits the concept of opposite $-a$ of number a. Having a generic number a_n obtained from a with a 2cp transformation over n bits, its opposite $-a_n$ is $-a_n = \bar{a}_n + 1$, where $\bar{a}$ is the bit-wise complement operator applied to the codeword a_n (1s and 0s are toggled in a_n). The immediate consequence is that the subtraction operation can be reduced to the addition one. In fact, given two numbers a_n, b_n in 2cp and defined on n bits, the subtraction $a_n - b_n$ becomes $a_n - b_n = a_n + (-b_n) = a_n + \bar{b}_n + 1$. Both addition and subtraction operators reduce to the algebraic sum, whose simple algorithm is that used for the addition operator.

In order to characterize the nature of the finite precision representation error, let us consider at first the truncation operator, chopping q bits from the n of the original representation. The limits of the truncation operator for $\mathbb{Z}$ are those presented for $\mathbb{N}$. Truncation should be intended as an operator transforming the n-dimensional space of the data into the reduced one of dimension $n - q$. Under this framework, the truncation error associated with truncated value $a(n - q)$ is always positive, assuming values $0 \leq a - 2^q a(n - q) < 2^q - 1$. The error introduced by the truncation operator is uniformly distributed in the interval $[0, 2^q - 1)$ and introduces a bias value. On the contrary, rounding introduces an unbiased error, a very welcome property in any computation: clearly, we would appreciate the outcome of a computation to be accurate, or, in the worst case, characterized by a small bias. Thus, rounding outperforms truncation in the 2cp representation. This makes it a very interesting operator for embedded systems despite the required extra computational cost. If rounding is applied, it can be shown that the representation error is uniformly distributed in interval $[-2^{q-1}, 2^{q-1})$.

3.2.4 Rational $\mathbb{Q}$ and Real $\mathbb{R}$ Numbers

As pointed out in the previous sections, the finiteness of the machine limits the number of codewords to 2^n if n is the number of available bits. As a consequence, we can only approximate a generic number a, either belonging to $\mathbb{Q}$ or $\mathbb{R}$, with the number $a(n)$ which, for its finite nature, belongs to $\mathbb{Q}$.

3.2.4.1 Fixed Point Representation

Any rational number $a \in \mathbb{Q}$ can be seen as composed of an integer part and a fractional one. A natural approximation $a(n)$ of a is a number where $l = n - k - 1$ bits are assigned to the integer part, one to the sign bit, and k to the fractional one. This notation is called fixed point since the "dot" separating the integer from the fractional part is conventionally fixed in the notation, being k bits leftwards from the least significant bit (note that the point is only virtual and such information is not stored). This said, we also note that the number $a(n)2^k$ is an integer number and, as such, it can be represented with a 2cp notation. De facto, there is no difference between a generic fixed point number and an integer one!

Example: Fixed Point Representation

Consider, as an example, decimal value $a = 1.56$ and say that we are willing to spend $n = 5$ bits to represent it in 2cp. We decide to use 2 bits for the fractional part ($k = 2$). The number can then be represented with the fixed point binary sequence [00110], e.g., codeword [001.10] associated with the approximated decimal number $a(n) = 1.5$. If we multiply the binary codeword by factor 2^k, the fractional point disappears and we obtain codeword [00110], which is associated to the binary value $a(n)2^2$. The introduced absolute error is $|e(q)| = |a - a(n)| = 0.06 < 2^{-2}$.

Let us consider now number a coded in 2cp over n bits with l bits associated with the non-fractional part (without including the sign bit). We wish to reduce the n bits to $n - q$ bits at first through a q least significant bits truncation (truncation might also affect the integer information, and not only the fractional one).

Multiply a by 2^{-l} so that $2^{-l}a$ becomes a totally fractional number. We have seen that if we keep k bits for the fractional part, then the introduced error is lower than 2^{-k}. Given the fact that we wish to keep $n - q$ bits and a bit is used for the sign, we have that the truncation error $e(q)$ is always positive (also for negative numbers) and satisfies the inequalities

$$0 \le e(q) < 2^l(2^{-(n-q-1)}).$$

As an interesting example, let us consider the decimal number 0.45 represented on $n = 5$ bits, $l = 0$. The 2cp representation becomes [00111]. We wish to represent the number on a smaller space by choosing $q = 2$. The obtained number after truncation is [001], e.g., decimal number 0.25. Since $l = 0$, we have that the representation error must satisfy $0 \le e(q) = 0.2 < 2^{-2} = 0.25$. If rounding is applied, then it can be shown that the error $e(q)$ satisfies

$$-2^l(2^{-(n-q)}) \le e(q) < 2^l(2^{-(n-q)}).$$

The error owed to rounding is independent of the binary representation and its mean is zero. Moreover, rounding introduces a lower variance compared to truncation.

Let us consider, as a second example, decimal number 6.9, to be represented in 2cp fixed point notation on $n = 7$ bits. The 2cp representation becomes [0110111]. We wish to represent the number on a smaller space by choosing $q = 2$ and rounding. The obtained number after rounding is [01110], e.g., decimal number 7. Since $l = 3$, $n = 7$, and $q = 2$, we have that the representation error $e(q) = 6.9 - 7$ has to be in magnitude smaller than 2^{-2}, as it is.

As a last example, consider decimal number -6.666 to be represented in a 2cp fixed point notation on $n = 7$ bits. The 2cp representation becomes [1001011]. We wish to represent the number on a smaller space by choosing $q = 1$ and rounding as space reduction technique. The codeword of the positive number is [0110101]; after rounding with $q = 1$ we get codeword [011011], to which the rounded negative codeword [100101] is associated. Since $l = 3$, $n = 7$, and $q = 1$, we have that the representation error $e(q) = -6.666 - (-6.75) = -0.084$ has to be of magnitude smaller than 2^{-3}, as it is.

Summarizing, in a 2cp notation, the error introduced by quantization is uniformly distributed [208] in the interval

$$[0, 2^{l-n+q+1})$$

for the truncation operator and uniformly distributed in the interval

$$[-2^{l-n+q}, 2^{l-n+q})$$

for the rounding operator. The above distributions should be used to test the effects of noise on the embedded computation as requested in Chap. 7 or to evaluate the intrinsicrobustness of the computational flow as detailed in Chap. 5.

3.3 Propagation of Uncertainty

We analyze in this section the way perturbations affecting sensor data propagate within a computational flow $y = f(x), x \in X \subset \mathbb{R}^d, y \in Y \subset \mathbb{R}$. The sensitivity analysis provides closed-form expressions for the linear function case and approximated results for the nonlinear one, provided that the perturbations affecting the inputs are small in magnitude compared to the inputs (small perturbation hypothesis). The analysis of *Perturbations in the large* i.e., perturbations of arbitrary magnitude, for the nonlinear case, cannot be obtained in a closed form unless $y = f(x)$ assumes a particular structure and has properties that make the mathematics amenable. The extended analysis dealing with the perturbation in the large framework will be presented in Chap. 7.

3.3.1 Linear Functions

Let us consider linear function $y = f(x) = \theta^T x$, where $\theta \in \Theta \subset \mathbb{R}^d$ and x are d-dimensional column vectors representing the parameters and the inputs of the linear function, respectively. In the sequel, the parameter vector θ is assumed to be constant and given, otherwise differently specified.

3.3.1.1 The Additive Perturbation Model

A perturbation δx affecting the inputs, say according to an additive signal plus noise perturbation model $x_p = x + \delta x$, generates the perturbed value $y_p = \theta^T x_p$. The pointwise error $\delta y = y_p - y$ can be rewritten, thanks to linearity, as

$$\delta y = \theta^T \delta x. \tag{3.1}$$

Note that the linear function, characterized by its parameter vector θ, is not structurally affected by perturbations, which only influence the function inputs. The (3.1) tells us that the propagated error at the function output is linear with the perturbation vector. The perturbation δx can be modeled as a random variable subject to pdf $f_{\delta x}(0, C_{\delta x})$, where $C_{\delta x}$ is the covariance matrix of the perturbation.

Characterization of the perturbation error δy, which also becomes a random variable, can be done by providing the mean and the standard deviation of the propagated error at the function output and then, where possible, its pdf. We have that

$$E_{\delta x}[\delta y] = E_{\delta x}[\theta^T \delta x] = \theta^T E_{\delta x}[\delta x] = 0$$

and

$$\text{Var}(\delta y) = E_{\delta x}[\theta^T \delta x \delta x^T \theta] = \theta^T E_{\delta x}[\delta x \delta x^T]\theta = \theta^T C_{\delta x}\theta = \text{trace}\left(\theta^T \theta C_{\delta x}\right).$$

Under the independence assumption for the perturbation affecting the inputs, $C_{\delta x}$ happens to be a diagonal matrix with the i-th entry characterized by variance $\sigma^2_{\delta x,i}$. Then, defining θ_i to be the i-th component of vector θ,

$$\text{Var}(\delta y) = \sum_{i=1}^{d} \theta_i^2 \sigma^2_{\delta x,i}.$$

In the particular case where all perturbations have the same variance $\sigma^2_{\delta x}$, e.g., perturbations are uniformly defined within the same bounded interval, the above expression becomes

$$\text{Var}(\delta y) = \sigma^2_{\delta x}\theta^T\theta. \tag{3.2}$$

The pdf of the propagated error cannot be evaluated a priori in a closed form unless we assume that the dimension d is large enough. In such a case, we can invoke the Central Limit Theorem (CLT) under the Lyapunov assumptions [35] and δy can be modeled as a random variable drawn from a Gaussian distribution.

CLT Under the Lyapunov Condition

Let $Y_i, i = 1 \ldots d$ a set of independent random variables characterized by finite expected value $E[Y_i]$ and variance $\text{Var}(Y_i)$. Denote $s_d^2 = \sum_{i=1}^d \text{Var}(Y_i)$ and $Y = \sum_i Y_i$. If there exists number $l > 0$ such that

$$\lim_{d\to\infty}\left(\frac{1}{s_d^{2+l}}\sum_{i=1}^{d} E\left[|Y_i - E[Y_i]|^{2+l}\right]\right) = 0,$$

then $Z = \frac{(Y-E[Y])}{\sqrt{Var(Y)}}$ converges to the standard normal distribution.

From the intuitive point of view, the CLT tells us that the sum of many, not-too-large, and not-too-correlated random terms, average out. The Lyapunov condition is one way for quantifying the not-too-large term request by inspecting the behavior on some $2 + l$ moments. In most cases, one tests the satisfaction of the condition for $l = 1$.

From the theorem, with the choice $Y_i = \theta_i \delta x_i$, δy can be approximated as a random variable drawn from Gaussian distribution $\delta y = \mathscr{N}(0, \sum_{i=1}^{d} \theta_i^2 \sigma_{\delta x,i}^2)$ provided that the Lyapunov condition holds. If all $\sigma_{\delta x,i}^2$ terms are identical to $\sigma_{\delta x}^2$, then $\delta y = \mathscr{N}(0, \sigma_{\delta x}^2 \theta^T \theta)$.

It is easy to show that the Lyapunov condition holds if each component of random variable δx is uniformly distributed within a given interval, as it happens in many application cases (think of the error distribution introduced by the rounding and truncation operators operating on binary 2cp codewords). Let us show it with an example that models the situation where all inputs of our embedded system are represented on the same number of bits and a 2cp notation is adopted. Rounding is the considered chopping operator for which we know that the induced error distribution is uniform and centered.

Example: The CLT Under the Lyapunov Condition

Let δx be an i.i.d random variable with each component uniformly defined in interval $[-1, 1]$ (if we consider the case of an embedded system, the error introduced by rounding is defined in such an interval). Assume that $\theta_m^2 \neq 0 \leq \theta_i^2 \leq \theta_M^2$, i.e., the generic parameter is bounded by the same minimum and maximum values. Let $\delta y = \theta^T \delta x$ and define $Y_i = \theta_i \delta x_i$. We have that $E[Y_i] = 0$ and variance $\mathrm{Var}[Y_i] = \frac{\theta_i^2}{3}$.

Denote

$$s_d^2 = \sum_{i=1}^{d} \mathrm{Var}[Y_i] = \frac{1}{3} \sum_{i=1}^{d} \theta_i^2.$$

Let us compute $E\left[|Y_i|^{2+l}\right]$ for $l = 2$

$$E\left[|Y_i|^4\right] = \theta_i^4 \int_0^1 \delta x_i^4 d\delta x_i = \frac{\theta_i^4}{5}$$

Since

$$\frac{1}{(s_d^2)^2} \sum_{i=1}^{d} E\left[|Y_i|^4\right] = \frac{1}{(s_d^2)^2} \sum_{i=1}^{d} \frac{\theta_i^4}{5}$$

and $\sum_{i=1}^{d} \frac{\theta_i^4}{5} \leq \frac{d}{5} \theta_M^4$ and $(s_d^2)^2 = \left(\frac{1}{3} \sum_{i=1}^{d} \theta_i^2\right)^2 \geq \frac{d^2 \theta_m^4}{9}$ we can bound

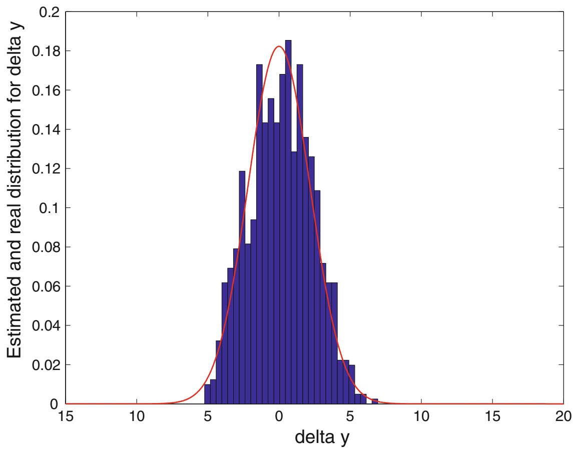

Fig. 3.1 The empirical distribution of δy compared with the set by the CLT. The dimension of the parameter space is $d = 5$

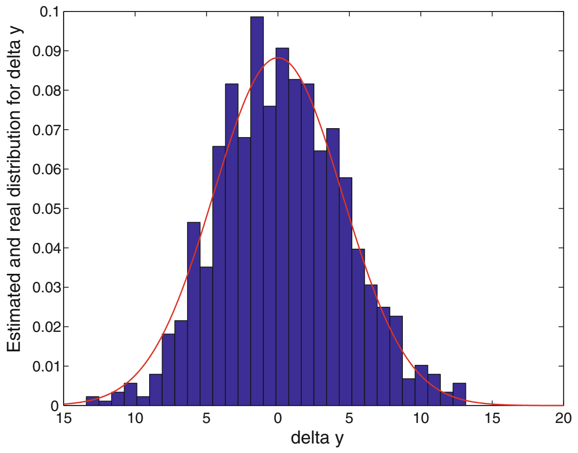

Fig. 3.2 The empirical distribution of δy compared with the set by the CLT. The dimension of the parameter space is $d = 15$

$$\left(\frac{1}{(s_d^2)^2} \sum_{i=1}^{d} E\left[|Y_i|^4 \right] \right) \leq \frac{9\theta_M^4}{5d\theta_m^4}$$

which scales as $O(\frac{1}{d})$ and, when $d \to \infty$, goes to zero, hence satisfying the Lyapunov condition. This grants that $\delta y = N(0, \sigma_{\delta x}^2 \theta^T \theta)$.

In the example above, a sufficiently large d, e.g., $d > 10$, proves to be a good approximation in many cases. Figures 3.1 and 3.2 present an example showing the quality of the approximation for $d = 5$ and $d = 15$, respectively. The application is configured so that $\theta_{d=5} = [2.47, -2.55, 0.52, 1.10, -0.50]$ and $\theta_{d=15} = [-2.81, 1.23, -2.65, -2.66, -1.99, -2.32, -1.50, 0.13, -1.48, -0.30, -1.67, -2.55, -2.89, 0.45, 2.47]$. 1000 δx vectors have been extracted from the $[-1, 1]^d$ hypercube according to the uniform distribution. The histogram for δy is plotted and contrasted with the Gaussian curve granted by the CLT ($\delta y = \mathcal{N}(0, \frac{1}{3}\theta^T\theta)$). We can observe that, also with a low d, the empirical distribution approximates well the Gaussian one.

3.3.1.2 The Multiplicative Perturbation Model

Within a multiplicative model δx affects inputs to yield perturbed value $x_p = x(1 + \delta x)$. The pointwise error $\delta y = y_p - y$ can be rewritten as

$$\delta y = \theta^T \, (x \circ \delta x)$$

where $\circ$ is the elementwise multiplication operator (multiplication is carried out at the component by component level). As done before, we are interested in characterizing the first two moments of the error distribution and assume that both inputs and perturbations, that are supposed to be independent, are drawn according to distributions $f_x(0, C_x)$ and $f_{\delta x}(0, C_{\delta x})$, respectively. We only require the covariance matrices C_x and $C_{\delta x}$ to be known (or that an estimate can be provided), but not the pdf. Inputs are zero centered only to ease the derivation (a zero mean subtraction can be introduced before carrying out the analysis). Expectation, now taken w.r.t. inputs and perturbations, leads to

$$E_{x,\delta x}[\delta y] = E_{x,\delta x}[\theta^T x \circ \delta x] = \theta^T E_x[x] \circ E_{\delta x}[\delta x] = 0$$

and variance

$$\mathrm{Var}(\delta y) = E_{x,\delta x}[\theta^T x x^T \circ \delta x \delta x^T \theta] = \theta^T C_x \circ C_{\delta x}\theta \tag{3.3}$$

$C_{\delta x}$ is diagonal under the assumption that perturbations affecting inputs are independent. If that is the case, the (3.3) becomes

$$\mathrm{Var}(\delta y) = \sum_{i=1}^{d} \theta_i^2 \sigma_{\delta x,i}^2 \sigma_{x,i}^2.$$

In the particular case that all the input variances are identical to σ_x^2 and perturbation variances to $\sigma_{\delta x}^2$, the variance at the output level simplifies as

$$\mathrm{Var}(\delta y) = \sigma_{\delta x}^2 \sigma_x^2 \theta^T \theta. \tag{3.4}$$

If we compare the variance of (3.4) with that given in (3.2), we see that the former, generated according to the multiplicative model, is equal to the latter (additive model) amplified by term σ_x^2. As with the additive model case, the error distribution can be approximated with a Gaussian one provided the Lyapunov condition is met.

3.3.2 Nonlinear Functions

Let now function $y = f(x)$ be at least twice differentiable w.r.t. x. Again, x is a column vector that, once affected by perturbation δx, assumes value x_p. By adopting the perturbation model, the effect of δx at the output δy is

$$\delta y = f(x_p) - f(x)$$

δy can be hardly described in a closed form unless strong hypotheses about the nature of function $f(\cdot)$ or the perturbation δx are assumed. Perturbation propagation analysis within a nonlinear function is carried out in the literature by assuming the small perturbation hypothesis, e.g., as done with the sensitivity analysis that studies the effects of perturbation affecting inputs on the function output, e.g., [129].

Although the small perturbation hypothesis might hold in several cases, it represents, in general, a strong assumption that needs to be weakened, e.g., as proposed in Chap. 5. However, the small perturbation assumption makes the mathematics amenable and we can expand $f(x_p) = f(x + \delta x)$ according to Taylor around x and stop the expansion at the quadratic term

$$f(x + \delta x) = f(x) + J(x)^T \delta x + \frac{1}{2}\delta x^T H(x)\delta x + \mathrm{o}(\delta x^T \delta x)$$

where $J(x) = \frac{\partial f(x)}{\partial x}$ is the gradient vector and $H(x) = \frac{\partial^2 f(x)}{\partial x^2}$ the Hessian matrix.

By discarding terms of order higher than two the perturbation propagated at the output takes the form

$$\delta y = J(x)^T \delta x + \frac{1}{2}\delta x^T H(x)\delta x. \tag{3.5}$$

Not much more can be said within a deterministic framework unless we introduce strong assumptions on $f(x)$ or δx. However, by moving to a stochastic framework, which considers x and δx mutually independent and identically distributed i.i.d random variables drawn from distributions $f_x(0, C_x)$ and $f_{\delta x}(0, C_{\delta x})$, respectively, the first two moments of the distribution of δy can be computed.

In fact, under the above assumptions and by taking expectation w.r.t. x and δx, the expected value of the perturbed output (3.5) becomes

$$E[\delta y] = \frac{1}{2}E[\delta x^T H(x)\delta x] = \frac{1}{2}\mathrm{trace}\left(E[H(x)\delta x \delta x^T]\right) = \frac{1}{2}\mathrm{trace}\,(E[H(x)]C_{\delta x}).$$

If the quasi-Newton approximation for the Hessian $H(x) = \frac{\partial f(x)}{\partial x}\frac{\partial f(x)^T}{\partial x}$ holds, then $H(x)$ is a semidefinite positive quadratic form and

$$E[\delta y] = \frac{1}{2}\,\mathrm{trace}\,(C_x C_{\delta x}). \tag{3.6}$$

From (3.6), each perturbation introduces an increase in $E[\delta y]$ if we consider the quadratic form expansion (a first-order approximation, obtained by solely maintaining the linear term, provides a null value). In order to compute $\text{Var}(\delta y)$, we consider only the first term of the expansion (the quadratic term does not allow us to advance the mathematics), which means that we only keep the linear approximation for function $f(x)$. Under the above assumptions and by taking expectation w.r.t. x and δx, the variance of the perturbed output becomes

$$\text{Var}(\delta y) = E\left[J(x)^T \delta x \delta x^T J(x)\right] = \text{trace}\left(E\left[J(x)J(x)^T\right] C_{\delta x}\right).$$

Obviously, if $f(x) = \theta^T x$ the derivation reduces to that of the linear function case.

3.4 Learning from Data and Uncertainty at the Model Level

This section studies the case where parameterized models are built from a series of noisy data. The use of a limited number of data to estimate the model, i.e., to determine an estimate of the optimal parameter configuration, introduces an extra source of uncertainty on the estimated parameters in addition to the noise (in previous sections, the parameters were given). In fact, given a different data set with the same cardinality, we will obtain a different parameter configuration with probability one, also in the linear model case. What happens when we select a non-optimal ("wrong") model to describe the data? Which is the relationship between the optimal parameter configuration, constrained by the selected model family, and the current one configured on a limited data set? Since the estimated parameter vector is a realization of a random variable centered on the optimal one, the model we obtain from the available data can be seen as a perturbed model induced by perturbations affecting the parameter vector. Which are then the effects of this perturbation on the performance of the model? This section aims at addressing the above aspects.

3.4.1 Basics of Learning: Inherent, Approximation, and Estimation Risks

Let $Z_N = \{(x_1, y_1), ..., (x_N, y_N)\}$ be the set composed of N (input-output) couples. The goal of machine learning is to build the simplest approximating model able to explain past Z_N data and future instances that will be provided by the data generating process.

Consider then the situation where the process generating the data (system model) is ruled by

$$y = g(x) + \eta, \tag{3.7}$$

where η is a noise term modeling the existing uncertainty affecting the unknown nonlinear function $g(x)$, if any. Once the generic data instance x_i is available, (3.7) provides value $y_i = g(x_i) + \eta_i$, η_i being a realization of the random variable η. In practical cases, the system for which we aim to create a model, by receiving input x_i, provides value y_i. We comment that both inputs and outputs are quantities measurable through sensors. The ultimate goal of learning is to build an approximation of $g(x)$ based on the information present in dataset Z_N through model family

$$f(\theta, x) \tag{3.8}$$

parameterized in the parameter vector $\theta \in \Theta \subset \mathbb{R}^p$. Selection of a suitable family of models $f(\theta, x)$ can be driven by some a priori available information about the system model. If data are likely to be generated by a linear model—or a linear model suffices—then this type of model should be considered. In this case, learning relies on vast results provided by the system identification theory, e.g., see [130]. The outcome of the the learning procedure is the parameter configuration $\hat{\theta}$ and, hence, model $f(\hat{\theta}, x)$, whose quality/accuracy must be assessed.

If the accuracy performance is not met, and margin for improvement exists, we have to select a new model family and reiterate the learning process. For instance, if the difference—residual— between the reconstructed value $f(\hat{\theta}, x)$ and the measured $y(x)$ one on a new data set in not a white noise (test procedure), then there is information that model $f(\hat{\theta}, x)$ was not able to capture. A new richer model family should be chosen and learning restarted. In this direction, feedforward neural networks have been shown to be universal function approximators [131], i.e., can approximate any nonlinear function, and are ideal candidates to solve the above learning problem [39]. However, the complexity of the neural model family must be balanced with the information content provided by the data, otherwise we might experience poor approximating accuracy. Such performance loss is either associated with overfitting (the degrees of freedom exposed by the family model are overdimensioned compared to the effective needs, so that noise affecting the data instances is learned as well) or underfitting (the model is underdimensioned w.r.t. the available data and the model cannot extract all the information present in the data).

In the sequel, we present the classic learning from data mechanism based on the statistical formulation set by Vapnik [132–134].

Define as *Structural risk* the function

$$\bar{V}(\theta) = \int L\left(y, f(\theta, x)\right) p_{x,y} dxy \tag{3.9}$$

where $L\left(y, f(\theta, x)\right)$ is a discrepancy loss function evaluating the closeness between $g(x)$ and $f(\theta, x)$ and $p_{x,y}$ is the probability density function associated with the i.i.d. (x, y) random variable vector. The structural risk (3.9) assesses the accuracy of a given model according to the loss function $L\left(y, f(\theta, x)\right)$.

The optimal parameter θ^o yielding the optimal model $f(\theta^o, x)$ constrained by the particular choice of the model family $f(\theta, x)$, is

$$\theta^{\mathrm{o}} = \arg\min_{\theta\in\Theta} \bar{V}(\theta).$$

However, we do not have access to $p_{x,y}$ and only the data set Z_N is available. Such an information allows us to construct the empirical distribution

$$\hat{p}_{x,y} = \frac{1}{N}\sum_{i=1}^{N} D_\delta(x - x_i, y - y_i) \tag{3.10}$$

where $D_\delta(x - x_i, y - y_i)$ is the Dirac function. The use of the estimate $\hat{p}_{x,y}$ of (3.10) in (3.9) leads to the *Empirical Risk*

$$V_N(\theta) = \frac{1}{N}\sum_{i=1}^{N} L(y_i, f(\theta, x_i)). \tag{3.11}$$

Finally, minimization of the empirical risk provides the estimate $\hat{\theta}$

$$\hat{\theta} = \arg\min_{\theta\in\Theta} V_N(\theta) \tag{3.12}$$

and, in turn, the model $f(\hat{\theta}, x)$ approximating $g(x)$ whose accuracy performance is $\bar{V}(\hat{\theta})$. Minimization of the empirical risk defined in (3.12) is also called the *learning process* and the minimization procedure *learning algorithm.*

Conditions granting $\hat{\theta}$ to converge to θ^{o} as well as observations regarding the speed of convergence will be given later in the chapter. Here, we introduce at first the concepts of *inherent risk, approximation risk*, and *estimation risk* relevant to subsequent analyses.

Define $V_I = \bar{V}(\theta^{\mathrm{o}})|_{g(x)=f(\theta^{\mathrm{o}},x)}$ to be the inherent risk, i.e., the a priori non-null intrinsic risk we have when unknown function $g(x)$ belongs to the chosen model family, i.e., $g(x) = f(\theta^{\mathrm{o}}, x)$. Rewrite the structural risk $\bar{V}(\hat{\theta})$ associated with model $f(\hat{\theta}, x)$, i.e., the performance of the obtained model, as

$$\bar{V}(\hat{\theta}) = \left(\bar{V}(\hat{\theta}) - \bar{V}(\theta^{\mathrm{o}})\right) + \left(\bar{V}(\theta^{\mathrm{o}}) - V_I\right) + V_I. \tag{3.13}$$

The risk associated with the model is composed of three terms

- The inherent risk V_I. The risk depends only on the structure of the learning problem and, for this reason, can be improved only by improving the problem itself, i.e., by acting on the process generating the data, (e.g., by designing a more precise sensor architecture). Nothing else can be done. This is the minimum risk we can have and we reach it—implying optimal accuracy performance in function approximation—when the other sources of uncertainty leading to the two other risks are null;
- The approximation risk $\bar{V}(\theta^{\mathrm{o}}) - V_I$. The risk depends on how close the model family (also named hypothesis space) is to the process generating the data. To

improve it we need to select model families that are more and more expressive, i.e., either contain or are very close to $g(x)$ according to the figure of merit $L(\cdot, \cdot)$. Given an unknown $g(x)$ function, we need to select families of approximating functions that are universal function approximator, e.g., feedforward neural networks.

- The estimation risk $\bar{V}(\hat{\theta}) - \bar{V}(\theta^{\circ})$. The risk depends on the ability of the learning algorithm to select a parameter vector $\hat{\theta}$ close to θ°. If we have an effective learning process, we hope to be able to get a $\hat{\theta}$ close to θ° so that the contribution to the model risk is negligible.

The theory allows us to understand the intrinsic limits of learning. A learning problem is affected by three sources of error; of these, the inherent one is determined by the nature of the problem and, for this reason, cannot be improved by learning. The remaining error sources, i.e., those introduced by the approximation and the estimation processes, are the true target of any learning procedure.

Asymptotically with the number of available data N, the approximation and the estimation errors can both be controlled if the learning method has some basic consistency features (which most practical methods have). But when the available data set is small, the dominating component of the learning error is determined if the method is consistent by the approximation error, i.e., by how well the model family $f(\theta, x)$ is close to the process generating the data $g(x)$. In other words, the model risk is mainly determined by the choice of the approximating function $f(\theta, x)$, rather than by the training procedure. As a consequence, in the absence of a priori information, we have no basis to prefer a consistent learning method to another one. Further details applied to the particular case where the learning problem is of classification type can be found in [135].

Example: Inherent, Approximation, and Estimation Risks

Consider a quadratic loss function $L(y, f(\theta, x)) = (y - f(\theta, x))^2$ and a process generating the data ruled by $g(x) = x, x \in [0, 1]$ affected by a Gaussian noise so that $\eta = \mathcal{N}(0, \sigma_\eta^2)$

$$y = x + \eta.$$

Consider the function family $f(\theta, x) = k, \theta = [k]$. The structural risk becomes

$$\bar{V}(\theta) = \int (x + \eta - k)^2 \frac{1}{\sqrt{2\pi}} e^{\frac{-\eta^2}{2}} dx d\eta \tag{3.14}$$

that, after some calculus, leads to

$$\bar{V}(\theta) = \frac{1}{3} + \sigma_\eta^2 + k^2 - k. \tag{3.15}$$

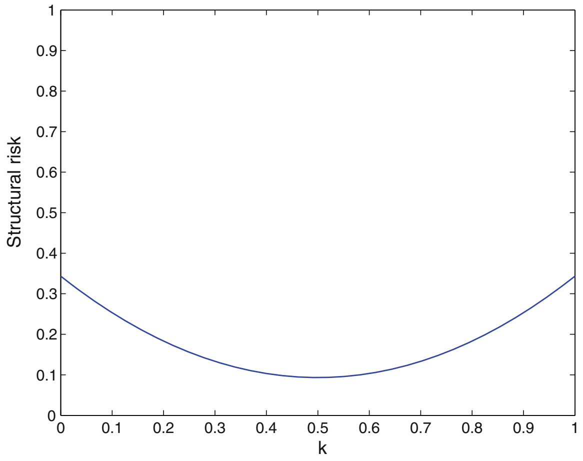

Fig. 3.3 The structural risk as function of k for the example. The structural risk presents a unique minimum θ^o in correspondence with $k = \frac{1}{2}$

Figure 3.3 presents the structural risk as function of k for the case $\sigma_\eta^2 = 0.01$. We see that the curve is characterized by a unique minimum θ^o.

The optimal point

$$\theta^o = \arg\min_{\theta \in \Theta} \bar{V}(\theta)$$

can be obtained by imposing the stationary relationship $\frac{\partial \bar{V}(\theta)}{\partial \theta} = 0$ and leads to $\theta^o = [\frac{1}{2}]$. Assume that the learning procedure has provided value $\hat{\theta} = [\frac{1}{4}]$. The learning situation is that of Fig. 3.4, where we have some points generated by the system model $y = x + \eta$, the optimal model minimizing the structural risk $y = \frac{1}{2}$, and the available one $y = \frac{1}{4}$.

It is now easy to derive from (3.13)

- The inherent risk $V_I = \bar{V}(\theta^o)|_{g(x)=\frac{1}{2}} = \sigma_\eta^2$;
- The approximation risk $\bar{V}(\theta^o) - V_I = \frac{1}{12}$;
- The estimation risk $\bar{V}(\hat{\theta}) - \bar{V}(\theta^o) = \frac{3}{48}$.

By adding the three risks we obtain the accuracy performance $\bar{V}(\hat{\theta}) = \sigma_\eta^2 + \frac{7}{48}$ of the available model, in line with what is expected by using (3.15) evaluated for $\hat{\theta} = [\frac{1}{4}]$.

3.4.2 *The Bias-Variance Tradeoff*

Following the previous section, consider now the case where we aim at determining the *expected prediction error* at a given—fixed—input value x for a quadratic loss

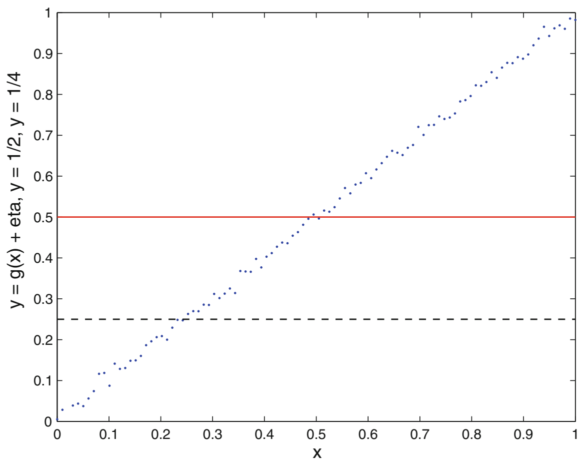

Fig. 3.4 The key elements of the learning process. The process generating the data $y = x + \eta$, the optimal model minimizing the structural risk $y = f(\theta^o, x) = \frac{1}{2}$, and the one provided by the learning procedure $y = f(\hat{\theta}, x) = \frac{1}{4}$

function Squared Error (SE), namely the error we should expect in test on an unknown instance x. The trained model $f(\hat{\theta}, x)$ has been derived by following the minimization of the empirical procedure (3.12). In the following, expectation is taken with respect to the noise on a *given* sample x, namely

$$SE_{PE}(x) = E\left[y(x) - f(\hat{\theta}, x)\right]^2. \tag{3.16}$$

We present a main result known in the literature as the bias-variance tradeoff. The SE can be seen as decomposed in the intrinsic error level, that introduced by the model or approximation error, and that introduced by the estimation procedure. Let us elaborate (3.16) as

$$\begin{aligned} SE_{PE} &= E\left[y(x) - f(\hat{\theta}, x)\right]^2 \\ &= E\left[y(x) - g(x) + g(x) - f(\hat{\theta}, x)\right]^2 \\ &= E\left[(y(x) - g(x))^2\right] + E[(g(x) - f(\hat{\theta}, x))^2] \\ &\quad + 2E\left[(y(x) - g(x))\,(g(x) - f(\hat{\theta}, x))\right] \\ &= E\left[\eta^2\right] + E\left[\left(g(x) - f(\hat{\theta}, x)\right)^2\right] \end{aligned}$$

since $E\left[(y(x)-g(x))\,(g(x)-f(\hat{\theta},x))\right]=0$. In fact, we can rewrite the term as

$$E\left[y(x)g(x)\right]+E\left[y(x)f(\hat{\theta},x)\right]-E\left[g(x)g(x)\right]+E\left[g(x)f(\hat{\theta},x)\right]$$

and

$$\begin{aligned}
E\left[y(x)g(x)\right]&=g^2(x)\\
E\left[y(x)f(\hat{\theta},x)\right]&=E\left[(g(x)+\eta)\,f(\hat{\theta},x)\right]=E\left[g(x)f(\hat{\theta},x)\right]\\
E\left[g(x)g(x)\right]&=g^2(x).
\end{aligned}$$

Thus, the SE can be decomposed in the variance of the noise and the SE between the true function and the estimate

$$E\left[\text{SE}\right]=\sigma_\eta^2+E\left[\left(g(x)-f(\hat{\theta},x)\right)^2\right]. \tag{3.17}$$

The second term of Eq. (3.17) can be further refined by using the same trick used above, which requires adding and subtracting $E\left[f(\hat{\theta},x)\right]$

$$\begin{aligned}
&E\left[\left(g(x)-E\left[f(\hat{\theta},x)\right]+E\left[f(\hat{\theta},x)\right]+f(\hat{\theta},x)\right)^2\right]\\
&\quad=E\left[\left(g(x)-E\left[f(\hat{\theta},x)\right]\right)^2\right]+E\left[\left(E\left[f(\hat{\theta},x)\right]-f(\hat{\theta},x)\right)^2\right]\\
&\qquad+2E\left[\left(g(x)-E\left[f(\hat{\theta},x)\right]\right)\left(E\left[f(\hat{\theta},x)\right]-f(\hat{\theta},x)\right)\right]
\end{aligned}$$

The double product cancels since

$$\begin{aligned}
E\left[g(x)E\left[f(\hat{\theta},x)\right]\right]&=g(x)E\left[f(\hat{\theta},x)\right]\\
E\left[g(x)f(\hat{\theta},x)\right]&=g(x)E\left[f(\hat{\theta},x)\right]\\
E\left[E\left[f(\hat{\theta},x)\right]^2\right]&=E\left[f(\hat{\theta},x)\right]^2\\
E\left[f(\hat{\theta},x)E\left[f(\hat{\theta},x)\right]\right]&=E\left[f(\hat{\theta},x)\right]^2
\end{aligned}$$

Equation (3.17) can be finally rewritten as

$$\text{SE}_{\text{PE}}=\sigma_\eta^2+E\left[\left(g(x)-E\left[f(\hat{\theta},x)\right]\right)^2\right]+E\left[\left(E\left[f(\hat{\theta},x)\right]-f(\hat{\theta},x)\right)^2\right]. \tag{3.18}$$

Equation (3.18) states that the accuracy performance of the approximating function $f(\hat{\theta}, x)$ can be described by means of three terms. The first one is the variance of the intrinsic noise and cannot be canceled, independently of how good our approximation is. The second term is the square of the bias and represents the quadratic error we have in approximating the true function $g(x)$ when our model generation procedure is able to provide the best model of the model family $f(\theta^{o}, x)$ (recall again that the best model is the one that minimizes the distance between the true function and the optimal approximation built in a noise-free environment having an infinite number of training points). De facto, the bias represents a discrepancy between the two functions according to the $\mathrm{SE}_{\mathrm{PE}}$ figure of merit. The last term is known as variance and it represents the variance introduced by having considered the approximating model $f(\hat{\theta}, x)$ instead of the optimal one within the family $f(\theta^{o}, x)$. Of course, if our model generating process is capable of providing a more accurate model, the variance term reduces.

Comments

Recall that the SE is the integral of the quadratic pointwise discrepancy only in the case that the distribution of the inputs is uniform. When this is not the case, the quadratic discrepancy is weighted by the pdf f_x to differentiate the interest of the error toward more likely inputs. For instance, if we consider interval $X = [0, 1]$, $g(x) = x^2$ and the approximating function $f(\hat{\theta}, x) = x$ then the quadratic discrepancy is SE $= 1/30$. Differently, if we induce the probability density function $f_x = 2$ if $x \leq 0.25$, $f_x = 2/3$ if $x > 0.25$, then the SE $\simeq 0.027$. Results related to convergence of the Mean Squared Error (MSE) to the SE are given in Chap. 4.

From Eq. (3.18) we observe that, in order to minimize the $\mathrm{SE}_{\mathrm{PE}}$, we need to minimize both the bias and the variance terms over the input space. However, this is not trivial. For instance, if the model family is rich in terms of degrees of freedom (say overdimensioned w.r.t. the problem) then $f(\hat{\theta}, x)$ would perfectly interpolate the training data. This will make the bias term vanish entirely at the cost of a high variance. More in general, finding an optimal bias-variance tradeoff is a difficult task but acceptable solutions can be found, e.g., by relying also on early stopping techniques, by cross-validation methods, or by introducing regularisers in the learning process [39].

3.4.3 Nonlinear Regression

The regression problem is a particular case of learning and aims at determining the best static model approximating an unknown static function. The framework assumes that there exists a time invariant model generating the couples (x_i, y_i) populating the data set Z_N. The learning framework is that of the empirical/structural risks presented in previous sections, here presented in the asymptotic formulation to give the reader a different prospective on how the learning framework can also be formalized.

Define the structural risk in the form

$$\bar{V}_N(\theta) = \frac{1}{N}\sum_{i=1}^{N} E\left[L(\varepsilon_i(\theta))\right]$$

and the empirical risk

$$V_N(\theta) = \frac{1}{N}\sum_{i=1}^{N} L(\varepsilon_i(\theta))$$

where $\varepsilon_i(\theta) = y_i - f(\theta, x_i)$ is the prediction error at sample (x_i, y_i). The optimal parameter point is defined as

$$\theta^{\mathrm{o}} = \arg\min_{\theta\in\Theta}\left[\lim_{N\to+\infty}\bar{V}_N(\theta)\right]$$

and estimated by

$$\hat{\theta} = \arg\min_{\theta\in\Theta} V_N(\theta).$$

The following analysis holds around the optimal point θ^{o} which minimizes the structural risk (note that, under the regularity hypothesis, $\bar{V}_N \to \bar{V}$ when $N \to \infty$). If local minima exist and we find ourselves in one of these, then we require that there exists a neighborhood for which the optimal point is unique. The analysis confines us within this neighborhood.

As with [137], we require that the approximating function $f(\theta, x)$ is Lipschitz and that the partial derivatives up to the third order w.r.t θ and ε are bound by a constant (regularity conditions). Under these hypotheses, $\hat{\theta}$ converges in probability to θ^{o} when $N \to \infty$ and the distribution of the parameter vector follows a multivariate Gaussian distribution

$$\lim_{N\to\infty}\sqrt{N}\Sigma_N^{-\frac{1}{2}}(\hat{\theta} - \theta^{\mathrm{o}}) \sim \mathscr{N}(0, I_p) \tag{3.19}$$

where, the Hessian of $\bar{V}_N$ is defined as $\bar{V}_N''$

$$\Sigma_N = \left[\bar{V}_N''(\theta^{\mathrm{o}})\right]^{-1} U_N \left[\bar{V}_N''(\theta^{\mathrm{o}})\right]^{-1},$$

and the squared matrix U_N of order d is

$$U_N = NE\left[\left(\frac{\partial V_N(\theta)}{\partial\theta}\right)\left(\frac{\partial V_N(\theta)}{\partial\theta}\right)^T\right].$$

I_p is the identity matrix of order p.

Comments

The framework designed for the nonlinear case is general but nonlinearity does not allow us to obtain closed-form results unless particular assumptions are made. In nonlinear regression, we identify a suitable nonlinear model family $f(\theta, x)$ characterized by enough expressive power to keep as small as possible the approximation risk. This can be achieved by resorting to universal function approximators, e.g., feedforward neural networks or radial basis functions [39, 131] to be used as $f(\theta, x)$. Then, we have to control the estimation error for the chosen neural network, an operation that can be carried out by selecting an effective learning algorithm, such as a second-order Levenberg–Marquardt, a DFP, or a BFGS one, to be applied to the empirical risk (see [125] for a comprehensive treatment). We might even need to run the algorithm several times to mitigate the presence of local minima in the V_N function.

Once training is perfected, the validity of the above must be intended in the neighborhood of the found local minimum that, under the regularity assumptions, is unique. However, it is thought that such univocity for the minimum does not exist for overdimensioned neural networks where the minimum provided by the learning procedure might be a saddle point. It should also be emphasized that if we run again the learning algorithm we will end up in a different minimum with probability one. This is a consequence of the complexity of the parameter space and the random selection for the initial weights. The final minimum also depends on the particular choice of Z_N. Although the derivation might seem to have a small impact in real applications, it is relevant when the approximating function is linear, either static or dynamic, as presented in the next two sections.

3.4.4 Linear Regression

The linear regression case is a particularly relevant case of nonlinear regression, where the system generating the data is linear

$$y = g(x) + \eta = \theta^{\mathrm{o}\,T} x + \eta \tag{3.20}$$

with θ^{o} being an unknown parameter vector to be estimated. $\eta \sim \mathcal{N}(0, \sigma_\eta^2)$ is a white noise of σ_η^2 variance. The Gaussian request for the noise is amply satisfied whenever data come from sensors. In other words, we assume that scattered data coming from the sensor (or otherwise available) can be optimally described by a linear model. However, only the data set Z_N is available and we wish to provide, starting from the finite data set, an estimate $\hat{\theta}$ of θ^{o}.

We choose the linear model family $f(\theta, x) = \theta^T x$ and the loss function to be an SE. All requested hypotheses leading to (3.19) are satisfied and a unique minimum θ^{o} is achieved if inputs are linearly independent (otherwise we simply have to remove the dependent inputs).

$\bar{V}_N$ and V_N are chosen as

$$\bar{V}_N(\theta) = \frac{1}{2N}\sum_{i=1}^{N} E\left[\varepsilon_i(\theta)^2\right]$$

$$V_N(\theta) = \frac{1}{2N}\sum_{i=1}^{N} \varepsilon_i(\theta)^2.$$

Since

$$\left.\frac{\partial V_N(\theta)}{\partial\theta}\right|_{Z_N} = -\frac{1}{N}\sum_{i=1}^{N}\varepsilon_i(\theta)x_i,$$

by exploiting the independence between x and ε and recalling that $E[\varepsilon_i\varepsilon_j] = 0$ since η is an i.i.d. random variable

$$\begin{aligned} U_N(\theta) &= \frac{1}{N}E\left[\sum_{i=1}^{N}\varepsilon_i(\theta)x_i\sum_{j=1}^{N}\varepsilon_j(\theta)x_j^T\right] = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{N}E\left[\varepsilon_i(\theta)\varepsilon_j(\theta)x_ix_j^T\right] \\ &= \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{N}E\left[\varepsilon_i(\theta)\varepsilon_j(\theta)\right]E\left[x_ix_j^T\right] = \frac{1}{N}\sum_{i=1}^{N}E\left[\varepsilon_i(\theta)^2\right]x_ix_i^T \end{aligned}$$

while

$$\bar{V}_N'' = \frac{\partial^2\bar{V}_N(\theta)}{\partial\theta^2} = \frac{1}{N}\sum_{i=1}^{N}x_ix_i^T.$$

Since $E\left[\varepsilon_i(\theta)^2\right] = \sigma_\eta^2$

$$U_N(\theta) = \sigma_\eta^2\frac{1}{N}\sum_{i=1}^{N}x_ix_i^T = \sigma_\eta^2\bar{V}_N''(\theta).$$

The expression of Σ_N simplifies as

$$\begin{aligned} \Sigma_N &= \left[\bar{V}_N''(\theta^{\mathrm{o}})\right]^{-1}U_N(\theta^{\mathrm{o}})\left[\bar{V}_N''(\theta^{\mathrm{o}})\right]^{-1} \\ &= \left[\bar{V}_N''(\theta^{\mathrm{o}})\right]^{-1}\sigma_\eta^2\bar{V}_N''(\theta^{\mathrm{o}})\left[\bar{V}_N''(\theta^{\mathrm{o}})\right]^{-1} \\ &= \sigma_\eta^2\left[\bar{V}_N''(\theta^{\mathrm{o}})\right]^{-1} = \sigma_\eta^2\left[\bar{V}_N''\right]^{-1} \end{aligned}$$

Also, in the linear case the distribution of the parameters is Gaussian following (3.19), and reduces to

$$\lim_{N\to\infty}(\hat{\theta}-\theta^{o}) \sim \mathcal{N}(0, \frac{\sigma_\eta^2}{N}\left[\bar{V}_N''\right]^{-1}).$$

The covariance depends on the variance of the noise and the inverse of the input Hessian $\bar{V}_N'' = \frac{1}{N}\sum_{i=1}^{N} x_i x_i^T$.

Comments

We comment that linearity must be intended w.r.t. parameters θ. As such, any model family $f(\theta, x) = \theta^T\phi(x)$, with ϕ a generic function, can be used, assumptions are automatically valid, and the results hold. This comment has some relevance to machine learning, where the ϕ function can be intended as a transformation of the inputs (feature extraction).

3.4.5 Linear Time-Invariant Predictive Models

One of the most common applications we can consider given a datastream is the design of dynamic models able to approximate it over time. This can be carried out by different methods, e.g., by limiting the analysis to the most common linear techniques, by considering space–state descriptions or predictive input–output models. In the following, we will focus on predictive models since they will be used in Chap. 10. Assume that a physical description for the system model is unavailable and that the unknown dynamic system generating the data is time invariant.

Following the notation set up by Ljung [130], given a parameter vector $\theta \in \Theta$, the input and the output sequences $u^t = (u(1), \ldots, u(t)) \in \mathbb{R}^{t\times m}$, $u(\cdot) \in \mathbb{R}^m$ and $y^{t-1} = (y(1), \ldots, y(t-1)) \in \mathbb{R}^{t-1}$, respectively, the predictive model in a one-step prediction form for output $y(t)$, at time t is

$$\hat{y}(t,\theta) = f(\theta, u^t, y^{t-1}).$$

The prediction error for model $f(\theta, u^t, y^{t-1})$, at time t is

$$\varepsilon(\theta, u^t, y^{t-1}) = \varepsilon(t,\theta) = y(t) - \hat{y}(t,\theta).$$

The structural risk $\bar{V}_N(\theta)$ can be defined as

$$\bar{V}_N(\theta) = \frac{1}{N}\sum_{t=1}^{N} E\left[L(\theta, \varepsilon(t,\theta))\right]$$

where $L(\cdot,\cdot) \in \mathbb{R}$ is a suitable loss function and expectation is taken w.r.t. the distribution of u and y.

The optimal parameter configuration is the one minimizing the structural risk

$$\theta^{\mathrm{o}} = \arg\min_{\theta \in \Theta} \left[\lim_{N \to +\infty} \bar{V}_N(\theta) \right].$$

Following the procedure presented in Sect. 3.4.1 and given the input–output training sequence $Z_N = \{(u(t), y(t))\}_{t=1}^{N}$, the empirical risk becomes

$$V_N(\theta, u^t, y^{t-1}) = \frac{1}{N} \sum_{t=1}^{N} L(\theta, \varepsilon(t, \theta)).$$

Minimization of the empirical risk leads to the parameter configuration $\hat{\theta}_N$

$$\hat{\theta}_N = \arg\min_{\theta \in \Theta} V_N(\theta, u^t, y^{t-1}).$$

In the following, we assume that the obtained model has been suitably chosen and does not degenerate in the identification phase (otherwise, it must be scaled to a smaller model).

By relying on the theoretical framework developed in [130, 136, 137], under the mild hypotheses that recent past data suffice to generate accurate approximations of $u(t)$ and $y(t)$, that $f(\cdot)$ is three time differentiable w.r.t. θ and satisfies Lipschitz conditions, and that the structural risk is a convex function, minimization of $W_N(\theta)$ provides a unique point θ^{o}, it is true that:

$$\lim_{N \to \infty} V_N - \bar{V}_N \to 0 \quad \text{w.p. } 1$$

thus

$$\lim_{N \to \infty} \hat{\theta}_N \to \theta^{\mathrm{o}} \quad \text{w.p. } 1$$

and

$$\lim_{N \to \infty} \sqrt{N} \Sigma_N^{-\frac{1}{2}} (\hat{\theta}_N - \theta^{\mathrm{o}}) \sim \mathcal{N}(0, I_p)$$

where Σ_N is the covariance matrix

$$\Sigma_N = \left[W_N''(\theta^{\mathrm{o}}) \right]^{-1} U_N \left[W_N''(\theta^{\mathrm{o}}) \right]^{-1},$$
$$U_N = N E \left[V_N{}'(\theta^{\mathrm{o}}) V_N{}'(\theta^{\mathrm{o}})^T \right]$$

$W_N'' = \frac{\partial^2 W_N}{\partial \theta^2}$ is the Hessian matrix of W_N and $V_N{}' = \frac{\partial V_N}{\partial \theta}$ is the gradient of V_N.

The theorem assures that, given a sufficiently large N, $\hat{\theta}_N$ follows a multivariate Gaussian distribution with mean vector θ^{o} and covariance matrix $\frac{\Sigma_N}{N}$.

The theorem contemplates the situation where there exists model bias between the optimal model and the process generating the data. In fact, under the aforementioned hypotheses, a unique optimal point $\theta^o \in \Theta$ exists even in case of model bias.

The above equations grant that the tolerated perturbation space for parameters is ruled by a Gaussian distribution. Having an application we can compute an estimate of matrix Σ_N, hence knowing the uncertainty we should expect on parameters. Having the uncertainty distribution on $\hat{\theta}$ we can now estimate the expected uncertainty in accuracy, e.g., $\bar{V}(\hat{\theta})$ by using randomization techniques as explained in Chap. 4.

3.4.6 Uncertainty at the Application Level

As a final note we comment that, in addition to the different sources of uncertainty we might experience, and that were summarized in the previous sections, we also have uncertainty at the application level. This source of uncertainty derives from the fact that, often, we do not know the solution to our application and we design it by exploiting some a priori information, whenever available. The outcome is a numerical algorithm describing the application.

However, we might have considered another solution, better, worse, or equivalent to the one we have. Clearly, we will keep the best performing solution also satisfying some extra application constraints such as computational complexity, power/energy consumption, or memory requirements, to name the few. The application solution might be complex, since it might come from a partitioning approach where the application is partitioned in parts, each of which has to be solved with the most appropriate signal/image processing or computational intelligence tool.

However, by looking at the problem from a high-level perspective the key elements are the unknown ideal algorithm, optimal in its own sense $g(x)$ and the best solution we found $f(x)$, x representing the input vector feeding my application. The uncertainty at the application level can be evaluated by introducing a discrepancy $L(\cdot, \cdot)$ between the two functions and computing the functional

$$\bar{V}(f, g) = \int L\left(g(x), f(x)\right) p_x dx$$

where p_x is the pdf induced on the input space. Function $g(x)$ is unknown but, as an Oracle, it provides the noisy value $y = g(x, \eta)$ once queried with input x. η is an unknown noise affecting the y generation process.

Clearly, the application bias increases if function $f(x)$ has not been suitably selected and it badly approximates $g(x)$. All aspects related to approximation and inherent risks can be extended as well to this framework if we assume that function $f(\cdot)$ belongs to a functional space F, possibly, but not necessarily, parameterized.

The problem requires now to estimate $\bar{V}(f, g)$ with $\hat{V}(f, g)$ having only data set Z_N but being able to invoke the Oracle all the needed time. Chapter 7 will identify the optimal N so that $\hat{V}(f, g)$ will be a good estimate of $\bar{V}(f, g)$. Let us now assume that we have found a set of solutions $F_f = \{f_1(x), f_2(x), \ldots, f_n(x)\}$. Since it is hard to guarantee that solutions are i.i.d we can only select the best solution $\bar{f}(x)$ as the one minimizing the discrepancy function over set F_f

$$\bar{f}(x) = \arg \min_{f_i(x) \in F_f} \hat{V}(f, g).$$

Not much more than this can be done unless a priori information is made available by someone.