# 2

# Regularisation

## 2.1 Introduction

The purpose of this chapter is to carry on with the analysis started in chapter 1. In the previous chapter, we have linked the problem of learning to (parametric) regression estimation. The underlying inductive principle consists in minimising a cost functional, such as the quadratic cost, depending on the parameters and the data, so as to identify the parameters of the model.

In this chapter, we show that minimising the simple cost calculated on the data does not yield good result in real, noisy situations. It is shown that the learning problem presented as a simple empirical risk minimisation, is an ill-posed problem. The concepts of *well-posed* and *ill-posed* problems and the technique of regularisation are introduced in a general context. Afterwards, we link regularisation and noise injection on the input. We derive an original equivalence between stopped training and regularisation in section 2.14. We introduce afterwards the dichotomy between *formal* and *structural* regularisation techniques, in the context of non-linear parametric neural networks models. Another original contribution of this chapter is the analysis in section 2.16 of a regularisation technique that combines both aspects, an analysis that will be carried out further in chapter 6.

## 2.2 Stability of non-regularised solutions

At the end of chapter 1, we have displayed an example suggesting that the minimisation of the empirical cost could lead to inappropriate models. We can carry this idea further by focusing on the case of polynomial regression. For a number $N$ of points, there exists a polynomial of degree $N-1$ that approximates these points arbitrarily closely (Stone-Weierstrass theorem).

Let us consider a simple parabolic interpolation involving 3 points. These 3 points determine a unique second-order polynomial $y(x) = ax^2 + bx + c$, as long as no two

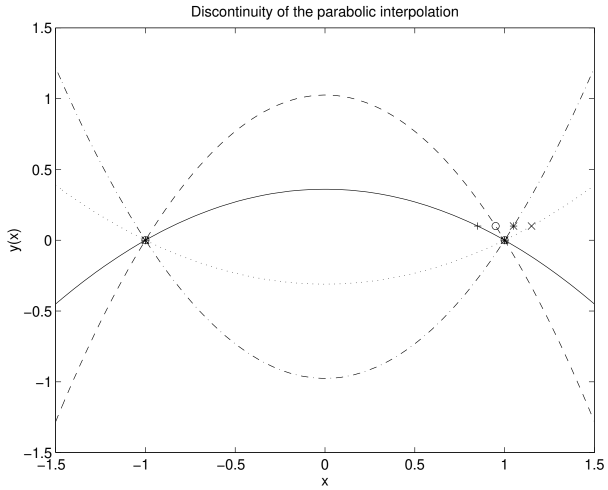


**Figure 2.1**: 4 different 3-points parabolic interpolations for four very close data sets display a discontinuity in the solution. Two points are held fixed, and very small variations in the third one provoke large variations of the solution.

abscissa are the same. To illustrate our point, we will keep the two first points fixed in $(-1;0)$ and $(1;0)$, respectively. The last point depends on a variable and is set to $(d;0.1)$, so that the solution is the simple symmetric parabola $y\left(x\right)=ax^2-a$, with $a=\frac{0.1}{d^2-1}$, for all values of $d \neq \pm 1$.

Figure 2.1 displays a couple of solutions corresponding to very close values in the data. A small discrepancy in the variable data point can lead to an arbitrarily large variation in the solution.

This artificial example shows that the learning process (polynomial regression in that case) is not necessarily continuous in the data. This can lead to obvious problems in the case where learning is applied to real, noisy data. In that case, noisy samples from the system will provide different data sets. However, the target function, i.e. the underlying process, is still the same so we would expect the learning procedure to yield similar, if not identical, results. On the other hand, a discontinuity in solution is the sign of a potentially large generalisation error.

The next section poses the problem in a more formal way, which leads to the definition of the concepts of *well-posed problems*, *ill-posed problems*, and *regularisation*.

## 2.3 Well-posed and ill-posed problems

The existence of ill-posed problems has been observed in the early 1900s by the French mathematician Hadamard. Let us consider a typical *inverse problem*. We wish to solve the equation:

$$\mathcal{A}f = F, \qquad F \in \mathcal{F} \tag{2.1}$$

where $\mathcal{A}$ is an operator and $F$ belongs to the metric space $\mathcal{F}$. $\mathcal{A}$ can be a linear as well as a non-linear operator. Typical examples include derivative or integral operators. For example, a system governed by a second order differential equation can be discretised and expressed as a linear equation $\mathbf{A}.f = F$, where $F$ is a set of discrete measurements and $\mathbf{A}$ is a known matrix representing the differential equation on $f$. E.g. the second derivative is expressed simply as a band diagonal matrix with $-2$ on the diagonal, and 1 on the upper and lower first bands.

In the context of parametric regression estimation, $F$ is the observed data, and $f \in \mathcal{W}$ is the unknown model, uniquely determined by a set of parameters. This is typical of inverse problems as we wish to invert the cause+system = effect. Knowing the cause (resp. the system) and the effect, we try to deduce the system (resp. the cause).

Hadamard noticed that in some cases, (2.1) is *ill-posed*: a small deviation of the right-hand side can result in a large deviation in the solution $f$. More precisely, let us define the concept of *Hadamard well-posedness*. (2.1) is a *well-posed* problem if the following conditions hold:

1. $\forall F \in \mathcal{F},\ \exists f \in \mathcal{W},\ \mathcal{A}f = F$. A solution to (2.1) exists.
2. $\forall (f_1, f_2) \in \mathcal{W}^2,\ \mathcal{A}f_1 = \mathcal{A}f_2 \Longrightarrow f_1 = f_2$. The solution is unique[1].
3. With $\mathcal{A}f = F$ and $\mathcal{A}\tilde{f} = \tilde{F}$, we have $\lim_{F \to \tilde{F}} f = \tilde{f}$. The solution is stable with small variations in the right-hand side of (2.1).

The third condition above is equivalent to writing that the inverse operator $\mathcal{A}^{-1}$ is continuous. In the context of this study, the inverse operator is the learning procedure. Simple learning procedures such as minimisation of the quadratic cost are *not* continuous as the example in section 2.2 shows, and the learning problem is thus ill-posed.

It is interesting to notice that Hadamard thought that ill-posed problems where restricted to mathematics and that real-life problems were well-posed. However, it was later found out that many actual inverse problems are ill-posed. This is true in a large number of fields, from mechanics to geophysics or statistics, as we will later show. A classical example for linear ill-posed problem is a Fredholm integral equation of the first kind:

$$\int_a^b K(x, u)\, f(u)\, du = F(x) \quad , a \leq x \leq b \tag{2.2}$$

where $K$ is a known squared integrable kernel, and $f$ is the sought solution.

## 2.4 More on ill-posed problems

The definition of Hadamard well-posedness does not accommodate a number of tasks such as parameter restoration. This called for an extension of the definition of ill-posed problem, that could only arise when the need to solve such problems appeared.

[1] $\forall (f_1, f_2) \in \mathcal{W}^2$ is equivalent to $\forall (F_1, F_2) \in \mathcal{F}^2$ according to the previous condition.

*Tikhonov well-posedness* restricts the definition in section 2.3 to a set $\mathcal{R} \subset \mathcal{W}$. The restriction made by Tikhonov is reflected in the following result: If the operator $\mathcal{A}$ is non-ambiguous and continuous on a compact set $\mathcal{R}$, then the inverse operator $\mathcal{A}^{-1}$ is continuous on the image $\mathcal{AR}$. Having a continuous operator with a continuous inverse on compact sets[2], the stability condition is guaranteed. Tikhonov well-posedness can be expressed by the following conditions: problem (2.1) is well posed if there exist a subset $\mathcal{R} \subset \mathcal{W}$, such that

1. It has a solution, $\forall F \in \mathcal{AR}\ \exists f \in \mathcal{R},\ \mathcal{A}f = F$.
2. The solution is unique $\forall (f_1, f_2) \in \mathcal{R}^2,\ \mathcal{A}f_1 = \mathcal{A}f_2 \Longrightarrow f_1 = f_2$.
3. For any sequence $f_i \in \mathcal{R}$ and $f \in \mathcal{R}$, such that $\lim_{i\to\infty} \mathcal{A}f_i = \mathcal{A}f$, we have $\lim_{i\to\infty} f_i = f$.

The last condition is especially interesting in the context of a learning procedure based on the minimisation of a given cost $C$. It means that is we have a series of models $f_i$ such that $\min C(f_i) \to \min C(f)$, then likewise $f_i \to f$. This is precisely what we were missing earlier. Indeed, the law of great numbers does guarantee the convergence of the empirical cost to the expected cost (section 1.2), but **only in the case where the problem is well-posed will the corresponding convergence be true for the *solution* of the minimisation problem**.

In the remainder of this chapter, we will set ourselves in the context of Tikhononv well-posedness. The original, ill-posed task is to solve the minimisation problem:

$$\widetilde{w} = \arg\min_{w} G(w) \tag{2.3}$$

As mentioned in the previous chapter, this task is unattainable, and we have to optimise an estimator of $G(w)$. The approximated problem is thus to minimise the empirical cost rather than the expected cost:

$$\widehat{w} = \arg\min_{w} S(w) \tag{2.4}$$

As will become clear in section 2.7, this minimisation is not a continuous function of the data. Small perturbations on the data can lead to high discrepancies in the estimated parameters $\widehat{w}$.

## 2.5 Tikhonov regularisation method

A main contribution of Tikhonov is that he proposes a *regularisation* method, i.e. a method turning an ill-posed problem into a close well-posed problem. The idea is to confine the problem to a restricted set by the use of a regularisation functional $R(f)$.

In the context of cost minimisation in parametric regression, this functional will typically depend on the parameters. Setting a constraint on this functional by $R(f) \leq c$

[2]The image of a compact set through a continuous operator is compact

defines a structure of subsets (defined via $c$) of the function set $\mathcal{W}$. In these conditions, the regularised minimisation problems we wish to solve are:

$$\widehat{w}_c = \arg\min_{R(f)\leq c} S(w) \tag{2.5}$$

Equation (2.5) is equivalent to seeking the function minimising the empirical cost in a small subset of $\mathcal{W}$. The problem here is that it is difficult to carry out a minimisation with inequality constraints. However, according to the Kuhn and Tucker theorem, there is an implicit equivalence between solving (2.5) and minimising a modified version of the cost:

$$\widehat{w}_\xi = \arg\min_{w} \left(S(w) + \xi R(f)\right) \tag{2.6}$$

This is reminiscent of the method of Lagrange multipliers. The *regularisation parameter* $\xi$ also called *hyper-parameter*[3] implicitly defines a structure on the possible models by constraining the model. Roughly, low values of $\xi$ in (2.6), i.e. weak constraint, correspond to high values of $c$ in (2.5). On the other hand, a large regularisation parameter gives higher importance to the minimisation of $R(f)$. It thus corresponds to a low value for $c$.

The regularised cost minimisation problem is a trade-off between fitting the data (minimising $S(w)$) and constraining the model to stay in a small compact subset (minimising $R(f)$). Furthermore, the balance between satisfying the constraint on the model and staying close to the data is governed by the regularisation parameter $\xi$.

## 2.6 Regularisation functionals

In the previous section, we have introduced the use of the regularisation functional $R(f)$ in order to make the learning problem well-posed. Not all functionals, however, are well suited to regularisation of an ill-posed problem.

As we have seen above, the regularisation constraint should actually define a structure of compact sets. In order for a functional $R$ to be suitable, it has to fulfill a number of conditions:

1. $R$ is semi-continuous on a dense subset of $\mathcal{W}$. This is the case for any continuous function on $\mathcal{W}$.
2. $R$ is positive: $\forall f \in \mathcal{W}, R(f) \geq 0$.
3. A solution of problem (2.1) exists in the domain of definition of $R$.
4. $R$ defines a structure of compact sets: $\forall c \geq 0, \{f | R(f) \leq c\}$ are all compact.

If all these conditions are met, $R$ deserves the name of *regularisation term*. For a regularisation term $R$, the minimisation problem (2.6) is a well-posed problem.

The above conditions are far from being restrictive. This allows for a large class of regularisers to be used. In particular, a common choice for $R$ consists in taking a

[3] This extension is due to the vocabulary used in Bayesian inference, see chapter 5.

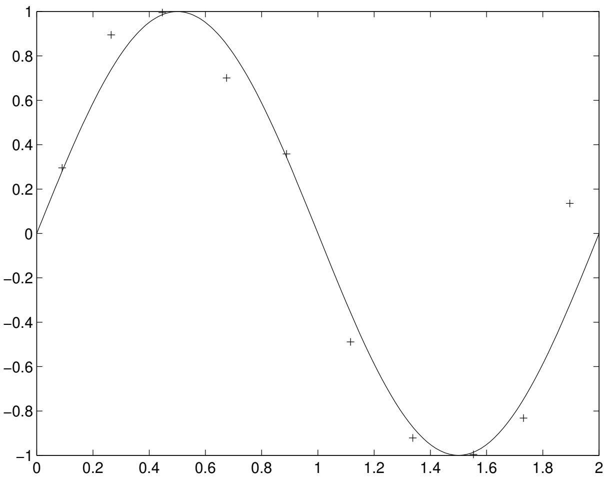


**Figure 2.2**: The 10 data points are displayed together with the sinusoid from which they were sampled, with low noise added (variance 0.025).

norm on $\mathcal{W}$, $R(f) = \|f\|_p$, or some power of this norm. In the same line of idea, another common choice is to use an operator $\mathcal{L}$, typically a derivative operator:

$$R(f) = \|\mathcal{L}f\|_p \tag{2.7}$$

From section 2.11 to section 2.16, a number of different regularisation techniques are introduced in the context of neural network models.

## 2.7 An example of ill-posed problem

Let us now consider a classical example of ill-posed problem in non-linear regression. The setting is extremely simple: we try to fit a sinusoid on 10 points with x-values generated in the interval $[0; 2]$ and y-values in $[-1; 1]$.

The model is a simple one-parameter function $y = \sin(a\pi x)$. It is non-linear and depends on a single parameter $a$, the frequency. Let us consider for example that we generate 10 slightly noisy points from one such sinusoid with $a = 1$. The reference and the associated points are presented in figure 2.2.

The noise level is rather low, with a sample variance of 0.025. In this case, the underlying mapping is taken as one of the possible models. The best choice for $a$ (i.e. the one that generalises best) would obviously be $\widehat{a} = 1$. In the context where we do not know the exact mapping, it is unnecessary to resort to multi-dimensional optimisation as in chapter 1, because the model depends on a unique parameter. A simple line search will suffice. The mean squared error is a function of a single parameter:

$$S(a) = \frac{1}{10} \sum_{x=1}^{10} \left( y^{(i)} - \sin(a\pi x) \right)^2 \tag{2.8}$$

Figure 2.3 displays the behaviour of the mean squared error $S$ as a function of $a$ on

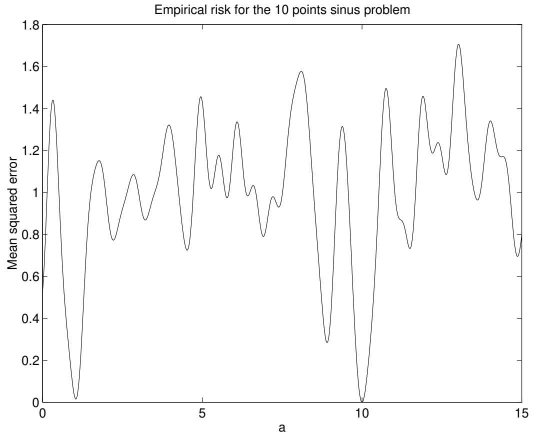


**Figure 2.3**: The mean squared error calculated on the 10 points for $a$ between 0 and 15 displays two deep minima: one for $a = 1.04$, the other one for $a = 10$.

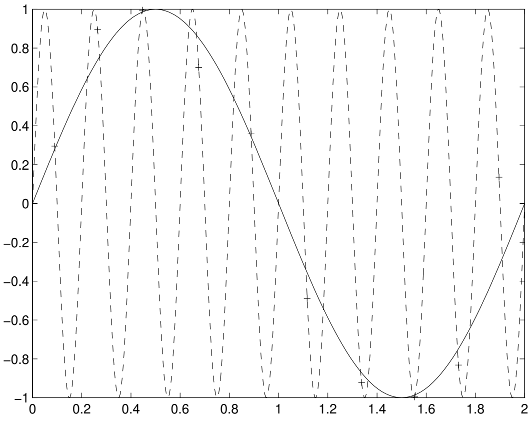


**Figure 2.4**: Solution of the mean squared error minimisation (dashed) together with the data points (+) and the underlying mapping (solid).

the interval $[0; 15]$. As expected, there is a sharp decrease around 1 (close to 1.04 to be precise), where the MSE value is 0.016. However, it appears that this is only a (good) local minimum. The global minimum is in $\widehat{a} = 10$, where[4]the MSE actually reaches 0. The resulting model is plotted together with the underlying function and the 10 points of figure 2.4. The empirical solution $\widehat{a}$ indeed minimises the distance to the data as these points are actually on the model.

It is clear though that this solution is a very bad one in terms of generalisation. Apart from a couple of places where the curves cross, $\widehat{a}$ produces wild guesses irrelevant to the underlying mapping. The reason for this is simple. Whatever set of $N$ point we get with ordinates in $[-1; 1]$, there exists a value of $a$ such that the associated sinusoid approximates the data arbitrarily closely. This is reflected in the fact that the sinusoid, even though it has a single parameter, has infinite capacity: it can interpolate with arbitrary precision any set of any number of points within its range.

In order to avoid this embarrassing behaviour, let us add a small regularisation term to the mean squared error: $R(a) = 0.01a^2$. This term is a regularisation term, as it obviously meets all the requirements listed in section 2.6. It corresponds to putting a penalty on large values of $a$. This favours low frequencies, i.e. smooth functions, a rather reasonable (and very common) constraint.

The resulting regularised cost is displayed on figure 2.5. The cost function has been "ironed" by the regularisation term, and only one clear minimum remains, for $\widehat{a} = 1.04$. Due to the regularisation effect, this value is slightly smaller than that corresponding to the first local minima above. Figure 2.6 displays the shape of the resulting model, together with the original mapping and the data points. Despite the limited amount of data available, the model provides a fairly good approximation of the underlying mapping in the domain of the data.

[4]The fact that we get $\widehat{a} = 10$ is of course not related to the number of data!

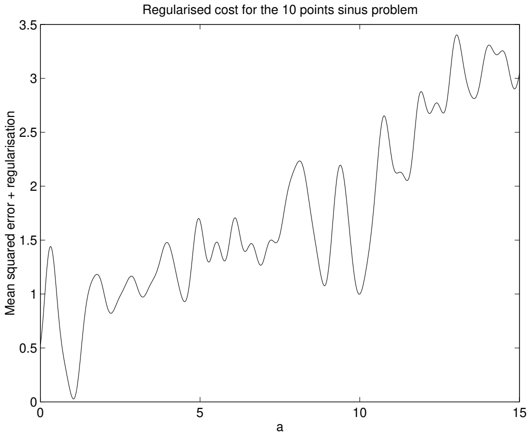


**Figure 2.5**: With the help of a regularisation term, the cost function favours low values of $a$, resulting in one clear minimum $a = 1.04$.

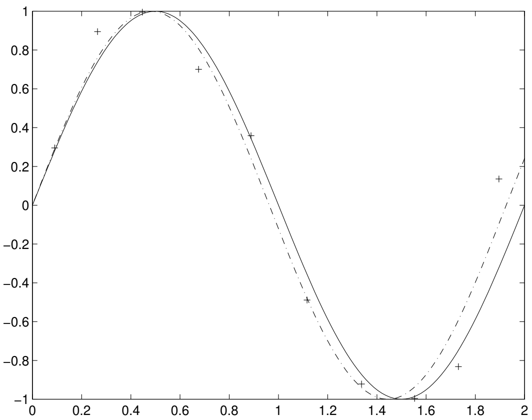

**Figure 2.6**: Solution of the regularised cost minimisation (dash-dotted) together with the data points (+) and the underlying mapping (solid).

## 2.8 Density estimation is an ill-posed problem

Let us recall that the learning problem is to obtain a function $f$ in a given set $\mathcal{W}$, that minimises the generalisation error:

$$G(w) = \int e(x, w)\, p(x)\; dx \tag{2.9}$$

Let us now consider the problem of estimating the density $p(x)$. If we estimate this density correctly, we could hope in turn to estimate $G(w)$. We now wish to solve the probability distribution problem, i.e. find density $p(u)$ satisfying:

$$\int_{-\infty}^{x} p(u)\; du = P(x)\,, \qquad \forall x \tag{2.10}$$

where $P(x)$ is an unknown probability distribution function, but we have a number $N$ of examples $x^{(i)}$ available, sampled from this distribution. The unknown p.d.f. can then be approximated by:

$$P_N(x) = \frac{1}{N} \sum_{i=1}^{N} H\left(x - x^{(i)}\right) \tag{2.11}$$

where $H$ is the Heavyside (step) function. Its derivative is the Dirac function $\delta$. According to the Glivenko-Cantelli theorem, the empirical distribution function (2.11) converges uniformly towards the actual p.d.f. The approximate problem of density estimation becomes:

$$\int_{-\infty}^{x} p_N(u)\; du = P_N(x)\,, \qquad \forall x \tag{2.12}$$

and the obvious solution to this problem is:

$$p_N(x) = \frac{1}{N} \sum_{i=1}^{N} \delta\left(x - x^{(i)}\right) \tag{2.13}$$

the empirical density estimate. Despite the (uniform) convergence of $P_N$ towards $P$, the solution $p_N$ of (2.12) does not converge towards the (unknown) solution $p$ of (2.10). The density estimation problem is thus ill-posed.

Notice that the use of the empirical density 2.13 as an estimate for $p(x)$ in (2.9) leads to the expression of the empirical risk or (unregularised) training error.

## 2.9 Regularisation parameter

In section 2.5 we have noticed that the regularisation parameter $\xi$ sets the balance between the fit to the data and the constraint on the model. In the previous section, the effect of regularisation has been displayed on a toy example. However, no justification has been given regarding the choice of the 0.01 as the regularisation parameter.

The choice of the proper amount of regularisation, i.e. the choice of the optimal $\xi$ is indeed a central preoccupation in regularisation. In the line of the definition of Tikhonov well-posedness, it corresponds either to the restriction we make on the possible subsets of solutions or (it is implicitly equivalent) to the strength of the constraint on the model.

Unfortunately, there is no general method to obtain the optimal level of regularisation. A number of methods exist that *estimate* this optimal level. It is one of the goals of this thesis to give such methods and exhibit their links and differences. A number of techniques are covered in chapter 3, such as cross-validation and prediction error estimation. Chapter 5 takes a different standpoint and presents the relationship between regularisation and Bayesian inference, with associated methods for the setting of $\xi$.

## 2.10 Input noise

It has been reported that adding noise to the input data during the optimisation procedure is a way of improving the generalisation abilities of the model. This effect is often called "jitter" to distinguish it from the effect of the usual, corrupting noise. In this section, we analyse the effect of jitter, and demonstrate its equivalence with a form of regularisation.

Let us first consider the linear case. The addition of a jitter $\varepsilon$ on the input produces a data matrix $(\mathbf{X} + \varepsilon)$. The jitter is sampled from a Gaussian distribution of 0 mean and variance $\sigma^2$, independent of both input and output values. For a large number of data[5], we have:

$$\left((\mathbf{X}+\varepsilon)^\top (\mathbf{X}+\varepsilon)\right) \approx \left(\mathbf{X}^\top \mathbf{X} + \varepsilon^\top \varepsilon\right) \tag{2.14}$$

thanks to the independence assumption. Furthermore, $\varepsilon^\top \varepsilon \approx N\sigma^2 I$. Under the same

[5] this assumption does not hold if the number of data is low and jitter is added once and for all. It *does* hold however, when each presentation of an input to the optimisation procedure is made with a different jitter $\varepsilon$. In the linear case, this can be done by replicating the data matrix several times with a different jitter.

assumptions, the cross product gives $(\mathbf{X}+\varepsilon)^\top \mathbf{Y} \approx \mathbf{X}^\top\mathbf{Y}$. The resulting estimator for the parameters is:

$$\widehat{w} \approx \left(\mathbf{X}^\top\mathbf{X} + N\sigma^2\mathbf{I}\right)^{-1}\mathbf{X}^\top\mathbf{Y} \tag{2.15}$$

Comparing (2.15) and (1.12) shows that the use of jitter is equivalent to adding a small diagonal term before inverting $\mathbf{X}^\top\mathbf{X}$. This is known as *ridge regression*, the most simple form of regularisation for linear models.

In order to study the effect of the noise on non-linear regression, let us perform a Taylor expansion around each data point:

$$f_w\left(x^{(i)}+\varepsilon\right) = f_w\left(x^{(i)}\right) + \varepsilon^\top\nabla f_w\left(x^{(i)}\right) + \frac{1}{2}\varepsilon^\top\mathbf{H}_{f_w}\left(x^{(i)}\right)\varepsilon + o\left(\varepsilon^\top\varepsilon\right) \tag{2.16}$$

In order to keep notations concise, we will note $\nabla^{(i)} = \nabla f_w\left(x^{(i)}\right)$ and $\mathbf{H}^{(i)} = \mathbf{H}_{f_w}\left(x^{(i)}\right)$. Using the expansion (2.16), we derive the expression of the empirical risk with jitter:

$$\begin{aligned} S_\varepsilon(w) &= \sum_{i=1}^{N}\left(y^{(i)} - f\left(x^{(i)}+\varepsilon\right)\right)^2 \\ &= \sum_{i=1}^{N}\left(y^{(i)} - f\left(x^{(i)}\right)\right)^2 + \nabla^{(i)\top}\varepsilon\varepsilon^\top\nabla^{(i)} \\ &\quad -2\left(y^{(i)} - f\left(x^{(i)}\right)\right)\left(\varepsilon^\top\nabla^{(i)} + \frac{1}{2}\varepsilon^\top\mathbf{H}^{(i)}\varepsilon\right) + o\left(\varepsilon^\top\varepsilon\right) \end{aligned} \tag{2.17}$$

As we focus on the *average* behaviour of jitter, we have $\varepsilon\varepsilon^\top \approx \sigma^2\mathbf{I}$, $\varepsilon^\top\mathbf{H}^{(i)}\varepsilon \approx \sigma^2\mathrm{tr}\left(\mathbf{H}^{(i)}\right)$, while the term with a single $\varepsilon$ in the last parenthesis, $\varepsilon^\top\nabla^{(i)}$, disappears. The final result is then:

$$S_\epsilon(w) = S(w) + \sigma^2 R(f_w) \tag{2.18}$$

where the expression of $R(f_w)$ is given by:

$$R(f_w) = \sum_{i=1}^{N}\left[\left(\nabla f_w\left(x^{(i)}\right)\right)^2 - \left(y^{(i)} - f_w\left(x^{(i)}\right)\right)\mathrm{tr}\left(\mathbf{H}_{f_w}\left(x^{(i)}\right)\right)\right] \tag{2.19}$$

Expression (2.18) looks very much like a form of regularisation, where $\sigma^2$ plays the role of a regularisation coefficient. The higher the level of noise, the more constrained the model. However, a closer look at (2.19) shows that function $R(f_w)$ does not qualify as a regularisation functional, according to section 2.6. It is continuous as long as the model is twice continuously differentiable, but it is certainly not necessarily positive. The curvature of the model as well as the estimation error can make the second term on the right-hand side of (2.19) negative.

It has been argued that the second term of (2.19) involving the second derivatives, vanishes to order $\sigma^2$. This approximation allows for a simplification of (2.19) into:

$$R(f_w) = \sum_{i=1}^{N}\sum_{k=1}^{P}\left(\frac{\partial f_w}{\partial w_k}\left(x^{(i)}\right)\right)^2 \tag{2.20}$$

This is indeed a regulariser, in the spirit of equation 2.7. With this approximation, the effect of noise added on the input during training becomes clear: the variance of this noise behaves as a regularisation parameter, and the regularisation term is linked to the first derivative of the model. This constraint favours slowly varying functions, leading to a smooth model.

It should be noted that the approximation involved in obtaining 2.20 has been criticized as unsound and potentially harmful if it is used directly during training. Indeed, it is only valid at the minimum or in a close neighbourhood. During most of the training, it is incorrect, and does not allow to characterise the actual regularising effect of noise injection. Even in very simple cases, such as a unique hidden neuron, it can lead to incorrect results. However, the derivation based on the Taylor expansion stays valid and gives an elegant explanation of the effect of noise injection on the input. The problem of estimating the actual regularising effect of noise injection, however, is still open.

## 2.11 Regularisation in neural networks

The simplest form of regularisation consists in adding a regular term $R(w)$ to the empirical cost $S(w)$, weighed by a regularisation coefficient $\xi$:

$$C(w) = S(w) + \xi R(f_w) \tag{2.21}$$

In the context of neural networks, this corresponds to *formal regularisation*. There are a number of choices for the functional $R(f_w)$, and some of them are investigated in sections 2.12 to 2.14.

Another widespread regularisation technique in neural computation is called *structural regularisation*. As the name indicates, it corresponds to limiting the capacity of a network by modifying the *structure* of the network. Sections 2.15 and 2.16 are relevant to this kind of regularisation.

The structural regularisation technique suppresses a number of parameters in the model. In doing so, they indeed reduce the capacity of the model[6]. However, this is not exactly in line with the definition of Tikhonov regularisation. If we recall sections 2.4 and 2.5, a problem is well-posed in Tikhonov's sense when the model is constrained to be in a subset of $\mathcal{W}$. Deleting one parameter amounts to limiting the search for the right model to a subspace (of dimension $P-1$) of the original, $P$-dimensional parameter space. .

It should be noted that such a subspace is not compact if the original space is not. This means that there is no guarantee that the inverse operator is continuous. In other words, limiting the search to such a subspace does not necessarily make the problem well-posed (in Tikhonov's sense). It can however improve generalisation by limiting the capacity of possible models.

[6]The capacity is not linked to the number of parameters in the sense that a model with one parameter can very well have infinite capacity (cf. section 2.7). However, *for a given model*, suppressing some parameters actually lowers the capacity

## 2.12 Weight-decay

The simplest way of performing formal regularisation is to use the second norm of the parameter vector:

$$R_{WD}\left(f_w\right) = \|w\|_2^2 = \sum_{k=1}^{P} w_k^2 \tag{2.22}$$

This technique is known as *weight-decay* for a simple reason. The derivative with respect to weight $w_k$ is $\frac{\partial R}{\partial w_k} = 2w_k$. If this constraint alone influences the update of the weight, its behaviour obeys the simple differential equation $\frac{\partial w_k}{\partial t} = -2w_k$, giving the solution $w\left(t\right) \propto \exp\left(-2t\right)$, corresponding to an exponential *decay* of the weight towards 0.

In a regularised cost minimisation problem, the use of a weight decay favours models with low weights. This has a natural explanation as large parameters (excluding biases) will lead the model to produce sharp variations in some regions, according to equation (1.21). The weight decay also corresponds in a way to an indirect smoothness constraint. It should be noted though that a given level of regularisation could be reached with a number of equal magnitude parameters, as well as one parameter of large magnitude, and the remainder with very low values. The regulariser used in section 2.7 is of the same kind.

An important information arises when we consider the derivative of the cost function with respect to parameter $w_k$:

$$\frac{\partial C}{\partial w_k} = \frac{\partial S}{\partial w_k} + 2\xi w_k \tag{2.23}$$

Consider now the minimum $\widehat{w}$ of the cost function, where (by definition) the gradient is zero. (2.23) becomes:

$$\frac{\partial S}{\partial w_k} = -2\xi \widehat{w}_k \tag{2.24}$$

The left-hand term in (2.24) represents the local variation of the quadratic cost with $w_k$ at the optimum $\widehat{w}$. Its absolute value represents in some sense the *sensitivity* of the data fit to variations in parameter $w_k$. With a weight-decay, this sensitivity is proportional to the weight value and can differ greatly among the parameters. Furthermore, this sensitivity is highly arbitrary: there is no reason why the sensitivity to a given weight should be higher than the sensitivity to a second weight, just because the first one happens to have a larger absolute value.

We will come back to this topic later in section 2.16 when we consider an alternative regulariser.

## 2.13 Regularisation and data normalization

We will now investigate the link between some formal regularisation functionals and the need to normalize data in a multi-layered perceptrons with linear output.

It is well-known that normalization is not a theoretical necessity for existence of a proper model. Indeed, let us consider that all data have been normalized by subtracting a quantity $\overline{\mu}$ (usually the mean) and dividing the result by $\overline{\sigma}$ (usually the standard deviation[7]). It is easy to check that there is an equivalence between a model with parameters $w$ applied to the non-normalized data and a model with parameters $\overline{w}$ applied to normalized data when the input weights obey:

$$\forall 1 \leq j \leq H, \left( \forall 1 \leq i \leq n,\ \overline{w}_{ji} = \overline{\sigma} w_{ji} \text{ and } \overline{w}_{j0} = w_{j0} + \sum_{i=1}^{n} \overline{\mu} w_{ji} \right) \tag{2.25}$$

and the output weights satisfy:

$$\forall 1 \leq j \leq H,\ W_j = \overline{\sigma}\overline{W}_j \quad \text{and} \quad W_0 = \overline{\sigma}\overline{W}_0 + \overline{\mu} \tag{2.26}$$

So normalization does not matter as far as the existence of the model goes. On the other hand, there can be some numerical problems when handling large data, due to the saturation in the sigmoid function. According to equations (1.22) and (1.25), the calculation of the gradient of the cost with respect to the input weights involves the derivative of the transfer function. When the input to the hidden cell, i.e. the weighted sum of the input, is large (in absolute value), this derivative is naturally zero. This means that the weight is "freezed": its derivative is so small that successive iterations of the learning algorithm cannot make it evolve.

This behaviour manifests itself by the presence of large plateaus on the cost surface, as illustrated on the right of figure 2.7. This figure displays the projection of a cost surface on the $111^{\text{th}}$ parameter of a neural network model. This model was obtained by training a 10 hidden cells network on the non-normalized Sunspots data set. The total number of parameters is 141 in that case, and parameter 111 corresponds to a connection between the seventh delay in the input layer and the ninth hidden cell.

According to this observation, normalizing the data has a practical aspect, as raw data can be more difficult, or practically impossible, to learn. In this context, the weight-decay constraint towards 0 allows to "de-freeze" the weight. The regularised cost is in dashed line in figure 2.7. It is clear that the weight decay gives some curvature to the cost surface in the plateau area. Accordingly, the derivative of the cost function with respect to parameter 111 is non zero and optimisation is carried on. The minimum on the regularised surface is in $-1$, close to the actual minimum of the non-regularised cost.

Saturation problems are likely to happen when the non-normalized data have large values. In such a case, $\sigma$ is large, and the input weight correspondence (2.25) requires small values for the weight of the non-normalized network. This is precisely the kind of constraint that a weight-decay regulariser imposes.

Finally, let us note that the modelling of non-normalised can be performed efficiently by initialising the weights properly (e.g. small initial values when input values are large). However, the use of a weight-decay frees us from the problem of choosing the proper initialisation, which is potentially different for the input and output layer.

[7] we will here suppose that the normalization is identical on the input and on the output. It is a reasonable claim in our case, as time-series modelling, for example, uses the same data on the input and on the output. This is by no means a restriction. the same results can be derived for arbitrary input and output. The same remark applies to the network structure: these results are not limited to one hidden layer perceptrons.

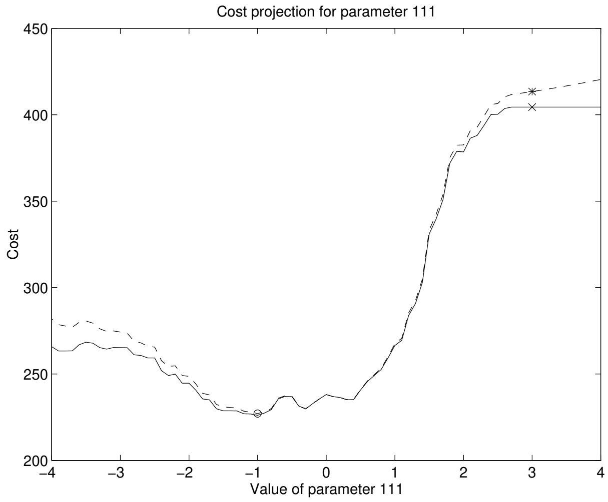


**Figure 2.7**: Projection of the regularised (dashed) and non-regularised (solid) cost surfaces on the subspace corresponding to parameter 111 (out of 141). For non-regularised cost, the parameter value is 3, situated in the flat zone (x), hence frozen. With regularisation, the induced curvature makes the parameter (*) slide to the minimum in $-1$ (o).

## 2.14 Stopped training

A widely used technique for improving the generalisation abilities of a model is to use *stopped training*. In order to analyse this technique, we will define it as:

- Perform a non-linear optimisation of the quadratic cost $S(w)$.
- At regular intervals, estimate the generalisation error $\widehat{G}_t(w)$.
- Stop the algorithm at step $t_{ST}$ such that $\widehat{G}_{t_{ST}}(w)$ is minimal.

This definition is slightly broader than the usual implementation. The basic method consists in splitting the available data in two sets, one for training, and one for assessing the generalisation abilities of the model ("validation error"). This assessment is made at every iteration, saving the parameters corresponding to successive minimum of the validation error and restoring these parameters when training is completed.

It will become obvious in chapter 3 that this fits in our definition, where the scheme known as "single validation" is used as a generalisation error estimator. The above definition allows for a broader range of generalisation assessment methods, and possibly several weight updates between each.

In the most primitive setting, training is stopped as soon as the generalisation estimate increases. This of course is a rather poor idea, as this estimate could very well decrease further as the optimisation continues.

We will now perform a bit of analysis in the linear case. Let us recall the notations from sections 1.5, 1.6 and 2.10. We will consider the case of a standard gradient descent optimisation, where the update step is made with constant step-size, by:

$$w_{t+1} = w_t - \eta \nabla S(w_t) \tag{2.27}$$

If this algorithm converges, it will do so to the unique[8] minimum $\widehat{w} = \left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top \mathbf{Y}$. We can also notice that:

$$\nabla S(w_t) = \frac{2}{N}\mathbf{X}^\top (\mathbf{X} w_t - \mathbf{Y}) = \frac{2}{N}\mathbf{X}^\top \mathbf{X}(w_t - \widehat{w}) \tag{2.28}$$

From (2.28), we easily obtain the expression of the parameter vector at step $t$:

$$w_t = \left(\mathbf{I} - \left(\mathbf{I} - \frac{2\eta}{N}\mathbf{X}^\top \mathbf{X}\right)^t\right)\widehat{w} + \left(\mathbf{I} - \frac{2\eta}{N}\mathbf{X}^\top \mathbf{X}\right)^t w_0 \tag{2.29}$$

Let us now use the eigen-decomposition of $\mathbf{X}^\top \mathbf{X} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^\top$, with $\mathbf{\Lambda}$ the diagonal matrix of eigen-values $\lambda_k$. We obtain the following result:

$$\left(\mathbf{I} - \frac{2\eta}{N}\mathbf{X}^\top \mathbf{X}\right) = \mathbf{Q}\left(\mathbf{I} - \frac{2\eta}{N}\mathbf{\Lambda}\right)\mathbf{Q}^\top = \mathbf{Q}\begin{bmatrix} \ddots & 0 & 0 \\ 0 & 1 - \frac{2\eta}{N}\lambda_k & 0 \\ 0 & 0 & \ddots \end{bmatrix}\mathbf{Q}^\top \tag{2.30}$$

As an aside, we notice that the algorithm (2.27) converges if and only if $\eta < \frac{N}{\lambda_k}$ for all $k$. The maximum step-size will then depend on the largest eigen-value (or rather on its inverse).

We then obtain the expression of the parameter vector at step $t$. In order to simplify the calculations, and as we are only interested in asymptotic behaviour, we will take $w_0 = 0$ to eliminate the second term in the right-hand side of (2.29):

$$w_t = \mathbf{Q}\begin{bmatrix} \ddots & 0 & 0 \\ 0 & \frac{1-\left(1-\frac{2\eta}{N}\lambda_k\right)^t}{\lambda_k} & 0 \\ 0 & 0 & \ddots \end{bmatrix}\mathbf{Q}^\top \mathbf{X}^\top \mathbf{Y} \tag{2.31}$$

As we shall prove, there is an asymptotic link between this result and the solution obtained by *ridge regression*, i.e. a form of linear weight decay:

$$\widehat{w}_\xi = \left(\mathbf{X}^\top \mathbf{X} + \xi \mathbf{I}\right)^{-1} \mathbf{X}^\top \mathbf{Y} = \mathbf{Q}\begin{bmatrix} \ddots & 0 & 0 \\ 0 & \frac{1}{\lambda_k + \xi} & 0 \\ 0 & 0 & \ddots \end{bmatrix}\mathbf{Q}^\top \mathbf{X}^\top \mathbf{Y} \tag{2.32}$$

Equations (2.31) and (2.32) can indeed be put in similar forms. Now if we want to exhibit any relationship between the ridge regression solution and the stopped training solution, we have to investigate the links between the diagonal terms.

For a rather large value of $\lambda_k$, we see that $\frac{1}{\lambda_k+\xi} \approx \frac{1}{\lambda_k}$, so the ridge term has no effect on the solution. In the case of stopped training, for large $\lambda_k$ in the limit of the convergence, $\frac{2\eta}{N}\lambda_k$ is close to 0. The diagonal term in (2.31) is approximately $\frac{1}{\lambda_k}$

[8] we are in the linear case

after only a few iterations, corresponding to the unregularised solution, as for the ridge estimate.

For small values of $\lambda_k$, we have $\frac{1}{\lambda_k+\xi} = \frac{1}{\xi} + o(1)$, and the following expansion for the main term in the diagonal of (2.31):

$$\left(1 - \frac{2\eta}{N}\lambda_k\right)^t = 1 - \frac{2\eta t}{N}\lambda_k + o(\lambda_k) \tag{2.33}$$

So in order to have an asymptotic equivalence between the stopped training and *ridge regression* solutions, we need:

$$t_{ST} \approx \frac{N}{2\eta\xi} \tag{2.34}$$

Let us now denote $\widehat{w}_T(t)$ the parameter estimate given by the stopped training at step $t$, and $\widehat{w}_\Xi(\xi)$ the *ridge regression* estimate for a regularisation level of $\xi$. Equation (2.34) tells us the we have $\widehat{w}_\Xi(\xi) \approx \widehat{w}_T\left(\frac{N}{2\eta\xi}\right)$. The optimal regularisation level corresponds to the minimum of the generalisation estimate. If we note $E(w)$ this estimate for a model of parameters $w$, it means that the optimum is reached for $\frac{\partial E}{\partial \xi} = 0$. We get by the derivation rule of composed functions:

$$\frac{\partial E\left(\widehat{w}_\Xi(\xi)\right)}{\partial \xi} = \frac{\partial\left(\frac{N}{2\eta\xi}\right)}{\partial \xi}\frac{\partial E\left(\widehat{w}_T(t)\right)}{\partial t} = -\frac{N}{2\eta\xi^2}\frac{\partial E\left(\widehat{w}_T(t)\right)}{\partial t} \tag{2.35}$$

This shows that when a level $\xi$ of *ridge regression* regularisation is optimal with respect to the generalisation estimate $E$, the corresponding number of step (2.34) is optimal for the stopped training algorithm. Equation (2.35) also shows that the derivatives w.r.t. $\xi$ and $t$ are of opposite signs. This is no surprise as those quantities are inversely proportional: an increase in $t_{TS}$ corresponds to a decrease in $\xi$.

In short the above derivations give the following insight into stopped training:

1. For any number $t$ of iteration in stopped-training, there exist a regularisation level $\xi$ with the same effect.
2. The model with best estimated generalisation error is the same for stopped training and ridge regression.

## 2.15 "Optimal" methods: OBD, OBS, ...

We will now present some results related to *structural regularisation*, in order to better appreciate the assets of the technique presented in the next (and last) section.

The main pruning techniques for neural networks are *Optimal Brain Damage*, or *OBD*, and *Optimal Brain Surgeon*, a.k.a. *OBS*, hence the title of the section.

Both methods rely on a second order approximation of the increase in error around the minimum. For the quadratic cost $S(w)$, we have according to (1.26):

$$S(w) - S(\widehat{w}) = \frac{1}{2}(w - \widehat{w})^\top \mathbf{H}(w - \widehat{w}) + o\left(\|w - \widehat{w}\|^3\right) \tag{2.36}$$

where $\mathbf{H}$ is the Hessian of $S$, as usual. Using (2.36), we wish to pick the weight(s) that can be deleted with an increase in error as small as possible.

If we denote by $u_k$ an orthonormal basis in parameter space, and $\widehat{w}_k$ the $k$-th component of the solution $\widehat{w}$, the modification $\delta w$ in weight associated with deleting parameter number $k$ in $\widehat{w}$ is $\delta w = \widehat{w}_k u_k$. In the case of OBD, the increase in error is calculated immediately, and is called the *saliency* $s_k$ of parameter $w_k$:

$$s_k = \frac{1}{2}\widehat{w}_k^2 u_k^\top \mathbf{H} u_k = \frac{1}{2}\widehat{w}_k^2 h_{kk} \tag{2.37}$$

where $h_{kk}$ is the $k$-th diagonal element of the Hessian $\mathbf{H}$. Hence the *OBD* algorithm:

1. Train network to a minimum $\widehat{w}$.
2. Compute saliencies $s_k$ for $1 \leq k \leq P$.
3. Delete a number of weights with small saliencies.
4. Go back to step 1 with the remaining network, until there is no weight left with small saliency.

OBD leads to efficient pruning of weights with limited increase in error. However, there are a couple of problematic issues. The weights are not independent: when weight $w_k$ is deleted, the network is usually not at a minimum in training error in the subspace of non-pruned parameters. In order to obtain the actual increase in error between the best network with parameter $w_k$ and the best network without parameter $w_k$, it should be necessary to re-train the network after each deletion, which is computationally difficult. Another way to look at this problem, is to say that OBD neglects the coupling between weights, i.e. makes the assumption that the Hessian matrix is diagonal.

This suggests that we need a way to incorporate some more information in the algorithm to take into account the effect of re-training in the calculation of the saliency and in the update of non-pruned weights. This can be done by the OBS algorithm. This algorithm calculates, on the basis of the Hessian, the expected increase in error *after re-training the pruned network*, and computes the associated adaptation for the other weights.

We want to find the modification $\delta w$ of the parameters that lead to the minimum increase of $S(w)$ subject to the constraint that $\widehat{w}_k + \delta w\, u_k = 0$. We use the method of Lagrange multiplier, and form $L = \frac{1}{2}\delta w\, \mathbf{H}\, \delta w + \mu\left(\widehat{w}_k + \delta w\, u_k\right)$, where $\mu$ is an unknown constant. The derivative of the Lagrangian w.r.t. $\delta w$ is zero, so $\mathbf{H}\delta w = -\mu u_k$. Plugging this into our constraint we infer $\mu$ which results in:

$$\delta w = -\frac{\widehat{w}_k}{g_{kk}}\mathbf{H}^{-1} u_k \tag{2.38}$$

where $g_{kk}$ is the $k$-th diagonal term of $\mathbf{H}^{-1} = [g_{ij}]$. The corresponding increase in error is the saliency:

$$s_k = \frac{\widehat{w}_k^2}{2\, g_{kk}} \tag{2.39}$$

We notice that OBD is a special case of OBS where the Hessian matrix is diagonal, and $\mathbf{H}^{-1} = [1/h_{kk}]$. The calculation of the full inverse Hessian information is very costly. However, the OBS algorithm uses the *Sherman-Morrison* inversion identity to compute approximate inverse Hessian information as in section 1.14. The *OBS* algorithm can thus be described as:

1. Train network to a minimum $\widehat{w}$.
2. Compute $\mathbf{H}^{-1}$.
3. Compute the saliencies $s_k$.
4. Find the weight with smallest saliency.
5. If saliency is sufficiently small, update $\widehat{w}$ with (2.38) and go back to step 2.
6. Otherwise terminate.

It is also possible to replace step 6 by a re-training of the network, in order to compute the saliencies again on a re-trained network and check that this does not modify the results.

With the approximate inverse Hessian calculation, the OBS scheme is computationally feasible, but demanding. In the next section, we analyse a different regulariser, which provides some pruning, at a much lower cost than both pruning schemes above.

Let us eventually note that we have not mentioned in this section any stop criterion. Tuning the extent of the pruning is similar to tuning a regularisation parameter. Typically, one will be interested in minimising generalisation error, and refer to chapter 3 for a review of generalisation error estimators.

## 2.16 Reconciling formal and structural regularisation

We will now analyse the effect of a new regulariser. It consists in using the 1-norm of the parameters, rather than the squared 2-norm:

$$R_L\left(f_w\right) = \|w\|_1 = \sum_{k=1}^{P} |w_k| \tag{2.40}$$

The effect of this regulariser appears when we perform a piece of analysis in a manner similar to the end of section 2.12.

The derivative of the regularisation functional is defined on $I\!R^*$ and takes values $\pm 1$ (the sign function). Equation (2.24) expressing the sensitivity of the data fit to parameter $w_k$ becomes:

$$\left|\frac{\partial S}{\partial w_k}\left(\widehat{w}\right)\right| = \xi, \qquad |\widehat{w}_k| > 0 \tag{2.41}$$

When $\widehat{w}_k = 0$, the derivative does not exist. Let us consider what happens as $\xi$ grows. For $\xi = 0$, the sensitivity is 0. As $\xi$ increases, parameter $w_k$ takes a value that allows the sensitivity to match the regularisation parameter according to (2.41). If parameter $w_k$ is not really necessary, it only fits the noise, and is thus not very sensitive to the data. When $\xi$ grows and can not be matched by the sensitivity any more, we have $\left|\frac{\partial S}{\partial w_k}(\widehat{w})\right| < \xi$.

Let us now assume that parameter $\widehat{w}_k$ is not zero. (2.41) would then hold, which is in contradiction with the previous consideration about low sensitivity. This simple contradiction shows that our assumption is wrong, and we have $\widehat{w}_k = 0$: the parameter is pruned.

This regulariser has been called a *pruning prior* for this reason[9]. It is a regulariser in the sense of section 2.6, which combines the effects of formal and structural regularisation. It actually splits the parameters into two sets: on one hand, those for which (2.41) verifies, which bear a small constraint in the style of formal regularisation. On the other hand, parameters the sensitivity of which is nowhere high enough are forced to zero as in structural regularisation.

The fact that (2.40) is not derivable in 0 is not a problem. First, it is extremely unlikely that a parameter takes a value of *exactly* 0 during a numerical optimisation. Even if it did, this would not cause any numerical problem. Furthermore, it is possible to approximate $r(x) = |x|$ arbitrarily closely by a series of function as:

$$r_\epsilon(x) = \epsilon \ln\left(2\cosh\left(\frac{x}{\epsilon}\right)\right) \tag{2.42}$$

The $r_\epsilon$ converge uniformly towards $r$ as $\|r_\epsilon - r\|_\infty = \epsilon \ln 2$. For the first and second derivatives, simple convergence is achieved on $\mathbb{R}^*$.

This approximation is not necessary for any numerical optimisation method we can think of. Furthermore, it leads to two main problems:

1. The division by a small quantity $\epsilon$ in (2.42) leads to an overflow in the hyperbolic cosine.
2. We have found empirically that the approximation leads to poorer convergence.

The convergence damage can be explained by the fact that the neighbourhood of 0 in the approximation is smooth (by construction). Instead of yielding a derivative of $\pm\xi$, the regularisation term using the approximate form will have a vanishing gradient around 0! This means that as the regularisation tries to prune a weight by pulling it to 0, the approximation prevents this effect to be completed.

The interesting properties of the 1-norm regulariser are investigated further in chapter 6.

## COMMENTS

**2.2** A classical example of the instability of polynomial interpolation is given by the "Runge phenomenon". It consists in approximating the function $f(x) = \frac{1}{1+25x^2}$

[9]the reason for the "prior" will be exposed in chapter 5

by polynomials of increasing degrees: the interpolation becomes unstable very fast as the degree increases. We have chosen to limit ourselves to a second degree polynomial as we believe this provides an even simpler example.

**2.3** The study of ill-posed problems originates in the work of Hadamard (1902). However, it stayed a mathematical curiosity until the sixties, with the work of Russian mathematicians. Well-posed and ill-posed problems in Hadamard's sense are defined e.g. in chapter 1 of (Badeva and Morozov, 1991).

**2.4** Tikhonov well-posedness is also defined by Badeva and Morozov (1991). The links between Hadamard well-posedness and Tikhonov well-posedness are studied in Dontchev and Zolezzi (1992).

**2.5** Badeva and Morozov (1991), already cited above, gives a long introduction to Tikhonov regularisation method. A brief description is also given by Vapnik (1995).

**2.9** Hansen (1996) explores briefly the use of different regularisation functionals, in the context of linear inverse problems. E.g. the use of the 2-norm or the 1-norm for the regularisation functional, and Backus-Gilbert regularisation.

**2.7** This is a classical example (Charton, 1994).

**2.8** The ill-posedness of density estimation is mentioned by Vapnik (1995) in section 1.8.2.

**2.9** The choice of the appropriate regularisation parameter is a topic explored in chapter 3.

**2.10** Calculations in this section follow those of Bishop (1995), although we apply these derivations directly to the empirical cost. Equation (2.19) corrects equation (11) of Bishop (1995). Other authors have published similar results. The inadequacy of neglecting the second order term to make $R(f_w)$ a Tikhonov regulariser has been argued by Grandvalet (1995), who gives simple example where regularisation with (2.20) fails to give good results.

**2.11** The distinction between *formal* and *structural* regularisation in neural networks was introduced by Denker et al. (1987).

**2.12** The use of weight-decay for improvement of the generalisation error has been a standard technique for a long time in neural computation (Plaut et al., 1986). It is analysed, e.g. by Krogh and Hertz (1992). The use of the 2-norm of the parameters as a regulariser is common in linear regression, under the name *ridge regression*, already mentioned.

**2.11** Ljung et al. (1992) gives an example of a practical case where regularisation helps handling non-normalized data. In that case, however, weight-decay is only used for a few iteration in order to "de-freeze" the data, so this experiment is not really representative of regularised training.

**2.14** The links between stopped training and regularisation with non-linear models are still under investigation. In a private communication, V. Morozov mentioned that the problem of stopped training and regularisation had been solved by mathematicians from the ex-Soviet union. Unfortunately, we have been unable to find their work in translated form. Ljung et al. (1992) addresses the problem, and Sjöberg and Ljung (1995) also considers the equivalence between the two methods.

**2.15** The OBD algorithm was suggested by Le Cun et al. (1990). The OBS algorithm was proposed by Hassibi and Stork (1993). Some improvements have been proposed since. An interesting elaboration is the validation set pruning methods $\gamma$OBD and $\gamma$OBS, proposed by Pedersen et al. (1996).

**2.16** The regularisation term discussed in this section has not been studied a lot yet. It has been proposed to the neural network community by Williams (1995). Besides chapter 6, further studies can be found in (Goutte and Hansen, 1997; Goutte, 1996). The study of the influence of different regularisers than the usual 2-norm is also of interest to linear systems, cf. (Hansen, 1996) already cited.

## References

Badeva, V. and Morozov, V. (1991). *Problèmes incorrectement posés*. Série automatique. Masson, Paris.

Bishop, C. M. (1995). Training with noise is equivalent to thikonov regularization. *NC*, 7(1):110–116.

Charton, F. (1994). Discussion on the denker example. Personal communication.

Denker, J., Schwartz, D., Wittner, B., Solla, S., Howard, R., Jackel, L., and Hopfield, J. (1987). Large automatic learning, rule extraction, and generalization. *Complex Systems*, 1(5):877–922.

Dontchev, A. L. and Zolezzi, T. (1992). *Well-posed optimization problems*. Number 1543 in Lecture notes in mathematics. Springer, Berlin.

Goutte, C. (1996). On the use of a pruning prior for neural networks. In *Neural Networks for Signal Processing VI – Proceedings of the 1996 IEEE Workshop*, number VI in NNSP, pages 52–61, Piscataway, New Jersey. IEEE.

Goutte, C. and Hansen, L. K. (1997). Regularization with a pruning prior. *Neural Networks*. to appear.

Grandvalet, Y. (1995). *Injection de bruit dans les perceptrons multicouches*. PhD thesis, Université Technologique de Compiègne, France.

Hadamard, J. (1902). Sur les problemes aux derivees partielles et leur signification physique. *Bul. Univ. Princeton*, 13(49).

Hansen, P. C. (1996). *Rank-Deficient and Discrete Ill-Posed Problems*. Doctoral Dissertation. Polyteknisk Forlag, Lyngby (Denmark).

Hassibi, B. and Stork, D. G. (1993). Second order derivatives for network pruning: Optimal brain surgeon. In Hanson, S., Cowan, J., and Giles, C., editors, *Advances in Neural Information Processing Systems*, volume 5 of *NIPS*, pages 164–171. Morgan Kaufmann.

Krogh, A. and Hertz, J. A. (1992). A simple weight decay can improve generalization. In Moody, J. E., Hanson, S. J., and Lippman, R. P., editors, *Advances in Neural Information Processing Systems*, volume 4 of *NIPS*.

Le Cun, Y., Denker, J. S., and Solla, S. A. (1990). Optimal brain damage. In Touretzky, D. S., editor, *Advances in Neural Information Processing Systems*, number 2 in NIPS, pages 598–605. Morgan-Kaufmann.

Ljung, L., Sjöberg, J., and McKelvey, T. (1992). On the use of regularization in system identification. Technical Report 1379, Department of Electrical Engineering, Linköping University, S-581 83 Linköping, Sweden.

Pedersen, M. W., Hansen, L. K., and Larsen, J. (1996). Pruning with generalization based weight saliencies: $\gamma$OBD, $\gamma$OBS. In Touretzky, D. S., Mozer, M. C., and Hasselmo, M. E., editors, *Advances in Neural Information Processing Systems*, number 8 in NIPS. MIT Press.

Plaut, D. C., Nowlan, S. J., and Hinton, G. E. (1986). Experiments on learning by backpropagation. Technical Report CMU-CS-86-126, Carnegie Mellon University, Pittsburgh, USA.

Sjöberg, J. and Ljung, L. (1995). Overtraining, regularization and searching for minimum with application to neural nets. *International Journal of Control*.

Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. Springer.

Williams, P. M. (1995). Bayesian regularization and pruning using a Laplace prior. *Neural Computation*, 7(1):117–143.