THÈSE DE DOCTORAT DE L'UNIVERSITÉ PARIS 6

Spécialité :

Informatique

présentée par :

Cyril Goutte

pour obtenir le grade de DOCTEUR DE l'UNIVERSITÉ PARIS 6.

Titre :

STATISTICAL LEARNING AND REGULARISATION FOR REGRESSION
Application to system identification and time series modelling

Apprentissage statistique et regularisation pour la régression
Application à l'identification de systèmes et à la modélisation des séries chronologiques

soutenue le 7 juillet 1997 devant le jury composé de :

Sylvie THIRIA	Président
Anne GUÉRIN-DUGUÉ	Rapporteur
Gérard GOVAERT	Rapporteur
Jan LARSEN	Examinateur
Patrick GALLINARI	Examinateur

Acknowledgements

I am indebted to Professor Patrick Gallinari, who took me under his supervision in 1992, and followed me throughout this work. I would like him to know that his friendly support during these years were largely instrumental in the completion of this thesis.

I would also like to thank Professor Lars Kai Hansen for welcoming me at the now-defunct Electronics Institute during the year 1995-1996, and later at the Institute of Mathematical Modelling. His enthusiastic comments and support had a major influence on my work.

I wish to express my gratitude to the members of the jury, who undertook to evaluate my work. Sylvie Thiria accepted to be the head of this jury; Anne Guérin-Dugué and Gérard Govaert reviewed and refereed this thesis; Jan Larsen showed great interest in my work and accepted to fly from Denmark to be part of this jury.

I also wish to express my gratitude to all those who helped me through many fruitful discussions, or just by creating a friendly and challenging working atmosphere. The PhD students or ex-PhD students of the connectionist group of Laforia (Paris 6 University), and of the Digital Signal Processing group (Technical University of Denmark).

Finally, special thanks are in order for my wife Kia. I don't know what I should thank her more for: her continued support and exaltation, or just putting up with me while I was writing this manuscript.

Technical thanks

I am especially indebted to those who provided advices, software and other gimmicks that helped the completion of this thesis. I owe Peter Toft as I borrowed some of his LaTeX style, and his `si.m` matlab visualisation function among other things. LaTeX also got me indebted to Peter S.K. Hansen and Jan Larsen.

I am also pleased to thank Xavier Driancourt and Neuristique S.A. for lending me a version of the TLisp/SN software while I was away, and taking care of my endless requests on top of it.

This document was written using a customised version of the LaTeX `book` format with packages `fancyhdr`, `graphicx`, `apalike` and `makeidx`.

Introduction

Notation

Here are a number of symbols that are used throughout the text, together with their meaning. Other symbols will be defined locally, and are thus not detailed here.

x Input of the system and model.

y Output of the system.

$\widehat{y}$ Estimator of the output of the system.

f System mapping from input to output: $y = f(x)$.

f_w Parametric model mapping from input to output: $\widehat{y} = f_w(x)$.

w Vector of parameters.

$\widehat{w}$ Estimator of the vector of the parameters.

$\widetilde{w}$ Optimal vector of parameter.

$e(y, \widehat{y})$ Local error associated with the estimation of y by $\widehat{y}$.

$x^{(k)}$ Sample input.

$y^{(k)}$ Observed sample output.

$\mathcal{D}$ Data set $= \left(x^{(k)}, y^{(k)}\right)$.

$\varepsilon^{(k)}$ Error on the sample output: $y^{(k)} = f\left(x^{(k)}\right) + \varepsilon^{(k)}$.

$p(x)$ Probability density of the input x.

$\widehat{p}(x)$ Estimated density of the input x.

$\delta(\cdot)$ Dirac function.

$H(\cdot)$ Heavyside function.

$\mathrm{E}_v(\cdot)$ Expectation over variable v.

$S(w)$ Empirical risk, or training error.

$G(w)$ Expected risk, or generalisation error.

$C(w)$ Cost function, usually the regularised empirical risk.

N Number of examples (sample size).

P Number of parameters.

$\nabla_v f$ Gradient of function f with respect to variable v. When the gradient is computed with respect to the argument of the function, the subscript may not be mentioned, i.e. $\nabla f(x) = \nabla_x f(x)$.

$\mathbf{H}_f$ Hessian matrix of multidimensional function f.

cte denotes a constant term with no relevance to the rest of the equation.

$R(f)$ Regularisation functional.

$R(w)$ Regularisation functional for a parametric model, where w is the set of parameters.

ξ Regularisation coefficient.

$\mathrm{card}(E)$ Number of elements in set E (Sometimes written as $\#(E)$ in the american literature).

$\mathrm{P}(A)$ Probability of A (in a probability space).

Content

This thesis is relevant to the study of neural networks models. This research has traditionally been linked to computer science and artificial intelligence. In the last few years however, two distinct lines of thoughts seem to diverge: the first one stays close to the biological origins of the term—we will call it "neuro-biological"; the second considers neural networks as a model, and studies them from a statistical point of view.

Our work concerns the second of these lines. Furthermore, we try to stay independent of any specific application, and keep a general approach to neural networks. We attempt to exhibit the links between neural networks and statistics, and show that neural computation can be naturally placed in the realm of traditional statistics. We thus compare neural networks to other models: linear regression as well as non-parametric estimators. We also consider some of the numerous learning techniques developed for neural models, and try to compare them, from both a theoretical and applied point of view.

This thesis is organised along these ideas. The authors' contributions are integrated to this presentation and duly signaled, but we refrained from using a state-of-the-art / contribution dichotomy.

As a general guideline, let us just say that chapters 1 and 5 (as well as appendix B are more pedagogical, and original contributions are limited apart from a few minor corrections, and the implementation of the non-parametric tool-box. In chapters 2 and 3, the original content is higher as we present some original pieces of analysis.

Chapters 4 and 6 present some entirely original work. The first one presents an application of the techniques presented in chapters 1 to 3 to a usefull problem: finding the relevant delays to include in a time-series modelling problem The second presents our work on the so-called "pruning prior", first on a toy problem where we present its joint regularisation and pruning abilities, then on more difficult tasks: time-series modelling and system identification.

We have been concerned into giving a balanced and thorough presentation has been a main concern. It sometimes require to go back to the basics. Furthermore, this work is relevant to several quite different domains. Concepts that are obvious for a statistician might not be so for a connectionist, and the reverse is also true.

Summary

This thesis deals with the problem of statistical learning. By "learning", we mean a process by which we obtain a model of a phenomenon using data measured on it. We focus on system identification and time series modelling, and our work is naturally slightly influenced by this concern. We will for example evoke only regression estimation, and no classification problem. Most theoretical and practical aspects presented herein can easily be adapted however.

Let us proceed with a detailed presentation of the document. Chapter 1 presents the basic regression problem. We present the concepts of training and generalisation error and the maximum likelihood method. We evoke linear models as well as non-linear models. Neural-networks are presented as a particular class of non-linear models, and we present a number of optimisation technique used for training these models. We conclude by deriving the bias-variance decomposition for least-squares regression and show the limits of simple regression.

Chapter 2 is dedicated to regularisation. It gives a fast general presentation of ill-posed problems and gives the idea of regularisation to solve these. After presenting a typical case of ill-posedness, we introduce a number of well-known and less well-known regularisation techniques. We mention in particular an original regularisation technique that will be used in later chapters. We recall the link between regularisation and noise injection and data normalisation, and study carefully the relationship between stopped-training and regularisation with a ridge term.

Chapter 3 will focus on generalisation and its use in the tuning of regularisation *hyper-parameters*. Two main classes of methods are presented. The validation methods, including *split-sample*, cross-validation and *leave-one-out*, and the algebraic estimates of generalisation error. We insist on the links between these methods and

show that most of them are asymptotically equivalent. We also compare them from a computational point of view.

Chapter 4 puts the content of the first chapters into practice. We show how it is possible to use generalisation error to extract relevant input information in a time-series modelling problem. In particular, we use non-parametric regression and regularised neural networks to perform some of the modelling.

In chapter 5, we present an attractive alternative to the above: the Bayesian methods. The *evidence* and the *MAP* methods are presented, analysed and compared.

Chapter 6 is of a more applied nature. It is dedicated to the study of a regulariser that provides automatic pruning of unnecessary weights. It is exemplified with the study of a small problem: the parameter location problem. We present a thorough analyse of a number of solution, spanning an array of techniques from Bayesian analysis to the generalisation-oriented solutions. A comparison is made with the weight-decay method. We also provide some analysis on more complicated problems of time-series modelling and system identification.

Each chapter closes with a section dedicated to comments. The bibliographical references have been included in this section, rather than scattered in the text.

This thesis ends with a number of appendices. Some of them insist on aspects that are not treated in the main part, others contain papers written in the course of this Ph.D. They address points that are evoked in the text, and can serve as reference.

But poursuivi et contribution

Traditionnellement, le domaine du connexionnisme, est plutôt rattaché à l'informatique, et généralement à l'intelligence artificielle. Au cours des dernières années, il semble que deux courants se dégagent: l'un d'inspiration neuro-biologique, toujours rattaché à l'ethymologie même du terme *réseaux de neurones* ; l'autre plus théorique, se dégage des fondations biologiques pour s'intéresser au *modèle* et l'étudier d'un point de vue statistique.

C'est dans ce dernier cadre que se situent nos travaux. Il nous a semblé important non pas d'étudier une application particulière et restrictive, mais de développer une approche plus générale du domaine. Nous nous sommes efforcés de mettre en évidence des liens entre connexionnisme et statistiques, et de montrer que les méthodes connexionnistes trouvent naturellement leur place dans le bestiaire statistique traditionnel. Ainsi, nous avons étudié les réseaux de neurones en les comparant à d'autres techniques : régression linéaire bien sûr, mais aussi estimateurs non-paramétriques. Enfin, nous nous sommes intéressés à certaines des déjà nombreuses techniques d'apprentissages appliquées aux modèles connexionnistes: montrer leurs points communs lorsque cela est possible, et les comparer sur des cas appliqués.

L'organisation de ce document reflette donc ces préoccupations plutôt qu'une présentation linéaire des travaux de l'auteur. Les contributions originales sont réparties au fil des chapitres, et signalées comme telles.

Brièvement, nous pouvons dire que dans les parties didactiques que sont les chapitres 1 et 5 (ainsi que l'annexe B), ces contributions sont plutôt faibles, mis à part quelques corrections mineures et la partie implémentation des algorithmes non-paramétriques. Les chapitres 2 et 3 contiennent une part plus importante de contributions originales, notamment en ce qui concerne l'analyse et la comparaison des éléments présentés.

Enfin, les chapitres 4 et 6, sont pratiquement entièrement originaux. Le premier présente une application originale des techniques introduites dans les trois premiers chapitres. Il s'agit d'extraire d'une série temporelle les délais nécessaires à la modélisation de cette série par un modèle donné. Le second présente nos travaux relatifs à une fonctionnelle de régularisation particulière, tout d'abord sur un problème très simple, puis sur des problèmes plus appliqués en modélisation de séries temporelles et identification de systèmes.

Description détaillée

On s'intéresse ici au problème de l'apprentissage statistique. On entend par apprentissage l'obtention à partir de données issues d'un phénomène, d'un modèle de celui-ci. Les domaines qui nous intéressent plus particulièrement sont ceux de l'identification de systèmes et de la modélisation de séries temporelles, l'ensemble des travaux présentés ici relèvent de cette direction de recherche privilégiée. Par exemple, on ne s'intéresse qu'aux problèmes d'estimation de la regression, et pas à la classification. Cependant, la plupart des méthodes et développements théoriques présentés ont une application plus large.

Dans le chapitre 1, on présente rapidement certains aspects concernant la régression non régularisée, la méthode du maximum de vraissemblance, dans le cas linéaire puis dans le cas non-linéaire. Plusieurs techniques d'optimisation sont présentées pour effectuer un apprentissage non-linéaire. On conclut le chapitre en évoquant les problèmes posés par l'utilisation de la régression simple, ce qui nous permet d'enchaîner sur le chapitre suivant.

Au chapitre 2, on définit les notions de problèmes correctement et incorrectement posés. On introduit la régularisation, et l'on étudie la convergence des solutions régularisées. Le lien est effectué avec les modèles connexionnistes, pour lesquels on présente un certain nombre de techniques de régularisation usuelles et moins usuelles. On étudie en particulier les liens entre régularisation et injection de bruit, normalisation et apprentissage interrompu (stopped training).

Le chapitre 3 étudie le problème des hyper-paramètres et de leur détermination. Plusieurs méthodes sont présentées pour permettre d'estimer l'erreur de généralisation et permettre d'optimiser la valeur des hyper-paramètres : validation simple, validation croisée, et validation croisée "leave-one-out", puis une série d'estimateurs algébriques. On s'intéresse alors plus particulièrement aux liens entre ces différentes méthode, et l'on montre qu'elles sont pour la plupart équivalentes. Une comparaison est effectuée, à la fois sur le plan de la charge de calcul requise et sur le plan des résultats obtenus.

Le chapitre 5 présente les méthodes d'apprentissage Bayésien. Deux techniques sont présentées, analysées, et comparées : l'évidence tout d'abord, puis la méthode MAP.

Le chapitre 6 est plus appliqué. Il est dédié à l'étude d'une fonctionnelle de régularisation qui permet d'effectuer de manière automatique l'élagage des poids. On étudie tout d'abord plusieurs techniques d'apprentissage sur une problème simple. Ceci permet de mener à bien la plupart des calculs de manière exacte et mène à des analyses intéressantes. On insiste plus particulièrement sur la comparaison de notre fonctionnelle de régularisation avec le "weight-decay" traditionnellement utilisé en connexionnisme. On présente ensuite des résultats qui illustrent le fonctionnement de cette fonctionnelle sur deux types de problèmes : modélisation de séries temporelles et identification de systèmes.

Au cours de cette présentation, nous étudions une classe de modèles non-linéaires particuliers : les réseaux de neurones, aussi appelés réseaux (ou modèles) connexionnistes. Il n'y a pas de chapitre consacré spécifiquement à ces modèles, dans la mesure ou la majorité des méthodes présentées ici s'appliquent à des classes de modèles plus larges.

Ce manuscrit se termine par quelques annexes. En particulier, l'annexe B présente de manière détaillée les techniques de régression non-paramétrique que l'on utilise à de nombreuses reprises, dans les chapitres 6 et 4.

Dans chaque chapitre, les sujets abordés ont été répartis par section. Plutôt que d'indiquer les références en apparté au cours du texte, il a été préféré une organisation différente. Le corps du chapitre est censé être auto-suffisant, et de nombreuses références sont indiquées, section par section, à la fin de chaque chapitre.

Contents

List of Figures

List of Tables

1

Parametric regression

1.1 Learning problem

Let us present briefly the learning problem we will address in this chapter and the following. The ultimate goal is the *modelling* of a mapping $f : x \longmapsto y$ from multi-dimensional input x to output y. The output can be multi-dimensional, but we will mostly address situations where it is a one dimensional real value.

Furthermore, we should take into account the fact that we scarcely ever observe the actual true mapping $y = f(x)$. This is due to perturbations such as e.g. observational noise. We will rather have a joint probability $p(x, y)$. We expect this probability to be peaked for values of x and y corresponding to the mapping.

We focus on automatic learning by example. A set $\mathcal{D} = \left(x^{(i)}, y^{(i)}\right)_{i=1\ldots N}$ of data sampled from the joint distribution $p(x, y) = p(y|x)\, p(x)$ is collected. With the help of this set, we try to identify a *model* of the data, parameterised by a set of parameters w: $f_w : x \longmapsto \widehat{y}$.

$\widehat{y}$ is the estimator of y provided by the model. In the framework of this chapter, it is provided by a specific model $f_{\widehat{w}}$. $\widehat{w}$ in turn is an estimator of $\widetilde{w}$, the set of parameters for which the model is the closest to the original mapping, or *system*, f.

1.2 Learning and optimisation

The *fit* of the model to the system in a given point x is measured using a criterion representing the distance from the model prediction $\widehat{y}$ to the system, $e(y, f_w(x))$. This is the *local risk*. The performance of the model is measured by the *expected risk*:

$$G(w) = \mathrm{E}_{x,y}\left(e\left(y, f_w(x)\right)\right) = \int\int e\left(y, f_w(x)\right)\, p(y|x)\, p(x)\; dxdy \qquad (1.1)$$

This quantity represents the ability to yield good performance for all the possible situations (i.e. (x, y) pairs) and is thus called *generalisation error*. The optimal set of parameters minimises the generalisation error:

$$\widetilde{w} = \underset{w}{\operatorname{argmin}}\, G(w) \tag{1.2}$$

It is clear that no calculation, let alone optimisation, of the generalisation error is possible (except for degenerate cases). For a finite sample however, the data set provides an estimator of $G(w)$, the *empirical risk*:

$$S(w) = \widehat{\mathrm{E}}_{x,y}\left(e\left(y, f_w(x)\right)\right) = \frac{1}{N}\sum_{i=1}^{N} e\left(y^{(i)}, f_w\left(x^{(i)}\right)\right) \tag{1.3}$$

This corresponds to estimating the joint probability by the empirical density [1]: $\widehat{p}(y,x) = \frac{1}{N}\sum_{i=1}^{N}\delta\left(x - x^{(i)}\right)\delta\left(y - y^{(i)}\right)$, where $\delta(\cdot)$ is the Dirac function. Minimising (1.3) is referred to as *training* the model. The data set $\mathcal{D}$ and the empirical risk $S(w)$ are the *training set* and *training error*, respectively.

According to the law of great numbers, $S(w)$ converges towards $G(w)$ when N grows. However, this is not true for their minima: $\underset{w}{\operatorname{argmin}} S(w)$ does not converge towards $\widetilde{w}$.

1.3 Maximum likelihood

The maximum likelihood method allows us to link the risk function $e(\cdot)$ and the assumption on the noise. Let us assume that the observed output $y^{(i)}$ is the actual system output $f\left(x^{(i)}\right)$ corrupted by additive, independent Gaussian noise: $y^{(i)} = f_w\left(x^{(i)}\right) + \varepsilon$. The Gaussian assumption originates from practical considerations and is a "safe bet" in most situations. The probability that a given example $\left(x^{(i)}, y^{(i)}\right)$ was sampled from a system modelled by f_w is:

$$\mathrm{P}\left(y^{(i)}|w, x^{(i)}\right) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{\left\|y^{(i)} - f_w\left(x^{(i)}\right)\right\|^2}{2\sigma^2}\right) \tag{1.4}$$

Remember here that y is one-dimensional, so σ^2 is the scalar variance rather than a covariance matrix. As the noise is assumed independent, we obtain the likelihood of the data set $\mathcal{D}$:

$$\mathrm{P}(\mathcal{D}|w) = \prod_{i=1}^{N}\mathrm{P}\left(y^{(i)}|w, x^{(i)}\right) = \left(2\pi\sigma^2\right)^{-\frac{N}{2}}\exp\left(-\frac{\sum_{i=1}^{N}\left\|y^{(i)} - f_w\left(x^{(i)}\right)\right\|^2}{2\sigma^2}\right) \tag{1.5}$$

The most likely solution is the one for which (1.5) is the highest, i.e. the *maximum likelihood* solution, $\widehat{w} = \operatorname{argmax}_w \mathrm{P}(\mathcal{D}|w)$. Let us now take $e(\cdot,\cdot) = \|\cdot - \cdot\|^2$, the Euclidean norm, which corresponds to the least squares approach:

$$\mathrm{P}(\mathcal{D}|w) = \left(2\pi\sigma^2\right)^{-\frac{N}{2}}\exp\left(-\frac{N}{2\sigma^2}S(w)\right) \tag{1.6}$$

[1] This topic will be explored further in section 2.8.

In other words, $S(w) \propto -\ln \mathrm{P}(\mathcal{D}|w)$, so the least square and the maximum likelihood solution coincide. The least square solution is here a special case of expected risk minimisation. Equation (1.6) links the design of the local risk criterion and the assumption on the noise.

1.4 Noisy output

We have shown the link between the assumed output noise distribution and the error function. Let us now analyse the influence of this noise on the generalisation performance. Let us assume again that the system is corrupted by additive, independent noise, with 0 average and σ^2 variance: $y = f(x) + \varepsilon$. We have $p(y|x) = p(\varepsilon)$ so for a squared error:

$$G(w) = \int_x \int_\varepsilon \left(f(x) + \varepsilon - f_w(x)\right)^2 p(x)p(\varepsilon)\, d\varepsilon\, dx \tag{1.7}$$

We now recall that $\int p(\varepsilon)\, d\varepsilon = 1$, $\int \varepsilon p(\varepsilon)\, d\varepsilon = 0$ (0 mean), and $\int \varepsilon^2 p(\varepsilon)\, d\varepsilon = \sigma^2$. After some algebra, we derive the final result:

$$G(w) = \sigma^2 + \int_x \left(f(x) - f_w(x)\right)^2 p(x)\, dx \tag{1.8}$$

The difference in generalisation error between the noisy case and the noise-free case is an additive constant. This gives the following insights:

- The noise level is a lower bound on generalisation error.
- The generalisation error of a perfect model is the variance of the output noise.
- Output noise can be neglected as far as *generalisation* error is concerned. This is of course not the case with *training* error (cf. section 2.10).

1.5 Linear regression

We will here handle the particular case where the model is linear:

$$f_w(x) = x^\top . w \tag{1.9}$$

where $^\top$ is the transpose operator. We also assume that the system is linear, corrupted by additive Gaussian noise, independent of the output. We sample a number N of input-output pairs as : $y^{(i)} = x^{(i)\top} \widetilde{w} + \varepsilon^{(i)}$. We wish to estimate the ideal set $\widetilde{w}$.

Let us denote by $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{E}$ the $N \times P$, $N \times 1$ and $N \times 1$ matrices (respectively) containing the transposed input, output and noise vectors:

$$\mathbf{X} = \begin{bmatrix} x^{(1)\top} \\ x^{(2)\top} \\ \vdots \\ x^{(N)\top} \end{bmatrix} \quad \mathbf{Y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(N)} \end{bmatrix} \quad \mathbf{E} = \begin{bmatrix} \varepsilon^{(1)} \\ \varepsilon^{(2)} \\ \vdots \\ \varepsilon^{(N)} \end{bmatrix} \tag{1.10}$$

The mean squared training error is expressed simply as:

$$S(w) = \frac{1}{N}\|\mathbf{X}.w - \mathbf{Y}\|^2 \tag{1.11}$$

The linear maximum likelihood estimator is obtained by minimising (1.11). As $\nabla S(w) = \frac{2}{N}\mathbf{X}^\top(\mathbf{X}w - \mathbf{Y})$, we obtain the well-known expression of the linear regression solution:

$$\widehat{w} = \left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{X}^\top\mathbf{Y} \tag{1.12}$$

$\left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{X}^\top$ is the pseudo-inverse of matrix $\mathbf{X}$.

1.6 Algebraic estimators of errors for linear regression

This section gives insight into assessing the quality of the solution, once we obtain it with (1.12). It also is crucial as it gives the idea of the framework behind obtaining algebraic estimates of generalisation, as presented in chapter 3.

In general, $\widehat{w}$ and $\widetilde{w}$ will be different. As $\mathbf{Y} = \mathbf{X}^\top\widetilde{w} + \mathbf{E}$, the linear estimator satisfies $\widehat{w} = \widetilde{w} + \left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{X}^\top\mathbf{E}$. This leads to the expression of the training and generalisation errors:

$$S(\widehat{w}) = \frac{1}{N}\left(\mathbf{E}^\top\mathbf{E} - \mathbf{E}^\top\mathbf{X}\left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{X}^\top\mathbf{E}\right) \tag{1.13}$$

$$G(\widehat{w}) = \sigma^2 + \int_x \left[x^\top\left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{X}^\top\mathbf{E}\mathbf{E}^\top\mathbf{X}\left(\mathbf{X}^\top\mathbf{X}\right)^{-1}x\right] p(x)\,dx \tag{1.14}$$

In order to get rid of the fluctuations introduced by the use of a particular sample as a training set, we will average (1.13) and (1.14) over all possible training sets $\mathcal{D}$ *of size* N. This means averaging over the input sample, and over the noise sample, with $\langle\cdot\rangle_{\mathcal{D}} = \langle\langle\cdot\rangle_\varepsilon\rangle_x = \langle\langle\cdot\rangle_x\rangle_\varepsilon$. In order to perform the calculations, recall that $\langle\mathbf{E}\mathbf{E}^\top\rangle_\varepsilon = \sigma^2\mathbf{I}_N$ and for a square matrix $\mathbf{M}$, $\langle\mathbf{E}^\top\mathbf{M}\mathbf{E}\rangle_\varepsilon = \sigma^2\mathrm{tr}(\mathbf{M})$. Furthermore, we note m_{kl} the elements of $\left(\mathbf{X}^\top\mathbf{X}\right)^{-1}$:

$$\mathrm{tr}\left(\mathbf{X}\left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{X}^\top\right) = \sum_{i=1}^{N}\sum_{k=1}^{P}\sum_{l=1}^{P} x_k^{(i)}x_l^{(i)}m_{kl} = P \tag{1.15}$$

This arises from the definition of the m_{kl}, and leads to the average training error:

$$\langle S\rangle_{\mathcal{D}} = \sigma^2\left(1 - \frac{P}{N}\right) \tag{1.16}$$

For the generalisation error, let us denote by $\mathbf{H}_S$ and $\mathbf{H}_G$ the Hessian matrices of the training and generalisation errors (respectively).

$$\langle G\rangle_{\mathcal{D}} = \sigma^2\left(1 + \int_x x^\top\left\langle\left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\right\rangle_x x\,p(x)\,dx\right) = \sigma^2\left(1 + \frac{\mathrm{tr}\left(\mathbf{H}_G\left\langle\mathbf{H}_S^{-1}\right\rangle_x\right)}{N}\right) \tag{1.17}$$

By definition of the generalisation error, $\langle \mathbf{H_S} \rangle_x = \mathbf{H_G}$. This equivalence translates to the inverse, to a first order approximation: $\left\langle \mathbf{H_S}^{-1} \right\rangle_x = \mathbf{H_G}^{-1} + \mathcal{O}\left(\frac{P+1}{N}\right)$, leading to the *Final Prediction Error* or *FPE*:

$$\langle G \rangle_{\mathcal{D}} = \sigma^2 \left(1 + \frac{P}{N}\right) = \frac{N+P}{N-P} \langle S \rangle_{\mathcal{D}} \tag{1.18}$$

This result calls for some clarification. Consider that each parameter is a *degree of freedom*. The theoretically optimal error is the noise variance σ^2. However, as the training set is finite, each of these parameters will tend to over-fit the data, leading to a *downward* bias of size σ^2/N in the training error. By reaction, the generalisation performance worsens, producing an *upward* bias of similar amplitude in the generalisation error.

1.7 Non-linear regression

Let us now switch to the more general case of a non-linear model f_w. Using the squared error, the empirical risk is expressed as:

$$S(w) = \frac{1}{N} \sum_{i=1}^{N} \left(y^{(i)} - f_w\left(x^{(i)}\right) \right)^2 \tag{1.19}$$

Unlike the linear case above, there is no analytical solution for the minimisation of (1.19). Finding $\widehat{w}$ is just a standard optimisation problem, i.e. a *non-linear regression*. We wish to either minimise $S(w)$ or find $\nabla S(w) = 0$. The gradient of the training cost (1.19) is:

$$\nabla S(w) = \frac{2}{N} \sum_{i=1}^{N} \mathbf{J}_{f_w}(w) \left(y^{(i)} - f_w\left(x^{(i)}\right) \right) \tag{1.20}$$

For a M-dimensional function f_w, $\mathbf{J}_{f_w}$ is the $M \times P$ *Jacobian* matrix calculated in w. For a one-dimensional model, $\mathbf{J}_{f_w} = \nabla f_w(w)$.

In the next section, we briefly present neural network models, a class of non-linear models for which the first order information is easily computed.

1.8 Neural Networks

Generally speaking, a neural networks is a set of inter-connected cells, called *neurons*, performing a simple non-linear transformation of their weighted inputs. They also go by the name of *artificial neural networks* or *connectionist models*.

We will here only consider a subset of neural networks: the multi-layer perceptrons (MLP). It can be represented by a number of ordered layers with forward connections between them. The estimation is straightforward as there are no backwards (i.e.

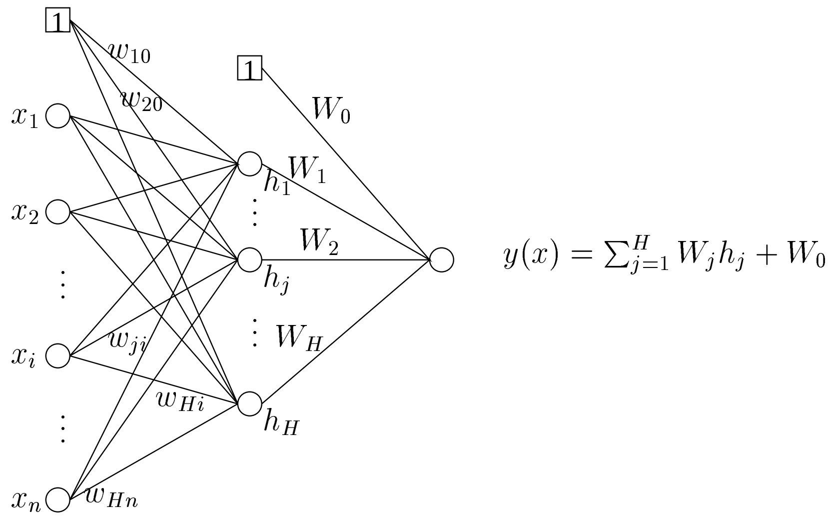


Figure 1.1: Multi-layer perceptrons with one hidden layer. The square cells are biases, the round cells have either linear (input and output layers) or sigmoïd (hidden layer) transfer functions.

recurrent) connections. A MLP with n inputs, one hidden layer containing H units and one output is represented in figure 1.1. It implements the mapping:

$$f_w(x) = \sum_{j=1}^{H} W_j h \left(\sum_{i=1}^{n} w_{ji} x_i + w_{j0} \right) + W_0 \tag{1.21}$$

The transfer function h in the input layer is usually a non-linear, increasing, bounded function such as the hyperbolic tangent (tanh), the error function (erf) or the simple sigmoïd $\frac{1}{1+\mathrm{e}^{-x}}$. For regression problems, the output cell is kept linear, while for classification it is customary to take a non-linear bounded function again.

All weights w_{ji} and W_j are gathered in weight vector w containing the $P = (n+2)H + 1$ parameters. The choice of the number of input and output units is guided by the problem. In time series modelling for example, we will have one output (the forecast) and as many inputs as past values we involve in the prediction. On the other hand, the number of hidden units is problem dependent, and a central preoccupation in neural computation. It is linked with the model capacity as shown in chapters 2 and 3.

Neural Networks are interesting for several reasons:

- They provide a convenient generic non-linear model.
- They can approximate any reasonable function arbitrarily closely.
- They are rather simple to implement.
- *Back-propagation* allows for a convenient estimation of $\widehat{w}$ through standard optimisation methods.

1.9 Back-Propagation

The *back-propagation* rule is the application of the chain rule of derivation to the MLP. We present the derivation for a one output, one-hidden layer case, but the generalisation is straightforward.

The local quadratic cost for example k is $S_k(w) = \left(y^{(i)} - y\left(x^{(k)}\right)\right)^2$. For ease of calculation, we introduce the output h_j of hidden unit j, and its input $p_j = h^{-1}(h_j)$. The derivative of the cost with respect to the input of the hidden layer and the input layer are:

$$\frac{\partial S_k}{\partial p_j} = \frac{\partial h_j}{\partial p_j}\frac{\partial y}{\partial h_j}\frac{\partial S_k}{\partial y} = 2h'(p_j)\, W_j \left(y^{(k)} - y\left(x^{(k)}\right)\right) \tag{1.22}$$

$$\frac{\partial S_k}{\partial x_i} = \sum_{j=1}^{H} \frac{\partial p_j}{\partial x_i}\frac{\partial S_k}{\partial p_j} = \sum_{j=1}^{H} w_{ji}\frac{\partial S_k}{\partial p_j} \tag{1.23}$$

The derivatives with respect to the parameters are:

$$\frac{\partial S_k}{\partial W_j} = \frac{\partial y}{\partial W_j}\frac{\partial S_k}{\partial y} = 2h_j \left(y^{(k)} - y\left(x^{(k)}\right)\right) \tag{1.24}$$

$$\frac{\partial S_k}{\partial w_{ji}} = \frac{\partial p_j}{\partial w_{ji}}\frac{\partial S_k}{\partial p_j} = x_i \frac{\partial S_k}{\partial p_j} \tag{1.25}$$

The calculation of the neural network estimation is done by a forward pass, giving h_j and $y(x)$ given the x_i, w_{ji} and W_j. The calculation of the derivatives of the cost with respect to the parameters is done by *back-propagation*, using (1.22) and (1.23), then (1.24) and (1.25).

The first derivatives with respect to the parameters allow the use of first order optimisation method to minimise $S(w)$ (and even some approximations of second order methods). The first derivatives with respect to the inputs allow the use of a neural network as a part in a modular system.

1.10 Quadratic approximation

For one-dimensional optimisation, the basic techniques are e.g. golden search or parabolic interpolation. These are not directly relevant to numerical learning, though they are crucial when performing a *line search*, i.e. minimisation along a given search direction. The methods presented below are relevant to the minimisation of a function of P parameter, i.e. in a P-dimensional space.

Given a cost function $C(w)$, let us consider the quadratic expansion around $\widehat{w}$:

$$C(w) = C(\widehat{w}) + \frac{1}{2}(w - \widehat{w})^\top \mathbf{H}\,(w - \widehat{w}) \tag{1.26}$$

$\mathbf{H}$ is the Hessian matrix of the cost function. This expression relates to the use of a *quadratic* cost. It is exact for linear models, approximate for others. However, just as the parabolic interpolation allows to minimise one-dimensional functions that are not necessarily parabolas, we expect the quadratic approximation to yield acceptable results in a fairly broad context.

1.11 Steepest descent

The idea of *steepest descent* is to search at every iteration along the direction of steepest descent, i.e. the gradient:

$$w_{t+1} = w_t - \eta_t \nabla C(w_t) \tag{1.27}$$

η_t is a gain parameter that can be set in several ways. Consider for example:

- setting η_t to a constant parameter.
- finding the "optimal" rate $\widehat{\eta}_t = \text{argmin}_\eta\, C(w_t - \eta \nabla C(w_t))$.

The rate of convergence of an algorithm is defined as the speed at which it gets closer to the solution, i.e. the ration between the distance to the solution at time t and the distance to the solution at time $t-1$. For the steepest descent algorithm, it will later give some insight into the inefficiency of this scheme:

Proposition 1.1 *Consider a quadratic error function with constant Hessian* $\mathbf{H}$ *of condition number* c. *The rate of convergence of* w_t *towards the minimum* $\widehat{w}$ *using the Steepest Descent algorithm is given by:*

$$\frac{C(w_t) - C(\widehat{w})}{C(w_0) - C(\widehat{w})} \leq \left(\frac{c-1}{c+1}\right)^{2t} \tag{1.28}$$

The condition number c of a matrix is the ratio of its largest eigenvalue and its smallest one. The Hessian $\mathbf{H}$ is usually rather ill-conditioned, i.e. has a large condition number, so that the ratio in (1.28) is close to one. Physically, the cost surface has a very narrow valley around $\widehat{w}$, so the algorithm jumps from one side to the other of the valley in very short orthogonal steps.

Several heuristic improvements of this schemes have been made to counter this effect, such the addition of a momentum term. However, rather than customising a poor learning technique, we advocate the use of a more decent optimisation technique.

1.12 Stochastic gradient descent

The stochastic version of the gradient descent algorithm, also called *on-line* gradient descent, can lead to drastic improvement in convergence when it is properly used.

The stochastic algorithm consists in updating the parameters on the basis of the gradient of the *local* cost:

$$w_{t+1} = w_t - \eta_{t+1} \nabla C_k(w) \tag{1.29}$$

where C_k is the local cost for pattern k, i.e. $\left\| y^{(k)} - f_{w_t}\left(x^{(k)}\right) \right\|^2$. The choice of the pattern k to use in step t is done in a stochastic manner, taking the training examples at random. This scheme is thus well-suited to on-line adaptation of the weights. On the other hand, the previous algorithm uses the computation of the gradient on the entire training set, making it better suited for *off-line* training.

A particularly sensitive issue in the stochastic design is the setting of the learning rate as the optimisation proceeds. The following proposition addresses this issue:

Proposition 1.2 *For a cost function with mild regularity conditions (satisfied by the quadratic cost), the stochastic gradient descent algorithm (1.29) converges when the sequence of positive gains η_t satisfies:*

$$\sum_{t=1}^{+\infty} \eta_t = \infty \quad \textit{and} \quad \sum_{t=1}^{+\infty} \eta_t^2 < +\infty \tag{1.30}$$

The *mild regularity conditions* mentioned in proposition 1.2 are the fact that the global cost has a number of (bounded) derivatives, and a couple of conditions on the derivative of the local cost.

The stochastic gradient descent proves very efficient when used in a suitable manner, especially when the η_t are set in a proper way. Unfortunately, this algorithm is commonly misused in the neural network literature, because it is often presented as an *ad hoc* procedure with no reference to the stochastic optimisation literature. It is for example common to find a constant setting for η_t, in contradiction with condition (1.30).

Finally, let us mention that stochastic gradient descent is recommended for problems such as large scale classification, where the training set is large, and often redundant. It is generally less well suited to small scale problems and regression estimation. In such cases, the conjugate gradient algorithm presented below usually yields better performance.

1.13 Conjugate gradient

The conjugate gradient method increases the efficiency of the optimisation by avoiding the oscillation of the steepest descent method in ill-conditioned cases.

Let us first recall that two vectors u and v are said *conjugate* with respect to matrix A if $u^\top \mathbf{A} v = v^\top \mathbf{A} u = 0$, i.e. they are orthogonal in the sense of the quadratic form $\mathbf{A}$.

As we have noticed earlier, the quadratic approximation (1.26) involves the Hessian matrix of the cost function. Furthermore, it would be convenient that once a minimisation step is completed, the next search direction does not interfere with the previous one (in order not to undo what has just been done) i.e. is restricted to the conjugate hyper-plane, which means that the old and the new search direction are kept conjugate with respect to the Hessian $\mathbf{H}$.

We will then construct a set h_t of successive conjugate search directions together with a set g_t of gradient directions. If w_t is the approximation of the location of the minimum of $C(w)$ at step t, we have:

$$g_{t+1} = \nabla C\left(w_t\right) = \mathbf{H}\left(w_t - \widehat{w}\right) \tag{1.31}$$

The new search direction will be taken as a combination of the gradient direction and the previous search direction[2]:

$$h_{t+1} = g_{t+1} - \gamma_{t+1} h_t \tag{1.32}$$

where γ_{t+1} is a scalar. Let us now recall that the h_i are a conjugate family of vectors, i.e. in particular ${h_{t+1}}^\top \mathbf{H} h_t = {h_{t-1}}^\top \mathbf{H} h_t = 0$. This property is used together with (1.32) at step $t+1$ and t to obtain:

$$\begin{aligned} 0 &= {g_{t+1}}^\top \mathbf{H} h_t - \gamma_{t+1} {h_t}^\top \mathbf{H} h_t \\ {h_t}^\top \mathbf{H} h_t &= {g_t}^\top \mathbf{H} h_t \end{aligned}$$

leading to the value of γ_{t+1}:

$$\gamma_{t+1} = \frac{{g_{t+1}}^\top \mathbf{H} h_t}{{g_t}^\top \mathbf{H} h_t} \tag{1.33}$$

This expression is still unsatisfactory as it involves the Hessian matrix, the calculation of which we seek to avoid. We just have to notice that, as w_t is obtained by a search along the direction h_t starting from w_{t-1}, there exists a value α_t such that $w_t - w_{t-1} = \alpha_t h_t$. From (1.31), we then get:

$$g_{t+1} - g_t = \mathbf{H}\left(w_t - w_{t-1}\right) = \alpha_t \mathbf{H} h_t \tag{1.34}$$

This leads to a more convenient expression for γ_{t+1}:

$$\gamma_{t+1} = \frac{{g_{t+1}}^\top \left(g_{t+1} - g_t\right)}{{g_t}^\top \left(g_{t+1} - g_t\right)} \tag{1.35}$$

Another consequence of the line search along h_t is that the gradient of C in w_t is orthogonal to the search direction, i.e. ${h_t}^\top g_{t+1} = 0$. Using (1.34) and (1.32), this can be extended to :

$${h_i}^\top g_j = {g_i}^\top g_j = 0 \qquad i \neq j \tag{1.36}$$

Equation (1.36) allows several rewritings of (1.35):

$$\gamma_{t+1} = -\frac{{g_{t+1}}^\top g_{t+1}}{{g_t}^\top g_t} \qquad \text{Fletcher-Reeves} \tag{1.37}$$

$$\gamma_{t+1} = -\frac{{g_{t+1}}^\top \left(g_{t+1} - g_t\right)}{{g_t}^\top g_t} \qquad \text{Polak-Ribière} \tag{1.38}$$

$$\gamma_{t+1} = -\frac{{g_{t+1}}^\top \left(g_{t+1} - g_t\right)}{{h_t}^\top g_t} \qquad \text{Hestenes-Stiefel} \tag{1.39}$$

According to (1.36), equations (1.38) and (1.39) could be simplified. The effects of the three formulas are indeed the same when the cost is exactly quadratic. On the other hand, in real cases such as non-linear regression, this is not the case, and the Hessian is not constant. (1.38) and (1.39) then have a better behaviour as they *restart* the algorithm by resetting the search direction to the gradient of the cost function. Indeed, when two successive gradient directions are too close, we have $g_{t+1} \approx g_t$ leading to $h_{t+1} \approx g_{t+1}$.

The conjugate gradient algorithm applied to a P-dimensional quadratic cost function finds the *exact* minimum in *at most* P iterations. Furthermore, the rate of convergence is given in the following proposition:

[2]What we actually want is the projection of the gradient direction on the hyper-plane conjugate with the previous search direction. This is done by subtracting the right component along h_t.

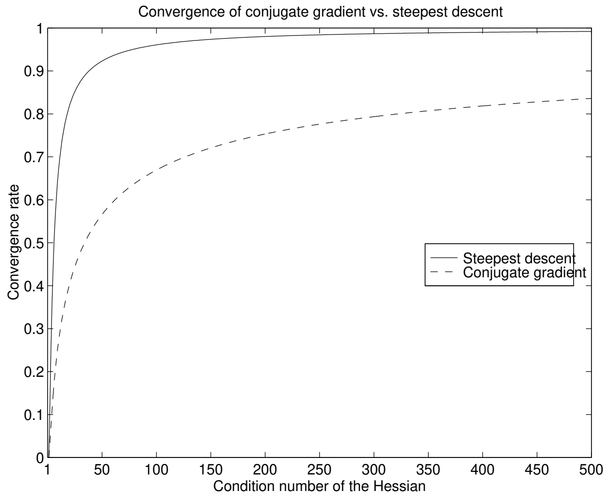


Figure 1.2: Comparison of the convergence rates of the conjugate gradient (dashed) and steepest descent (solid) algorithms, as a function of the condition number. If the condition number is 200, w_t gets 2% closer to $\widehat{w}$ with every steepest descent iteration, and 25% closer with a conjugate gradient iteration.

Proposition 1.3 *Consider a quadratic error function with constant Hessian* $\mathbf{H}$ *of condition number* c*. The rate of convergence of* w_t *towards the minimum* $\widehat{w}$ *using the Conjugate Gradient algorithm is given by:*

$$\frac{C(w_t) - C(\widehat{w})}{C(w_0) - C(\widehat{w})} \leq 4\left(\frac{\sqrt{c}-1}{\sqrt{c}+1}\right)^{2t} \tag{1.40}$$

Proposition 1.3 shows that beside an initial factor of 4, the convergence of the conjugate gradient algorithm is much faster than using a steepest descent. This is illustrated by figure 1.2. For high condition numbers, there is virtually no convergence of the steepest descent: the rate $\frac{c-1}{c+1}$ is close to 1, so that according to (1.28) the distance to the solution decreases extremely slowly. On the other hand, the conjugate gradient keeps decent convergence properties even with very ill-conditioned matrices.

As an example, let us consider a Hessian with a condition number of $c = 350$. After 10 iterations, the steepest descent algorithm leads to a decrease in excess error of $1 - \left(\frac{c-1}{c+1}\right)^{20} \approx 0.1$ i.e. 10%. Using a conjugate gradient, we get $1 - 4\left(\frac{\sqrt{c}-1}{\sqrt{c}+1}\right)^{20} \approx 0.53$ hence a reduction of more than half the possible decrease in error. It takes 21 iterations to get within 5% of the minimum with conjugate gradient, more than 500 using the steepest descent.

Finally, it should be noted that the conjugate gradient optimisation scheme does not depend on the setting of an extra parameter such as the learning rate in both steepest descent and stochastic gradient descent.

1.14 Newton and quasi-Newton

We will now go back to the quadratic approximation formula (1.26). Let us first recall from (1.31) the derivative in w_t of the cost:

$$\nabla c(w_t) = \mathbf{H}(w_t - \widehat{w})$$

leading to the direct estimation:

$$\widehat{w} = w_t - \mathbf{H}^{-1}\nabla c(w_t) \tag{1.41}$$

This is similar to the Newton method for finding roots in one-dimensional problems. Formula (1.41) is valid for an exactly quadratic cost function. In the general case, the *Newton algorithm in multi-dimensions* consists in choosing the search direction accordingly:

$$w_{t+1} = w_t - \eta_{t+1}\mathbf{H}^{-1}\nabla c(w_t) \tag{1.42}$$

where η_{t+1} is set by a simple *line search.*

Expressed as it is, this second order method requires the calculation of the Hessian of the cost. Unfortunately, the full Hessian calculation and inversion is usually too costly to be reasonable in the context of non-linear models such as Neural Networks. A few shortcuts, however, allow to use the basic idea of the Newton algorithm.

The *quasi-Newton* method consists in approximating the Hessian so it is easier to calculate. Indeed, consider the expression of the cost with quadratic error in (1.19). We can rewrite this quantity as $S(w) = \frac{1}{N}\sum_{i=1}^{N} \varepsilon_i(w)^2$, so that the Hessian becomes:

$$\mathbf{H}_S(w) = \frac{2}{N}\sum_{i=1}^{N}\left(\varepsilon_i(w)\,\mathbf{H}_\varepsilon(w) + \nabla\varepsilon_i(w)\,\nabla\varepsilon_i(w)^\top\right) \tag{1.43}$$

We will now neglect the first term under the sum, which is hard to estimate. Furthermore, it is potentially problematic, as it could lead to a non positive definite Hessian.

This is the *Gauss-Newton approximation*, and the Hessian becomes:

$$\mathbf{H}_S(w) = \frac{2}{N}\sum_{i=1}^{N}\nabla\varepsilon_i(w)\,\nabla\varepsilon_i(w)^\top \tag{1.44}$$

The inverse of the approximate Hessian can be computed using the *Sherman-Morrison* inversion identity by the following iterative formula:

$$\mathbf{H}_i^{-1} = \mathbf{H}_{i-1}^{-1} - \frac{\mathbf{H}_{i-1}^{-1}\nabla\varepsilon_i(w)\,\nabla\varepsilon_i(w)^\top\mathbf{H}_{i-1}^{-1}}{1+\nabla\varepsilon_i(w)^\top\mathbf{H}_{i-1}^{-1}\nabla\varepsilon_i(w)} \qquad i = 1\dots N \tag{1.45}$$

and $\mathbf{H}_0$ is chosen as a matrix the inverse of which is easy to calculate. After N iterations, $\mathbf{H}_N^{-1}$ approximates the inverse of the approximate Hessian.

Another method for computing the approximate inverse Hessian information needed for the Newton algorithm is the *one-step memory-less Broyden-Fletcher-Goldfard-Shanno* method (BFGS). Let us first introduce $y_t = \nabla S(w_{t+1}) - \nabla S(w_t)$ and $s_t = w_{t+1} - w_t$. The *positive definite secant update* for the inverse Hessian is:

$$\mathbf{H}_{t+1}^{-1} = \mathbf{H}_t^{-1} + \frac{s_t s_t^\top}{y_t^\top s_t}\left(1 + \frac{y_t^\top\mathbf{H}_t^{-1}y_t}{y_t^\top s_t}\right) - \frac{\mathbf{H}_t^{-1}y_t s_t^\top + s_t y_t^\top\mathbf{H}_t^{-1}}{y_t^\top s_t} \tag{1.46}$$

This expression ensures that the Hessian and its inverse are symmetric positive definite. The costly aspect of the update is to keep the entire inverse Hessian in memory. To avoid this, the update rule (1.46) is applied at each time step with the (crude) approximation that $\mathbf{H_t} \approx \mathbf{I}$, leading to an actually memory-less method.

For an exact line search, BFGS is equivalent to the conjugate gradient algorithm.

1.15 Bias–Variance decomposition

The post-learning generalisation error, as defined in (1.1), is dependent upon the training set. It is indeed the minimisation of the training error for this specific set that leads to the estimate of the parameters, $\widehat{w}$.

In order to assess the generalisation abilities of a given model independently of the training sample, it is suitable to consider the expected generalisation error[3]:

$$\langle G(\widehat{w}) \rangle_{\mathcal{D}} = \int \left\langle \left(f(x) - f_{\widehat{w}}(x) \right)^2 \right\rangle_{\mathcal{D}} p(x)\ dx \tag{1.47}$$

where the average is performed over all possible training sets, as in section 1.6.

Let us now introduce $f_{\mathcal{D}}(x) = \langle f_{\widehat{w}}(x) \rangle_{\mathcal{D}}$ in the squared term:

$$\langle G(\widehat{w}) \rangle_{\mathcal{D}} = \int \left\langle \left(f(x) - f_{\mathcal{D}}(x) + f_{\mathcal{D}}(x) - f_{\widehat{w}}(x) \right)^2 \right\rangle_{\mathcal{D}} p(x)\ dx \tag{1.48}$$

When this expression is developed, the cross-product disappears when taking its expectation, leaving:

$$\begin{aligned} \langle G(\widehat{w}) \rangle_{\mathcal{D}} &= \int \left(f(x) - f_{\mathcal{D}}(x) \right)^2 p(x)\ dx \\ &+ \int \left\langle \left(f_{\widehat{w}}(x) - f_{\mathcal{D}}(x) \right)^2 \right\rangle_{\mathcal{D}} p(x)\ dx \end{aligned} \tag{1.49}$$

The first part is the squared deviation between the system and the expectation of the model over all possible training. It is called the squared *bias* as it measures the intrinsic bias the model has. The second part is the average of the squared fluctuations of the model around its expectation, i.e. the *variance*.

Accordingly, the generalisation error can be separated into three components:

1. The noise level (cf. section 1.4),
2. The squared bias induced by the choice of a model,
3. The variance coming from the sampling.

Let us notice that if there exist a set of parameters w^* such that for all x, $f_{w^*} = \langle f_{\widehat{w}}(x) \rangle$, then by direct application of the above formulas, w^* minimises the generalisation error and f_{w^*} is the optimal model. However, there is no guarantee that such a model exists, as the models do not necessarily constitute a vector space. Furthermore, even if such a model exists, it may not be possible to obtain it by minimisation of the empirical cost on a given set of data.

[3]the noise term has been discarded here according to section 1.4. It will be taken into account in the analysis below.

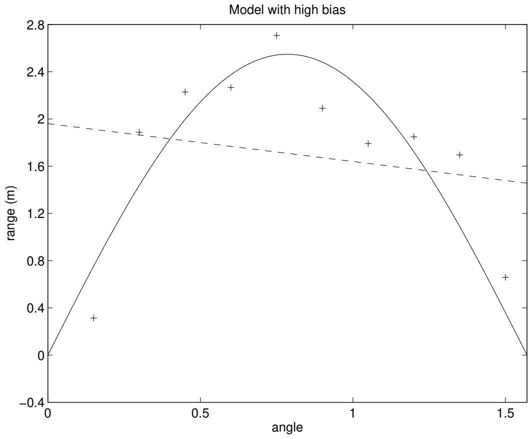


Figure 1.3: Linear modelling of the relationship between the initial angle and the range. The crosses are the data, the solid line is the theoretical relationship, and the dashed line is the linear model obtained from the data.

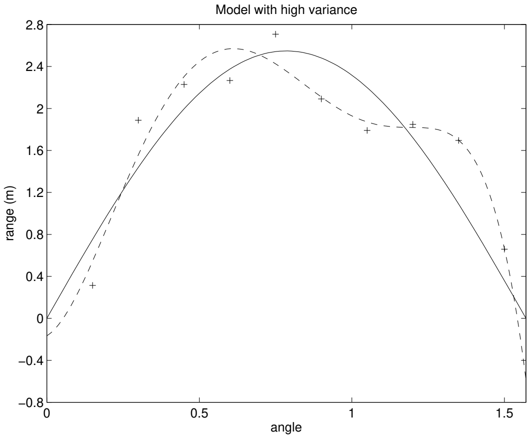


Figure 1.4: Same as figure 1.3, but using a 7th order polynomial modelling. The model is now highly dependent on the data.

1.16 Limits of a simple regression

The bias–variance decomposition gives insight into a possible pitfall of the simple empirical cost minimisation as suggested in this chapter.

Let us consider that we perform 10 experiments where we measure the range of a projectile shot at various angles with a given speed of $5\,m.s^{-1}$. With no perturbation, the theoretical link between the angle α_i and the range d_i is given by the formula $d_i = \frac{25}{g}\sin 2\alpha_i$ with $g = 9{,}81\,m.s^{-2}$.

However, consider now that we attempt to model this relationship using a simple linear model. The result for a given experiment is displayed in figure 1.3. It is likely that the slope of the line would not change drastically for different sets of points: the variance of the estimate is very low. On the other hand, the bias is high because the model is intrinsically incapable of modelling the right relationship.

On the other hand, when we perform a polynomial regression with a 7th order polynomial, the model seems to somehow get closer to the data. So close that we would expect the average estimation to be the average of the data, i.e. the actual theoretical curve. The bias is thus very low, but the variance is high as the curve tends to over-fit the data.

This simple example shows, with the help of the bias-variance decomposition, that optimisation of the empirical cost does not guarantee the success of the learning procedure. In the next chapter, this will be detailed, and regularisation is introduced as a possible solution.

COMMENTS

1.1 The importance of modelling appears in many fields. Let us cite for example system identification (Nørgaard, 1996), control (Goutte and Ledoux, 1995), or time series modelling (Kouam, 1993). This thesis will focus on system identification and time series. That is why we deal here with univariate outputs.

1.2 The use of optimisation for learning is connected to the study of adaptive systems such as e.g. (Tsypkin and Nikolic, 1971). See also (Bottou, 1991) in the context of neural networks. The definition of empirical and expected risk is at the basis of statistical learning theory (see e.g. Vapnik, 1995). Some authors favour *off-training-set error* (see Wolpert, 1996) (in our case however the two notions meet). The consistency of the minimum of the empirical risk has been studied by White (1989).

1.3 The maximum likelihood method was developed by Fisher in the 1920's for estimating the unknown parameters of a density. Maximum likelihood estimators are used in numerous fields such as parameter estimation, signal processing, equalization, etc. The probabilistic point of view will be extended in chapter 5.

1.4 This is a standard result. A similar derivation was presented by Bishop (1995b).

1.5 The topic of linear regression is extensively covered in many textbooks on statistics. It is a basic tool, sometimes used in different contexts. In section B.2 for example, we evoke the use of a local linear fit in a non-parametric estimator.

1.6 The error estimator derived in this section is due to Akaike (1969). Similar estimators are derived in the non-linear case, using a locally linear approximation (i.e. a quadratic approximation to the cost function). For a fast derivation, see e.g. Rasmussen (1993).

1.8 Despite a wealth of literature on neural networks, few books offer a truly sound introduction to this class of models. (Hertz et al., 1991) has been acclaimed for some time as *the* reference in the field. It has now been joined by the books by Bishop (1995a) and Ripley (1996). Incidentally, these books adopt a statistical perspective.

1.9 The *back-propagation* rule is usually credited to Werbos (1974) or Rumelhart et al. (1986). However, it was discovered earlier in different contexts. Vapnik (1995) mentions its use in (Bryson et al., 1963) for solving some control problems. Bottou (1991) cites Amari (1967) in the context of adaptive systems and notes that it is nothing more than proper application of the derivation rules invented by Leibnitz in the 17th century.

1.10 Optimisation methods are covered in many books such as e.g. (Fletcher, 1987) or (Press et al., 1992) for one-dimensional and multi-dimensional techniques. The quadratic approximation is necessary to handle non linear problems. However, methods presented here are also used in the linear case. Indeed, iterative minimisation methods are a common alternative to the computationally expensive matrix inversion in equation (1.12).

1.11 According to Press et al. (1992), the steepest descent algorithm dates back to Cauchy. Its convergence rate is a standard consideration in numerical analysis. It is derived, e.g. by Møller (1993a). The convergence problem for ill-conditioned Hessians is illustrated e.g. in (Møller, 1993a), page 16 and (Press et al., 1992), page 421.

1.12 The stochastic algorithm and the associated convergence conditions date back to Robbins and Munro (1951). This algorithm was applied to neural networks at the end of the eighties by Rumelhart et al. (1986). A good presentation including demonstrations of the convergence of the algorithm is given in (Bottou, 1991).

1.13 The conjugate gradient algorithm is studied extensively in the linear optimisation literature. A neural network perspective is offered by Møller (1993a). The same author mentions several extensions of the conjugate gradient algorithm designed to handle stochastic training, and proposes one in (Møller, 1993b).

1.14 Second order methods are presented by Battiti (1992), who confirms the equivalence of BFGS and conjugate gradient. Equation (1.46) is a simplification of the update presented by Battiti (1992), page 157. It corrects the expression given by Møller (1993a), page 43.

1.15 The bias-variance decomposition is a classical tool is statistics, see chapter B for some non-parametric cases. It has been introduced to the neural networks literature by Geman et al. (1992). Contrary to the regression estimation case, the expression of this decomposition for classification problems is an open, and very active, area of research, see (Tibshirani, 1996; Friedman, 1996) among others.

1.16 The trade-off between bias and variance and its influence on learning will also be illustrated in chapter B for non-parametric models.

References

Akaike, H. (1969). Fitting autoregressive models for prediction. *Annals of the Institute of Statistical Mathematics*, 21:243–247.

Amari, S. I. (1967). A theory of adaptive pattern classifiers. *IEEE Transactions on Electronic Computers*, 16:279–307.

Battiti, R. (1992). First- and second-order methods for learning: Between steepest descent and newton's method. *Neural Computation*, 4(2):141–166.

Bishop, C. M. (1995a). *Neural Networks for pattern recognition.* Clarendon Press, Oxford.

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

Bottou, L. (1991). *Une approche théorique de l'apprentissage connexionniste : application à la reconnaissance de la parole.* Thèse de doctorat, Université Paris XI, Orsay.

Bryson, A., Denham, W., and Dreyfuss, S. (1963). Optimal programming problem with inequality constraints. I: Necessary conditions for extremal solutions. *AIAA journal*, 1:25–44.

Fletcher, R. (1987). *Practical Methods of Optimization.* Wiley.

Friedman, J. H. (1996). On bias, variance, 0/1 - loss, and the curse-of-dimensionality. Technical report, Department of Statistics, Stanford University. ftp://playfair.stanford.edu/pub/friedman/curse.ps.Z.

Geman, S., Bienenstock, E., and Doursat, R. (1992). Neural networks and the bias/variance dilemna. *Neural Computation*, 4(1):1–58.

Goutte, C. and Ledoux, C. (1995). Synthèse des techniques de commande connexionniste. Technical Report 95/02, LAFORIA.

Hertz, J., Krogh, A., and Palmer, R. G. (1991). *Introduction to the theory of neural computation.* Addison-Wesley.

Kouam, A. (1993). *Approches connexionnistes pour la prévision des séries temporelles.* PhD thesis, Université de Paris Sud.

Møller, M. (1993a). *Efficient Training of Feed-Forward Neural Networks.* PhD thesis, Computer Science department, Aarhus University.

Møller, M. (1993b). Supervised learning on large redundant training sets. *International Journal of Neural Systems*, 4(1):15–25.

Nørgaard, P. M. (1996). *System identification and control with neural networks.* PhD thesis, Department of Automation, Technical University of Denmark.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (1992). *Numerical Recipes in C.* Cambridge University Press, 2nd edition.

Rasmussen, C. E. (1993). Generalization in neural networks. Master's thesis, Electronics institute, Technical University of Denmark.

Ripley, B. D. (1996). *Pattern Recognition and Neural Networks.* Cambridge University Press.

Robbins, H. and Munro, S. (1951). A stochastic approximation method. *Annals of Math. Stat.*, 22:400–407.

Rumelhart, D., Hinton, G., and Williams, R. (1986). Learning internal representation by error propagation. In Rumelhart, D. and McClellan, J., editors, *Parallel Distributed Processing : exploring the microstructure of cognition*, volume 1, pages 318–362. MIT Press.

Tibshirani, R. (1996). Bias, variance and prediction error for classification rules. Technical report, University of Toronto. http://utstat.toronto.edu/pub/tibs/biasvar.ps.

Tsypkin, Y. Z. and Nikolic, Z. J. (1971). *Adaptation and learning in automatic systems*, volume 73 of *Mathematics in science and engineering*. Academic Press, New York and London.

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

Werbos, P. J. (1974). *Beyond regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, Harvard University, Cambridge, MA.

White, H. (1989). Learning in artificial neural networks: A statistical perspective. *Neural Computation*, 1:425–464.

Wolpert, D. H. (1996). The lack of a priori distinctions between learning algorithms. *Neural Computation*, 8(8):1341–1390.

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

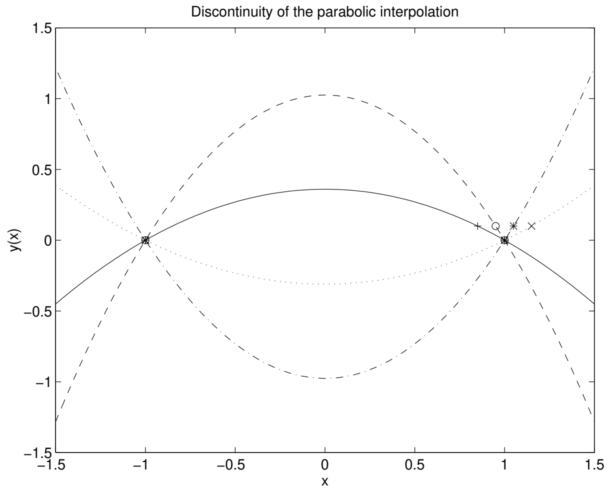


Figure 2.1: 4 different 3-points parabolic interpolations for four very close data sets display a discontinuity in the solution. Two points are held fixed, and very small variations in the third one provoke large variations of the solution.

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 $\min C\left(f_i\right) \rightarrow \min C\left(f\right)$, then likewise $f_i \rightarrow f$. 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:

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

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:

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

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 $R\left(f\right)$.

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

[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* ξ also called *hyper-parameter*[3] implicitly defines a structure on the possible models by constraining the model. Roughly, low values of ξ 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 ξ.

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.

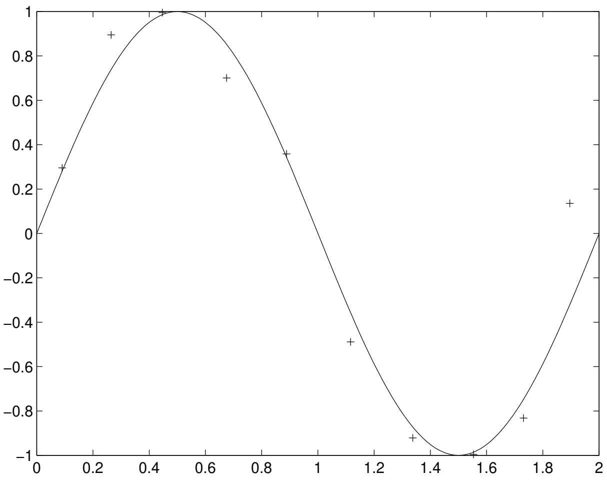


Figure 2.2: The 10 data points are displayed together with the sinusoid from which they were sampled, with low noise added (variance 0.025).

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

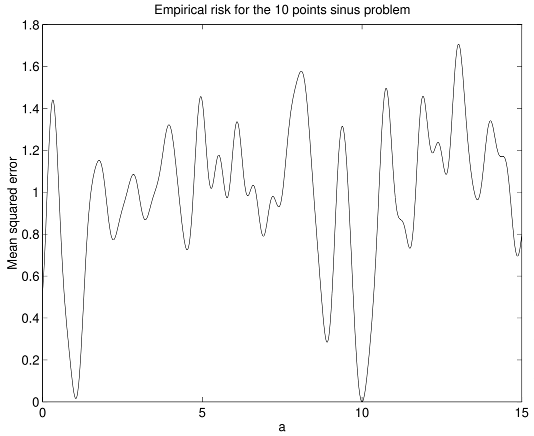


Figure 2.3: The mean squared error calculated on the 10 points for a between 0 and 15 displays two deep minima: one for $a = 1.04$, the other one for $a = 10$.

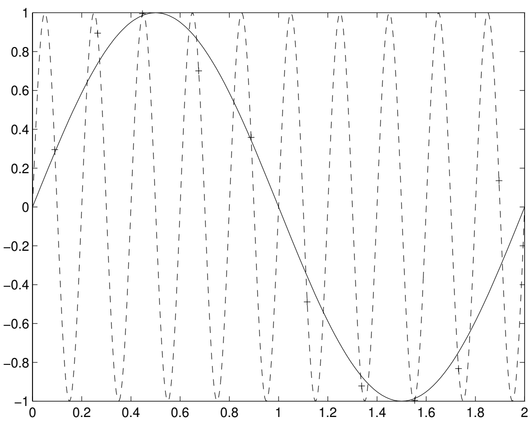


Figure 2.4: Solution of the mean squared error minimisation (dashed) together with the data points (+) and the underlying mapping (solid).

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!

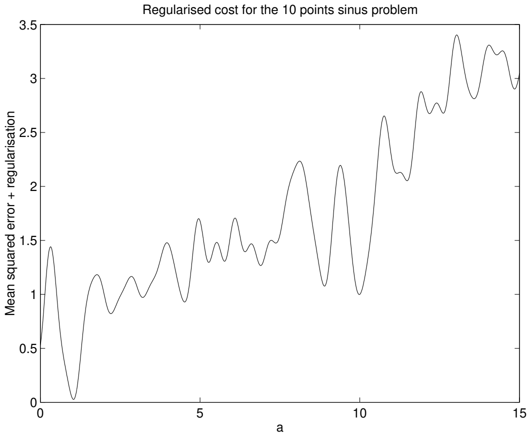


Figure 2.5: With the help of a regularisation term, the cost function favours low values of a, resulting in one clear minimum $a = 1.04$.

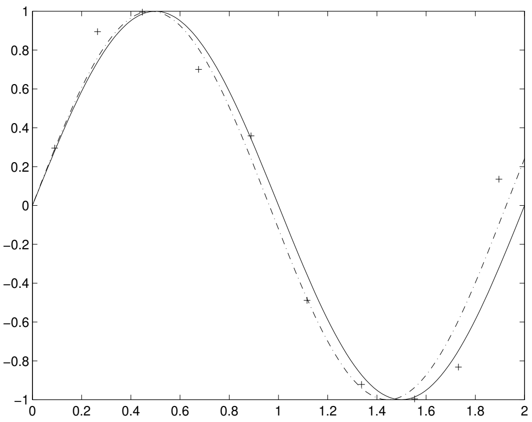


Figure 2.6: Solution of the regularised cost minimisation (dash-dotted) together with the data points (+) and the underlying mapping (solid).

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 δ. 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 ξ 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 ξ 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 ξ.

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 ε on the input produces a data matrix $(\mathbf{X}+\varepsilon)$. The jitter is sampled from a Gaussian distribution of 0 mean and variance σ^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 ε. 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 ε 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 σ^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 σ^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 ξ:

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

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, σ 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.

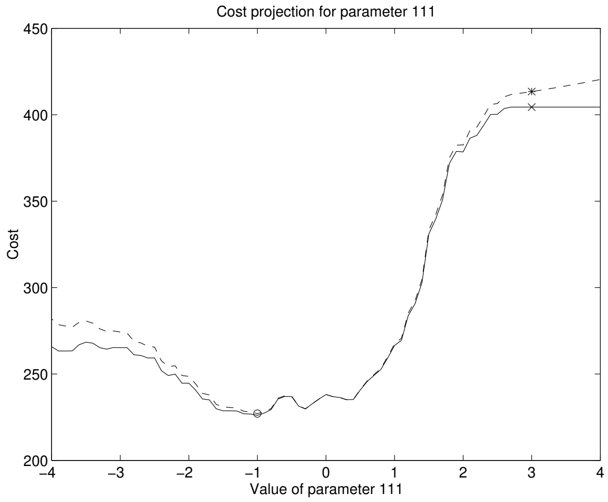


Figure 2.7: Projection of the regularised (dashed) and non-regularised (solid) cost surfaces on the subspace corresponding to parameter 111 (out of 141). For non-regularised cost, the parameter value is 3, situated in the flat zone (x), hence frozen. With regularisation, the induced curvature makes the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

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

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:

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

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 λ_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 λ_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 λ_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 λ_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 ξ. 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:

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

This shows that when a level ξ 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. ξ 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 ξ.

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 ξ 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 δw 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 δw 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 μ is an unknown constant. The derivative of the Lagrangian w.r.t. δw is zero, so $\mathbf{H}\delta w = -\mu u_k$. Plugging this into our constraint we infer μ 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 $\mathbb{R}^*$ and takes values ± 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 ξ grows. For $\xi = 0$, the sensitivity is 0. As ξ 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 ξ 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_ϵ 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 ϵ 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 γOBD and γOBS, 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: γOBD, γOBS. 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.

3

Hyper-parameters

3.1 Introduction

We have introduced earlier the use of regularisation in the learning procedure. It should now be understood that regularisation is most often a necessity to increase the quality of the results. Even when the unregularised solution is acceptable, it is likely that *some* regularisation will produce an improvement in performance.

There does not exist any method giving directly the best value for the regularisation parameter ξ, even in the linear case. The topic of this chapter is thus to propose some methods to *estimate* the best value. The best ξ being the one that leads to the smallest generalisation error, the methods presented and compared here propose estimators of the generalisation error. This estimation can then be used to approximate the best regularisation level.

In sections 3.2 to 3.4 we present validation-based techniques. They estimate the generalisation error on the basis of some extra data. In sections 3.6 to 3.9, we deal with algebraic estimates of this error, that do not use any extra data, but rely on a number of assumptions.

The contribution of this chapter is to present all these techniques and analyse them on the same ground. We also present some short derivations clarifying the links between different estimators of generalisation error, as well as a comparison between them.

During the course of this chapter, the error will be the quadratic difference. For the validation-based methods, it is possible to consider any kind of error without modification of the method. On the other hand, the algebraic estimates are specific to the quadratic cost. Adapting them to another cost function would require to derive new expressions for the estimators.

The general setting for this chapter is similar to chapters 1 and 2. A model f_w depending on a set of parameters w is used to estimate the regression of x on y. $\widehat{w}$ is obtained by optimising the regularised cost:

$$C(w) = S(w) + \xi R(w) \tag{3.1}$$

where $S(w) = \sum_{i=1}^{N} \left(y^{(i)} - f_w \left(x^{(i)} \right) \right)^2$ is the quadratic cost and $R(w)$ is the regularisation functional, weighed by the regularisation level ξ. Typically, $R(w)$ involves a sum of absolute or quadratic values of the weights. However, this is not a restriction, and more general terms involving e.g. first and second derivatives can be found in the literature.

The application of the results in this chapter to other regularisation techniques such as pruning is straightforward.

3.2 Single validation method

The single validation method is a degenerate case of the cross-validation method presented later. The data available for learning (i.e. excluding a possible "test set" on which the final performance is checked) is split in two : the training set and the validation set. The training set is used for training, as can be expected. The validation set is used to assess the performance of the model for various values of the hyper-parameter.

The validation set should *not* be used for training the actual model, i.e. either identifying the parameter of the model, or calculating a non-parametric estimate. On the other hand, it *is* part of the learning procedure, as it guides the choice of the value of the hyper-parameter that will lead to the final model. It differs in that way with the "test set", which should not be touched at any rate during the entire learning.

When V data are reserved for validation, the single validation method divides the available data in three sets:

1. The training set contains data $V+1$ to N. It is used to calculate and optimise the cost function $C(w)$, giving an estimate $\widehat{w}$ of the parameters.
2. The validation set contains data 1 to V. It is used to assess the generalisation performance of $\widehat{w}$.
3. The test set contains data after N. It is used to assess the final performance of the entire learning procedure.

The validation set provides an estimate of the generalisation error that can be expressed as:

$$\widehat{G}_{\mathrm{SV}} = \frac{1}{V} \sum_{k=1}^{V} \left(y^{(k)} - f_{\widehat{w}} \left(x^{(k)} \right) \right)^2 \tag{3.2}$$

V regulates the trade-off between training and validation data. The balance between the size of the training set and the size of the validation set is a matter of personal judgment. According to the law of great numbers, $\widehat{G}_{\mathrm{SV}}$ is an unbiased estimator of the generalisation error.

The main drawbacks of the single validation method is that all data are not available during training, and the results are highly dependent on the data reserved for validation. This trade-off is a typical no-win situation. If we want the validation error to be a fair estimate, V must be large, and the number of remaining data for training is correspondingly small. If we want to keep a large number of data for training, the estimate (3.2) will be poor.

3.3 Cross-validation methods

The *cross-validation* method consists in averaging the effect of the choice of a particular validation set over several such sets of identical size. Theoretically, we would like to consider *all* such sets. Unfortunately, the number of validation sets of size V taken from a total number N of data is $\mathsf{C}_{\mathrm{N}}^{\mathrm{V}}$. The commendable desire of taking into account *all* sets of a given size does not survive combinatorial considerations. As soon as the validation size is larger than a couple of elements, training is no longer feasible.

The special case where $V = 1$ is still possible, and is treated in section 3.4.

The practical application of cross-validation overcomes the combinatorial limitations by considering only a subset of disjoint validation sets. Let us split the available data in Q sets S_j of size V each, such that $N = Q.V$. For practical reason, we will here consider that V actually divides N, and that the sets are taken in the order of the data: set S_1 contains examples 1 to V, set S_2 contains examples $V+1$ to $2V$, etc.

Q different models $\hat{f}_j$ are trained leaving out every set S_j in turn. The cross-validation estimator is the average of the performance of these models on the left-out subset:

$$\widehat{G}_{\mathrm{CV}} = \frac{1}{N} \sum_{j=1}^{V} \sum_{k \in S_j} \left(f_j \left(x^{(k)} \right) - y^{(k)} \right)^2 \tag{3.3}$$

Note that as we perform an average over a number of different models, each trained on parts of the available data, we actually obtain an estimator of the *average generalisation error*, rather than the generalisation error as in the single validation case.

This estimation scheme is guided by the number Q of subsets to leave out. An equivalent scheme consists in setting the size V of each of the subsets, rather than their number. In such a case, the cross-validation method is known under the name *leave-V-out* cross-validation, or *Q-fold* cross-validation.

The cross-validation method limits the dependency to the validation data by averaging over several validation sets. It is a factor Q more costly to perform than a single validation, as Q models have to be estimated. This is much less costly though than the full cross-validation involving all sets of size V. The choice of Q or V still reflects a trade-off, between available data and learning time. Large values of Q correspond to small, but numerous validation sets. Training on the remaining data will then be close to training on the full set, but many trainings are necessary.

The main drawback of this method is that for moderate values of Q, the available data is reduced in a non-negligible proportion. E.g. for $Q = 10$, a common setting, only 90% of the examples are available during each parameter identification. The next section addresses the use of a limit case of this cross-validation, where $Q = N$.

3.4 Leave-one-out cross validation

A special case of cross-validation arises when the size of the validation set is only $V = 1$ element. This technique is generally known as *leave-one-out* cross-validation, although it sometimes goes under other names.

Let us denote by $\hat{f}_k$ the model obtained by leaving out example number k. The limit case of (3.3) leads to the following estimator:

$$\widehat{G}_{\text{LOO}} = \frac{1}{N} \sum_{k=1}^{N} \left(\hat{f}_k \left(x^{(k)} \right) - y^{(k)} \right)^2 \tag{3.4}$$

Leave-one-out cross validation is still hard to apply *as is* to models requiring expensive calculation to provide the numerous $\hat{f}_k$ estimates. In the case of neural networks, for example, a full calculation of (3.4) would require to train N different networks, which is often infeasible.

On the other hand, the computational burden is not as high for other models such as linear models or non-parametric estimators. In the case of non-parametric regression, leave-one-out cross-validation is a standard tool for setting the value of the smoothing parameters, as exemplified in sections B.8 to B.12.

In the linear case, simplifications arise. Using the same notation as section 1.5, we recall that the linear parameter vector leaving example k out is given by:

$$\widehat{w}_k = \left(\mathbf{X}_k{}^\top \mathbf{X}_k \right)^{-1} \mathbf{X}_k{}^\top \mathbf{Y}_k \tag{3.5}$$

where $\mathbf{X}_k$ and $\mathbf{Y}_k$ are the input and output matrices leaving the k-th example out. It is convenient to notice that $\mathbf{X}_k{}^\top \mathbf{X}_k = \mathbf{X}^\top \mathbf{X} - x^{(k)} x^{(k)\top}$ and $\mathbf{X}_k{}^\top \mathbf{Y}_k = \mathbf{X}^\top \mathbf{Y} - x^{(k)} y^{(k)\top}$. Using the matrix inversion lemma, we get:

$$\left(\mathbf{X}_k{}^\top \mathbf{X}_k \right)^{-1} = \left(\mathbf{X}^\top \mathbf{X} \right)^{-1} + \frac{\left(\mathbf{X}^\top \mathbf{X} \right)^{-1} x^{(k)} x^{(k)\top} \left(\mathbf{X}^\top \mathbf{X} \right)^{-1}}{1 - x^{(k)\top} \left(\mathbf{X}^\top \mathbf{X} \right)^{-1} x^{(k)}} \tag{3.6}$$

Using (3.6), the LOO expression (3.4) simplifies to the rather straightforward expression:

$$\widehat{G}_{\text{LOO}} = \frac{1}{N} \sum_{k=1}^{N} \frac{\left(\hat{w}^\top x^{(k)} - y^{(k)} \right)^2}{\left(1 - x^{(k)\top} \left(\mathbf{X}^\top \mathbf{X} \right)^{-1} x^{(k)} \right)^2} \tag{3.7}$$

where $\hat{w}$ is the optimal linear parameter vector, i.e. $\hat{w} = \left(\mathbf{X}^\top \mathbf{X} \right)^{-1} \mathbf{X}^\top \mathbf{Y}$. Notice that the expression (3.7) stays valid for a regularised linear model. In the case of ridge regression, one just has to replace $\left(\mathbf{X}^\top \mathbf{X} \right)^{-1}$ with $\left(\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_P \right)^{-1}$.

The use of (3.7) greatly speeds up the evaluation of the leave-one-out error. Indeed, the straightforward application of (3.4) requires N model estimations, which boils down to N matrix inversion. Only one matrix inversion is necessary for $\left(\mathbf{X}^\top \mathbf{X} \right)^{-1}$, and the x-dependent terms in the denominator of (3.4) are the diagonal elements of $\mathbf{X} \left(\mathbf{X}^\top \mathbf{X} \right)^{-1} \mathbf{X}^\top$.

Unfortunately, there is no such exact expression for non-linear models in general, and neural networks in particular. An approximation in the same line will be studied in section 3.9.

3.5 Against cross validation

Single and cross validation (the validation-based methods) are widely used methods of assessing the generalisation abilities of a given model. The widespread use of cross-validation is often justified by the belief that, though computationally expensive, it gives sound results—in our case, a proper choice of ξ.

We will try to show here that cross-validation also poses a number of problems that are not easily addressed. We will first focus on the trade-off between computational demand and estimation accuracy, then address the topic of the results obtained.

Cross-validation poses a serious computational problem. As noted above, leave-one-out cross-validation requires to train as many models as there are data. Carrying out N non-linear regression, even with an efficient algorithm such as conjugate gradient or Levenberg-Marquardt method, is extremely time-consuming. For simple models the solution can be obtained faster, making this scheme computationally viable. However, it is likely in such cases, as we have seen for linear regression in section 3.4, that the LOO expression can be turned into a more convenient estimator such as (3.7).

Single validation, on the other hand, poses a serious reliability problem. The size of the validation set has to be limited in order to keep a large portion of the training data available for optimisation. It is easy to conceive that the choice of a different validation set would lead to a different optimisation solution and, predictably, another assessment of the generalisation error. It is indeed in order to compensate for the variability of the data that the cross-validation method suggests to average the assessments made with different validation sets.

Practical application of cross-validation is then subject to an unpleasant trade-off between getting noisy estimates and computational cost. Unfortunately, this is not the only limitation of this method. Recent developments in computational learning theory have led to the so-called "no free lunch" theorem (Wolpert and Macready, 1995), showing that no learning strategy is, on average, better than random guessing. This result is not so much a drawback as it is a plea for insisting on the assumptions that make a given algorithm efficient. The usefulness of a learning strategy lies in a number of assumptions that have to be fulfilled if any gain is to be expected.

Cross-validation methods are especially problematic in that respect. It is not yet clear what the underlying assumptions are, and thus in which case one should expect an efficient behaviour. The fact that many problems have been successfully tackled with these techniques suggests that these unknown hypotheses verify in a number of practical case. However, as long as they have not been identified precisely, there is no guarantee whatsoever that a new problem will meet these requirements.

3.6 Algebraic estimators of generalisation error

Under the proper set of hypothesis, it is possible to derive asymptotic estimates of the generalisation error averaged over all possible training sets of size N, in a manner similar to section 1.6. The main assumptions on the problem are:

1. The (output) data are corrupted by white stationary noise with zero mean and variance σ^2, independent of the input.
2. The model is *complete*, i.e. there exist a "true" set of parameters $\widetilde{w}$, such that $f_{\widetilde{w}} = f$.
3. The empirical solution is close to the "true" solution so that a Taylor expansion can be written.

A simple analytic measure of the generalisation abilities of a (linear) model is the adjusted residual squared error:

$$\widehat{G}_{\mathrm{AR}} = \frac{N}{N - 2P} S\left(\widehat{w}\right) \tag{3.8}$$

where P is the number of parameters in the model, and $N\,S\left(\widehat{w}\right)$ is the sum of residuals, hence the name. This estimator enjoys some asymptotic properties for linear models, but it is not so impressive when it comes to non-linear models, let alone regularised models.

With the assumptions above, we use a Taylor expansion of the empirical and expected costs $S\left(w\right)$ and $G\left(w\right)$, after which the calculations are similar to the linear case in section 1.6. The derivation basically lead us to derive a link between average training error (or average generalisation error) and the noise level σ^2. This results in Akaike's *Final Prediction Error* or FPE:

$$\widehat{G}_{\mathrm{FPE}} = \frac{N + \widehat{P}}{N - \widehat{P}} S\left(\widehat{w}\right) \tag{3.9}$$

where $\widehat{P}$ is the estimated *effective number of parameters* (see next section). This correction takes into account the fact that regularisation tends to "disable" some parameters, so not all of them will use up one degree of freedom. We will come back to the topic of the effective number of parameters, including their expression for several types of regularisers, in section 3.7. Note that equation (3.9) is also known under the name GPE.

By expanding the expression of the inverse regularised Hessian, it is possible to push derivation further and exhibit the *FPE for a Regularised cost* or FPER:

$$\widehat{G}_{\mathrm{FPER}} = \frac{N + \widehat{P}_2}{N - 2\widehat{P}_1 + \widehat{P}_2} S\left(\widehat{w}\right) \tag{3.10}$$

We now have two quantities $\widehat{P}_1$ and $\widehat{P}_2$ related to the effective number of parameters. The following section will elaborate on this concept and exhibit the links between the two quantities and $\widehat{P}$ mentioned earlier.

One should also be aware of the fact that the algebraic estimates (3.9) and (3.10) are theoretically valid for averaged errors, i.e. the training or generalisation error averaged over the fluctuation of the training set. However, we expect this property to stay valid as we plug $S(w)$ as the only estimate we have of the average training error.

3.7 Effective number of parameters

The expression of the *effective number of parameters* appears in the derivation of the estimators above. For a regularised system, let us denote $\mathbf{H}_S$ and $\mathbf{H}_C$ the Hessians of the quadratic and regularised cost (respectively). These Hessians appear in the Taylor expansions performed around $\widetilde{w}$.

In this context, a common expression for the effective number of parameters is:

$$\widehat{P} = \mathrm{tr}\left(\mathbf{H}_S \mathbf{H}_C^{-1}\right) \tag{3.11}$$

For a weight-decay regulariser, equation 3.11 gives the well-known[1]expression:

$$\widehat{P} \approx \sum_{k=1}^{P} \frac{\lambda_k}{\lambda_k + \xi} \tag{3.12}$$

where λ_k are the eigenvalues of $\mathbf{H}_S$, and ξ is the weight-decay coefficient. Roughly speaking, a large eigenvalue ($\lambda_k \gg \xi$) will contribute 1 to the parameter count, while a small eigenvalue ($\lambda_k \ll \xi$) will have negligible contribution.

Remarkably, this is not the only expression of the desperately-sought *effective number of parameters*. Indeed, this expression depends on the derivation of the estimator. Different derivations are likely to lead to different expressions for $\widehat{P}$.

For Laplace regularisation, on the other hand, expression (3.11) does not lead to any analytical solution, as the second derivative of the regularisation term is undefined in 0. However, this regulariser has a pruning effect. The *effective number of parameters* in this case will then be approximated by the number of parameters that have not been pruned:

$$\widehat{P} = \mathrm{card}\left\{w_k \neq 0\right\} \tag{3.13}$$

In practice, a parameter will be considered pruned not only when it is actually zero, but also when it is lower than a given threshold, depending on the training algorithm.

In the case of FPER, we have mentioned two quantities $\widehat{P}_1$ and $\widehat{P}_2$. $\widehat{P}_1$ is defined in the same way as $\widehat{P}$ in (3.11), while $\widehat{P}_2$ is defined as:

$$\widehat{P}_2 = \mathrm{tr}\left(\mathbf{H}_S \mathbf{H}_C^{-1} \mathbf{H}_S \mathbf{H}_C^{-1}\right) \tag{3.14}$$

For weight-decay, we have $\widehat{P}_2 \approx \sum_{k=1}^{P} \frac{\lambda_k^2}{(\lambda_k + \xi)^2}$.

[1]This expression is easily derived from the eigen-decomposition of $\mathbf{H}_S = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^\top$, with $\mathbf{\Lambda}$ the diagonal matrix of the eigenvalues λ_k. In the same condition, $\mathbf{H}_C = \mathbf{Q}\left(\mathbf{\Lambda} + \xi\mathbf{I}\right)\mathbf{Q}^\top$ leading to (3.12).

It should be noted that when $\widehat{P}_1$ and $\widehat{P}_2$ are close to each other, (3.10) reduces to (3.9). It is the case for weight-decay when the eigenvalues are well-separated around ξ: $\frac{\lambda_k}{\lambda_k+\xi} \approx \frac{\lambda_k^2}{(\lambda_k+\xi)^2} \in \{0,1\}$.

3.8 Information criteria

In this section we deal with different but related estimates of the generalisation error. These estimates rely on maximising some *information criterion* rather than minimising the expected prediction error. The information criteria involve the natural logarithm of the maximum likelihood. According to section 1.3, the maximum likelihood ML is linked with the minimum training error by:

$$-\ln ML = \frac{N\,S\left(\widehat{w}\right)}{2\sigma^2} + cte \tag{3.15}$$

Please note that as σ^2 is usually unknown, it will be replaced by an estimator. Recall e.g. from section 3.6 that one such estimator is $\widehat{\sigma}^2 = \frac{N\,S(\widehat{w})}{N-\widehat{P}}$.

One of the first estimates introduced was Mallows' C_p statistics, which is a particular case of Akaike's first Information Criterion, dubbed AIC. The original definition of AIC is $2\ln ML - 2P$. In our context, AIC leads to the following estimator of the generalisation abilities of our model:

$$\widehat{G}_{\text{AIC}} = S\left(\widehat{w}\right) + \frac{2\widehat{P}}{N}\widehat{\sigma}^2 \tag{3.16}$$

where $\widehat{\sigma}^2$ is the estimated variance of the noise and $\widehat{P}$ is still the *effective number of parameters*. Taking $\widehat{\sigma}^2 = \frac{N}{N-\widehat{P}}S\left(\widehat{w}\right)$ as mentioned above, (3.16) reduces to the same expression as the FPE in section 3.6.

A similar criterion, is Schwarz's Bayesian Criterion, or BIC. Using the same notation, it leads to the following estimate of generalisation error:

$$\widehat{G}_{\text{BIC}} = S\left(\widehat{w}\right) + \frac{\widehat{P}\ln N}{N}\widehat{\sigma}^2 \tag{3.17}$$

Please note however that this criterion is not obtained on the basis of the average generalisation error, but using the Bayesian approach of the evidence (see chapter 5).

One can see by comparing (3.16) and (3.17) that BIC leans more towards smaller models (fewer parameters) as soon as there are at least 8 examples. Another interesting feature of BIC is that it is a consistent criterion. As a consequence, one can infer that AIC or C_p do not lead to correct models when $N \to \infty$. This is most distressing as both criteria are asymptotic, hence valid in the limit of the large number of examples. However, it should be noted that in our approach, generalisation error is indeed the quantity of interest. The fact that AIC would favour an over-parameterised model is thus not problematic as long as it minimises the generalisation abilities.

3.9 Algebraic estimate of the LOO error

Another approach combines the LOO approach and algebraic estimation. The difference between the minimum of the quadratic cost $\widehat{w}$ and each of the minimum of the one-example-out errors $\widehat{w}_k$ is approximated by doing the usual first-order Taylor expansion around $C(\widehat{w})$.

After expanding the LOO generalisation error estimator (3.4) linearly in $\Delta\widehat{w}_k = \widehat{w}_k - \widehat{w}$, we get the approximation:

$$\widehat{G}_{\text{LOO}} \approx \frac{1}{N}\sum_{k=1}^{N}\left(\hat{f}_{\widehat{w}}\left(x^{(k)}\right) - y^{(k)}\right)^2\left[1 + 4\nabla_w^\top f_{\widehat{w}}\left(x^{(k)}\right)\mathbf{H}_k^{-1}\nabla_w f_{\widehat{w}}\left(x^{(k)}\right)\right] \tag{3.18}$$

where $\mathbf{H}_k$ is the Hessian of the *regularised* one-left-out cost, that is:

$$\mathbf{H}_k = \mathbf{H}_C - 2\nabla_w f_{\widehat{w}}\left(x^{(k)}\right)\nabla_w^\top f_{\widehat{w}}\left(x^{(k)}\right)$$

where we have used the Gauss-Newton approximation of the Hessian, leaving only gradients of the model. After using the matrix inversion lemma to calculate $\mathbf{H}_k^{-1}$, (3.18) becomes the *linear unlearning leave-one-out* estimate:

$$\widehat{G}_{\text{LULOO}} = \frac{1}{N}\sum_{k=1}^{N}\left(\hat{f}_{\widehat{w}}\left(x[k]\right) - y^{(k)}\right)^2\frac{1+h_k}{1-h_k} \tag{3.19}$$

This corresponds to doing a Taylor expansion around the full-sample minimum for each of the LOO samples, hence the name of “linear unlearning”.

where $h_k = 2\nabla_w^\top f_{\widehat{w}}\left(x^{(k)}\right)\mathbf{H}_C^{-1}\nabla_w f_{\widehat{w}}\left(x^{(k)}\right)$. Predictably, this estimator involves the inverse of the Hessian, just like other algebraic estimators. In the linear case, $\nabla_w f_{\widehat{w}}\left(x^{(k)}\right) = x^{(k)\top}$ and $\mathbf{H}_C = 2\left(\mathbf{X}^\top\mathbf{X}\right)$, leading to $h_k = x^{(k)\top}\left(\mathbf{X}^\top\mathbf{X}\right)^{-1}x^{(k)}$. Notice in these condition the similarity between (3.19) and the exact expression (3.7) in the limit of small h_k.

3.10 Comparing the estimators

In this section we establish some links between the estimators of generalisation error from the above sections.

First let us consider that we use $\widehat{\sigma}^2 = \frac{N\,S(\widehat{w})}{N-\widehat{P}}$ as an estimator of the noise variance. The use of this estimator in (3.16) leads to:

$$\widehat{G}_{\text{AIC}} = \left(1 + \frac{2\widehat{P}}{n-\widehat{P}}\right)S(\widehat{w}) = \left(\frac{N+\widehat{P}}{N-\widehat{P}}\right)S(\widehat{w}) = \widehat{G}_{\text{FPE}} \tag{3.20}$$

Let us now consider the expression of the adjusted residual estimator, using the effective number of parameters $\widehat{P}$ in place of the raw number of parameters P in (3.8):

$$\widehat{G}_{\text{AR}} = \frac{N}{N-2\widehat{P}}S(\widehat{w}) = \left(\frac{N+\widehat{P}}{N-\widehat{P}}\right)\left(1 + o\left(\frac{1}{N}\right)\right)S(\widehat{w}) \approx \widehat{G}_{\text{FPE}} \tag{3.21}$$

which proves the first order equivalence between the adjusted residual estimator modified with effective number of parameters and the FPE (and thus AIC) estimate.

Let us now consider the linear *leave-one-out* situation. According to (3.7), the cross-validation estimate is:

$$\widehat{G}_{\mathrm{LOO}} = \frac{1}{N}\sum_{k=1}^{N}\left(\frac{y^{(k)} - f_{\widehat{w}}\left(x^{(k)}\right)}{1-h_k}\right)^2 \tag{3.22}$$

where h_k is the k-th diagonal element of matrix $\mathbf{X}\left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{X}^\top$. Approximating these terms by their average $\bar{h} = \frac{1}{N}\sum_{k=1}^{N} h_k$ leads to the *generalised cross-validation* estimator:

$$\widehat{G}_{\mathrm{GCV}} = \frac{1}{N}\sum_{k=1}^{N}\left(\frac{y^{(k)} - f_{\widehat{w}}\left(x^{(k)}\right)}{1-\bar{h}}\right)^2 = \frac{S\left(\widehat{w}\right)}{\left(1-\bar{h}\right)^2} \tag{3.23}$$

Let us now use the well-known first-order approximation $\left(1-\bar{h}\right)^{-2} \approx 1 + 2\bar{h}$ to obtain the following approximate expression for GCV:

$$\widehat{G}_{\mathrm{GCV}} \approx S\left(\widehat{w}\right) + 2\bar{h}S\left(\widehat{w}\right) \tag{3.24}$$

This is an approximation to the C_p statistics and AIC, where $S\left(\widehat{w}\right)$ is taken as a first-order approximation of the noise level, and $N\bar{h} = \mathrm{tr}\left(\mathbf{X}\left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{X}^\top\right)$ is the effective number of parameters.

We already have pointed out in section 3.9 the similarity between the linear LOO and the LULOO estimator 3.19. Let us also notice that h_k plays the role of the P/N ratio. In that respect, the linear unlearning LOO in (3.19) can be seen as a “local” FPE, where the error correction occurs at each individual example's level.

This completes our study of the relationships between the above estimators. We have represented these links in a schematic way in figure 3.1.

3.11 Against algebraic estimators

The use of algebraic estimators in the context of non-linear regularised regression is of course not free of all worries.

The first concern is the number of hypothesis necessary to derive the estimators. It is common statistical practice to assume that the noise is rather well-behaved as in section 3.6. On the other hand, the *completeness* assumption is only valid to a certain extent, and for fairly large models. As multi-layered perceptrons are universal approximators, a neural network model could be considered complete. However, the situation is not that clear. First it could require many hidden units, and thus a large number of parameters to have a *complete* model. Even with an unlimited number of hidden unit, the network architecture is not necessarily adapted to the system. This is particularly the case in system identification if either e.g. the set of inputs is wrong. This can happen when some delays on the input/output are not taken into account.

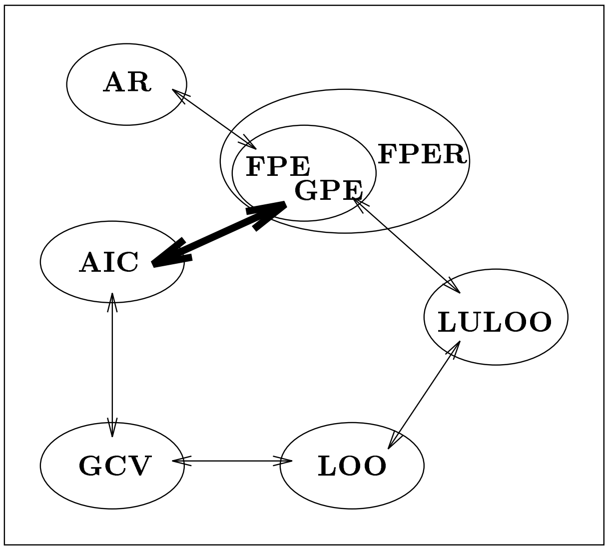


Figure 3.1: The relationships between algebraic estimators are represented on this figure. The bold line between FPE and AIC represents the especially close relationship between these two estimators.

Another concern is the fact that these estimators, even when the assumptions are justified, are asymptotic. In order to be statistically sound, they require a large number of examples during the learning process. However, as noted above, it is rarely the case that we have that much data. In practical cases, data are in short supply, and the accuracy of statistical estimators is thus questionable.

The last concern is the fact that most proper estimators above require the calculation of the *effective number of parameters*. This is a crucial aspect, for the assessment of the algorithm's computational demand (as illustrated below) as well as for the accuracy of the result. Equations 3.12 and 3.13 are only approximations and it can be seen from expression (3.9), for example, that the algebraic estimation is quite sensitive to variations in $\widehat{P}$, especially when the ratio $\widehat{P}/N$ is not small. Indeed, $\frac{\partial \widehat{G}_{\text{FPE}}}{\partial \widehat{P}} = \frac{2N}{\left(N-\widehat{P}\right)^2}$

3.12 Comparing cross validation and estimators: resources

In order to compare the resources needed using different techniques for optimising the regularisation level, let us first consider the time taken by the optimisation algorithm. According to chapter 1, the conjugate gradient algorithm is an efficient way of performing the optimisation of the regularised cost.

One iteration of conjugate gradient can be performed in $\mathcal{O}(NP)$ operations. The number of iterations needed is problem-dependent, but it is reasonable to consider that we need an order P of iterations. For a *linear* problem, as noted in section 1.13, exact minimisation requires at most P iterations. For a *non-linear* problem, we recall that this is no longer true: the algorithm will need to restart a number of times before eventually converging. However, there is no reason to believe that this need to restart is linked to either the number of parameters or the number of examples. The optimisation of the regularised cost takes an order $\mathcal{O}\left(P^2N\right)$ operations.

In the LOO scheme, each term in (3.4) requires to learn the appropriate model $\widehat{f}_k$, meaning that there will be N non-linear optimisation processes necessary. The generalisation assessment is made just by combining the squared error produced by each model on the left-out example, which is at most $\mathcal{O}\left(PN\right)$, hence negligible. The overall complexity of the leave-one-out scheme is then N times larger than that of a simple optimisation, that is:

$$\mathcal{C}_{LOO} = \mathcal{O}\left(P^2N^2\right) \tag{3.25}$$

The *linear unlearning* scheme (cf. section 3.9) avoids retraining all the models. However, the expression of the estimate (3.19) involves the calculation of the h_k which require the computation of the inverse Hessian information. Using the Gauss-Newton approximation, the inverse Hessian can be obtained by a recursive method in $\mathcal{O}\left(P^2N\right)$ operations. The remaining calculation for each k is only $\mathcal{O}\left(P^2\right)$ so the overall computation cost of the linear unlearning leave-one-out estimate is:

$$\mathcal{C}_{LU} = \mathcal{O}\left(P^2N\right) \tag{3.26}$$

For other algebraic estimates, the dominating cost is the estimation of the *effective number of parameters*. As above, it involves the extraction of the inverse Hessian information. Indeed, both (3.11) and (3.14) require computing the trace of a product of non-regularised Hessian matrix and inverse of regularised Hessian matrix. In the same conditions as above (Gauss-Newton approximation), the dominant cost is the recursive estimation of the inverse Hessian, i.e. $\mathcal{O}\left(P^2N\right)$ operations. Please note that full multiplication of $P \times P$ matrices would require $\mathcal{O}\left(P^3\right)$ operation. Only the trace is necessary, however, bringing the computational cost down to a quadratic $\mathcal{O}\left(P^2\right)$.

The computational complexity of a learning algorithm based on algebraic estimation of the generalisation error is then:

$$\mathcal{C}_{GEN} = \mathcal{O}\left(NP^2\right) \tag{3.27}$$

This complexity is valid for FPE/GPE, $FPER$, AIC and BIC.

The operation counts presented above in (3.25), (3.26) and (3.27) is valid for *one* training step, i.e. optimisation of the regularised cost for a given value of ξ, and estimation of the generalisation abilities of the resulting model. A typical learning process will involve several such evaluations for different values of ξ in order to find the optimal generalisation level. This can be done by any one-dimensional minimisation method such as golden search, dichotomy, or Brent's method. At any rate, the number of iterations, and thus the number of different training steps required, depends on the accuracy needed. It is independent of either the number of parameters or the number of examples.

This last remark validates the estimates given for the full learning scheme. Predictably, the complexity of the linear unlearning and the other algebraic estimators are similar. This is no big surprise if we consider the relationships between these estimators, exhibited in section 3.10.

In the limit of the large number of examples, the LOO method with its quadratic complexity proves far too expensive to be viable. The other two schemes have only linear complexity in the number of examples, which is makes them much better options.

It is unfortunately seldom the case, however, that P is negligible before N. In many real applications, data tend to be sparse, and both P and N are of the same order. The computational costs derived in this section show that even then, leave-one-out cross-validation loses in flop counts compared to the other estimators. It should be noted however, that in such a case, the reliability of these estimates is questionable.

3.13 How can I choose the hyper-parameter value

In this chapter we have presented two families of methods to chose the extent of the regularisation. They rely on estimating the generalisation error, and choosing the regularisation level that yields the lowest generalisation error.

We insisted on the fact that cross-validation is a computationally expensive method, leading to rather noisy estimates. Furthermore, the assumption under which this method is efficient are not well understood.

On the other hand, the algebraic estimators rely on a number of more visible assumptions. Their calculation is much less demanding in terms of computational resource. However, they are only asymptotic estimators, thus not well-suited to practical cases when the number of data is limited.

Considering the above, and when it is needed to chose one among these two classes of methods, we will favour the choice of the algebraic estimate. In the practical applications presented later in this work, we will adopt this standpoint.

COMMENTS

3.2 Despite its drawbacks and inaccuracy, the single validation method has been used extensively and is still widespread. It is sometimes necessary to use it to compare with older results. e.g. Goutte (1996) and Weigend et al. (1990). Some studies also explore the way the validation set should be used to yield maximum efficiency, e.g. Larsen and Hansen (1995); Larsen et al. (1996), or Kearns (1996). However there does not seem to be any consensus yet on the optimal split ratio.

3.4 The *leave-one-out cross-validation* scheme has been named e.g. *hold-out* method by Weigend et al. (1990). It also shares some similarity with the *jackknife* statistical procedure, but differs in the purpose (see Efron, 1982).

3.5 A more precise study of the computational cost is given in section 3.12. The "no free lunch" theorem is due to (Wolpert and Macready, 1995). This result has sparked heated debates in the computational learning community. The cross-validation procedure in particular has been in the spotlight, and several efforts try to study this method in the light of NFL (Zhu and Rohwer, 1996; Goutte, 1997).

3.6 The FPE in the linear case is due to Akaike (1969). The generalisation to regularised neural networks and the associated notion of *effective number of parameters* was developed e.g. by Moody (1991). Ljung et al. (1992) propose a similar estimate, but with an *effective number of parameters* closer to $\widehat{P}_2$ in (3.14). The FPER is specific to regularised complete models and is due to Larsen and Hansen (1994). Other estimators take into account the bias in the model, i.e. suppress the completeness assumption (Murata et al., 1994; Larsen, 1992). The counting method for estimating the effective number of parameters for Laplace/L_1 regularisation is illustrated by Goutte (1996).

3.8 Mallows' C_p was introduced by Mallows (1973), soon followed by Akaike's AIC in (Akaike, 1974). The Bayesian Information Criterion is introduced by Schwartz (1978), together with a proof of its asymptotic optimality.

3.9 Linear unlearning for leave-one-out cross-validation estimates is suggested by Hansen and Larsen (1996), and applied to system identification in a recent communication by Sørensen et al. (1996).

3.10 Apart from these asymptotic considerations, it is interesting to have a operational comparison of different estimators. A brief empirical comparison between FPE (with raw parameter number), GPE and FPER is included at the end of (Larsen and Hansen, 1994). It illustrates the effect of the *number of effective parameters*, and shows that GPE and FPER provide close estimates (in that case).

3.11 The universal approximation of neural networks is demonstrated in the classical (Hornik et al., 1989).

3.12 The computational cost of several optimisation techniques applied to neural networks has been studied by Møller (1993). The resource asymptotics mentionned in this section relate to efficient optimisation methods such as the *scaled conjugate gradient*. This method was originally developed for use with neural networks models, but has wider application possibilities.

References

Akaike, H. (1969). Fitting autoregressive models for prediction. *Annals of the Institute of Statistical Mathematics*, 21:243–247.

Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723.

Efron, B. E. (1982). *The Jacknife, the Bootstrap and Other Resampling plans*, volume 38 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. SIAM.

Goutte, C. (1996). On the use of a pruning prior for neural networks. In NNSP96 (1996), pages 52–61.

Goutte, C. (1997). Note on free lunches and cross-validation. *Neural Computation*, 9(6). to appear.

Hansen, L. K. and Larsen, J. (1996). Linear unlearning for cross-validation. *Advances in Computational Mathematics*, 5(2,3):269–280.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–368.

Kearns, M. (1996). A bound on the error of cross validation using the approximation and estimation rates, with consequences for the training-test split. In Touretzky, D. S., Mozer, M. C., and Hasselmo, M. E., editors, *Advances in Neural Information Processing Systems*, number 8 in NIPS. MIT Press.

Larsen, J. (1992). A generalization error estimate for nonlinear systems. In Kung, S. Y., Fallside, F., and Sørensen, J. A., editors, *Neural Networks for Signal Processing – Proceedings of the 1992 IEEE Workshop*, number II in NNSP, pages 29–38, Piscataway, New Jersey. IEEE.

Larsen, J. and Hansen, L. K. (1994). Generalized performance of regularized neural networks models. In Vlontzos, J., Hwang, J. N., and Wilson, E., editors, *Neural Networks for Signal Processing IV – Proceedings of the 1994 IEEE Workshop*, number IV in NNSP, pages 42–51, Piscataway, New Jersey. IEEE.

Larsen, J. and Hansen, L. K. (1995). Empirical generalization assessment of neural network models. In Girosi, F., editor, *Neural Networks for Signal Processing V – Proceedings of the 1995 IEEE Workshop*, number V in NNSP, pages 42–51, Piscataway, New Jersey. IEEE.

Larsen, J., Hansen, L. K., Svarer, C., and Ohlsson, M. (1996). Design and regularization of neural networks: the optimal use of a validation set. In NNSP96 (1996), pages 62–71.

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.

Mallows, C. (1973). Some comments on c_p. *Technometrics*, 15:661–675.

Møller, M. (1993). A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, 6(4):525–533.

Moody, J. (1991). Note on generalization, regularization and architecture selection in nonlinear learning systems. In Juang, B. H., Kung, S. Y., and Kamm, C. A., editors, *Proceedings of the first IEEE Workshop on Neural Networks for Signal Processing*, number I in NNSP, pages 1–10, Piscataway, New Jersey. IEEE.

Murata, N., Yoshizawa, S., and Amari, S. (1994). Network Information Criterion—determining the number of hidden units for an artificial neural network model. *IEEE Transactions on Neural Networks*, 5(6):865–872.

NNSP96 (1996). *Neural Networks for Signal Processing VI – Proceedings of the 1996 IEEE Workshop*, number VI in NNSP, Piscataway, New Jersey. IEEE.

Schwartz, G. (1978). Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464.

Sørensen, P. H., Nørgård, M., Hansen, L. K., and Larsen, J. (1996). Cross-validation with luloo. In *Proceedings of 1996 International Conference on Neural Information Processing, ICONIP'96*.

Weigend, A. S., Huberman, B. A., and Rumelhart, D. E. (1990). Predicting the future: a connectionist approach. *International Journal of Neural Systems*, 1(3):193–210.

Wolpert, D. H. and Macready, W. G. (1995). The mathematics of search. Technical Report SFI-TR-95-02-010, Santa Fe Institute.

Zhu, H. and Rohwer, R. (1996). No free lunch for cross validation. *Neural Computation*, 8(7):1421–1426.

4

Lag-space estimation in time-series modelling

4.1 Situation and purpose

In this chapter, we focus on an original application of generalisation error and its estimation. The purpose of this work is to show that generalisation error plays a crucial role, and that its use can solve interesting and challenging problems.

This chapter deals with time series modelling. From a series of measurements x_1, $x_2, \ldots x_L$, we wish to identify a model $\widehat{f}$ that allows to predict the future behaviour of the time series from a number of past measurements:

$$\widehat{x}_t = \widehat{f}\left(x_{t-i_1}, x_{t-i_2}, \ldots, x_{t-i_m}\right) \tag{4.1}$$

Notice that in the general case, the past delays $i_1, i_2, \ldots, i_m$ need not be consecutive.

The estimation of the model itself is a parametric or non-parametric regression problem; a general treatment of these topics is given in chapter 1 and appendix B.

Before this estimation can be efficiently performed, however, it is necessary to set the input and output spaces. The output is well-known: it is the prediction of the time-series behaviour at time t, i.e. $\widehat{x}_t$. On the other hand, the input information is not know *a priori*. It is of crucial importance to determine this information, i.e. the proper *lag-space*. If some relevant delays are missing, the model will be unable to capture the underlying mapping, and thus fail to provide good prediction. On the other hand, if irrelevant inputs are included, the input space will have too high a dimension. From the curse of dimensionality, we expect poor predicting abilities. In the case of parametric estimation in particular, we will have an over-parameterised model, and thus poor performance.

4.2 Input selection

It should be understood that the problem of selecting the proper inputs is a central problem for time series modelling. This is a special case of variable selection, which in turn can be seen as part of the more general problem of analysing the structure in the data. However, our problem has a number of special features.

For one, there is an obvious, structural input correlation. More interesting, the search of the relevant delays fulfills one of the basic requirements of conventional variable selection. In such a case, all necessary variables must be available, so that a sufficient subset of inputs actually exists. This assumption is trivially met, as long as the system is predictable, because we have the past delays. This argument breaks down in the case where a long-term delay is needed, that ranges further than the period actually spanned by the data. However, the relevance of such a case is questionable as there would be no data to identify the associated parameter(s) anyway.

The path of statistical variable selection will be explored in section 4.6, and the solution we propose in section 4.8 is inspired from these classical techniques.

It should be noted that these methods rely on the use of a model and the identification of this model for a given lag-space. Another approach seeks this information in the data itself, in a non-parametric way. It originates in the study of the *embedding dimension* of a non-linear dynamical system. The next two sections present this approach and a practical implementation.

4.3 The embedding dimension

Assume that we have a deterministic dynamical system described as a smooth mapping:

$$\begin{aligned} h : M &\longrightarrow M \\ s_t &\longmapsto s_{t+1} \end{aligned}$$

where s_t is a state vector, and M is the phase space. We observe the system and record a time series $x_1,\ x_2,\ \ldots x_L$. We wish to reconstruct the dynamics of the system by finding a mapping involving only these outputs. We thus form vectors[1] of observed data $y_{t-1} = (x_{t-1}, x_{t-2}, \ldots, x_{t-m})^\top$, and model the dynamic by a mapping on these vectors:

$$\begin{aligned} g : I\!R^m &\longrightarrow I\!R^m \\ y_{t-1} &\longmapsto y_t \end{aligned}$$

Such a mapping always exists, and the dimension of y_t is the *embedding dimension*. It means that g is embedded in a space of dimension m, the *embedding space*.

Once we have found the embedding dimension, the dynamics of the system can be reconstructed by a mapping from the embedding space onto itself. Furthermore, all

[1] In this presentation, we consider only consecutive delays for the sake of clarity. However the same arguments apply to the case where the mapping implements non consecutive delays as in the rest on the chapter.

components in y_t except the first one are well known from the past. Projecting the mapping g onto its first coordinate, we can write:

$$x_t = f(x_{t-1}, x_{t-2}, \ldots, x_{t-m}) \tag{4.2}$$

We can see that this is the standard time series modelling problem. Therefore, the search for the embedding dimension is crucial to our problem, as **finding the embedding dimension is equivalent to finding the necessary delays** for the ideal mapping.

4.4 Geometric methods

A number of methods have been proposed in the nonlinear dynamics literature. All those we are aware of rely on some kind of geometrical argument, based on the observation that the mapping g (or f) is continuous. Thus close inputs should correspond to close outputs for the proper mapping.

If the number of delays is insufficient, close inputs could lead to an arbitrarily large difference in the output, due to the effect of the missing delays. On the other hand, when the number of delays is too high, a large difference in input (along the unnecessary dimensions) gives close outputs.

This is exemplified in figures 4.1 and 4.2. In these figure, we have generated data using the *Hénon map* :

$$x_t = 1 - 1.4x_{t-1}^2 + 0.3x_{t-2} \tag{4.3}$$

In that case, the embedding space is 2, and the primary delays that we expect to select are x_{t-1} and x_{t-2}. Each dot on the figure represents the input and output distance between two data, i.e (y_{t-1}, y_t) and $(y_{\ell-1}, y_\ell)$. The value on the x-axis is therefore $\|y_{t-1} - y_{\ell-1}\|$, while the value on the y-axis is $\|y_t - y_\ell\|$. When the input y_{t-1} (resp. $y_{\ell-1}$) contains only one delay x_{t-1} (resp. $x_{\ell-1}$), we obtain figure 4.1. The data that have very close inputs (left of the figure) can have arbitrarily distant outputs.

On the other hand, let us include the first two delays in the input vector, x_{t-1} and x_{t-2} (resp. $x_{\ell-1}$ and $x_{\ell-2}$). The situation is now that of figure 4.2: for every pair of data, if the inputs are close, the outputs are necessarily close too.

This example shows clearly that a continuous mapping between input and output can not be implemented unless we include at least the first two delays. Hence the geometrical approach would select x_{t-1} and x_{t-2} as relevant delays in this case.

4.5 A practical method

The δ-test method uses the geometric idea in a straightforward manner, by estimating empirically the probability that the output distance is smaller than a given ϵ when the input distance is smaller that δ. If we denote the input distance (along

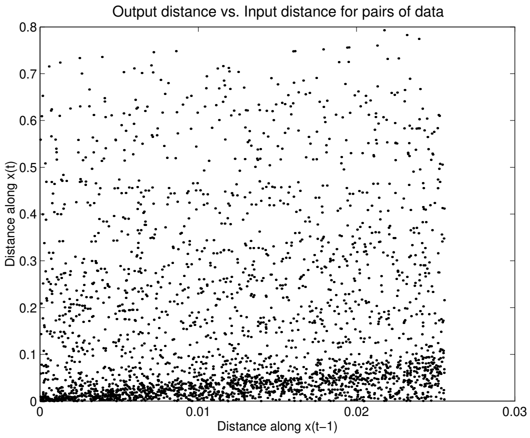


Figure 4.1: When the input contains only the first delay x_{t-1}, close inputs can correspond to arbitrarily distant outputs (left). Data are generated with the Hénon map; each dot corresponds to one input-output couple.

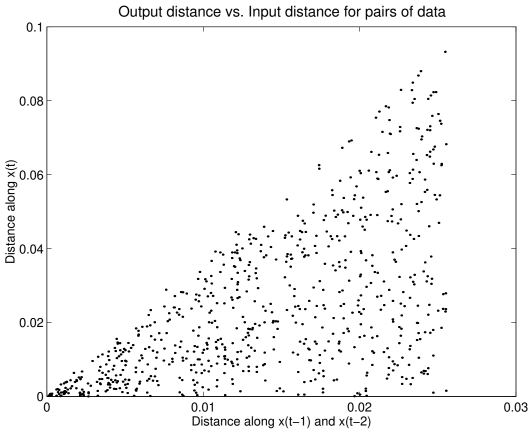


Figure 4.2: When the input contains the two first delays x_{t-1} and x_{t-2}, close inputs lead to close outputs. Data are generated with the Hénon map; each dot corresponds to one input-output couple.

each coordinate) by $\mathbf{d}$ and the output distance by d_t, we can write this probability as $P\left(d_t \leq \epsilon | \mathbf{d} \leq \delta\right)$, or simply $\mathrm{P}\left(\epsilon|\delta\right)$.

For a given input dimension m and a number of candidate delays i_m, consider the following sets:

$$A_m\left(\delta\right) = \left\{\left(j,k\right), j \neq k, \forall i \in \left\{i_1, \ldots, i_m\right\} \left|x_{j-i} - x_{k-i}\right| \leq \delta\right\} \tag{4.4}$$

$$B_m\left(\epsilon, \delta\right) = \left\{\left(j,k\right) \in A_m\left(\delta\right), \left|x_j - x_k\right| \leq \epsilon\right\} \tag{4.5}$$

We can now define the following empirical probabilities, where N_m is the total number of distinct (j,k) pairs:

$$\begin{aligned}
\mathrm{P}_m\left(\delta\right) &= \frac{\mathrm{card}\left(A_m\left(\delta\right)\right)}{N_m} = \widehat{P}\left(\mathbf{d} \leq \delta\right) \\
\mathrm{P}_m\left(\epsilon, \delta\right) &= \frac{\mathrm{card}\left(B_m\left(\epsilon, \delta\right)\right)}{N_m} = \widehat{P}\left(d_t \leq \epsilon, \mathbf{d} \leq \delta\right) \\
\mathrm{P}_m\left(\epsilon|\delta\right) &= \frac{\mathrm{P}\left(\epsilon, \delta\right)}{\mathrm{P}\left(\delta\right)} = \widehat{P}\left(d_t \leq \epsilon | \mathbf{d} \leq \delta\right)
\end{aligned}$$

In the special case where $m = 0$ (no delays), we set $A_0\left(\delta\right) = \left\{\left(j,k\right), j \neq k\right\}$, such that $\mathrm{P}_m\left(\epsilon|\delta\right)$ depends only on ϵ. The estimated probabilities above become quite noisy when the number of elements in set A_m and B_m are small. For this reason, we estimate the standard deviation of $\mathrm{P}_m\left(\epsilon|\delta\right)$. Notice that this estimate is the empirical average of a binomial variable (either a given couple satisfied the conditions on δ and ϵ, or it does not). The standard deviation is then estimated easily by:

$$\widehat{\sigma}_m\left(\epsilon|\delta\right) = \sqrt{\frac{\mathrm{P}_m\left(\epsilon|\delta\right)\left(1 - \mathrm{P}_m\left(\epsilon|\delta\right)\right)}{\mathrm{card}\left(A_m\right) - 1}} \tag{4.6}$$

This allows us to keep track of cases where the estimation of $\mathrm{P}_m(\epsilon|\delta)$ is made on an insufficient amount of data.

Generally speaking, $\mathrm{P}_m(\epsilon|\delta)$ increases with ϵ (laxer output test), and when δ approaches 0 (stricter input condition). Let us now define by $\mathrm{P}_m(\epsilon)$ the maximum over δ of $\mathrm{P}_m(\epsilon|\delta)$: $\mathrm{P}_m(\epsilon) = \max_{\delta>0} \mathrm{P}_m(\epsilon|\delta)$. The *dependability index* is defined as:

$$\lambda_m(\epsilon) = \frac{\mathrm{P}_m(\epsilon) - \mathrm{P}_{m-1}(\epsilon)}{1 - P_0(\epsilon)} \tag{4.7}$$

$P_0(\epsilon)$ represents how much data passes the continuity test when no input information is available. This dependability index measures how much of the remaining continuity information is associated with involving input i_m. This index is then averaged over ϵ with respect to the probability $(1 - P_0(\epsilon))$:

$$\overline{\lambda}_m = \int \lambda_m(\epsilon)\,(1 - P_0(\epsilon))\; d\epsilon \tag{4.8}$$

It is clear that $\lambda_m(\epsilon)$, and therefore its average, should be positive quantities. Furthermore, if the system is deterministic, the dependability is zero after a certain number of inputs, so the sum of averages saturates. If the system is also noise-free, they sum up to 1. For any m greater than the embedding dimension: $\sum_{j=1}^{m} \overline{\lambda}_j = \sum_{j=1}^{m} \lambda_m(\epsilon)$. In the presence of noise, this relation is only valid for values of ϵ that are larger than a given threshold ϵ_0.

In the examples below, any reference to the embedding dimension refers to results obtained using this method.

4.6 Statistical variable selection

Statistical variable selection (or feature selection) encompasses a number of techniques aimed at choosing a relevant subset of input variables in a regression or a classification problem. As in the rest of this document, we will limit ourselves to considerations related to the regression problem, even though most methods discussed below apply to classification as well. Variable selection can be seen as a part of the data analysis problem: the selection (or discard) of a variable tells us about the relevance of the associated measurement to the modelled system.

In a general setting, this is a purely combinatorial problem: given V possible variables, there is 2^V possible subsets (including the empty set and the full set) of these variables. Given a performance measure, such as prediction error, the only optimal scheme is to test all these subset and choose the one that gives the best performance. It is easy to see that such an extensive scheme is only viable when the number of variables is rather low, identifying 2^V models, when we have more than a few variable, requires too much computation.

A number of techniques have been devised to overcome this combinatorial limit. Some of them use an iterative, locally optimal technique to construct an estimate of the relevant subset in a number of steps. We will refer to them as *stepwise selection methods*, not to be confused with *stepwise regression*, a subset of these methods that we will address below.

In *forward selection*, we start with an empty set of variables. At each step, we select a candidate variable using a *selection criteria*, check whether this variable should be added to the set, and iterate until a given *stop condition* is reached.

On the contrary, *backward elimination* methods start with the full set of all input variables. At each step, the least significant variable is selected according to a *selection criteria*. If this variable is irrelevant, it is removed and the process is iterated until a *stop condition* is reached.

It is easy to devise examples where the inclusion of a variable causes a previously included variable to become irrelevant. It thus seems appropriate to consider running a backward elimination each time a new variable is added by forward selection. This combination of both approaches is known as *stepwise regression* in the linear regression context, even though the name seems rather misleading (forward selection and backward elimination are themselves stepwise methods).

In all cases, each step is composed of three operation: selecting a variable, checking whether or not it is relevant, and adding/removing it in accordance with its relevance. The first operation can be done with respect to a number of criteria: e.g. residual or adjusted mean squared error, squared correlation coefficient R^2, estimated prediction error or variance thereof. These criteria are among the most commonly used, but there is a wide range of possible criteria.

A number of *pruning schemes* have been adapted to the problem of variable selection. The generally calculate some kind of *saliency* (see section 2.15) associated with each variable, and prune those inputs (i.e. variables) that have the lowest saliency. As this description indicates, these are typical examples of *backward elimination*, even though the criteria involved are rather original in a statistical context.

In the experiments below, we will use a forward selection method that we will refer to as F-*inclusion*. Given p already selected variables and the associated f_p model, a variable i is added to the set of p variables if and only if the F-ratio for this variable exceeds a given level F_{in}:

$$F_i = \left(\frac{\sum_{k=1}^{N} \left(y^{(k)} - f_p\left(x^{(k)}\right)\right)^2 - \sum_{k=1}^{N} \left(y^{(k)} - f_{p+i}\left(x^{(k)}\right)\right)^2}{\widehat{\sigma}_i^2} \right) > F_{in} \tag{4.9}$$

where f_{p+i} is the model including variable i, $\widehat{\sigma}_i^2$ is an estimate of the noise level using the adjusted residuals $\frac{N\,S(w)}{N-(p+1)}$ as in sections 3.6 and 3.10. Equation 4.9 is an generalisation to non-linear models of the classical formula for linear regression. Given a confidence level[2] α, we take $F_{in} = F(\alpha, 1, n-p-1)$.

Let us add that regularisation is a convenient method to deal with the problem of variable selection, in at least two ways. First usual regularisation constraint such as weight-decay pull weights towards 0. In linear regression, there is a one-on-one relationship between weights and parameters. The analysis of the magnitude of the weights gives information on how the corresponding input contribute to the estimation, and thus on its relevance. With non-linear models it is not so easy, but inputs can be selected nonetheless. Second, a properly chosen regression level with *all* the

[2]In the experiments below, the confidence level will be written in subscript: F_{99}-inclusion refers to the scheme with $\alpha = 0.99$.

inputs is expected to provide better predictive abilities than a non-regularised model using a subset of the inputs. Prediction performance is, after all, what we are after. As appealing as this idea may be, it does not fit within the scope of the present study.

4.7 The case of time series

After these general considerations, let us narrow our study down to the case of time series modelling. As noticed above, this special case fits in the more general problem of variable selection, but it has a number of original features.

All possible variables are delays, and are thus available for selection.

On the other hand, the maximum number of delays is not known in advance, i.e. we do not start with an extensive set—apart from the case where we would start with all possible delays, hardly a practical suggestion. This feature makes our problem a rather bad candidate for backward elimination methods, and pruning schemes in particular.

For the same reason, the optimal, exhaustive method is ill-suited to time series. Furthermore, it is likely than any non-trivial time series modelling problem would be computationally too expensive to consider with this method. As an example, consider that the well-known sunspots benchmark problem, with 12 inputs (the widely accepted standard) would require 4096 different model identifications.

Both the extensive scheme and backward elimination would become are more attractive as soon as we have an upper bound on the possible relevant delays. Such a value can be estimated in a non-parametric fashion, either using the δ-test above, or some cruder, faster method. However, as discussed in section 4.13, such a non-parametric estimate is not necessarily useful in the context of parametric modelling.

However, there is a natural ordering in the possible inputs, and it seems intuitively reasonable to consider candidate delays from the most recent by going further in the past. This is by no means a guarantee that the resulting scheme is optimal: indeed our experiments below seem to show that it is not, but not by a wide margin.

4.8 The use of generalisation

As mentioned in the previous chapters, the ultimate goal in a modelling procedure is to minimise the generalisation error. For a model f_m predicting x_t from a set of past delays i_1, i_2, $\dots i_m$ as in (4.1), we denote by $\mathbf{x}^{(t)}$ the vector of past delays $(x_{t-i_1}, \dots, x_{t-i_m})$. The generalisation error or expected risk is defined as:

$$G(f_m) = \int \left(x_t - f_m\left(\mathbf{x}^{(t)}\right)\right)^2 p(x_t, \mathbf{x}^{(t)})\, \mathrm{d}\mathbf{x}^{(t)} \mathrm{d}x_t \tag{4.10}$$

Our goal is to produce a model that minimises this risk. Thus, the optimal set of delays (with respect to generalisation error) will be one that minimises (4.10). This suggest an alternative criterion to use in conjunction with a stepwise selection method.

Accordingly, we propose the following algorithm (σ_x^2 is the variance of the data):

1. Initialise: $d = 0$; $G_{min} = \sigma_x^2$; no input selected.
2. Model: $d = d + 1$; add delay $t - d$ to selected inputs; estimate generalisation error $\widehat{G}$ for resulting model.
3. Test: if $\widehat{G}$ is significantly smaller than G_{min},
   then keep delay $t - d$; $G_{min} = \widehat{G}$.
   Discard otherwise.
4. Iterate: Go to step 2 until stop condition is reached.

Contrary to the non-parametric methods relying on geometric arguments to estimate the embedding dimension, this algorithm is model-based and depends on a proper way of estimating generalisation error. Furthermore, it is a typical *forward selection* procedure. The topics of estimating this error and assessing a significant decrease will be addressed in the next section.

Finally, let us mention that the algorithm requires a stop condition. It can be similar to those used with a classical variable selection. It can also be some argument based on an estimated maximum delay, or a clever heuristic, e.g. stopping the algorithm when a sufficient time frame has been spanned without adding any delay. In the quasi-periodic data below, it seems reasonable to stop the algorithm when more than a period has been passed without inclusion.

4.9 Generalisation estimates and statistical significance

As noted in chapter 1, calculating the value of the generalisation error in (4.10) is usually not feasible. We than have to resort to the use of an estimator of this quantity. Following chapter 3, we consider two such estimates.

With *leave-one-out* cross-validation, the generalisation estimator is calculated by averaging over all examples the prediction error on each one of them for the model trained on the rest of the sample. If we have N input-output pair, and $f_m^{(t)}$ is the model trained after leaving example t out, the cross-validation score is:

$$\widehat{G} = \frac{1}{N} \sum_{t=1}^{N} \left(f_m^{(t)} \left(\mathbf{x}^{(t)} \right) - x_t \right)^2 \tag{4.11}$$

This expression can be generalised easily to the case of v-fold cross-validation.

Algebraic estimators of the average generalisation error are the obvious other choice. As noted in chapter 3, there are a number of different estimators. For the sake of simplicity, we will here consider the use of the *Final Prediction Error* using the effective number of parameters, i.e. an expression similar to GPE:

$$\widehat{G} = \left(\frac{N + \widehat{P}}{N - \widehat{P}} \right) \frac{1}{N} \sum_{t=1}^{N} \left(x_t - f_m \left(\mathbf{x}^{(t)} \right) \right)^2 \tag{4.12}$$

For cross-validation estimators, the generalisation estimate is an average of N terms, cf. (3.3) and (4.11). For algebraic estimators, it can also be written as such an average. Noting $e^{(t)} = \left(x_t - f_m\left(\mathbf{x}^{(t)}\right)\right)^2$, we have $\widehat{G} = \frac{1}{N}\sum_{t=1}^{N}\left(\frac{N+\widehat{P}}{N-\widehat{P}}\right)e^{(t)}$.

We take advantage of this fact by noticing that both estimators can be written as the empirical average of a statistics. In both cases, the terms under the sum are samples from an unknown distribution, the mean of which is the estimated generalisation error. Furthermore, for two models f_1 and f_2, a large number of the data on which these estimators are based are common. Typically, in two successive steps of the algorithm in section 4.8, the model having one more delay will have one less training example.

Using the common data, the significance of an apparent decrease in generalisation error can be checked using a *paired t-test*. This statistical test is used to check whether two sets of paired data have significantly different mean. It relies on the assumption that the paired difference between the two sets has a Gaussian distribution, and checks whether the mean of this difference is significantly different from 0.

For cross-validation, we just check whether the residuals in (4.11) for the two models. For algebraic estimators, we use the corrected residuals under the sum of (4.12): for two models f_1 and f_2, we test $\left(\frac{N+\widehat{P}_1}{N-\widehat{P}_1}\right)e_1^{(t)}$ against $\left(\frac{N+\widehat{P}_2}{N-\widehat{P}_2}\right)e_2^{(t)}$ using the paired t-test.

4.10 Experiment 1: the Hénon map

The first experiments are carried out on a small dynamical mapping from the non-linear dynamics literature, the *Hénon map*:

$$x_t = 1 - 1.4x_{t-1}^2 + 0.3x_{t-2} \tag{4.13}$$

With this artificial problem, we can generate a fairly large generalisation set of 10000 elements, that will hopefully provide a reliable unbiased estimate of the true generalisation error.

We experiment on both on non-noisy and noisy data, adding a Gaussian noise of variance $\sigma_\varepsilon^2 = 0.1$. We use both a linear model and a non-parametric kernel smoother, with Gaussian kernel. A linear model is obviously a poor choice here considering the non-linear nature of the system, but it serves quite well the purpose of demonstrating the workings of our delay selection scheme.

We generate a training set containing 500 data, and experiment with four different selection schemes:

1. A generalisation-based forward selection method using a large, 10000 elements validation set distinct from the test set.
2. The delay selection algorithm from section 4.8 using the FPE as a generalisation estimator for the linear model, and the LOO cross-validation estimator for the kernel smoother.

Hénon map:		No noise		Noisy	
		Linear	Kernel	Linear	Kernel
Large validation set	Delays	1–7	1–2	1–7	1–3
	MSE	0.376	0.000	0.503	0.214
	Gener.	0.379	0.000	0.389	0.067
Generalisation based algorithm	Delays	1,3–6	1–2	1,3,4	1–3
	MSE	0.376	0.000	0.523	0.214
	Gener.	0.378	0.000	0.409	0.067
F_{99}-inclusion	Delays	1–6	1–2	1–6,10	1–8,11–13, 16,17,19,20
	MSE	0.376	0.000	0.499	0.032
	Gener.	0.379	0.000	0.389	0.294
δ-test	Delays	1–2		1–2	
	MSE	0.457	0.000	0.567	0.266
	Gener.	0.455	0.000	0.459	0.097

Table 4.1: Results on the noisy and non-noisy Hénon map data, for two models: a linear model and a non parametric Kernel smoother.

3. The F_{99}-inclusion, as explained in section 4.6.
4. The δ-test outlined in section 4.5.

In both the noisy and non-noisy case, the δ-test selects the physically appropriate delays: 1 and 2. As noted above, this non-parametric test is independent of the model (linear or kernel) used for the prediction.

The results are gathered in table 4.1. For each experiment, we provide three pieces of information:

Delays The list of delays obtained,

MSE the mean squared error on the training set for the resulting delays,

Gener. the generalisation error, estimated from the 10000 elements generalisation set, for the resulting delays.

The full use of the first method (generalisation based with a large validation set) with the kernel smoother model requires the non-parametric estimation of 10000 data from 500 points. This is computationally too expensive (and requires a large amount of memory), so we will limit our validation set to 3000 data in that case.

4.11 Discussion of the Hénon map experiment

We can see on table 4.1 that all three forward selection methods outperform the δ-test in our experiments with the linear model. This is of course no surprise as a linear combination of the first two inputs is obviously not sufficient to represent the Hénon map accurately. The generalisation-based algorithm (using the FPE as a generalisation estimate) favours a parsimonious model, with only 5 inputs on the non-noisy data, while keeping generalisation abilities at the same level as the other two parametric methods.

For the Kernel smoother, we notice that in the non-noisy case all method produce the expected results. The noisy case is more interesting. The F_{99} inclusion scheme leads to an extreme case of curse of dimensionality. It selects far too many delays, leading to a clear problem of over-fitting, as shown by the generalisation error. On the other hand, the training error is suspiciously low, far below the noise level.

The generalisation-based algorithm selects an additional, “unnecessary” delay, but this leads to an actual decrease in the generalisation error. This is a well known effect of algebraic estimates of the average generalisation error. They tackle the effect of a limited training set size by selecting slightly over-parameterised models. The important aspect however is that it leads to good generalisation performance. It is indeed the case here, and we obtain slightly better prediction abilities both on the training and on the generalisation set.

The large validation set method is supposed to act as a reference for generalisation-based methods. In the linear case, the selection of the last delay $t-7$ is due to finite sample size. The results of the other two do not correspond exactly, but the overall performance is very similar anyway. For the kernel smoother, it leads to the same results as our generalisation-based algorithm, which is comforting. However, it is mostly relevant to the ability of the chosen generalisation estimator (LOO or FPE) to accurately represent the variations in the actual generalisation error.

This example displays two important features of our generalisation method. First its ability to add delays that are further in the past when it is necessary. This contrasts with the non-parametric technique which selects the same delays regardless of the model. Second, its emphasis on the generalisation error, which is indeed the quantity of interest when we are interested in e.g. time series prediction. This is also an interesting contrast with the embedding dimension, which reflects the “physical” structure of the data.

4.12 Experiment 2: the Fraser river data

We will now apply our delay selection scheme to a real time series. We have chosen a freely available dataset. It contains the mean monthly flow of the *Fraser River* in Hope, British Columbia, from march 1913 to December 1990. Due to natural cycles, there is a rough periodicity in the data, with maxima every 11 to 13 months. It is partly displayed in figure 4.3. There are 946 measurements that we split in two subsets: 315 ($\sim$ one third) of the available data for training, and the remaining 613 ($\sim$ two thirds) as a test set to estimate the generalisation abilities of the resulting model. This rather bold choice will allow us to analyse the behaviour of different methods in a case where data is rather scarce.

We will consider the same modelling techniques as above, and add non-linear neural networks models. In that case, the generalisation error will be estimated as in (4.12).

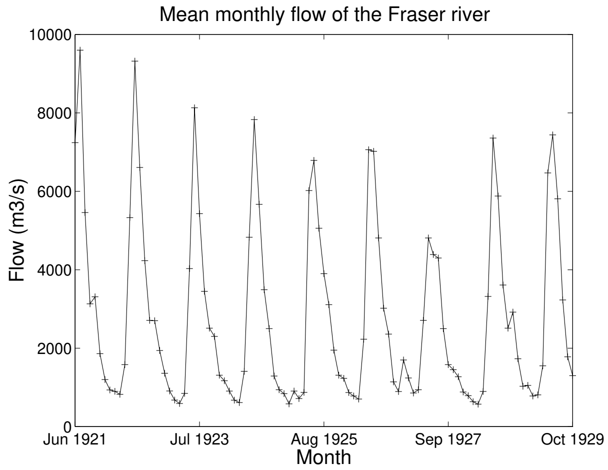


Figure 4.3: Portion of the Fraser river data set, between June 1921 and October 1929. The rough yearly periodicity in the data appears clearly.

For all models, the significance level in the paired t-test is set to 95% as before.

Performing a δ-test on the Fraser river data leads us to select delays 1, 2, 4, 7, 8 and 11 as relevant inputs to our models.

Among the parametric methods mentioned above, the "large validation set" method is obviously not applicable in this case: there is no such thing as a large validation set. If there were, we would probably be better off using part of it for estimating the models. The disappointing F-inclusion method will also be discarded, so that we test our delay selection algorithm against the inputs selected by the δ-test.

In the linear case, we select 19 delays ranging from 1 to 49, i.e. four "periods" of the time series. With the kernel smoother, only 14 delays are selected, from 1 to 27, i.e. only two periods of the data. Finally, the neural network model leads to a selection of only 9 parameters: 1 to 4, 7, 10 and 11 and 23.

The results of these experiments are summarised in figure 4.4. Clearly, the best models are the linear model and the non-linear neural networks, with inputs selected using the generalisation based algorithm. Only with the kernel smoother does this algorithm lead to a decrease in performance compared to the embedding dimension method. In that case, the low performance is due to the inability of the simple kernel smoother to handle high dimensional inputs, leading to a severe case of the so-called *curse of dimensionality*. In upper-right of the plot, the linear model using the 6 inputs corresponding to the embedding space displays a possible under-fit.

The prediction of the linear and neural network models are presented on figure 4.5.

4.13 Discussion and conclusions

The methods based on estimating the embedding dimension yield homogeneous results and depends only on the data. The specification of a model is actually not even

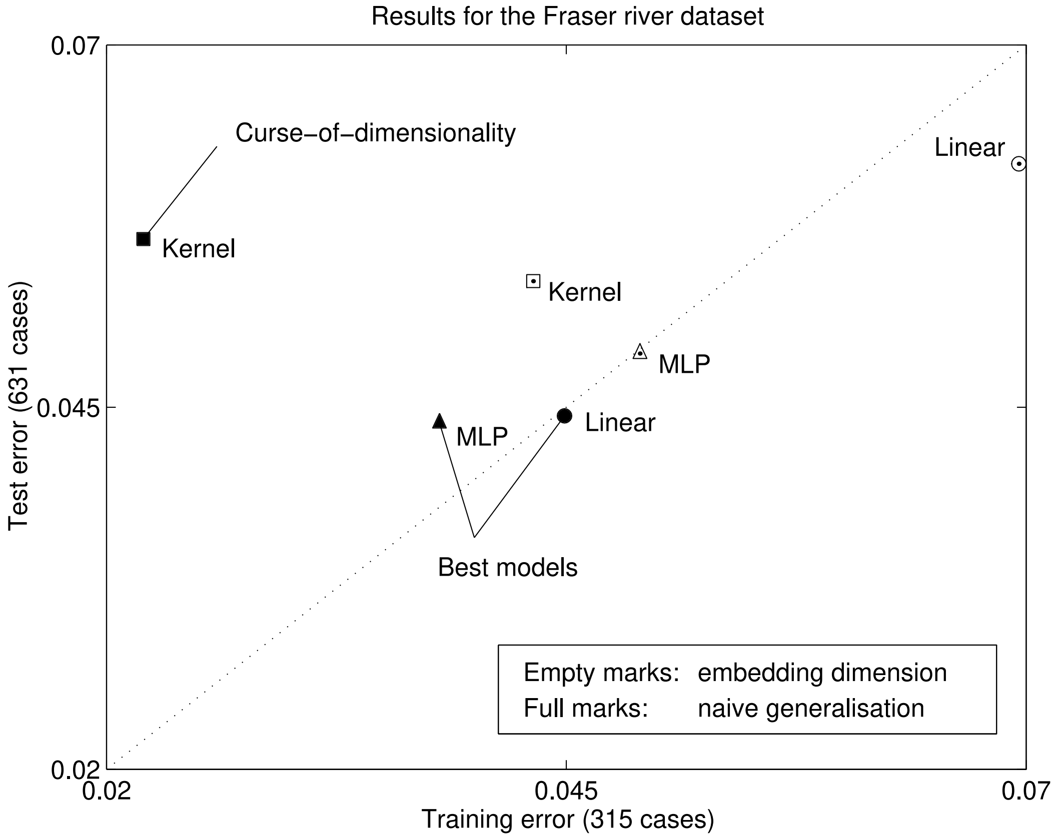


Figure 4.4: Results for the Fraser river time series for three different models: linear, kernel smoother and neural network. The lower the point, the better the prediction accuracy. The black square in the upper left shows a bad case of *curse-of-dimensionality*, which is typical for kernel smoother in high dimensions.

necessary. A consequence is that it performs badly for non-flexible models.

On the other hand, the generalisation based algorithm is model dependent. It has to be applied for each model and can prove really time-consuming. However, it optimises directly the quantity of interest: the generalisation error.

In our experiments, the generalisation-based methods tend to select more delays, further in the past, as long as the (estimated) generalisation error decreases. Typically, the linear model uses delays over 4 periods of the time series, and provides excellent predictive performance. A consequence is that the selected delays have nothing to do with the actual, "physical" embedding dimension.

The non-parametric δ-test needs a large amount of data to provide reliable results. On the other hand, the generalisation-based method uses the available data to probe further into the lag-space (e.g. delay 49 for the linear model).

The generalisation-based method is a typical *forward selection* procedure. Performing a *backward elimination* step along the same lines on the set of inputs selected for the Neural Networks, we realise that deleting inputs 4, 7, 10 and 23 actually leads to a decrease in (estimated) generalisation error. The resulting neural network has

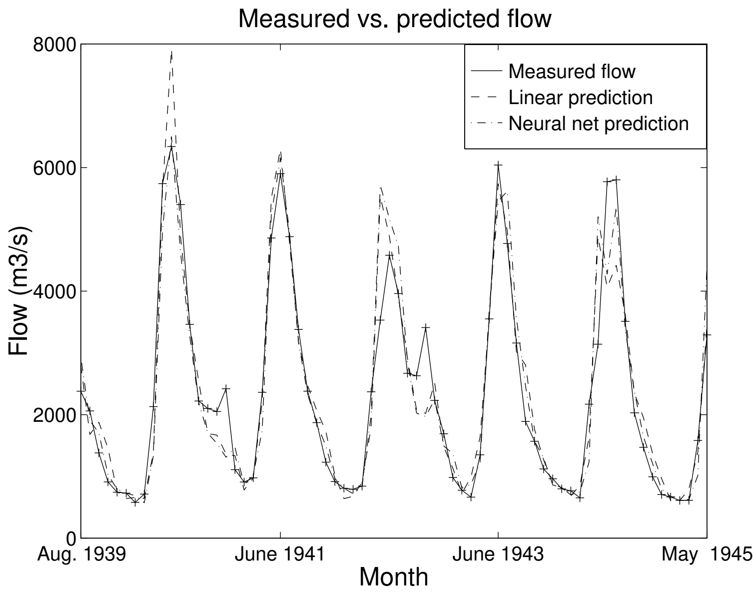


Figure 4.5: Another portion of the Fraser river data set, between August 1939 and May 1945. Notice the way the linear model overestimates the first peak, and both models seem to smooth secondary peaks.

only 5 inputs, and $0,0423$ / $0,0542$ / $0,0425$ as training, estimated generalisation and validation performance (respectively).

COMMENTS

The investigations in this chapter were mainly published as (Goutte, 1997b,a).

4.4 Apart from the δ-test presented below, geometrical approaches include those of Aleksić (1991) or Savit and Green (1991). Molina et al. (1996) also uses a geometrical argument, similar to that used earlier by He and Asada (1993) in the context of non-linear input-output system identification.

4.5 The δ-test presented in this section was developed by Pi and Peterson (1994). It was later refined to estimate the level of noise in a time series without doing regression. It is also used for comparison by Molina et al. (1996)

4.6 The topic of variable/feature selection is a very fruitful research field in statistics. A good introduction to classical methods is given by Hocking (1976). For a neural networks perspective, see Leray (1996). Cibas et al. (1994) use a method related to OBD to select variables in a neural network model.

4.8 This approach has been advocated in (Goutte, 1997b) in a coarser form. For references on cross-validation and algebraic estimators, see chapter 3.

4.9 The use of the paired t-test to check whether two distribution have different variance is explained very well by Press et al. (1992). The use of this test to check the significance of a decrease in error is mentioned by Larsen and Hansen (1995), and has been applied to delay selection by Goutte (1997a).

4.10 The Hénon map is a classic example in the non-linear dynamics literature. It is used for example (and among others) by Aleksić (1991), Pi and Peterson (1994) or Molina et al. (1996), already cited.

4.12 The dataset used in this section is available from `statlib`, in the `datasets` directory. It was first used by McLeod (1994).

References

Aleksić, Z. (1991). Estimating the embedding dimension. *Physica D*, 52:362–368.

Cibas, T., Fogelman Soulié, F., Gallinari, P., and Raudys, S. (1994). Variable selection with Optimal Cell Damage. In *Proceedings of ICANN'94*, pages 727–730.

Goutte, C. (1997a). Extracting the relevant delays in time series modelling. In *Neural Networks for Signal Processing VII – Proceedings of the 1997 IEEE Workshop*, number VII in NNSP, Piscataway, New Jersey. IEEE.

Goutte, C. (1997b). Lag space estimation in time series modelling. In *Proceedings of ICASSP'97*. IEEE.

He, X. and Asada, H. (1993). A new method for identifying orders of input-output models for nonlinear dynamic systems. In *American Conference on Control*, San Francisco, California.

Hocking, R. R. (1976). The analysis and selection of variables in linear regression. *Biometrics*, 32:1–49.

Larsen, J. and Hansen, L. K. (1995). Empirical generalization assessment of neural network models. In Girosi, F., editor, *Neural Networks for Signal Processing V – Proceedings of the 1995 IEEE Workshop*, number V in NNSP, pages 42–51, Piscataway, New Jersey. IEEE.

Leray, P. (1996). La sélection de variables. Technical report, Laforia.

McLeod, A. I. (1994). Diagnostic checking of periodic autoregression models with application. *Journal of Time Series Analysis*, 15(2):221–233.

Molina, C., Sampson, N., Fitzgerald, W. J., and Niranjan, M. (1996). Geometrical techniques for finding the embedding dimension of time series. In *Neural Networks for Signal Processing VI – Proceedings of the 1996 IEEE Workshop*, number VI in NNSP, pages 161–169, Piscataway, New Jersey. IEEE.

Pi, H. and Peterson, C. (1994). Finding the embedding dimension and variable dependences in time series. *Neural Computation*, 6(3):509–520.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (1992). *Numerical Recipes in C*. Cambridge University Press, 2nd edition.

Savit, R. and Green, M. (1991). Time series and dependent variables. *Physica D*, 50(1):95–116.

5

Bayesian estimation

5.1 Introduction

This chapter takes a different standpoint to address the problem of learning. We will here reason only in terms of probability, and make extensive use of the chain rule known as "Bayes' rule".

A fast definition of the basics in probability is provided in appendix A for quick reference. Most of this chapter is a review of the methods of Bayesian learning applied to our modelling purposes. Some original analyses and comments are also provided in section 5.8, 5.11 and 5.12.

There is a latent rivalry between "Bayesian" and "Orthodox" statistics. It is by no means our intention to enter this kind of controversy. We are perfectly willing to accept orthodox as well as unorthodox methods, as long as they are scientifically sound and provide good results when applied to learning tasks. The same disclaimer applies to the two frameworks presented here. They have been the object of heated controversy in the past 3 years in the neural networks community. We will not take side, but only present both frameworks, with their strong points and their weaknesses.

In the context of this work, the "Bayesian frameworks" are especially interesting as the provide some continuous update rules that can be used during regularised cost minimisation to yield an automatic selection of the regularisation level. Unlike the methods presented in chapter 3, it is not necessary to try several regularisation levels and perform as many optimisations. The Bayesian framework is the only one in which training is achieved through a *one-pass* optimisation procedure.

5.2 Bayes' rule

The simple application of the conditional probability definition above allows us to write that for two events A and B, we have:

$$\mathrm{P}(A \cap B) = \mathrm{P}(A|B)\,\mathrm{P}(B) \tag{5.1}$$

And we also have:

$$\mathrm{P}(B \cap A) = \mathrm{P}(B|A)\,\mathrm{P}(A) \tag{5.2}$$

As we obviously have $\mathrm{P}(A \cap B) = \mathrm{P}(B \cap A)$, we get from (5.1) and (5.2) the following formula:

$$\mathrm{P}(B|A) = \frac{\mathrm{P}(A|B)\,\mathrm{P}(B)}{\mathrm{P}(A)} \tag{5.3}$$

Bayes' rule was supposedly discovered in an unpublished manuscript by Thomas Bayes in the 18th century. It was rediscovered by Laplace later in the same century, but it is only in the past decades that Bayesian statistics became a hot topic (in many ways!).

As we can see, this rule derives from very simple considerations in statistics. It is also very useful as it allows to combine inferences. Indeed, it is common to introduce in (5.3) an additional term C. This term represents the "context", conditioning all the probabilities. It also allows to produce several levels of inference, as probabilities conditioned on C can in turn be calculated using Bayes' rule. If we write for example that $\mathrm{P}(A|BC) = \frac{\mathrm{P}(B|AC)\mathrm{P}(A|C)}{\mathrm{P}(B|C)}$, we can in turn combine this with the result of an other inference in the form of $\mathrm{P}(A|C) = \frac{\mathrm{P}(C|A)\mathrm{P}(A)}{\mathrm{P}(C)}$.

This formalism is very useful to incorporate new knowledge in our inference, or to update the results once new information is available.

5.3 Regression estimation

Let us now link the above with our regression estimation problem. An unknown system has produced a set of data $\mathcal{D} = \left(x^{(i)}, y^{(i)}\right)$ sampled independently. The model f_w is parameterised by a weight vector w.

Our goal of the estimation is to obtain an approximation of the output y, given the input x of the system. In terms of probability, this means that we are interested in the predictive distribution:

$$\mathrm{P}(y|x, \mathcal{D}) = \int \mathrm{P}(y|x, w, \mathcal{D})\,\mathrm{P}(w|\mathcal{D})\; dw \tag{5.4}$$

This predictive distribution is obtained by integrating over the model parameters (i.e. over all possible models) the conditional predictive distribution $\mathrm{P}(y|x, w, \mathcal{D})$ given by the error model. If the noise is assumed to be Gaussian with σ^2 variance, for example, we have:

$$\mathrm{P}(y|x, w, \mathcal{D}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\|y - f_w(x)\|^2}{2\sigma^2}\right) \tag{5.5}$$

where σ is assumed known. If it is not the case, the conditional predictive distribution is obtained by integrating (5.5) over the posterior noise distribution $\mathrm{P}(\sigma|\mathcal{D})$. This calculation involves Bayes' rule again and is similar to the treatment of the hyper-parameter that we will address later.

Knowledge of the predictive distribution allows to estimate the optimal output $\widehat{y}$. This can be done for example by:

$$\widehat{y} = \int y \, \mathrm{P}(y|x, \mathcal{D}) \; dy \tag{5.6}$$

which is the average of y.

The predictive distribution (5.4) involves the conditional predictive distribution, as well as the posterior weight distribution $\mathrm{P}(w|\mathcal{D})$. In order to obtain this value, we will use Bayesian inference, and Bayes' rule.

5.4 Bayesian inference on the parameters

We will now switch our interest from the distribution of the output to the distribution of the weights. The natural thing for estimating $\mathrm{P}(w|\mathcal{D})$ is to use Bayes' rule to infer the posterior distribution of the weights:

$$\mathrm{P}(w|\mathcal{D}) = \frac{\mathrm{P}(\mathcal{D}|w)\,\mathrm{P}(w)}{\mathrm{P}(\mathcal{D})} \tag{5.7}$$

Now the term $\mathrm{P}(\mathcal{D}|w)$ is of course the *likelihood* of the model that we recall from equation (1.6).

$$\mathrm{P}(\mathcal{D}|w) = \prod_{i=1}^{N} \mathrm{P}\left(y^{(i)}|x^{(i)}, w\right) \tag{5.8}$$

where $\mathrm{P}\left(y^{(i)}|x^{(i)}, w\right)$ is the likelihood of example i. This equation results directly from the independence assumption. In the case of a Gaussian assumption on the output noise, we get exactly equation (1.6):

$$\mathrm{P}(\mathcal{D}|w) = \left(2\pi\sigma^2\right)^{-\frac{N}{2}} \exp\left(-\frac{N}{2\sigma^2} S(w)\right) \tag{5.9}$$

The second term is the prior distribution on the weights. This distribution is parameterised by the *hyper-parameter* α. The prior on the weights becomes $\mathrm{P}(w|\alpha)$, which is linked with $\mathrm{P}(w)$ by:

$$\mathrm{P}(w) = \int \mathrm{P}(w|\alpha)\,\mathrm{P}(\alpha) \; d\alpha \tag{5.10}$$

$\mathrm{P}(\alpha)$ is the prior on the hyper-parameter, usually a *non-informative prior*. The setting of such priors will be discussed in section 5.10. In some cases, as we will see in section 5.9, it is extremely easy to calculate the exact weight prior $\mathrm{P}(w)$ by performing the integration of (5.10).

There remains a problematic issue, however, as we do not have the expression of $\mathrm{P}(\mathcal{D})$, the denominator of (5.7). It can be obtained by integrating over the weights by:

$$\mathrm{P}(\mathcal{D}) = \int \mathrm{P}(\mathcal{D}|w)\,\mathrm{P}(w)\; dw \tag{5.11}$$

The two quantities involved are the likelihood and the weight prior, which we have already tackled. Unfortunately, even though the expression of both probabilities are known, the integral is intractable even for simple cases.

The intractability of $\mathrm{P}(\mathcal{D})$ also deprives us of an attractive estimate of the model parameters. Indeed, given the posterior distribution $\mathrm{P}(w|\mathcal{D})$, we could compute a number of statistics on the parameters. For example, given a quadratic loss function on the parameters, the best estimate would be:

$$\widehat{w} = \int w\,\mathrm{P}(w|\mathcal{D})\; dw \tag{5.12}$$

As (5.7) can not be calculated exactly, neither the optimal Bayesian regression estimate $\widehat{y}$, nor the best set of parameters can be derived. Numerical integration could tackle this problem. However, several levels of integration are needed, so this method is not viable. Another solution evoked later is to rely on the mode of the distribution rather than on the mean.

Not surprisingly, Bayesian inference applied to our learning problem does not give any simple answer. It will be necessary to use some approximations to escape the deadlock. This is the purpose of section 5.6 and 5.9.

5.5 Prior on the weights and regularisation

Let us now leave the calculations to focus on the links between the above derivations and the regularised regression addressed in the previous chapters.

The idea of a *prior* on the parameters does not mean that we have to guess the values of the weights in order to be able to actually calculate them. Rather, the prior represents the state of our knowledge on a given quantity. If we have no reason to favour a particular value, we will use a uniform prior. This will typically be the case for the bias parameters in a neural network. Indeed, we have shown in section 2.13 that a linear transformation of the input (resp. output) values leads to a linear transformation of the input (resp. output) biases. There is thus no reason why a given value should *a priori* (i.e. before seeing the data) be preferred.

As for the other parameters of a neural network model, the fact that small weights should be privileged and analytical convenience led to the choice of a *Gaussian prior*:

$$\mathrm{P}(w|\alpha) = \left(\frac{\alpha}{\pi}\right)^{\frac{P}{2}} \exp\left(-\alpha \left\|w\right\|_2^2\right) \tag{5.13}$$

Other priors can be used. In particular, if we argue that what matters is the absolute value of the weights rather than their mean *and* variance, the maximum entropy distribution is a *Laplace prior*:

$$\mathrm{P}(w|\alpha) = \left(\frac{\alpha}{2}\right)^{P} \exp\left(-\alpha \left|w\right|\right) \tag{5.14}$$

The idea here is to introduce our prior belief on the weight distribution. Hopefully, it will guide the learning procedure towards a solution in accordance with these beliefs. Now replace "prior belief" by "constraint" and "guides" by "constrains", and the resulting scheme sounds very much like regularisation. Let us write the posterior distribution of the weight knowing the hyper-parameter using Bayes' rule:

$$\mathrm{P}\left(w|\mathcal{D},\alpha\right) = \frac{\mathrm{P}\left(\mathcal{D}|w\right)\mathrm{P}\left(w|\alpha\right)}{\mathrm{P}\left(\mathcal{D}|\alpha\right)} \tag{5.15}$$

Taking the minus logarithm of (5.15), we have:

$$-\ln\mathrm{P}\left(w|\mathcal{D},\alpha\right) = -\ln\mathrm{P}\left(\mathcal{D}|w\right) - \ln\mathrm{P}\left(w|\alpha\right) + \mathrm{cte} \tag{5.16}$$

We know since section (1.3) that there is a link between the log-likelihood and the empirical risk. In the Gaussian case, $-\ln\mathrm{P}\left(\mathcal{D}|w\right) = \frac{N}{2\sigma^2}S\left(w\right) + \mathrm{cte}$. As for the log-prior, in both cases above, it can be written as $-\ln\mathrm{P}\left(w|\alpha\right) = \alpha R\left(w\right) + \mathrm{cte}$. Rewriting (5.16), we have:

$$-\ln\mathrm{P}\left(w|\mathcal{D},\alpha\right) \propto S\left(w\right) + \frac{2\alpha\sigma^2}{N}R\left(w\right) + \mathrm{cte} \tag{5.17}$$

Up to an irrelevant constant[1], this is proportional to the expression of the regularised cost, such as in equation (2.21) for example.

Minimising the regularised cost will thus lead us to the *maximum posterior* solution. Furthermore, we see that this links the regularisation functionals used earlier to actual priors on the weight distribution. This clarifies the name of "Laplace prior" mentioned in connection with the regulariser used in section 2.16.

Furthermore, as we shall see in the following sections, this link allows us to use the results of the Bayesian estimation framework to provide means of setting the regularisation parameter in a regularised cost minimisation procedure.

5.6 The evidence framework

In order to avoid the intractability of the direct calculation of the posterior weight distribution, it is possible to consider estimating $\mathrm{P}\left(w|\mathcal{D}\right)$ by integrating over α:

$$\mathrm{P}\left(w|\mathcal{D}\right) = \int \mathrm{P}\left(w|\alpha,\mathcal{D}\right)\mathrm{P}\left(\alpha|\mathcal{D}\right)\, d\alpha \tag{5.18}$$

where $\mathrm{P}\left(\alpha|\mathcal{D}\right)$ is called the *evidence*, and $\mathrm{P}\left(w|\alpha,\mathcal{D}\right)$ can be obtained using Bayes' rule like in (5.15).

The *evidence procedure* considers that probability $\mathrm{P}\left(\alpha|\mathcal{D}\right)$ is sharply peaked about its maximum $\widehat{\alpha}$. In that condition, the integral (5.18) can be collapsed as if $\mathrm{P}\left(\alpha|\mathcal{D}\right)$ was a Dirac distribution $\delta\left(\alpha-\widehat{\alpha}\right)$:

$$\mathrm{P}\left(w|\mathcal{D}\right) \approx \mathrm{P}\left(w|\mathcal{D},\widehat{\alpha}\right) \tag{5.19}$$

[1]because independent of w.

According to (5.15), this leads to:

$$\mathrm{P}(w|\mathcal{D}) \propto \mathrm{P}(\mathcal{D}|w)\,\mathrm{P}(w|\widehat{\alpha}) \tag{5.20}$$

Following the remarks in the last section, we notice that finding the maximum posterior weight is equivalent to a regularised cost minimisation, where the regularisation parameter is linked to the value $\widehat{\alpha}$ maximising the evidence.

5.7 Bayesian inference on the hyper-parameter

The crucial step in the evidence procedure is to find $\widehat{\alpha}$, the value of the hyper-parameter that maximises the evidence $\mathrm{P}(\alpha|\mathcal{D})$.

The evidence can be found using Bayes' rule again:

$$\mathrm{P}(\alpha|\mathcal{D}) = \frac{\mathrm{P}(\mathcal{D}|\alpha)\,\mathrm{P}(\alpha)}{\mathrm{P}(\mathcal{D})} \propto \mathrm{P}(\mathcal{D}|\alpha)\,\mathrm{P}(\alpha) \tag{5.21}$$

The evidence is proportional to the product of the likelihood of the hyper-parameter and the prior on the hyper-parameter. We assume a flat prior on $\ln\alpha$, since α is a scale parameter on which we have no *a priori* information. The change from α to $\ln\alpha$ does not influence the calculation of the likelihood, as conditioning on α or $\ln\alpha$ is similar.

It should be noted that in some cases, as the likelihood (and not the posterior) is maximised, the likelihood itself has been labeled the "evidence". This abuse is very acceptable in this case, as the flat prior over $\ln\alpha$ means that one is equivalent to the other with respect to maximisation.

The likelihood/evidence is estimated by integrating the likelihood of the weights over the parameters:

$$\mathrm{P}(\mathcal{D}|\alpha) = \int \mathrm{P}(\mathcal{D}|w)\,\mathrm{P}(w|\alpha)\;dw \tag{5.22}$$

Another approximation is involved here as we recall that the integrand is proportional to the regularised cost (cf. section 5.5). Using the quadratic approximation of this cost in the minimum $\widehat{w}$, we recall that $C(w) = C(\widehat{w}) + \frac{1}{2}(w-\widehat{w})^{\top}\mathbf{H}(w-\widehat{w})$, where $\mathbf{H}$ is the Hessian as usual. Equation (5.22) then turns into a tractable Gaussian integral:

$$\mathrm{P}(\mathcal{D}|\alpha) = \int \frac{\exp\left(-\gamma C(\widehat{w})\right)}{Z_{\mathcal{D}} Z_W} \exp\left(-\frac{1}{2}(w-\widehat{w})^{\top}\gamma\mathbf{H}(w-\widehat{w})\right) dw \tag{5.23}$$

where γ comes from the fact that the term in the exponential is proportional to our previous definition of the regularised cost, typically involving the *average* squared error. In that case, $\gamma = \frac{N}{2\sigma^2}$. The normalizing terms $Z_{\mathcal{D}}$ and Z_W depend on the normalization of the likelihood, and the normalization of the prior, respectively.

For a Gaussian likelihood, the normalization is $Z_{\mathcal{D}} = (2\pi\sigma^2)^{\frac{N}{2}}$. As for the prior, if it is a Gaussian distribution, it is normalized by $Z_W = (\pi/\alpha)^{\frac{\widehat{P}}{2}}$, whereas for a Laplace distribution, the normalization constant is $Z_W = (2/\alpha)^{\widehat{P}}$, according to equations (5.13) and (5.14). It should be noted that $\widehat{P}$ here corresponds to the number of parameters with the associated prior. It is not necessarily the total number of weights. Indeed, as noted above (cf. section 5.5), we might want to use a flat prior over the bias, in which case the value of $\widehat{P}$ has to be adapted consequently.

Integrating the Gaussian integral (5.23), and taking the minus-log of the result, we get the log-likelihood:

$$-\ln \mathrm{P}\left(\mathcal{D}|\alpha\right) = \ln Z_{\mathcal{D}} + \ln Z_W + \frac{NS\left(\widehat{w}\right)}{2\sigma} + \alpha R\left(\widehat{w}\right) + \frac{1}{2}\ln\det\gamma\mathbf{H} \tag{5.24}$$

The expression of Z_W, $R\left(w\right)$ and $\mathbf{H}$ depend on the prior we use, so the expression of the log-likelihood, and hence that of the maximum $\widehat{\alpha}$ is prior-dependent.

The basic evidence procedure is there, however. With a first setting of α, we minimise the regularised cost to obtain $\widehat{w}$. The quadratic approximation around $\widehat{w}$ leads to a calculation of the evidence, the maximum of which gives a new estimate for α. In the case of a Gaussian weight likelihood of known variance σ^2 and a Laplace prior on weights, the update formula is:

$$\widehat{\alpha} = \frac{\widehat{P}}{R\left(\widehat{w}\right)} \tag{5.25}$$

In terms of regularised cost minimisation, the evidence reduces to a continuous update of the regularisation parameter ξ by $\widehat{\xi} = \frac{2\sigma^2\widehat{P}}{NR\left(\widehat{w}\right)}$.

If the noise variance is not known, it too can be treated as a hyper-parameter, e.g. $\beta = \frac{1}{2\sigma^2}$. It can be estimated by maximising the evidence with respect to β, leading to the following update formula for the regularisation parameter:

$$\widehat{\xi} = \frac{\widehat{P}\,S\left(\widehat{w}\right)}{\left(N-P\right)R\left(\widehat{w}\right)}. \tag{5.26}$$

Notice the use of $\widehat{P}$ in the numerator, and P in the denominator.

Once found, the maximum evidence hyper-parameter can be used to obtain regression estimates by integrating over the weights and combining (5.4) with the evidence approximation (5.19):

$$\widehat{y} = \int f_w\left(x\right)\mathrm{P}\left(w|\widehat{\alpha},\mathcal{D}\right)\,dw \tag{5.27}$$

5.8 Bayesian criticism of the evidence framework

The evidence framework is an attractive method. It shares some very direct similarities with straight regularised regression, in the way that it tries to infer the optimal hyper-parameter/regularisation level.

Furthermore, the evidence is a seductive quantity, allowing to perform several levels of Bayesian inference. In a first level, for a given model, $\widehat{\alpha}$ is chosen by maximising the evidence $\mathrm{P}\left(\mathcal{D}|\alpha\right)$. On a higher level, among a number of models, the best one $\widehat{M}$ is also chosen by maximising the evidence of the model $\mathrm{P}\left(\mathcal{D}|M_l\right)$.

This leads us to consider one of the critics against this framework. As noted above, the evidence is often taken as the likelihood of the hyper-parameter or the model. Maximising the evidence is then nothing more than finding the maximum likelihood solution, hardly a Bayesian ideal.

Another cause for worries is that the evidence framework relies on a number of assumptions in the course of the calculation: The evidence approximation (5.19), and the Gaussian assumption (5.23). The Gaussian assumption can be rather poor when the posterior weight distribution is not Gaussian, e.g. when it has a sharp mode away from the maximum likelihood. However, it is possible to effectively integrate (5.22) using e.g. the Metropolis algorithm or better, the Hybrid Monte Carlo method.

We will thus address mainly the problem of the first approximation. Let us assume that we have a Gaussian likelihood $\mathrm{P}(\mathcal{D}|w)$ and a Gaussian distribution for the weights conditioned on the hyper-parameter $\mathrm{P}(w|\alpha) = \sqrt{\frac{\alpha}{\pi}} \exp\left(-\alpha \|w\|^2\right)$. Integrating this distribution over the hyper-parameter in a manner similar as section 5.9 later, we get the prior on the weights $\mathrm{P}(w)$. This integration is done with a non-informative prior proportional to $1/\alpha$, leading to:

$$\mathrm{P}(w) \propto \frac{1}{\|w\|} \tag{5.28}$$

Now consider the expression of the posterior weight distribution (5.7) given by Bayes' rule. It is proportional to the product of the likelihood, a Gaussian, and the prior weight distribution, which, as (5.28) shows well, is nowhere near a Gaussian. The posterior distribution will thus *not* be Gaussian. This contradicts the evidence approximation (5.19). Indeed, according to Bayes' rule in (5.15), the evidence approximation for the posterior, $\mathrm{P}(w|\mathcal{D}, \widehat{\alpha})$, is a product of two Gaussian distributions, and thus Gaussian.

A simple difference can of course not been interpreted as a flagrant failure of the method. It has been argued that even though the distributions are not similar, the evidence approximation puts "most of the mass in the right place". It is beyond our scope to investigate thoroughly the basis of this claim. However, we will mention that sources cited in the COMMENTS section have argued against it, and proposed upper and lower bounds of the evidence error, i.e. the difference between e.g. the estimate given by the evidence in (5.27) and the Bayesian estimate obtained using the correct posterior distribution $\mathrm{P}(w|\mathcal{D})$ in the same conditions.

5.9 The MAP framework

Let us now go back to equation (5.7) in section 5.4. It gives the posterior distribution of the parameters. We have noted that the likelihood is a well known quantity, and that the prior on the weights can be estimated by integrating over the hyper-parameter:

$$\mathrm{P}(w) = \int \mathrm{P}(w|\alpha)\, \mathrm{P}(\alpha)\; d\alpha \tag{5.29}$$

Let us consider a neural network model with a Laplace prior on the weights such as $\mathrm{P}(w|\alpha) = \left(\frac{\alpha}{2}\right)^2 \exp(-\alpha\,|w|)$. As α is a *scale parameter*, we will use the improper, *non-informative prior* $\mathrm{P}(\alpha) = 1/\alpha$ (cf. section 5.10. Integrating (5.29), we obtain:

$$\mathrm{P}(w) = \int_0^{+\infty} \left(\frac{\alpha}{2}\right)^{\widehat{P}} \exp\left(-\alpha\left|w\right|\right) \frac{1}{\alpha}\, d\alpha = \frac{\left(\widehat{P}-1\right)!}{2^{\widehat{P}}} R(w)^{\widehat{P}} \tag{5.30}$$

where $R(w)$ is the corresponding regularisation functional, i.e. $\sum_j |w_j|$ in that case, and $\widehat{P}$ the number of regularised parameters as in section 5.6.

For a Gaussian prior, the derivation is slightly different, but we still get $\mathrm{P}(w) \propto R(w)^{\widehat{P}}$.

The likelihood $\mathrm{P}(\mathcal{D}|w)$ is known when the noise level σ^2 is known. If it is not so, it can be obtained in the same way. With a Gaussian assumption on the noise:

$$\mathrm{P}(\mathcal{D}|w) = \int_0^{+\infty} \left(\frac{\beta}{\pi}\right)^{\frac{N}{2}} \exp\left(-\beta N\, S(w)\right) \frac{1}{\beta}\, d\beta = \frac{\Gamma\left(\frac{N}{2}\right)}{\pi^{-\frac{N}{2}}} \left(N\, S(w)\right)^{-\frac{N}{2}} \tag{5.31}$$

As noticed in section 5.4, the full calculation of the posterior weight distribution $\mathrm{P}(w|\mathcal{D})$ is intractable. However, from (5.7), we see that $\mathrm{P}(w|\mathcal{D}) \propto \mathrm{P}(\mathcal{D}|w)\,\mathrm{P}(w)$, so we can consider maximising the posterior. The search for this maximum gives its name to the method: Maximum A Posteriori. We do estimate the model parameters by:

$$\widehat{w}_{MAP} = \arg\max_w \mathrm{P}(w|\mathcal{D}) \tag{5.32}$$

Let us consider the minus-logarithm of the posterior, that we infer from (5.30) and (5.31):

$$-\ln \mathrm{P}(w|\mathcal{D}) = \frac{N}{2} \ln S(w) + \widehat{P} R(w) + \mathrm{cte} \tag{5.33}$$

where the constant contains a sum of terms independent of w. Maximising the posterior with respect to w is equivalent to minimising (5.33). Derivating this expression, we get:

$$\nabla\left(-\ln \mathrm{P}(w|\mathcal{D})\right) = \frac{N}{2S(w)} \nabla S(w) + \frac{\widehat{P}}{R(w)} \nabla R(w) \tag{5.34}$$

Considering the MAP parameter estimate $\widehat{w}$ where this derivative is zero results in the following expression:

$$\nabla S(\widehat{w}) + \frac{2\widehat{P}\, S(\widehat{w})}{N\, R(\widehat{w})} \nabla R(\widehat{w}) = 0 \tag{5.35}$$

Note that (5.35) is the same as the gradient of the regularised cost (2.21) where we have set the regularisation parameter to:

$$\widehat{\xi} = \frac{2\widehat{P}\, S(\widehat{w})}{N\, R(\widehat{w})} \tag{5.36}$$

Once again, the Bayesian framework provides us with an update rule for the regularisation parameter. The MAP learning can thus be performed by minimising the regularised cost with a continuous update of the regularisation parameter given by (5.36).

5.10 Non-informative prior

The use of a *non-informative prior* is crucial to reflect our lack of knowledge about a given quantity. It is understood that the prior should serve the purpose of biasing the solution with some of the *a priori* information we have on it. However, we wouldn't want this prior to introduce some information we do *not* have on the problem!

The intuitive choice for a non-informative prior on α would be to chose a uniform distribution on a given range $[a;b]$. Indeed, we have no reason to privilege one value of parameter α more than the other. Contrary to intuitive belief, depending on the type of parameter, the uniform prior *does* introduce, against our will, some (unwanted) information in the problem.

This is clear if we consider what happens in a change of variable. Indeed, if the parameter is a *scale parameter*, we have no information on the parameter, but identically, we have no information on its inverse. Consider the case of the hyper-parameter on the weight distribution above. Whether we write the distribution $\exp\left(-\alpha R\left(w\right)\right)$ or $\exp\left(-\frac{R(w)}{\beta}\right)$ should not influence the result.

Let us now take an arbitrary distribution $\mathrm{P}\left(X|\alpha\right)$, that we wish to integrate over our parameter on the range $[a;b]$:

$$\mathrm{P}\left(X\right) = \int_a^b \mathrm{P}\left(X|\alpha\right)\mathrm{P}\left(\alpha\right)\, d\alpha \tag{5.37}$$

With a uniform prior, we simply get $\mathrm{P}\left(X|\left[a;b\right]\right)$. If $\mathrm{P}\left(X|\alpha\right)$ is in turn uniform with value μ, this amounts to $\mathrm{P}\left(X\right) = \mu\left(b-a\right)$.

α being a scale parameter, it is equivalent to write
$ProbX = int_{1/b}^{1/a}\mathrm{P}\left(X|1/\beta\right)\mathrm{P}\left(\beta\right)\, d\beta$. Using the same uniform prior, according to the above, and performing the change of variable $\alpha = 1/\beta$, we obtain:

$$\mathrm{P}\left(X\right) = \int_a^b \mathrm{P}\left(X|\alpha\right)\frac{1}{\alpha^2}\, d\alpha \tag{5.38}$$

It is clear that (5.38) and (5.37) will give different results. Using the same uniform distribution, the result will be $\mathrm{P}\left(X\right) = \frac{\mu}{3}\left(\frac{b^3-a^3}{a^3b^3}\right)$.

In order for the $1/x$ change of variable to have no effect on the use of the prior, we see that we need to have $\alpha^2\mathrm{P}\left(a\right) = \mathrm{P}\left(1/\alpha\right)$. The obvious solution in that case is:

$$\mathrm{P}\left(\alpha\right) = \frac{1}{\alpha} \tag{5.39}$$

This justifies the use of the non-informative prior in section 5.9, for all *scale* parameters.

Note that there is no prior that will be invariant under *any* variable change, so the type of parameter (i.e. type of change invariance) we desire is of prime importance.

As noted above, this prior is *improper*, as it does not integrate to one on $[0;+\infty[$ (which contradict our definition of a probability, definition A.6). On any more restricted interval $[a;b]$, it is possible to normalize. Integration can e.g. be performed on $[1/a;a]$, with $a \rightarrow +\infty$.

5.11 Criticism of the MAP framework

The main criticism about the MAP framework is that it does not estimate what really matters. Indeed, finding the maximum a posteriori parameters according to (5.32) corresponds to getting the mode $\widehat{w}$ of distribution $\mathrm{P}(w|\mathcal{D})$.

Typically, the estimate in x will then be given by using the model with $\widehat{w}$:

$$\widehat{y} = \int f_w(x)\,\mathrm{P}(w|\mathcal{D})\; dw \approx f_{\widehat{w}}(x) \tag{5.40}$$

However, this contradicts the real Bayesian method, in which we would obtain $\widehat{y}$ by integrating over the posterior weight distribution. The approximation might be correct if the resulting weight $\widehat{w}$ indeed represents the posterior distribution well. (5.40) is in spirit very close to the evidence approximation (5.19). Indeed, it corresponds to collapsing the integral over the weights in $\widehat{w}$.

Accordingly, this framework functions in the opposite way as the evidence framework. In the Evidence framework, we optimise the hyper-parameter value in order to integrate over the parameters using an approximate posterior distribution. In the MAP method, on the contrary, one integrates over the hyper-parameters, then optimises the weight (i.e. takes the mode instead of the expectation).

It is a criticism of the MAP method to notice that when faced to a choice of integrating over some parameters and optimising others, it is advisable, for the sake of accuracy, to integrate over as many parameters as possible. Typically, there are a large number of parameters, and possibly a handful of hyper-parameters. In the case of neural networks, for example, one hyper-parameter typically handles the weight of one layer, or of one input. The number of hyper-parameters will then be low compared to the number of parameters, and it sound intuitively reasonable to optimise this handful of parameters and integrate over the weights, rather than the opposite.

Finally, it has been argued that in the case where a large number of parameters are ill-determined, the MAP method leads to over-regularisation, and fails to produce sensible inference (cf. the COMMENTS section). However, we are not aware of any convincing example evidencing this behaviour.

5.12 Choice of a Bayesian framework

The arguments for or against each framework are of course interesting, but they are not necessarily relevant to the practitioner. Indeed, the evidence framework might rely on a maximum likelihood estimation of the hyper-parameter, and rely on some possibly wild assumptions, and the MAP method could very well fail to grasp where the actual mass of the posterior is.

On a more philosophical basis, let us quote an extract from Vapnik (1995):

> The only (but significant) shortcoming of the Bayesian approach is that it is restricted to the case where the set of functions of the learning machine coincides with the set of problems that the machine has to solve. [...] For example it cannot be applied to the problem of approximation of

the regression function by polynomials if the regression function is not polynomial, since the *a priori* probability $\mathrm{P}(\alpha)$ for any function from the admissible set of polynomials to be the regression is equal to zero.

On a practical basis though, many applications have shown that the evidence framework is susceptible to provide good results. The proof of that is the victory of D. MacKay in a time-series prediction contest, as well as the celebrated application of the evidence in Thodberg (1996).

The MAP framework on the other hand, appears to us to have some theoretical advantage. It was also used successful in some real life application, such as Williams (1995).

It is not our purpose to judge one method or the other. Even if we did on a theoretical basis, this would not necessarily have consequences on the practical application to neural networks. However, it should be noted that one strength of the Bayesian method(s) is that it allows to express the assumption on a problem very clearly.

Perhaps the main asset of Bayesian methods is the availability of a continuous update rule. This allows the design of a learning procedure that requires only one optimisation. It has been reported that the continuous update rules, (5.26) and (5.36), pose some convergence problems. However, it is our experience that when carefully applied they prove very efficient. The features to watch are the different weight categories (typically input and output weights will be two different regularisation classes), and the count of the number $\widehat{P}$ of regularised parameters, especially when pruning arises.

We use the Bayesian setting of the regularisation parameter to obtain some of the results in the two last chapters. In chapter 6, we carry out full Bayesian calculation on a simple example, using both frameworks.

COMMENTS

5.1 The philosophical as well as practical differences between the Bayesian and the frequentist (or Fisherian) approaches to statistics is well illustrated in e.g. (Efron, 1986), and the comments and references therein.

5.2 For a rather general (if partial) presentation of Bayesian reasoning, see Jaynes (1985) and enclosed references. Most textbooks on statistics and probability also present the Bayes inference rule.

5.7 The use of the evidence for neural networks was pioneered by MacKay (1992a,b). See also Thodberg (1993).
Neal (1992) advocates the use of a hybrid Monte Carlo method to perform the integration over the parameters, rather than the use of the evidence procedure. The computational cost of the method is rather high, but the results are reported very close to the Bayesian ideal.

5.8 On criticism of the evidence procedure, few sources can match Wolpert (1995, 1993). Many results including the bounds on the evidence error and sufficiency conditions for the evidence to work can be found there.

5.9 An application of the MAP framework using a Laplace prior weight distribution is given by Williams (1995). We have used the same prior, and demonstrated its effect on a toy problem in chapter 6.

5.10 It is surprisingly hard to find a source that provides a convincing treatment of the non-informative prior. In the context of inverse problems, let us mention (Scales and Smith, 1994, page 49).

5.11 Several authors have argued against the use of the MAP method. A simple example of incorrect inference produced by MAP is presented by MacKay (1993), although it is not clear whether it is really relevant. Thodberg (1996) also briefly argues that one should "integrate over as many parameters as possible".

5.12 The quote from V. Vapnik is in section 4.11.2 of (Vapnik, 1995).

References

Efron, B. (1986). Why isn't everyone a bayesian? *The American Statistician*, 40(1):1–11. with comments.

Jaynes, E. T. (1985). Bayesian methods: general background. In Justice, J., editor, *Maximum Entropy and Bayesian Methods in Applied Statistics*, pages 1–25. Cambridge University Press.

MacKay, D. (1992a). Bayesian interpolation. *Neural Computation*, 4:415–447.

MacKay, D. (1992b). A practical bayesian framework for backprop networks. *Neural Computation*, 4:448–472.

MacKay, D. (1993). Hyperparameters: Optimize or integrate out? In Heidbreder, G., editor, *Maximum entropy and Bayesian Methods*. Kluwer, Dordrecht.

Neal, R. M. (1992). Bayesian training of backpropagation network by the hybrid monte carlo method. Technical Report CRG-TR-92-1, Connectionist Research Group, Department of Computer Science, University of Toronto.

Scales, J. A. and Smith, M. L. (1994). *Introductory Geophysical Inverse Theory*. Samizdat Press, available via FTP form hilbert.mines.colorado.edu.

Thodberg, H. H. (1993). Ace of Bayes: application of neural network with pruning. Technical Report 1132-E, Danish meat research institute, Roskilde, Danmark.

Thodberg, H. H. (1996). A review of bayesian neural networks with an application to near infrared spectroscopy. *IEEE Transactions on Neural Networks*, 7(1):56–72.

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.

Wolpert, D. (1995). What Bayes has to say about the evidence procedure. In Heidbreder, G., editor, *1993 Maximum Entropy and Bayesian Methods Conference*. Kluwer.

Wolpert, D. H. (1993). On the use of evidence in neural networks. In Hanson, S. J., Cowan, J. D., and Giles, C. L., editors, *Advances in Neural Information Processing Systems*, number 5 in NIPS, pages 539–546. Morgan Kaufmann.

6

The effect of a pruning prior

6.1 Introduction

The aim of this chapter is two-fold:

- to provide an example of the application of statistical learning on a very simple problem. This very simple case allows us to gain insight into the way different learning schemes address the problem and work towards the same goal.
- to display the effect of an regularisation alternative to the usual "weight-decay" on this problem, and on real life time-series modelling and system identification.

The first part is developed in sections 6.2 to 6.13: this part is quite long as we have tried to go into the details of the derivations. The second part is the body of sections 6.14 to 6.19.

6.2 The parameter location problem

We address the problem of estimating the value of a quantity based on noisy measurements of it. The target value will be noted $\widetilde{w}$ and called the *teacher parameter*, or *teacher weight*, by analogy with neural network parameters.

A number of noisy examples are available, of the form:

$$y^{(k)} = \widetilde{w} + \varepsilon^{(k)} \qquad k = 1, \dots, N \tag{6.1}$$

The noise is taken with 0 mean and known variance σ^2. The training set $\mathcal{D}$ is here the set of all the available examples[1]: $\left\{y^{(k)}, k = 1 \dots N\right\}$.

The *real system* is thus $y = \widetilde{w} + \epsilon$, and we try to *model* it by finding an estimate $\widehat{w}$ of the teacher weight. This estimate will be called the *student parameter*.

[1]In this problem, there is no input-output relationship.

This simple problem should not lead the reader to think that these derivations are trivial. On the contrary, we will demonstrate that even in such a simple case, some important differences arise between different learning methods. Indeed, the results obtained are far from being obvious.

6.3 Maximum likelihood solution

The maximum likelihood solution depends on the shape of the noise ϵ. We will address the two simple cases of Gaussian and Laplacian noise. In the case of Gaussian noise, the likelihood of the *student parameter* w associated with the training set $\mathcal{D}$ is given by:

$$\mathrm{P}\left(\mathcal{D}|w\right) = \prod_{k=1}^{N} \frac{\exp\left[-\frac{\left(y^{(k)}-w\right)^2}{2\sigma^2}\right]}{\sqrt{2\pi\sigma^2}} \tag{6.2}$$

Maximising (6.2) with respect to w leads to the maximum likelihood (ML) estimator. This estimator is, as we know, the empirical mean:

$$w_{\mathrm{ML}} = \frac{1}{N}\sum_{k=1}^{N} y^{(k)} = \overline{y} \tag{6.3}$$

In the case of Laplacian noise, the likelihood of parameter w associated with the training set $\mathcal{D}$ is given by:

$$\mathrm{P}\left(\mathcal{D}|w\right) = \prod_{k=1}^{N} \frac{\exp\left[-\frac{\left|y^{(k)}-w\right|}{\mu}\right]}{2\mu} \tag{6.4}$$

where μ is the mean absolute value, which relates to σ^2 as $\sigma^2 = 2\mu^2$. Maximising (6.4) with respect to w leads to the maximum likelihood estimator in the case of Laplacian noise, the median:

$$w_{\mathrm{ML2}} = \mathrm{med}\left(y^{(k)}\right) \tag{6.5}$$

In the following, we will make the assumption that the noise ϵ on the teacher parameter is Gaussian. This does not mean that it actually is, only that we believe it to be so.

6.4 Regularised solution

Let us now apply a least mean square method to our simple problem. The quadratic cost is given by the average square distance between the student parameter and the data:

$$S\left(w\right) = \frac{1}{N}\sum_{k=1}^{N}\left(y^{(k)} - w\right)^2 = \left(\overline{y} - w\right)^2 + \sigma_y^2 \tag{6.6}$$

where σ_y^2 is the empirical variance, i.e. $\overline{\left(y^{(k)} - \overline{y}\right)^2}$. The value of w that minimises (6.6) is the empirical mean, i.e. the maximum likelihood estimator. The quality of this estimation is assessed by the expected risk, or generalisation error:

$$G\left(w\right) = \int_{-\infty}^{+\infty} \left(\widetilde{w} + \epsilon - w\right)^2 p\left(\epsilon\right)\, d\epsilon = \left(\widetilde{w} - w\right)^2 + \sigma^2 \tag{6.7}$$

One can see in (6.7) a very simple version of the bias-variance decomposition. σ^2 is the irreducible noise level, $\left(\widetilde{w} - w\right)^2$ is the variance of the estimate (due to the observational noise). The bias does not exist here as there is no modelling error: our one-parameter model is, by construction, perfect for the task at hand.

The regularised cost is obtained by adding a weighted constraint to the quadratic cost:

$$C\left(w\right) = S\left(w\right) + \lambda R\left(w\right) \tag{6.8}$$

where $R(w)$ can take different forms. The most common one is a quadratic term, which relates to ridge regression in the case of linear models and weight decay in the case of neural networks. $R(w)$ is then the L_2 norm of the parameter vector, i.e. $R(w) = w^2$. As will be seen later, this is equivalent to having a *Gaussian prior* on the weights. Beside the quadratic regularisation, we will here consider the case of another type of regularisation, introduced in section 2.16. In that case, recall that $R(w)$ is the L_1 norm of the parameter vector, $R(w) = |w|$. This relates to the use of a *Laplace prior* on the weights and will sometimes be referred to as *Laplace regularisation.*

Minimising (6.8) with respect to w in the case of the weight decay leads to:

$$w_{\mathrm{G}} = \frac{1}{1+\lambda} w_{\mathrm{ML}} \tag{6.9}$$

The index G refers to the use of a Gaussian prior on weights (weight-decay).

As we can see, the solution is proportional to the maximum likelihood estimator. When this estimator is small, it is likely that the teacher parameter itself is small. The regularised estimate is then slightly biased towards 0. On the other hand, for high values of the teacher parameter, the maximum likelihood will be high, and (6.9) can then be highly biased.

In the case of the Laplace regularisation, minimising (6.8) with respect to w leads to the regularised solution:

$$w_{\mathrm{L}} = \begin{cases} \left(1 - \frac{\lambda}{2|w_{\mathrm{ML}}|}\right) w_{\mathrm{ML}} & |w_{\mathrm{ML}}| > \lambda/2 \\ 0 & |w_{\mathrm{ML}}| \leq \lambda/2 \end{cases} \tag{6.10}$$

The index L now refers to the use of Laplace prior.

The behaviour of this estimate is entirely different. For small values of the maximum likelihood, it leads to pruning of the parameter. For higher values, the regularisation introduces a constant bias. This estimator is thus never highly biased, contrary to (6.9).

Predictably, both estimates depend on the regularisation parameter λ. This parameter has to be set using one of the methods discussed in chapters 3 and 5. We will now proceed with the presentation of the generalisation method and we will later present the derivation in the case of the Bayesian frameworks.

6.5 Generalisation method

The idea behind the *generalisation method* is to average the generalisation error over all possible training sets, and choose the value of the regularisation parameter that leads to a minimum of this average. Let us first consider the unregularised case. We recall (6.6) and (6.7), plug the maximum likelihood solution estimator $\overline{y}$ and integrate over its distribution:

$$S\left(w_{ML}\right)=\sigma_y^2 \qquad \Longrightarrow \qquad \left\langle S\left(w_{ML}\right)\right\rangle=\left(1-\frac{1}{N}\right)\sigma^2 \tag{6.11}$$

$$G\left(w_{ML}\right)=\left(\widetilde{w}-\overline{y}\right)^2 \qquad \Longrightarrow \qquad \left\langle G\left(w_{ML}\right)\right\rangle=\left(1+\frac{1}{N}\right)\sigma^2 \tag{6.12}$$

as $\overline{y}$ is normally distributed around $\widetilde{w}$ with variance σ^2/N, and $\langle\cdot\rangle$ is the average of a quantity over all possible training sets of size N.

In the case of weight decay, we seek the expression of w_{GG}, the solution of the **G**eneralisation method with a **G**aussian prior. Recall that $w_{\mathrm{ML}} \sim \mathcal{N}\left(\widetilde{w}, \sigma^2/N\right)$. Hence we have[2] $w_{\mathrm{G}} \sim \mathcal{N}\left(\frac{1}{1+\lambda}\widetilde{w}, \frac{1}{N(1+\lambda)^2}\sigma^2\right)$. According to (6.7), the average generalisation error is $\left\langle G\left(w_{\mathrm{G}}\right)\right\rangle=\left\langle\left(\widetilde{w}-w_{\mathrm{GG}}\right)^2\right\rangle+\sigma^2$, which leads to:

$$\left\langle G\left(w_{\mathrm{G}}\right)\right\rangle=\left(\frac{\lambda\widetilde{w}}{1+\lambda}\right)^2+\left(1+\frac{1}{N(1+\lambda)^2}\right)\sigma^2 \tag{6.13}$$

The optimal value of λ corresponds the the minimum of (6.13). It is reached for:

$$\lambda=\frac{\sigma^2}{N\widetilde{w}^2} \tag{6.14}$$

Remember that in our simple problem, σ is known and does not have to be estimated from the data. Inserting this expression in (6.9) leads to the "optimal" estimator with respect to the average generalisation error:

$$w_{\mathrm{GG}}=\frac{\widetilde{w}^2}{\widetilde{w}^2+\sigma^2/N}w_{\mathrm{ML}} \tag{6.15}$$

The problem with this estimator is that it involves the teacher parameter, $\widetilde{w}$ which by essence is unknown. A natural solution is to use an estimator to fill in the role. Replacing $\widetilde{w}$ with w_{ML} leads to the first generalisation method estimator for a Gaussian prior:

$$w_{\mathrm{GG1}}=\frac{w_{\mathrm{ML}}^3}{w_{\mathrm{ML}}^2+\sigma^2/N} \tag{6.16}$$

Another possibility is to insert the estimator itself in place of $\widetilde{w}$, leading to a self-consistent estimator:

$$w_{\mathrm{GG2}}=\frac{w_{\mathrm{GG2}}^2}{w_{\mathrm{GG2}}^2+\sigma^2/N}w_{\mathrm{ML}} \tag{6.17}$$

[2] we use the fact that if x is a random variable of distribution $\mathcal{N}\left(m,\sigma^2\right)$, then $a.x+b$ is a random variable of distribution $\mathcal{N}\left(a.m+b, a^2\sigma^2\right)$

In this case, it is possible to calculate the exact expression:

$$w_{\mathrm{GG2}} = \begin{cases} \left(1+\sqrt{1-\frac{4\sigma^2}{N w_{\mathrm{ML}}^2}}\right)\frac{w_{\mathrm{ML}}}{2} & w_{\mathrm{ML}}^2 > 4\sigma^2/N \\ 0 & w_{\mathrm{ML}}^2 \leq 4\sigma^2/N \end{cases} \tag{6.18}$$

This expression illustrates the way the weight decay constrains the solution: It "pulls" parameters towards 0. When the estimate is lower than a given noise-related level, the learning procedure, confident with this constraint, concludes that the parameter is indeed 0 and that the value of the maximum likelihood estimator reflects only noise. Because of this bold attitude, w_{GG2} was dubbed the *brave generalisation student*.

Let us now study the case of the Laplace regularisation. The expression of (6.10) does not allow to obtain $\langle G\rangle$ as easily as before. Integrating over w_{ML} leads to:

$$\begin{aligned} \langle G\left(w_L\right)\rangle &= \int_{-\infty}^{-\frac{\lambda}{2}} \left(\widetilde{w} - w_{\mathrm{ML}} - \frac{\lambda}{2}\right)^2 P(w_{\mathrm{ML}})\, dw_{\mathrm{ML}} + \int_{-\frac{\lambda}{2}}^{\frac{\lambda}{2}} \widetilde{w}^2\, P(w_{\mathrm{ML}})\, dw_{\mathrm{ML}} \\ &+ \int_{\frac{\lambda}{2}}^{+\infty} \left(\widetilde{w} - w_{\mathrm{ML}} + \frac{\lambda}{2}\right)^2 P(w_{\mathrm{ML}})\, dw_{\mathrm{ML}} + \sigma^2 \end{aligned} \tag{6.19}$$

After some simple but tedious algebra, we obtain the somewhat complicated expression:

$$\begin{aligned} \langle G\left(w_{\mathrm{L}}\right)\rangle &= \left(1+\frac{1}{N}\right)\sigma^2 + \frac{\lambda^2}{4} \qquad (6.20) \\ &+ \frac{\sigma^2}{N}\left[\left(AB-1\right)\left(\Phi(B)-\Phi(A)\right) + \frac{1}{\sqrt{2\pi}}\left(A\mathrm{e}^{-\frac{B^2}{2}} - B\mathrm{e}^{-\frac{A^2}{2}}\right)\right] \end{aligned}$$

where $A = \frac{\sigma}{\sqrt{N}}\left(\widetilde{w} - \frac{\lambda}{2}\right)$ and $B = \frac{\sigma}{\sqrt{N}}\left(\widetilde{w} + \frac{\lambda}{2}\right)$. Φ is the area under the normal curve, i.e.:

$$\Phi\left(x\right) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{x} \mathrm{e}^{-\frac{t^2}{2}}\, dt \tag{6.21}$$

The optimal λ has a value that minimises (6.20). There is no analytical solution to such a minimisation problem, but it is easy to obtain a numerical solution. Minimising (6.20) with respect to λ is a one dimensional optimisation problem, which can be solved by any standard technique such as golden search and/or quadratic minimisation. Furthermore, the shape of the average generalisation error is simple in this case, so that we are guaranteed to find the global minimum. Plugging this empirical value of λ into (6.10) gives w_{GL}, the generalisation method solution for a Laplace prior.

Another problem is that, as we noticed earlier for the weight decay, the above expression involves the teacher parameter $\widetilde{w}$. Thus, the same approximation can be performed, replacing $\widetilde{w}$ with its maximum likelihood estimator w_{ML}, and minimising the resulting expression. The estimator obtained is then equivalent in spirit to w_{GG1}, and we will denote it by w_{GL1}. The self consistent estimator cannot be computed directly. It can be approximated by iterating the approximation of $\widetilde{w}$ in (6.20). w_{GL1} can be used as an estimate of $\widetilde{w}$ to obtain a new numerical estimator that we denote w_{GL2}. Iterating this approximation-minimisation process, we obtain w_{GL3}, w_{GL4}, etc. The equivalent of the self-consistent estimator would then be $w_{\mathrm{GL}\infty}$.

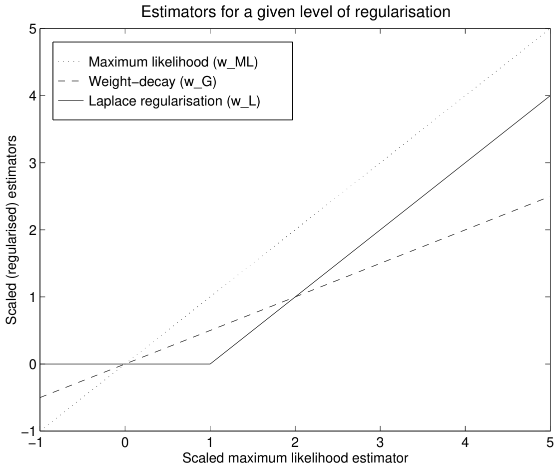


Figure 6.1: Comparison of two regularisation constraints for a given level of regularisation. The dashed line corresponds to the weight decay, equivalent to a Gaussian prior on the parameter (L_2 norm). The solid line corresponds to Laplace regularisation. The dotted line is the maximum likelihood estimator. All quantities are scaled by σ^2/N.

6.6 Results for generalisation method

Let us first compare the effect of the two regularisation methods for a given regularisation level.

Figure 6.1 illustrates the remarks made in section 6.4. In this figure, the regularisation level has been set to $\lambda = 1$. The solid line is the estimator w_{GG} of (6.9), obtained using a weight decay. It is clear that the bias between the regularised solution and the maximum likelihood increases with the value of the maximum likelihood. This is because the regularising penalty term grows quadratically with the size of the parameter. The dashed line corresponds to the estimator obtained using Laplace regularisation. Small parameters are forced to 0 and elsewhere there is a constant bias. Indeed, the regularising function is only linear in this case. The part where the estimator takes a value of zero will be called the *pruning section*.

Let us now analyse the case where the regularisation level is set using the generalisation method. Figure 6.2 displays three estimators obtained with this method, together with the reference maximum likelihood. The dash-dotted line is w_{GG1}, the estimator corresponding to weight decay regularisation and obtained by replacing $\widetilde{w}$ with w_{ML}. For small values of the maximum likelihood, the parameter is softly constrained to be close to 0. Otherwise, there is a slowly decreasing bias.

The brave generalisation student, in dashed line, has a very broad pruning section. Otherwise the bias is also slowly decreasing. This example shows that even though the weight decay, by itself, does not lead to pruning (cf. equation 6.9), the optimisation of the regularisation parameter in the light of the data can lead to such a pruning effect.

On the other hand, the Laplace prior prunes no matter what. The estimator resulting

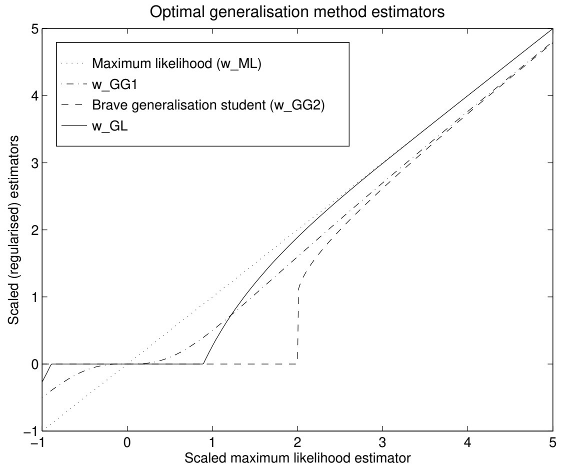


Figure 6.2: Comparison of estimators for two different types of regularisation, with the generalisation method. The solid line is for Laplace regularisation. The dotted and dash-dotted lines are for weight decay. All quantities are scaled by σ^2/N.

from the generalisation method, in solid line in figure 6.2, shows that the pruning section of w_{GL} is somewhat smaller than that of the brave generalisation student. An important improvement over the two previous estimators is that the bias is reduced extremely fast and is almost non-existent for moderate size parameters. When iterating the approximation as explained in section 6.5, the width of the pruning section increases, but the fast bias reduction is maintained.

6.7 Bayesian analysis

We will now analyse this simple problem with a Bayesian point of view. The two regularisation constraints studied above have their counterparts in prior over the parameter:

- a Gaussian prior $\mathrm{P}(w|\alpha) = \sqrt{\frac{\alpha}{2\pi}} \exp\left(-\frac{\alpha w^2}{2}\right)$ for the weight decay,
- or a Laplace prior $\mathrm{P}(w|\alpha) = \frac{\alpha}{2} \exp(-\alpha|w|)$.

With our Gaussian hypothesis on noise, the likelihood is:

$$\mathrm{P}(D|w) = \left(2\pi\sigma^2\right)^{-N/2} \exp\left(-\frac{N}{2\sigma^2} S(w)\right) \tag{6.22}$$

As noted above, the likelihood is maximised when the parameter is set to the empirical average of the examples.

6.8 Evidence and evidence framework

The posterior distribution is given by applying Bayes' rule:

$$\mathrm{P}\left(w|D,\alpha\right)=\frac{\mathrm{P}\left(D|w,\alpha\right)\mathrm{P}\left(w|\alpha\right)}{\mathrm{P}\left(D|\alpha\right)} \tag{6.23}$$

where the likelihood is independent of α: $\mathrm{P}\left(D|w,\alpha\right)=\mathrm{P}\left(D|w\right)$. This expression involves α, and is the equivalent to the regularised error displayed in (6.8). The maximum of (6.23) depends on the value of the hyper-parameter α. In order to optimise this hyper-parameter, we will maximise its likelihood in the light of the data $\mathrm{P}\left(D|\alpha\right)$. This likelihood is obtained by integrating over the parameter w, and is called the *evidence*:

$$\mathrm{P}\left(D|\alpha\right)=\int_{-\infty}^{+\infty}\mathrm{P}\left(D|w,\alpha\right)\mathrm{P}\left(w|\alpha\right)\,d\alpha \tag{6.24}$$

The value of (6.24) will of course depend on the prior on the parameters.

With a weight decay, the integral is Gaussian, and after some careful algebra, we derive:

$$-2\ln\mathrm{P}\left(D|\alpha\right)=-\ln\left(\frac{\alpha}{2\pi}\right)+\ln\left(\frac{N+\alpha\sigma^2}{2\sigma^2}\right)-\frac{\left(Nw_{\mathrm{ML}}\right)^2}{2\sigma^2\left(N+\alpha\sigma^2\right)}+c \tag{6.25}$$

where c is a constant with respect to α.

The evidence reaches its maximum for the following value of α:

$$\alpha_{\mathrm{ML}}=\begin{cases}\frac{1}{w_{\mathrm{ML}}^2-\sigma^2/N} & w_{\mathrm{ML}}^2>\frac{\sigma^2}{N}\\ +\infty & w_{\mathrm{ML}}^2\leq\frac{\sigma^2}{N}\end{cases} \tag{6.26}$$

Going back to (6.23), we write the expression for the posterior distribution:

$$\mathrm{P}\left(w|D,\alpha\right)\propto\sqrt{\frac{\alpha}{2\pi}}\exp\left(-\frac{\alpha w^2}{2}\right)\left(2\pi\sigma^2\right)^{-N/2}\exp\left(-\frac{N}{2\sigma^2}S(w)\right) \tag{6.27}$$

The posterior distribution is maximised for:

$$w_{\mathrm{MPG}}\left(\alpha\right)=\frac{N}{N+\alpha\sigma^2}w_{\mathrm{ML}} \tag{6.28}$$

We notice that this solution is, as expected, the same as (6.9) with the relation $\lambda=\alpha\sigma^2/N$. λ is equivalent α, rescaled by the variance of the maximum likelihood estimator.

We combine the solution (6.28) with the most likely value of α in (6.26) to obtain the final maximum posterior estimator for a Gaussian prior:

$$w_{\mathrm{MPG}}=\begin{cases}\left(1-\frac{\sigma^2}{Nw_{\mathrm{ML}}^2}\right)w_{\mathrm{ML}} & w_{\mathrm{ML}}^2>\frac{\sigma^2}{N}\\ 0 & w_{\mathrm{ML}}^2\leq\frac{\sigma^2}{N}\end{cases} \tag{6.29}$$

It is interesting to note that pruning arises here explicitly from the data. When optimising the hyper-parameter in the light of the data, it is sometimes set to $+\infty$. This setting forces the most probable estimator to take a value of zero. This is different from the Laplace prior. As noticed in section 6.3, this prior always produces a pruning effect, regardless of the value of the hyper-parameter.

Let us now switch to the case of the Laplace prior on the parameter. The calculation of the "evidence" is now less straightforward, because the integral (6.24) is not a Gaussian integral. Because of the simplicity of the parameter location problem, there are two possibilities:

1. Performing a Gaussian approximation of (6.24), and optimising it with respect to α: this is the traditional "evidence procedure".
2. Full calculation of (6.24), even if the resulting expression can only be optimised numerically: this is much closer to the Bayesian ideal, and we will call it the "evidence calculation".

The evidence calculation is not as easy as in the case of the Gaussian prior, due to the presence of the absolute value in the expression of the Laplace prior. After some algebra, we obtain the following expression:

$$\mathrm{P}\left(D|\alpha\right) = Z_p \frac{\alpha}{2}\left[\mathrm{e}^{\frac{A^2}{2}}\Phi\left(A\right) + \mathrm{e}^{\frac{B^2}{2}}\Phi\left(-B\right)\right] \tag{6.30}$$

where Z_p is a constant with respect to α, $A = \frac{\sqrt{N}}{\sigma}\left(w_{\mathrm{ML}} - \frac{\alpha\sigma^2}{N}\right)$ and $B = \frac{\sqrt{N}}{\sigma}\left(w_{\mathrm{ML}} + \frac{\alpha\sigma^2}{N}\right)$. Predictably, there is no analytical maximum to expression (6.30). Numerical optimisation leads to the most likely value of α, α_{ML} that is used to infer the most probable value of the parameter through (6.23).

The evidence procedure involves a Gaussian approximation to calculate the evidence. This comes down to making a quadratic approximation of the (regularised) cost function around its minimum. With a Laplace prior on the parameter, the posterior distribution is:

$$\mathrm{P}\left(w|D,\alpha\right) = \frac{\alpha\exp\left(-\alpha|w|\right)}{2\mathrm{P}\left(D|\alpha\right)}\prod_{k=1}^{N}\frac{1}{2\pi\sigma^2}\exp\left(-\frac{\left(y^{(k)}-w\right)^2}{2\sigma^2}\right) \tag{6.31}$$

This can also be written as:

$$\mathrm{P}\left(w|D,\alpha\right) = \frac{\left(2\pi\sigma^2\right)^{-N/2}}{\mathrm{P}\left(D|\alpha\right)}\frac{\alpha}{2}\exp\left(-\alpha|w| - \frac{N}{2\sigma^2}S(w)\right) \tag{6.32}$$

One can already feel in (6.32) the relationship between α and λ in (6.8). The posterior distribution is maximised for:

$$w_{\mathrm{MPL}} = \begin{cases} \left(1 - \frac{\alpha\sigma^2}{N|w_{\mathrm{ML}}|}\right) w_{\mathrm{ML}} & |w_{\mathrm{ML}}| > \frac{\alpha\sigma^2}{N} \\ 0 & |w_{\mathrm{ML}}| \leq \frac{\alpha\sigma^2}{N} \end{cases} \tag{6.33}$$

It is now clear that this solution is the same as the minimum of the regularised cost in (6.10), with the following relationship between α and λ: $\lambda = \frac{2\alpha\sigma^2}{N}$. If we define

$M(w) = \alpha|w| + \frac{N}{2\sigma^2}S(w)$, we can write a second order Taylor expansion around w_{MP}. Noticing that w_{MP} is the minimum of M, we write:

$$M\left(w\right) = M\left(w_{\mathrm{MPL}}\right) + \frac{h}{2}(w - w_{\mathrm{MPL}})^2 \tag{6.34}$$

where h is the second derivative of M with respect to w. Using this approximation, the integral (6.24) becomes Gaussian, and the minus log-likelihood of the hyper-parameter becomes:

$$-\ln \mathrm{P}\left(D|\alpha\right) = -\ln\alpha - \frac{\alpha^2\sigma^2}{2N} + \alpha|w_{\mathrm{MP}}| + c \tag{6.35}$$

in the case where w_{MP} is non zero, where c is a constant with respect to α. Equation (6.35) is minimised for:

$$\alpha_{\mathrm{ML}} = \frac{|w_{\mathrm{ML}}|}{2\sigma^2/N}\left(1 - \sqrt{1 - \frac{4\sigma^2}{Nw_{\mathrm{ML}}^2}}\right) \qquad w_{\mathrm{ML}}^2 > \frac{4\sigma^2}{N} \tag{6.36}$$

which leads to the evidence procedure estimator:

$$w_{\mathrm{EVL}} = \begin{cases} \left(1 + \sqrt{1 - \frac{4\sigma^2}{Nw_{\mathrm{ML}}^2}}\right)\frac{w_{\mathrm{ML}}}{2} & w_{\mathrm{ML}}^2 > 4\sigma^2/N \\ 0 & w_{\mathrm{ML}}^2 \leq 4\sigma^2/N \end{cases} \tag{6.37}$$

This is the *brave generalisation student* that was derived with the generalisation method in the case of a Gaussian prior on the parameters.

6.9 MAP solution

Let us now perform the Bayesian inference from a different standpoint. Using Bayes' rule, we write the posterior as:

$$\mathrm{P}\left(w|D\right) = \frac{\mathrm{P}\left(D|w\right)\mathrm{P}\left(w\right)}{\mathrm{P}\left(D\right)} \tag{6.38}$$

In order to obtain $\mathrm{P}\left(w\right)$, we have to get rid of the influence of the hyper-parameter α by integrating over it:

$$\mathrm{P}\left(w\right) = \int_0^\infty \mathrm{P}\left(w|\alpha\right)\mathrm{P}\left(\alpha\right)\, d\alpha \tag{6.39}$$

The result of this integration depends of course on the prior on w. The prior on α, on the other hand, will be the standard, non-informative, improper prior $\mathrm{P}\left(\alpha\right) = 1/\alpha$. With a Laplace prior, the result is:

$$\mathrm{P}\left(w\right) = \int_0^\infty \frac{\alpha}{2}\exp\left(-\alpha|w|\right)\frac{1}{\alpha}\, d\alpha = \frac{1}{2}\left[-\frac{\mathrm{e}^{-\alpha|w|}}{|w|}\right]_0^\infty = \frac{1}{2|w|} \tag{6.40}$$

While the Gaussian prior leads to:

$$\mathrm{P}\left(w\right) = \int_0^\infty \sqrt{\frac{\alpha}{2\pi}}\exp\left(-\frac{\alpha w^2}{2}\right)\frac{1}{\alpha}\, d\alpha = \frac{2}{\sqrt{2\pi}}\int_0^\infty \mathrm{e}^{-\frac{w^2u^2}{2}}du = \frac{1}{|w|} \tag{6.41}$$

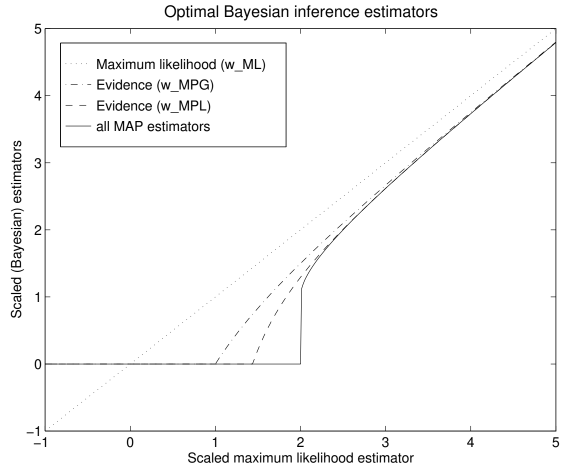


Figure 6.3: Comparison of estimators for two different priors on the parameter, with various Bayesian frameworks. The dashed and dash-dotted line are the estimators maximising the evidence for a Laplace and Gauss prior (respectively). The solid line is the MAP estimator, and also the *brave generalisation student*. All quantities are scaled.

We notice two important facts: both expressions are similar, and they correspond to a *non-informative prior* on the parameter!

Considering that they differ only by a multiplicative constant, they will lead to the same results, the MAP estimator for both Gaussian and Laplace prior. Invoking the negative log of (6.38), we now need to find the minimum of:

$$-\ln \mathrm{P}\,(w|D) = -\ln \mathrm{P}\,(D|w) + \ln|w| + cte = \frac{N}{2\sigma^2} S(w) + \ln|w| + c \tag{6.42}$$

where the constant c is independent of w.

The MAP solution is the minimum of (6.42). This minimisation must be handled cautiously, though. Indeed, the negative log probability is not bounded below: it is not defined in zero, where it goes to $-\infty$. This is caused by the use of the improper prior, which leads to a probability density that is not integrable. The MAP solution is given by another (local) minimum of (6.42): Differentiating (6.42) with respect to w leads to:

$$\frac{\partial}{\partial w}\left(-\ln \mathrm{P}\,(w|D)\right) = \frac{N}{\sigma^2}\left(w - w_{\mathrm{ML}}\right) + \frac{1}{w} \tag{6.43}$$

which leads to the following solution:

$$w_{\mathrm{MAP}} = \begin{cases} \left(1 + \sqrt{1 - \frac{4\sigma^2}{N w_{\mathrm{ML}}^2}}\right) \frac{w_{\mathrm{ML}}}{2} & w_{\mathrm{ML}}^2 > 4\sigma^2/N \\ 0 & w_{\mathrm{ML}}^2 \leq 4\sigma^2/N \end{cases} \tag{6.44}$$

This happens to be the *brave generalisation student* found above in the case of Gaussian noise and generalisation method. It is remarkable that this estimator is actually independent of the prior on the parameters.

In order to compare the Bayesian estimators in the same conditions as the generalisation method estimators, we plot the estimators on figure 6.3.

Prior	Maximum Likelihood	Generalisation Method		Bayes		
				Evidence		MAP
		approx.	self-cons.	Exact	Approx.	
Gauss	w_{ML}	w_{GG1}	BGS	w_{MPG}	w_{MPG}	BGS
Laplace	w_{ML}	w_{GL}	-	w_{MPL}	BGS	BGS

Table 6.1: Results on the location parameter problem. BGS denotes the *brave generalisation student* that was obtained in several conditions. All other estimators have the same syntax as in the text.

6.10 Overall results

The results of the Bayesian analysis of the location parameter problem are summarised in table 6.1.

The *brave generalisation student* is a very popular estimator: it has been derived in three different frameworks and appears for both priors.

All estimators present roughly the same shape. They display a pruning section of variable width for small parameters, and a slowly decreasing bias when the value of the maximum likelihood increases. This has a very simple explanation. The prior on the parameter gives the *student* some information saying that the teacher parameter has a tendency to be close to zero. When the maximum likelihood is "sufficiently close" to 0, the student tend to use this information and decide that the observed data could very well result from a teacher that is actually 0. The difference between the three estimators in figure 6.3 is their appreciation of what "sufficiently close" is. For that matter, the *brave generalisation student* displays a very brave behaviour, leading to the largest pruning section of all estimators.

6.11 Analysis

All the results above relate to the value of the maximum likelihood estimator. Let us now consider performance from another standpoint. We wish to evaluate the behaviour of the estimators as a function of the actual value of the teacher parameter. Indeed, a maximum likelihood value of $w_{\mathrm{ML}} = 1$ could easily come from a teacher parameter $\widetilde{w} = 1$, in which case all our regularised estimators would commit a small error. If the number of examples is small, the same maximum likelihood could be observed for $\widetilde{w} = 0$, in which case most estimators would perform quite well. It could also (though with extremely small probability) result from a teacher parameter $\widetilde{w} = 10^{10}$, and all estimators seen here would be completely misled.

We measure this effect by computing the *average excess error*, i.e. the squared difference between the estimator and the actual teacher parameter, averaged over all possible samples, i.e. over the distribution of the maximum likelihood estimator. All estimators are given as a function of w_{ML}. For unbiased Gaussian noise of variance σ^2, the average of the observation is $\mathcal{N}\left(\widetilde{w}, \frac{\sigma^2}{N}\right)$ so the average excess error for a

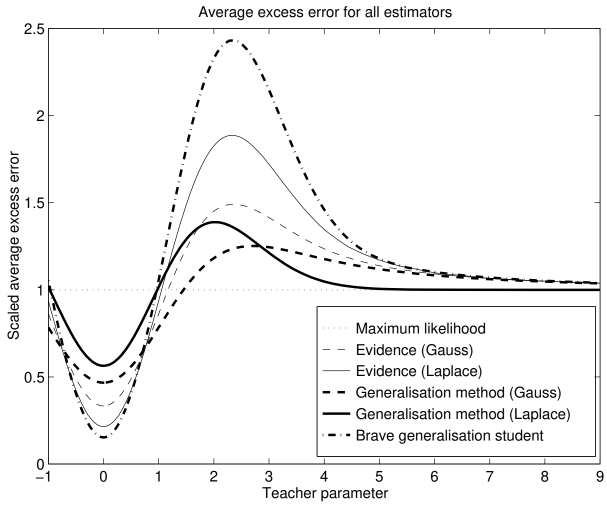


Figure 6.4: Average excess generalisation error computed as a function of the teacher parameter, for all the estimators reviewed above. The solid lines are for the Laplace prior, the dashed lines for the Gaussian prior. The bold lines are for the generalisation method, the standard ones for Bayesian inference. The dash-dotted line is the *brave generalisation student* which arises for both priors. The dotted line is the maximum likelihood estimator.

generic estimator w_{E} is:

$$\overline{\Delta}_E\left(\widetilde{w}\right) = \sqrt{\frac{N}{2\pi\sigma^2}} \int_{-\infty}^{+\infty} \left(\widetilde{w} - w_{\mathrm{E}}\right)^2 \mathrm{e}^{-\frac{N\left(w_{\mathrm{ML}}-\widetilde{w}\right)^2}{2\sigma^2}} \, dw_{\mathrm{ML}} \tag{6.45}$$

In the case of the maximum likelihood estimator, $\overline{\Delta}_{\mathrm{ML}}\left(\widetilde{w}\right) = \sigma^2/N$ according to (6.12). In order to simplify matters, we will scale all average excess errors by σ^2/N. If the scaled quantity is above 1, it performs worse (the error is higher) than the maximum likelihood estimator for this value of $\widetilde{w}$. On the other hand, it performs better when the scaled average excess error is smaller than 1.

Figure 6.4 presents the (scaled) average excess error for all the estimators studied in this chapter. They all display the same general shape. There is a sharp decrease in error around 0, where the prior is rightfully used. For large teacher parameters, all estimators are asymptotically equivalent to maximum likelihood. The good performance for small teacher parameters compensates with an overshoot for parameters of intermediate size. This is the logical consequence of the use of our priors. A large part of the maximum likelihoods generated for such intermediate parameters are mistaken by our estimators for averages arising from small teacher parameters, leading to inadequate pruning, and high estimation error.

6.12 Influence of the prior

The above analysis tells us what estimator is best for which value of the teacher parameter. On the other hand, we might be interested in finding a way of comparing estimators regardless of the teacher parameter.

Integrating the scaled average excess error difference between an estimators and the maximum likelihood with a *uniform* prior on parameters gives a quantity that we will call the *uniform increase in error* for estimator w_E:

$$\mathcal{U}_E = \int \left(\overline{\Delta}_E\left(\widetilde{w}\right) - \overline{\Delta}_{ML}\left(\widetilde{w}\right)\right) U(\widetilde{w})\, d\widetilde{w} \tag{6.46}$$

where $U(\widetilde{w})$ is the uniform distribution on $\widetilde{w}$.

$\mathcal{U}_E$ measures the overall degradation of performance that our regularised estimators produce when confronted with a system for which the prior they use is wrong. The uniform increases in error for all regularised and Bayesian estimators are given in the following table:

Estimator	w_{MPG}	w_{MPL}	w_{GG1}	w_{GL}	BGS
Total difference	2,37	4,07	1,27	0,91	6,35

It is no surprise that all estimators perform, with a uniform measure, worse than the maximum likelihood and thus present a positive uniform increase in error. w_{ML} is, after all, the optimal unbiased estimator. The increase is roughly proportional to the size of the overshoot in figure 6.4. The noticeable exception to this rule is the estimator obtained with the generalisation method for the Laplace prior. The fast reduction in bias noted in section 6.6 leads to the smallest uniform increase in error, even though the overshoot is fairly high.

When the distribution of teacher parameters has significant mass around 0, the picture is entirely different. With an improper distribution proportional to $1/\widetilde{w}$, the *brave generalisation student* performs best. For a standard Gaussian distribution of teacher parameters, it all depends on the variance $\tilde{\sigma}^2$ of the distribution. Very large variance spreads the mass across the teacher parameter axis and results will be similar to the uniform case above. On the other hand, when the variance is low, the mass is localised around 0 and favours the most conservative estimators. To illustrate this effect, we define as above the *Gaussian increase in error* by integrating the average excess error over $\widetilde{w}$ with a Gaussian distribution of $\widetilde{w}$.

$$\mathcal{G}_E(\sigma) = \int \left(\overline{\Delta}_E\left(\widetilde{w}\right) - \overline{\Delta}_{ML}\left(\widetilde{w}\right)\right) \mathcal{N}\left(\widetilde{w}, \sigma^2\right)\, d\widetilde{w} \tag{6.47}$$

Figure 6.5 presents the evolution of this quantity with the variance of the teacher parameter. For small variance, all estimators predictably lead to a *decrease* in error. The estimators linked to the weight decay perform well, as the teacher distribution corresponds here to the prior they use. When the scaled teacher parameter[3] has variance $\tilde{\sigma}^2 = 1$, the Gaussian increase in error is:

Estimator	w_{MPG}	w_{MPL}	w_{GG1}	w_{GL}	BGS
Total difference	-0,280	-0,248	-0,295	-0,122	-0,157

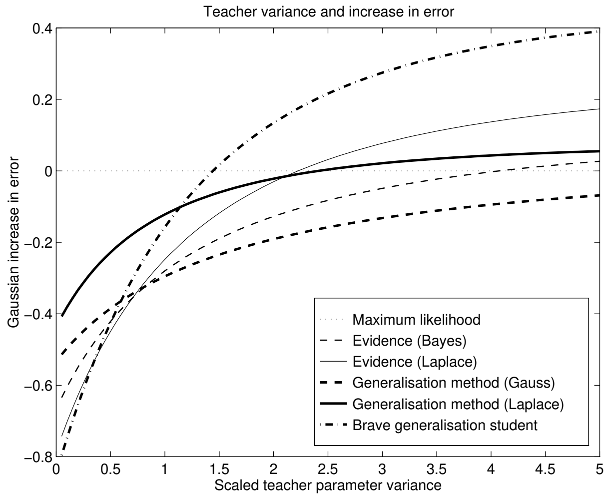


Figure 6.5: Influence of the teacher parameter variance on the increase in average excess generalisation error. For small variance, all estimators perform better than maximum likelihood. As the variance grows, performance tend to get worse.

6.13 Influence of the hypothesis

An important issue is the influence of the hypotheses on the results, or rather the influence of these hypotheses being wrong. All the calculation above have been performed for the case of a Gaussian noise on the data. This hypothesis influences the calculation by shaping the likelihood that is the basis for all methods. Of course, it influences the maximum likelihood. It is also of great importance during the Bayesian inference as the likelihood is involved in both the evidence framework and the MAP calculations. Lastly, it influences the choice of a quadratic cost function, a very important hypothesis when using the generalisation method.

An exhaustive study of this influence is difficult. There are many possible noise distributions, and the Gaussian distribution is one of the few for which the distribution of the average has an easy analytical expression. If we take the example of a Lapace noise distribution, it is hard to derive the expression of the distribution of the the average of a number of sample. This expression is necessary to integrate the error and produce the average excess errors on which our conclusions are based.

However, we know from the central limit theorem that the distribution of the average of the observation sampled from a distribution converges to a Gaussian distribution. As the average and variance of the noise are supposed well-known in our problem, we expect the main part of our results to stay valid.

[3]The teacher parameter is scaled by $\sqrt{\sigma^2/N}$, so this can be the case if one can think, for example, that the variance of the teacher parameter is 10 times that of the noise, and in the same time 10 examples are available.

6.14 Applications

From this section on we will address the problem of using this pruning prior together with neural networks models on operational problems. First we will use the sunspots benchmark problem as an example of time series modelling, then perform system identification on a small artificial system.

There are mainly two ways to improve the generalisation abilities of a neural network. *Regularisation* is most often used through the addition of a *weight-decay* term to the cost function. *Pruning* simply suppresses parameters that are believed to be useless. These methods are presented in chapter 2, and have been applied to a number of problems, including time series modelling and system identification.

The regularisation functional that we study in this chapter is simply the L_1 norm of the regularised parameters, and we will demonstrates how it provides automatic pruning of irrelevant parameters, without the use of a specific pruning technique. Combining two ways of improving generalisation error, it is a powerful tool for neural networks modelling.

Let us recall the expression of the regularised cost in that case:

$$C(w) = S(w) + \xi \sum_{k=1}^{P} |w_k| \tag{6.48}$$

where $S(w)$ is the standard (e.g. quadratic) cost, and ξ is the regularisation parameter. This kind of regularisation is sometimes referred to as *formal regularisation*, while pruning goes by the name of *structural regularisation*. As discussed earlier in this chapter, this kind of regularisation corresponds to the use of a Laplace prior on the parameters, while *weight-decay* is equivalent to a Gaussian prior.

The interesting property of this regulariser is that for any *non-zero* weights w_i at the minimum of $C(w)$, the first derivatives of (6.48) are zero, so that:

$$\left| \frac{\partial S(\hat{w})}{\partial w_i} \right| = \xi \qquad |w_i| > 0 \tag{6.49}$$

This means that the value of any *non zero* parameter is found at a point where the sensitivity of the data misfit to this parameter is in accordance with (6.49). If this is not possible (i.e. the sensitivity is nowhere that high), the parameter is forced to 0 and pruned out of the network.

The fact that this regulariser is not analytical in 0 is discussed in 2.16.

6.15 Time series

In order to illustrate the use of the pruning prior for time series processing, we choose the well known sunspots data. These have been historically the first time series studied using an autoregressive model. They have been established as one of the benchmarks for time series prediction algorithms using neural networks. This series is a yearly record of average sunspot activity. It has been recorded since 1700,

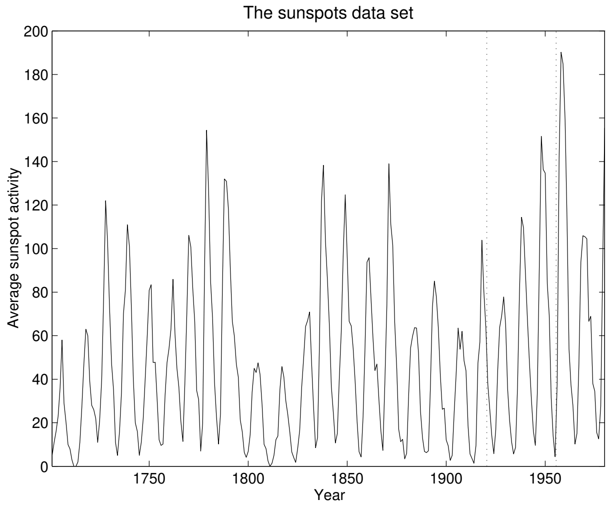


Figure 6.6: The well-known sunspots data. The dotted line indicate the limits of the three sets (training, validation and test).

and displays a cyclic pattern with maxima. The time between these ranges from 7 to 17 years, with a median of 11 years. It is displayed on figure 6.6.

In accordance with previous work, we attempt to predict one value using the twelve previous ones, leading to a network with 12 input units and one output unit. The available data constitutes 268 input-output pairs which are split into three sets. In the training set, we try to predict activities from 1712 to 1920, amounting to 209 examples, the validation set uses data from 1921 to 1955 (35 pairs), and the test set runs from 1956 to 1979 (24 examples).

No delay selection scheme was used during these experiments, all models and results mentioned later thus use the exact same network structure.

Performance is measured in terms of *average relative variance* (*arv*), which is the ratio between the average squared error of the model and the variance of the data. The widely used definition of the average relative variance of the prediction on a set D taken from the entire set S is:

$$arv(D) = \frac{\frac{1}{|D|}\sum_{i\in D}\left(y^{(i)} - f\left(x^{(i)}\right)\right)^2}{\frac{1}{|S|}\sum_{i\in S}\left(y^{(i)} - \langle y\rangle_S\right)^2} \tag{6.50}$$

It should be noted here that the more widely used definition scales by the *overall* variance of the data. This gives artificially high values of the error on the validation set, where the intrinsic data variance is around twice as high as in the other sets. A more "proper" definition of the *arv* quantity would relate the error of the model and the variance of the data calculated on the same set D:

$$arv(D) = \frac{\sum_{i\in D}\left(y^{(i)} - f\left(x^{(i)}\right)\right)^2}{\sum_{i\in D}\left(y^{(i)} - \langle y\rangle_D\right)^2} \tag{6.51}$$

In order to ease comparison with other papers, we will stick to the widely used

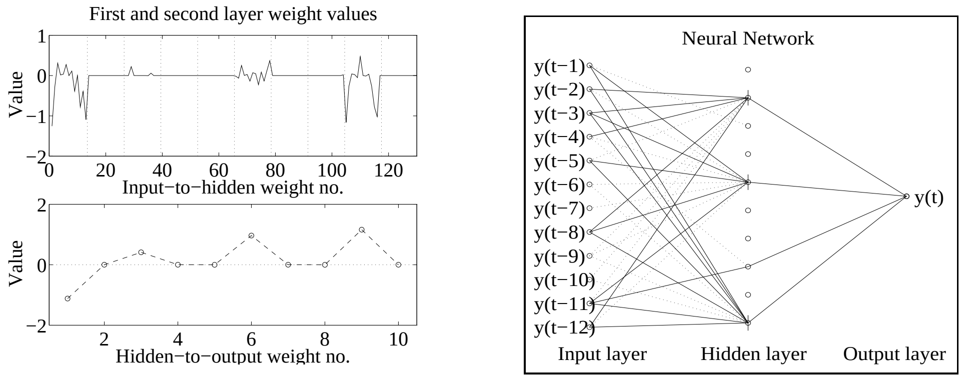


Figure 6.7: Top-left: input-to-hidden parameters, by hidden unit. Every hidden unit has 13 incoming connections: weights 1–13 go to HU 1, weights 14–26 to HU 2, etc. Bottom-left: hidden-to-output parameters. Right: the network. Dotted lines correspond to connections that are one order of magnitude lower than those in solid line. A vertical line through a cell corresponds to an active threshold.

definition. The proper numbers can be obtained by multiplying the given values by 1.277, 0.903 and 0.496 for the training, validation and test sets respectively.

6.16 Experiments and results

The model we used is a 12-10-1 neural network, containing 141 parameters, which can be considered as highly over-parameterised compared to the 209 data in the training set. Training is performed by minimising the cost function (6.48) as mentioned above. The optimisation technique is a standard conjugate gradient, implemented under the SN simulation software, from Neuristique. The regularisation parameter is set with the help of the validation set, by choosing the value of ξ that minimises the validation error, and the results are evaluated on the test set. As discussed in chapter 3, this scheme is not very satisfying, but has been adopted because it makes use of the 3 available sets in a manner similar to that of most studies.

The parameters of one of the networks we obtained are displayed on figure 6.7. It can be seen that in the course of the learning procedure, 6 of the hidden units have been disabled, and effectively pruned. Some of the input connections of the remaining hidden units have also been driven to 0 (e.g. in the third hidden unit). The solution displayed here is typical of those we obtained: 3 units perform most of the work, with one (sometimes two) additional units contributing to a lesser extent. In the inputs, the biggest contributions come from the first units, one middle unit ($t-8$ here) and the last two units. This pattern has already been observed by Svarer et al. (1993) using a completely different method.

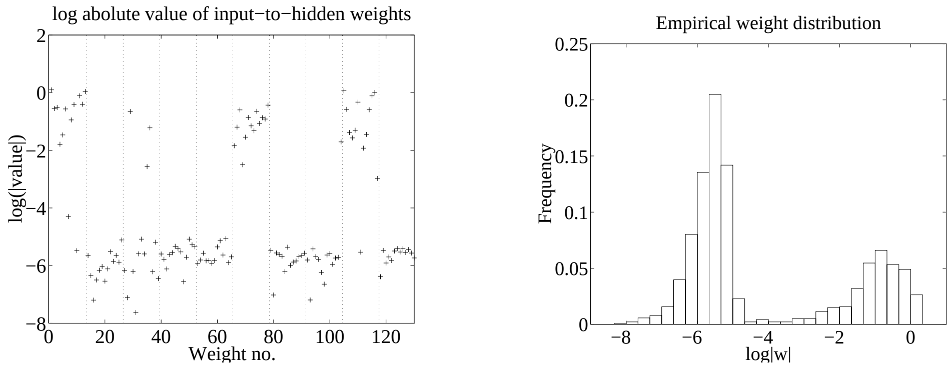


Figure 6.8: Left: magnitude of the input-to-hidden weights. Notice the logarithmic y-axes. Right: distribution of the log absolute value of the parameters. Notice the two modes corresponding to the pruned and not pruned weights.

6.17 Parameter distribution

Figure 6.8 corroborates the pruning effect by displaying the distribution of the parameters. On the left panel, the magnitude plot shows that input weights of the same network have been more or less clustered in two categories: roughly, the active weights are greater than 10^{-2}, when the rest are gathered below 10^{-4}. Again, we see that six hidden units have been effectively discarded. On the right, the empirical distribution of the absolute values of the weights of 10 networks trained with the pruning prior. With a logarithmic X-axis, it clearly displays two modes, one corresponding to the pruned parameters, and the other to those that are still in use. Apart from its illustrative purpose, this provides a way of empirically counting the number of effective parameters. This number can be used to obtain an algebraic estimate of the generalisation error.

This leads us to discussing the empirical distribution of the weights. A natural idea would be to test this empirical distribution against the Laplace distribution that we used as a prior. It should be clear from the plots presented here that the solution parameters do not follow a Laplace distribution, and that is to be expected. As Bayesians are well aware, one should indeed not confuse the probability of the parameters (the posterior, i.e. regularised cost) with the assumption on their distribution (the prior, i.e. the regularisation functional).

Of higher interest is the consideration of the actual empirical distribution. We used a Kolmogorov-Smirnov test to check each of the 10 network solutions obtained against the empirical distribution obtained by the combination of the rest of them. The result of this test is that only two networks have significance level between 1 and 5 percents against the others. The rest has higher significance level, up to 90%.

One could wonder why none of these "pruned" weights actually have a 0 value. It should be understood that (6.49) is valid for the exact minimum, obtained after a continuous optimisation. In the case of numerical learning, we proceed by discrete optimisation, minimising (6.48) in a number of steps, until the gradient is considered sufficiently small. An unnecessary parameter will typically endure oscillations around

Model/arv	Train (1712-1920)	Test (1921-1955)	Validation (1956-1979)
Linear	0.131	0.128	0.36
Weigend & al. 90	0.082	0.086	0.35
Svarer & al. 93	0.090 ± 0.001	0.082 ± 0.007	0.35 ± 0.05
Pruning prior	0.082 ± 0.001	0.082 ± 0.002	0.357 ± 0.013

Table 6.2: Comparison of the results obtained using different methods. The linear model includes an intercept. The results reported for Svarer & al. are averaged over 9 *retrained* networks; for the pruning prior we averaged over 10 network solutions.

0 with an amplitude decreasing with the step size. It will hence get closer to 0 but it is extremely unlikely that it will reach this exact value. This leads to one more remark: the technique known as "early stopping" is a poor combination with the pruning prior. By stopping before the minimum is reached, (6.49) will not stand, and therefore we have no guarantee on the pruning effect.

Table 6.2 displays the overall results obtained using the above procedure compared with previously released results. The figures reported by Svarer et al. are averaged over 9 successful trainings, and the results reported for the pruning prior are averaged over 10 trainings.

6.18 System identification

In this section, we consider the example of a simple, artificial system:

$$y(t) = \frac{u(t) + y(t-2)}{1 + y^2(t-1) + u^2(t-1)} \tag{6.52}$$

Figure 6.9 displays the behaviour of the system for two different kinds of input signal: a step signal of low frequency, and a random step signal, which corresponds to the training sequence we use to identify the parameters of the models below.

The random step signal is made out of steps of random lengths and amplitude. This type of signal allows for a better representation of the frequency domain than a purely random signal. For training, we generate 200 data using such a signal as input $u(t)$. The output $y(t)$ is then corrupted by Gaussian noise, $\sigma = 0.5$. This is a rather high value of the noise. In order to evaluate the results, we generate a large test set of $10,000$ data to get a hopefully fairly accurate estimate of the generalisation error. In all models below, we use the two last values of both u and y as input: $[y(t-1), y(t-2), u(t), u(t-1)]$ is the input vector of the model that attempts to predict $y(t)$.

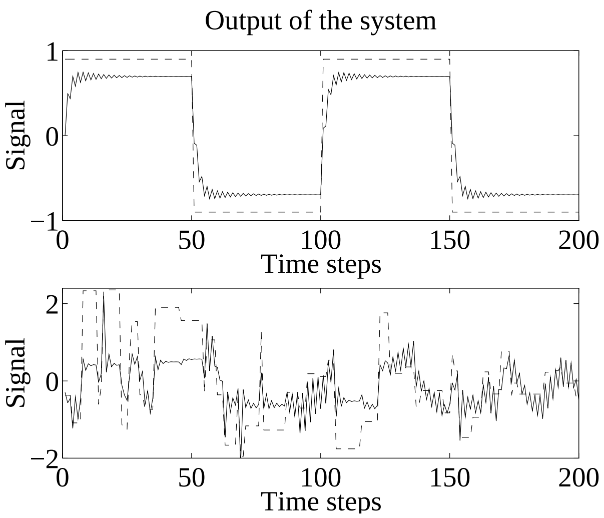


Figure 6.9: Response of the system 6.52 for two different input signals. The dashed line is the input signal and the solid line is the system response.

6.19 Experiments and results

We first perform a linear identification on our 200 noisy data. The values of the parameters are gathered in the following table. We also mention the average and standard deviation of the coefficients obtained in $10,000$ experiments using as many different sequences of 200 noisy data.

	$y(t-1)$	$y(t-2)$	$u(t)$	$u(t-1)$	1
Training sequence	0.053	0.247	0.469	-0.198	-0.060
Average of 10000	0.055	0.321	0.449	-0.117	-0.000
St. dev. of 10000	0.060	0.082	0.073	0.079	0.051

Predictably, the main linear influences come from $u(t)$ and $y(t-2)$. Figure 6.10 displays the behaviour of the linear model for the same signals as before. It is clear that the linear model is unsatisfactory.

The second step is to use a neural network to perform the identification. We use a 4-10-1 network to try to fit the data. This neural network model contains 61 parameters (including bias), which can be considered as slightly over-parameterised, compared to the 200 data at hand.

Training is performed by minimising the regularised cost (6.48), with MAP update of the regularisation parameter as in equation (5.36). The performance of all models is evaluated on the $10,000$ data of the test set.

On figure 6.11 we display a plot comparable to figure 6.7, for our system identification case. One can see that among the 10 hidden units originally in the model, only 1 remains in use. All the other hidden units have been pruned out of the network. This network is the smallest one we obtained, with only 7 parameters. Other solutions have up to 13 active parameters, and usually involve 2 hidden units. The number

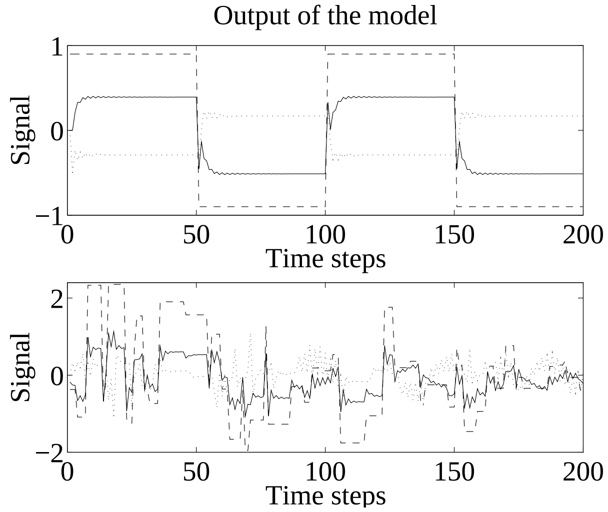


Figure 6.10: Response of the linear model for two different input signals. The dashed line is the input signal, the solid line is the system response, and the dotted line is the modelling error.

of active parameters can vary from a solution to another, depending on the initial conditions. However, it stays well short of the 61 weights of the full network.

The table below summarises the results obtained by both the linear model and a number of non-linear neural network models on the noisy training set and on the non-noisy test set. In this table, NN means Neural Networks. The numbers indicated are the mean squared error (MSE) over the training (resp. test) set. The performance on the test set is higher as we use only non-noisy data, when the training set has a relatively high level of noise.

Model/MSE	Number of parameters	Training set (200 data)	Validation set (10,000 data)
Linear model	5	0.310	0.120
Unregularised NN	61	0.190	0.244
Regularised NN	7–13	0.288 ± 0.01	0.105 ± 0.006

These results show that the non-linear regularised model predictably achieves performance slightly above the linear model. Furthermore, the use of the pruning prior limits the number of parameters to a number comparable to linear model. The parameters show the same pattern as in the linear case: the main contribution arises from $u(t)$ and $y(t-2)$. The influence of the other inputs and the non-linearity of the model allow for an error that is half that of the linear model.

COMMENTS

The part of this chapter related to the "parameter location problem" (from section 6.2 to 6.11 is related to results published in Hansen and Rasmussen (1994)

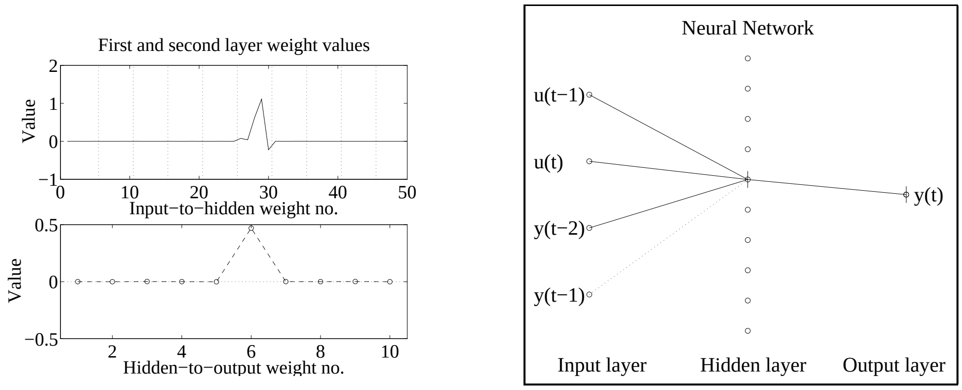


Figure 6.11: Top-left: input-to-hidden parameters, by hidden unit. Every hidden unit has 5 incoming connections: weights 1–5 go to HU 1, weights 6–10 to HU 2, etc. Bottom-left: hidden-to-output parameters. Right: the network. Dotted lines correspond to connections that are one order of magnitude lower than those in solid line. A vertical line through a cell corresponds to an active threshold.

and Goutte and Hansen (1997). These papers study the case of Gaussian and Laplace prior on the same toy problem, using the evidence framework and the generalisation method. The Gaussian case is studied by Hansen and Rasmussen (1994), who coined the term *brave generalisation student*. The remark relative to the $\mathrm{P}(\widetilde{w}) = 1/\widetilde{w}$ distribution is also theirs.

Sections 6.12 and 6.13 contain unpublished material.

The application to time series modelling and system identification is mostly taken from Goutte (1996). Other comments follow:

6.14 The effect of weight-decay is shown e.g. by Krogh and Hertz (1992), while pruning techniques originate in the work of Le Cun et al. (1990). Formal and structural regularisation are discussed by Denker et al. (1987).

6.15 The sunspots data have first been studied by Yule (1927) with an AR model. The have first been used in the context of neural networks prediction by Weigend et al. (1990).

6.16 The SN simulation software from Neuristique is discussed in Bottou and Le Cun (1988). The conjugate gradient method is detailed in section 1.13; see also Press et al. (1992). The results refer to Weigend et al. (1990) and Svarer et al. (1993) in the list of references below.

6.18 The simple system to identify is taken from Ljung et al. (1992).

References

Bottou, L. and Le Cun, Y. (1988). SN: A simulator for connectionist models. In *NeuroNîmes 88*, pages 371–382, Nîmes, France.

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.

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.

Hansen, L. K. and Rasmussen, C. E. (1994). Pruning from adaptive regularization. *Neural Computation*, 6:1223–1232.

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.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (1992). *Numerical Recipes in C*. Cambridge University Press, 2nd edition.

Svarer, C., Hansen, L., and Larsen, J. (1993). On design and evaluation of tapped-delay neural network architectures. In et al., H. B., editor, *IEEE International Conference on Neural Networks*, ICNN, pages 46–51, Piscataway, NJ. IEEE.

Weigend, A. S., Huberman, B. A., and Rumelhart, D. E. (1990). Predicting the future: a connectionist approach. *International Journal of Neural Systems*, 1(3):193–210.

Yule, G. U. (1927). On a method of investigating periodicities in disturbed series with special reference to Wolfer's sunspot numbers. *Philos. Trans. R. Soc. Lond. Ser. A*, 226:267.

A

Probability

We will here present the axiomatic definition of probability. This can be found in any textbook on statistics and probability. The aim of this appendix is to provide a quick reference for some of the concepts used in the thesis, particularly chapter 5.

Let us start with a couple of definitions. In the following, Ω is a set, and $\mathcal{A}$ is a collection of subsets A of Ω: $\forall A \in \mathcal{A},\ A \subset \Omega$. For a subset A, $\overline{A}$ represents the complement of A in Ω, i.e. $\overline{A} = \Omega \setminus A$.

Definition A.1 *A nonempty collection $\mathcal{A}$ of subsets of Ω is a* field *if:*

- $A \in \mathcal{A} \Rightarrow \overline{A} \in \mathcal{A}$.
- $A_1, A_2 \in \mathcal{A} \Rightarrow (A_1 \cup A_2) \in \mathcal{A}$.

Definition A.2 *A field $\mathcal{A}$ is a σ-field if $\{A_i\}_{i=1}^{\infty} \in \mathcal{A} \Rightarrow \bigcup_{i=1}^{\infty} A_i \in \mathcal{A}$.*

The definition of a *σ-field* extends that of a *field* by considering an infinite family of subsets of Ω. We can now proceed and define the concepts of *measure* and *measure space*.

Definition A.3 *A function $\mu : \mathcal{A} \to [0, +\infty[$ is a* measure *if:*

- $\mu(\emptyset) = 0$.
- $\forall \{A_i\}_{i=1}^{\infty}$ *mutually disjoint,* $\mu\left(\bigcup_{i=1}^{\infty} A_i\right) = \sum_{i=1}^{+\infty} \mu(A_i)$.

Definition A.4 *If Ω is a set and $\mathcal{A}$ a σ-field, $(\Omega, \mathcal{A})$ is a* measurable space.

Definition A.5 *If $(\Omega, \mathcal{A})$ is a measurable space and μ a measure, $(\Omega, \mathcal{A}, \mu)$ is a* measure space.

Definition A.6 *If* $(\Omega, \mathcal{A}, \mu)$ *is a measure space, and* $\mu(\Omega) = 1$*, then* $(\Omega, \mathcal{A}, \mu)$ *is a* probability space*, and* μ *is a* probability *on* $(\Omega, \mathcal{A})$*.*

This is the *axiomatic definition of probability*. We will now refer to a subset A from a probability space as an *event*, without mentioning the actual existence of the underlying set Ω and σ-field $\mathcal{A}$. In that context, the measure μ is a probability, and will be noted $\mathrm{P}(A)$.

From the above definitions, we deduct the following basic properties of probabilities:

1. $\mathrm{P}(\emptyset) = 0$.
2. $\mathrm{P}\left(\overline{A}\right) = 1 - \mathrm{P}(A)$.
3. $\forall A_1, A_2 \in \mathcal{A},\ A_1 \subset A_2 \Rightarrow \mathrm{P}(A_1) \leq \mathrm{P}(A_2)$.
4. $\mathrm{P}(A_1 \cup A_2) = \mathrm{P}(A_1) + \mathrm{P}(A_2) - \mathrm{P}(A_1 \cap A_2)$.
5. $\forall A_i \in \mathcal{A},\ \mathrm{P}\left(\bigcup A_i\right) \leq \sum_i \mathrm{P}(A_i)$.

Let us define the concept of *conditional probability*.

Definition A.7 *If* A *and* B *are two events such that* $\mathrm{P}(B) \neq 0$*, the* conditional probability of A given B *is:*

$$\mathrm{P}(A|B) = \frac{\mathrm{P}(A \cap B)}{\mathrm{P}(B)} \tag{A.1}$$

The conditional probability is indeed a probability, with respect to definition A.6.

Definition A.8 *Two events* A *and* B *are* independent *iff* $\mathrm{P}(A \cap B) = \mathrm{P}(A)\,\mathrm{P}(B)$*.*

We can also define the concept of independence by using the conditional probability. A is independent of B iff $\mathrm{P}(A|B) = \mathrm{P}(A)$. This means that the outcome of B does not influence the outcome of A. It is equivalent to say that A is independent of B, B is independent of A, or A and B are independent. We prolong definition A.8 to a set of events:

Definition A.9 *A set* $\{A_i\}_{i=1}^{N}$ *of events are mutually independent iff:*

$$\mathrm{P}\left(\bigcap_{i=1}^{N} A_i\right) = \prod_{i=1}^{N} \mathrm{P}(A_i) \tag{A.2}$$

B

Non-parametric regression

B.1 Smoothing regression

We deal here with a type of regression different from the one sudied earlier in chapters 1. Non-parametric methods have been developped in the sixties to surpass the limits of classical parametric methods.

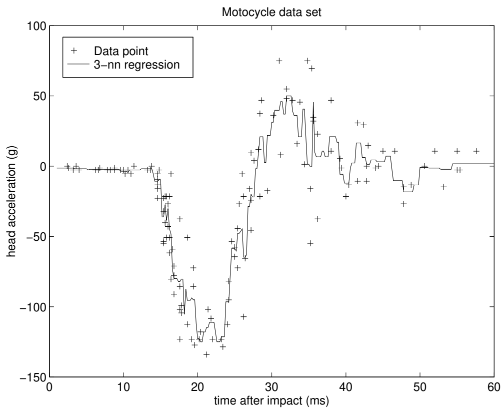
Recall that we try to estimate the regression of input x to output y based on a set of N examples $\left(x^{(i)}, y^{(i)}\right)$. Rather than parameterise the mapping f from x to y, we consider that for an input x, it is likely that the data set features some input vector $x^{(i)}$ that is close to x. With reasonable assumptions on the smoothness of f, we expect the regression estimate $\hat{y}$ to be close to $y^{(i)}$.

The non-parametric methods thus provide a regression estimation $\hat{y}$ that is data-driven, given as a weighted combination of the data:

$$\hat{y}(x) = \frac{1}{N} \sum_{i=1}^{N} W_i(x) . y^{(i)} \tag{B.1}$$

Even when not *defined* in such terms, non-parametric methods accept an equivalent expression of this form. We will here present two basic methods, k-nearest neighbours and kernel smoothing, and combine them with more advanced features. For both methods, we present examples of implementation, including for the smoothing parameter tuning. The chapter ends with a presentation of GRNN, a popular model in the field of neural computation. This links these non-parametric techniques to the field of neural networks.

Throughout this chapter, we will apply the methods discussed to the well-known motorcycle data set. It is a one dimensional set giving the head acceleration of a post mortem human test object as a function of the time after a simulated impact with a motorcycle. A plot of the data set is given on most figures in this chapter.



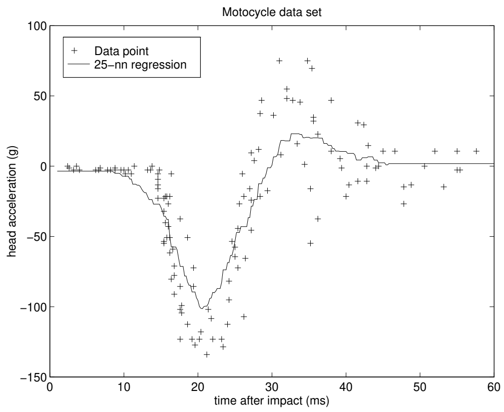


Figure B.1: Regression curve for a k-NN estimation with too few neighbours, $k = 3$.

Figure B.2: Regression curve for a k-NN estimator with too many neighbours, $k = 25$.

B.2 k-nearest neighbours

The k-nearest neighbours algorithm generalises the idea of moving average to regression estimation. It performs an average of the output $y^{(i)}$ of the k data points for which $x^{(i)}$ is closer to x, giving the following *uniform weight sequence*:

$$W_{ki}(x) = \begin{cases} \frac{N}{k} & i \in V_k(x) \\ 0 & \text{otherwise} \end{cases} \tag{B.2}$$

where $V_k(x) = \left\{ i : x^{(i)} \text{ is one of the } k \text{ data inputs closest to } x \right\}$ is the set of the k data points closest to x. The k-NN regression estimation is then:

$$\hat{y}_k(x) = \frac{1}{N} \sum_{i=1}^{N} W_{ki}(x)\, y^{(i)} = \frac{1}{k} \sum_{j \in V_x} y^{(j)} \tag{B.3}$$

Many alternatives exist to the *uniform weight sequence*, such as *local linear fit*, or the application of a *kernel* (cf. section B.10).

Figures B.1 and B.2 display two examples of k-NN regressions applied to the motorcycle data. In figure B.1, we used a small number of neighbours, leading to a typical case of *under-smoothing*. The behaviour of the estimate is highly influenced by the noise on the data. The second regression, in figure B.2, is made by using too many neighbours, and the regression curve is clearly over-smoothed. The observational noise has been greatly reduced, but the estimation bias is important. Notice e.g. that the region between 30 and 40 ms has been flattened.

B.3 Error decomposition for k-NN estimates

The balance between the reduction of observational noise and the estimation of the "true" regression curve is illustrated by the *bias–variance decomposition*. We have addressed this issue for parametric models in section 1.15.

The asymptotic expressions of bias and variance for one dimensional k-NN estimation using a weight setting given by (B.2) are given in the following proposition:

Proposition B.1 (Lai, 1977) *When $k \to \infty$, $N \to \infty$ and $k/N \to 0$, the bias and variance of the k-NN estimate with uniform weight sequence are:*

Bias: $\mathrm{E}\left(\hat{y}_{kNN}(x)\right) - f(x) \sim \left(\frac{k}{N}\right)^2 \frac{f''(x)p(x)+2p'(x)f'(x)}{24p(x)^3}$

Variance: $var\left(\hat{y}_{kNN}(x)\right) \sim \frac{\sigma^2}{k}$

To understand these expressions fully, let us recall that f, f' and f'' are the regression function and its first two derivatives, p is the density of the input, and σ^2 is the variance of the noise on the observations.

When k is low, the bias will tend to be low (for a given N), but the variance, inversely proportional to k, will be high. It is the opposite when k is too large, as illustrated on the example above. Furthermore, Proposition B.1 allows us to find the asymptotically optimal trade-off between squared bias and variance. In order for both those quantities to converge towards 0, let us impose that both should be asymptotically equivalent. If we focus on the quantities of interest, i.e. k and N, the equivalence between squared bias and variance gives $(k/n)^4 \sim 1/k$.

The asymptotical setting of k should thus be $k = \mathcal{O}\left(N^{\frac{4}{5}}\right)$ in order to obtain reduction of both variance and bias. The variance being inversely proportional to k, this setting makes the mean squared error converge to 0 like $N^{-\frac{4}{5}}$. The above proposition ensures the convergence of the estimate towards the true regression curve, in the limit of the great number of examples, and provided that k is chosen in accordance with the above rate.

B.4 k-NN implementation and computational issues

the basic implementation of the uniform weight sequence is pretty straightforward, and is given here in MATLAB code. `x` is the input where the estimation is performed, `Xdata` and `Ydata` are the matrix of the input and output vectors (respectively) in the training set, and `k` is the number of neighbours that we use.

```
function y = knn(x, Xdata, Ydata, k)
[P N] = size(Xdata) ;
distance = zeros(N, 1) ;
for j = 1:N
   distance(j) = (Xdata(:, j) - x)' * (Xdata(:, j) - x) ;
end
[sorted_dist, Index]  = sort(distance) ;
y = sum((Ydata(:, Index(1:k)))')' / k ;
```

In such a basic implementation, the dominant cost arises from sorting the distances for every estimation. For N examples and P-dimensional inputs, such an estimation can be done in $\mathcal{O}\left(N\left(\log N + P\right)\right)$ time.

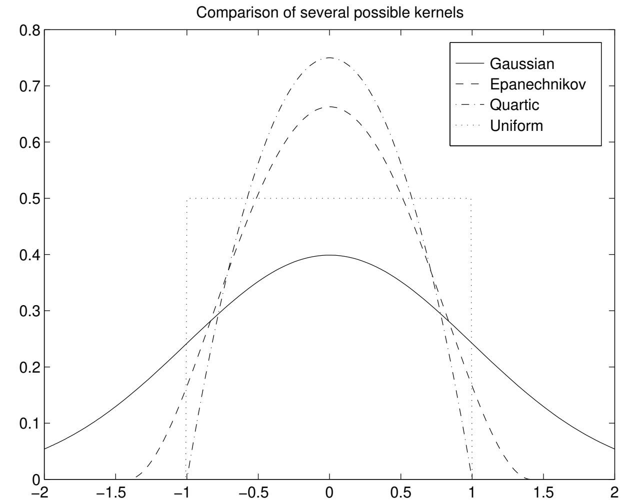


Figure B.3: Several kernels are plotted in order to compare their shapes. The uniform kernel is also plotted for comparison.

In the case of one-dimensional data, simplifications arise. On pre-sorted data, the estimation can be performed by a single pass over the data, bringing the cost to $\mathcal{O}(N)$, i.e. linear in the number of examples. Furthermore, repeated estimation are also very fast: computing the estimates in all N data points $x^{(i)}$ is also linear. Lastly, formulas can be derived for a fast update of the estimation in case the training set is modified.

This shows that the k-NN estimator is an extremely fast regression method. Furthermore, with a sound choice of neighbours, k-NN regression usually performs well. The crude averaging of Eq. B.3 can be replaced by more sensible methods as explained later in section B.10. The speed and ease of use of the k-nearest neighbours method make it an interesting basis for comparison.

B.5 Kernel estimators

Let us now consider that rather than taking a fixed number of data, we will weigh the data points according to their distance to the point x where we estimate. A simple solution is to use a *kernel* function to weigh the influence of each data point. This is the idea behind the well known Nadaraya-Watson estimator, which uses the weight sequence:

$$W_{di}(x) = N \frac{K_d\left(x - x^{(i)}\right)}{\sum_{k=1}^{N} K_d\left(x - x^{(k)}\right)} = \frac{K_d\left(x - x^{(i)}\right)}{\hat{p}_d(x)} \tag{B.4}$$

Function $K_d(x)$ is the kernel of width d, defined as $K(x/d)/d$, where K is the standard *kernel*. Expression B.4 is related to density estimation, as $\hat{p}_d(x)$ is the Parzen window estimation of the input density in x. Figure B.3 presents the shape of some of the most used kernels:

- The Gaussian kernel is a Gaussian function of mean 0 and variance 1: $K(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$.
- The Epanechnikov kernel is a quadratic function with compact support: $K(x) = \frac{3}{4}\left(1 - x^2\right) I\left(x^2 < 1\right)$.
- The Quartic kernel is a fourth degree polynomial with compact support: $K(x) = \frac{15}{16\sqrt{2}}\left(1 - \frac{x^2}{2}\right)^2 I\left(x^2 < 2\right)$.
- The uniform kernel given for comparison corresponds to a moving average around x : $K(x) = \frac{1}{2} I(|x| < 1)$.

I is the indicator function which value is 1 in the interval defined by the condition and 0 outside.

The kernel estimation in x is then:

$$\hat{y}_d(x) = \frac{1}{N} \frac{\sum_{i=1}^{N} K_d\left(x - x^{(i)}\right) y^{(i)}}{\hat{p}_d(x)} \tag{B.5}$$

Apart from the Gaussian kernel, notice that all kernels presented on figure B.3 have compact support. This gives interesting computational properties, discussed in section B.7. It also raises a critical point: the estimate is not defined when $\hat{p}_d(x) = 0$. This is the case in two situations:

1. when the kernel estimation is performed outside of the data range;
2. whenever the kernel size is so small that in some points, no data can be encompassed in the kernel.

At any rate, $\hat{y}(x)$ can be defined as being the average of the outputs in such cases.

Figures B.4 and B.5 present two different estimation curves obtained on the motorcycle data using an Epanechnikov (i.e. quadratic) kernel. In figure B.4, the kernel size is much too small and the estimation is under-smoothed: it follows the data much too closely. On the other hand, figure B.5 displays a typical case of over-smoothing. The regression curve is obviously badly estimated in the range 10–40ms.

The trade-off now depends on the kernel size d. The following section investigates this in the light of the bias–variance asymptotic expressions.

B.6 Error decomposition for kernel estimates

We present here a result similar to Proposition B.1 in the case of kernel smoothing.

Proposition B.2 *When $d \to 0$, $N \to \infty$ and $Nd \to \infty$, the bias and variance of the one dimensional kernel estimate with Epanechnikov weight sequence are:*

Bias: $\mathrm{E}\left(\hat{y}_k(x)\right) - f(x) \sim d^2 \frac{f''(x)p(x) + 2p'(x)f'(x)}{6p(x)}$

Variance: $var\left(\hat{y}_k(x)\right) \sim \frac{\sigma^2}{5Ndp(x)}$

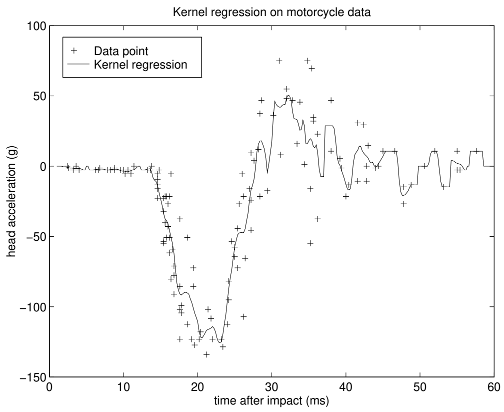


Figure B.4: Regression curve for a kernel estimation with a small kernel, $d = 0.6$.

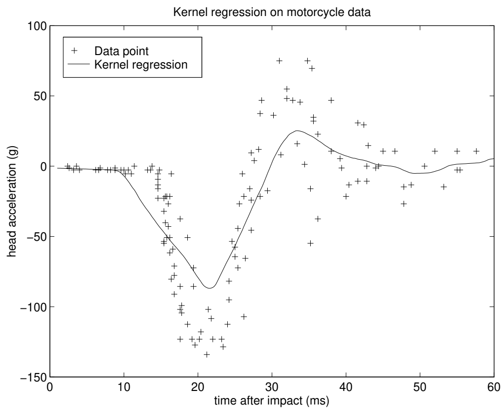


Figure B.5: Regression curve for a kernel estimator with a large kernel, $d = 4.0$.

As in proposition B.1, f and p are the regression function and the input density (respectively).

Proposition B.2 clearly shows that d and k play asymptotically equivalent roles, with $k \sim 2Ndp(x)$. This is intuitively natural, as $2Ndp(x)$ is approximately the number of examples to be expected in the vicinity of x. The proposition also gives the asymptotic setting of the kernel size. Using the same argument as in section B.3, we have $d = \mathcal{O}\left(N^{-1/5}\right)$. When this asymptotical setting is adopted, the mean squared error between the true regression curve and the kernel estimate decreases like $N^{-4/5}$, as before.

B.7 Kernel implementation and computational issues

The implementation of a kernel smoother is also simple, as presented in the MATLAB routine below. We compute the kernel estimate in `x` based on the data in `Xdata` and `Ydata`, with a kernel of width `d`, using a Gaussian kernel shape.

```
function y = kernel(x, Xdata, Ydata, d)
[P N] = size(Xdata) ;
sq_dist = zeros(N, 1) ;
for j = 1:N                         %% Squared distances
   sq_dist(j) = (Xdata(:,j) - x)' * (Xdata(:,j) - x) ;
end
s2 = 2 * d * d ;
W = exp(- sq_dist / s2) ;           %% Weights calculation
y = Ydata * W / sum(W) ;            %% Regression estimation
```

There are two main conceptual differences between this case and the k-NN algorithm described in section B.4:

1. no sorting of the distances is necessary before the estimation;

2. *all* the examples are weighed and used in the estimation.

This last point is important, as it reveals the complexity of the estimation. For a single point, the complete kernel estimation requires linear $\mathcal{O}(N)$ time. This is comparable with the k-NN method, though the complex operations involved here (such as exponentiation) yield a multiplicative factor.

Contrary to the k-NN algorithm, no sorting is here necessary. However, for repeated estimation, it is necessary to re-calculate the entire weight sequence for each point, making the regression estimation on N points $\mathcal{O}(N^2)$.

In order to lower the calculation time, it is convenient to use a kernel with compact support. Thus, only a (hopefully small) number of data will actually contribute to the estimate. We present here the implementation of the Epanechnikov kernel, the shape of which can be seen from figure B.3. The following code replaces the last two lines above:

```
Idx = find(sq_dist < s2) ;          %% Gets data in support
W = (1 - sq_dist(Idx) / s2) ;
if sum(W) > 0
   y = Ydata(Idx') * W / sum(W) ;    %% Normal estimation
else
   y = mean(Ydata')' ;              %% No data in the support
end
```

The only difference, apart from the kernel shape itself, is that we are now required to test whether there were data in the support.

When the number of data is large and the kernel width is small (these two characteristics usually go together according to proposition B.2), the speed-up resulting from the use of a compact support kernel can be very important. For an Epanechnikov kernel of size 1, the average number of data involved in the estimation is 5 instead of 133 for the Gaussian kernel, yielding an expected speed-up of around 25.

B.8 Smoothing parameter tuning

The above techniques allow for an easy estimation of the regression curve. Furthermore, they are consistent, and we have some asymptotical convergence results. The reliability of the estimate in a practical case, however, relies on a proper setting of the *smoothing parameter*, either the number of neighbours or the kernel size.

We will here again invoke the concept of generalisation. In non-parametric regression as well as for parametric method, the quality of the estimation is measured by the expected error. It is common to assess the generalisation error by computing the leave-one-out cross-validation estimate, as presented in section 3.4. The LOO estimate corresponds to the average error on a data point when the estimation is performed on the basis of the remaining points:

$$E_{LOO} = \frac{1}{N} \sum_{j=1}^{N} e\left(\hat{y}_{-j}\left(x^{(j)}\right), y^{(j)}\right) \tag{B.6}$$

where $e(\cdot,\cdot)$ is a measure of the error between the estimation and a data point, usually the squared difference (with Gaussian assumption on the noise). $\hat{y}_{-j}$ represents the estimation performed without observation j in the dataset. It is rather convenient to calculate E_{LOO} for non-parametric regression:

$$E_{LOO} = \frac{1}{N}\sum_{j=1}^{N}\left\|\frac{1}{N-1}\sum_{i\neq j} W_i\left(x^{(j)}\right)y^{(i)} - y^{(j)}\right\|^2 \tag{B.7}$$

Technical remark: Non-parametric methods usually compute an estimate based on data *surrounding* a given point. This is a weakness when attempting to estimate the regression curve at e.g. the boundaries. In such a case, many data points lie on one side of the estimation point, but virtually none on the other side, producing weak estimates. This can prove troublesome when trying to assess the value of the smoothing parameter. The poor estimate at the boundaries is weakened by the fact that the LOO discards each point when trying to estimate performance around it. This will lead to choosing too small a value of this parameter, producing an under-smoothed regression curve. To compensate for this undesirable behaviour, it is customary to weigh the examples in (B.6) in order to avoid involving extreme data points. Boundary data will thus be involved when estimating the regression, but no LOO estimation is performed there, as it would surely lead to poor results. A common technique consists in weighing to 0 a certain percentage of the points that are closest to the boundaries. However, if the notion of boundary is well-defined in the one-dimensional case, it is not the case as the number of dimension increases. In high dimensions, virtually *all* points can be considered as being "close to the boundaries" and it is not clear how to adapt this correction.

B.9 Practical implementation and computational issues

We present below the basic implementation of the LOO calculation for nearest neighbours regression. It is given in MATLAB code, and the variables are the same as in section B.4. The weighing technique mentioned above is implemented by not counting the LOO contribution from `pct` percent of the data at each end of the training set. It is left to the user to have the boundary data at the boundaries of the set.

```
function E = knnloo(k, Xdata, Ydata, pct)
[P N] = size(Xdata) ;
[T I] = sort(dist2(Xdata)) ;   %% Sorted square distances
E = 0 ;
imin = 1 + round(pct*N) ;
imax = N - round(pct*N) ;
for i = imin:imax
   Yhat = sum( Ydata(:, I(2:(k+1), i))' )' / k ;
   E = E + (Ydata(:, i) - Yhat)' * (Ydata(:, i) - Yhat) ;
end
E = E / (imax - imin + 1) ;
```

The function `dist2` on line 2 corresponds to the square distance matrix calculation. The detail of the calculation can be found in the kernel smoothing implementation

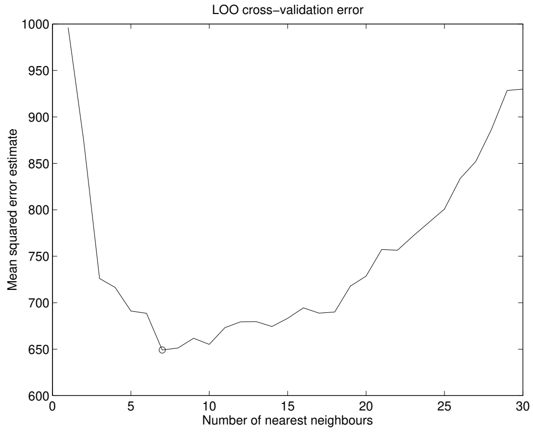


Figure B.6: LOO estimate for various values of k with 10% of the data left out. Minimum is obtained for $k = 7$.

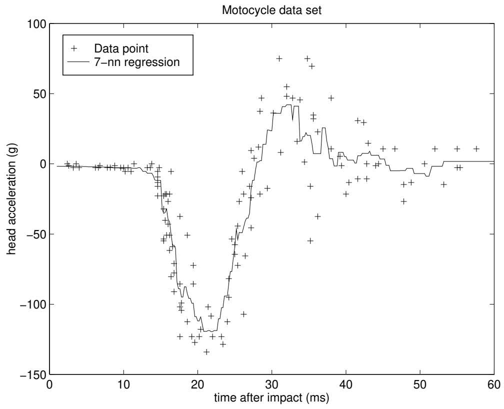


Figure B.7: LOO-optimal regression curve obtained using a 7-NN estimator.

(section B.7). This simple implementation is rather costly: calculating and sorting the distance matrix is already $\mathcal{O}\left(N^2 \log N\right)$, and the rest of the calculation is "only" $\mathcal{O}\left(N^2\right)$.

When repeated estimation are necessary, as is the case when we tune the smoothing parameter, the sorted distance matrix can be calculated once and for all. With sorted data, all individual estimates $\hat{y}_{-j}$ are computed in a single pass over the data. The LOO estimation for a k nearest neighbours algorithm on sorted data is thus only $\mathcal{O}(N)$.

The LOO generalisation error estimate is computed for 1 to 30 neighbours in figure B.6. We have here weighted the first and last 5% of the data to 0. The shape of the curve in the figure shows how the estimation error grows in case of under- or over-smoothing. The LOO scheme selects 7 neighbours as the optimal number. The estimation curve corresponding to this number is shown in figure B.7.

B.10 Variable kernel method

The reliability of the kernel estimation depends greatly on the number of data encompassed by the kernel. As seen above, the expected number of points encompassed is proportional to the density. If the density is halved, so is the expected number of points involved in the estimation. For the motorcycle data, the empirical density is plotted on figure B.8. It explains why the performance of the kernel smoother (e.g. figure B.10) is poor on the second half of the estimation.

A pleasant alternative is offered by *variable kernel* methods, which are a hybrid between kernel smooth and k-NN. The idea is to locally modify the size of the kernel in order to keep a fair number of data in the estimation. Let us recall the set

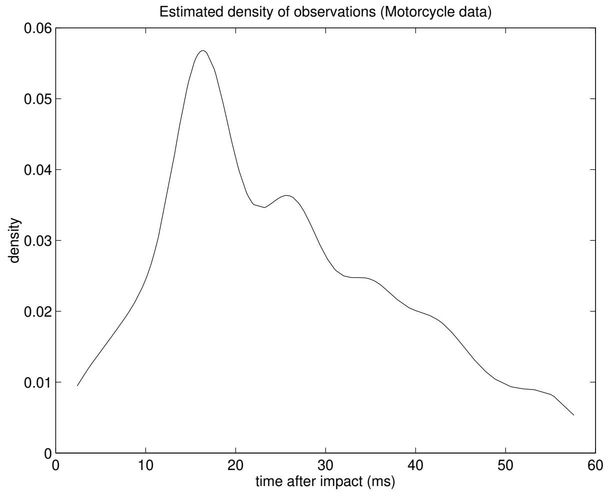


Figure B.8: Smoothed estimation of the density of the motorcycle data. The density is pretty high in the range 10–30 but decreases greatly afterwards.

$V_k(x)$ of the k nearest neighbours of x. The weight sequence is given by:

$$W_{ki}(x) = K_{\hat{d}(x)}\left(x - x^{(i)}\right) \tag{B.8}$$

where the setting of $\hat{d}(x)$ depends on the neighbourhood $V_k(x)$. A popular (and convenient) setting is:

$$\hat{d}(x) = \frac{\alpha}{k} \sum_{l \in V_k(x)} \left\| x - x^{(l)} \right\| \tag{B.9}$$

where α is a constant deciding on the ratio between the kernel size and the average distance of the k nearest neighbours to the current estimation point. In this setting, two types of optimisation are possible:

1. Setting α to a "meaningful" value, such as 1, and optimising the number k of neighbours. $\alpha = 1$ means that the average neighbour will be situated at one standard deviation, i.e. around the steepest slope of the Gaussian.

2. Optimising α with k fixed. k can be taken as a portion of the total number of data, e.g. half or a quarter.

In the first case, we perform a discrete optimisation, in a manner similar to the tuning of the number of neighbours in k-NN. In the second case, we perform a continuous optimisation. Furthermore, it is easy to derive the expression of the derivative of the LOO error (B.7) with respect to α:

$$\frac{\partial E_{LOO}}{\partial \alpha} = \frac{2}{N} \sum_{j=1}^{N} \left(\hat{y}_{-j} - y^{(j)}\right) \frac{\sum\limits_{i \neq j} \left(y^{(i)} - \hat{y}_{-j}\right) \dfrac{\partial K_{\hat{d}}\left(x^{(j)} - x^{(i)}\right)}{\partial \alpha}}{\sum\limits_{i \neq j} K_{\hat{d}}\left(x^{(j)} - x^{(i)}\right)} \tag{B.10}$$

where it should be noted that the the sum in the numerator in (B.10) is calculated during the evaluation of $\hat{y}_{-j}$. The derivative of the kernel weight with respect to α for a Gaussian Kernel is:

$$\frac{\partial K_{\hat{d}}\left(x^{(j)}-x^{(i)}\right)}{\partial \alpha} = \frac{\left\|x^{(j)}-x^{(i)}\right\|^2}{\alpha \hat{d}^2} K_{\hat{d}}\left(x^{(j)}-x^{(i)}\right) \tag{B.11}$$

B.11 Variable metric method

The estimation is also greatly improved by using a *variable metric* scheme. Both k-NN and kernel smoother rely on a metric to estimate either the proximity of a point, or its contribution to the smoothing. Let us define a generic metric based on a matrix $\mathbf{D}$. The squared distance between the data point $x^{(i)}$ and the estimation point x is:

$$\left\|x^{(i)}-x\right\|_{\mathbf{D}}^2 = \left(x^{(i)}-x\right)^\top \mathbf{D} \left(x^{(i)}-x\right) \tag{B.12}$$

If $\mathbf{D} = \mathbf{I}_P$ the identity matrix, we have the Euclidian distance. When $\mathbf{D}$ is the diagonal matrix of inverse variances on each component of the input space, we have a *scaling distance*. The distance is important as some components of the input might typically be less important (regardless of their variance) than others. E.g. if an input is just random noise, it will nevertheless influence the distance calculation, with possibly harmful side-effects.

The variable metric method parameterises the diagonal elements of the distance matrix:

$$\mathbf{D} = \begin{bmatrix} \mu_1^2 & 0 & \dots & 0 \\ 0 & \mu_2^2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \mu_P^2 \end{bmatrix} \tag{B.13}$$

and optimises the P distance coefficients[1] over the LOO error. This is possible as the derivative of the LOO generalisation estimate with respect to each distance coefficient is easily calculated:

$$\frac{\partial E_{LOO}}{\partial \mu_k} = \frac{2}{N} \sum_{j=1}^{N} \left(\hat{y}_{-j} - y^{(j)}\right) \frac{\sum_{i \neq j} \left(y^{(i)} - \hat{y}_{-j}\right) \frac{\partial K_d\left(x^{(j)}-x^{(i)}\right)}{\partial \mu_k}}{\sum_{i \neq j} K_d\left(x^{(j)}-x^{(i)}\right)} \tag{B.14}$$

The derivation of the weight sequence with respect to the distance coefficient depends on whether the kernel size is fixed or variable as in section B.10. For a fixed setting:

$$\frac{\partial K_d\left(x^{(j)}-x^{(i)}\right)}{\partial \mu_k} = -\frac{\mu_k \left(x_k^{(j)}-x_k^{(i)}\right)^2}{d^2} K_d\left(x^{(j)}-x^{(i)}\right) \tag{B.15}$$

[1] Many other forms are possible for the distance coefficients. This one is especially convenient as it insures both that the coefficient will stay positive, and that it can be driven to 0 if necessary.

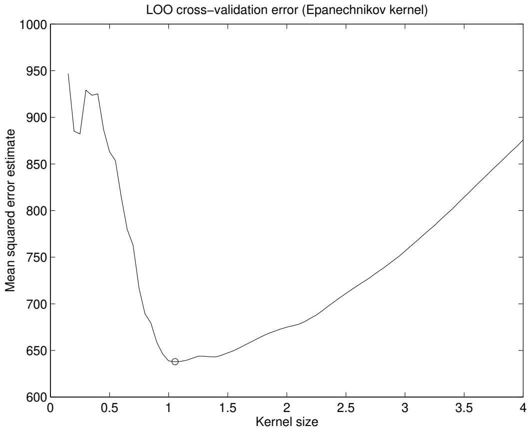


Figure B.9: LOO estimate as a function of the kernel size, with 10% of the data left out. The minimum is obtained for $d = 1.0545$.

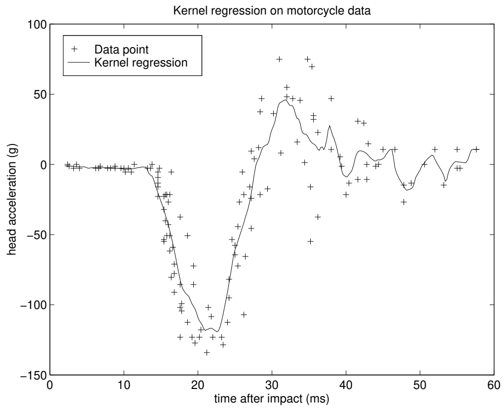


Figure B.10: LOO-optimal regression curve obtained using an Epanechnikov kernel of size 1.0545.

In the case of a variable kernel given by (B.9), the derivative turns into:

$$\frac{\partial K_{\hat{d}}}{\partial \mu_k} = \left[\frac{\alpha \left\| x^{(j)} - x^{(i)} \right\|_{\mathbf{D}}^2}{k\hat{d}} \sum_{l \in V_k^{(j)}} \left(\frac{\left(x_k^{(j)} - x_k^{(l)} \right)^2}{\left\| x^{(j)} - x^{(l)} \right\|_{\mathbf{D}}} \right) - \left(x_k^{(j)} - x_k^{(i)} \right)^2 \right] \frac{\mu_k K_{\hat{d}}}{\hat{d}^2} \quad \text{(B.16)}$$

where $V_k^{(j)}$ is $V_k\left(x^{(j)}\right)$.

B.12 Tuning by minimisation

The smoothing parameter is similar in spirit in both k-NN and kernel smoothing. However, it takes discrete values in the case of k-NN, and continuous values for the kernel smoother. The LOO is thus a real function of a discrete or continuous variable.

For a single smoothing parameter, the parameter tuning is a minimisation problem that can be solved by the usual line minimisation algorithms. Furthermore, the error curve is usually rather well-behaved: high and low values of the smoothing parameter produce high errors, and their are usually few local minima. We advocate the use of a quadratic search or a golden search in order to obtain the optimal value of d.

For (fixed-size) kernel smoothing of the motorcycle data, it takes 13 iterations of parabolic search to obtain a 4-digit approximation of the kernel size that minimises the leave-one-out error with an Epanechnikov kernel: $d_{opt} = 1.0545$. This LOO estimate was computed with 5% of the data left out at each end of the data set. This parameter selection takes around 2 seconds on a HP 9000/735.

In some cases, there is more than one parameter to optimise. This is noticeably the case for multi-dimensional input using a variable metric as explained in section B.11.

In such a case, the number of parameters to optimise is equal to the input dimension. Notice though that the LOO error is easily differentiated, so we are back to a non-linear optimisation problem (cf. Chapter 1).

B.13 GRNN: a non-parametric connectionist model

Kernel estimators have been introduced to the neural network literature under the name General Regression Neural Network or GRNN. The GRNN is a straightforward application of the Nadaray-Watson estimator using Gaussian weights.

It can be seen as a neural network-oriented implementation of a kernel estimator, basically in the form of a 3-layer network. These three layers have the following roles:

1. Computation of the kernel value $K_d\left(x - x^{(i)}\right)$ for each $x^{(i)}$. This involves a non linear transformation depending on the kernel shape (*pattern units*).
2. Calculation of the numerator of (B.5) and the Parzen density estimate (*summation unit*).
3. Final estimation (*output unit*).

Putting this non-parametric estimator in network form allows for a flattering comparison. As it reflects a non-parametric method, no parameter estimation is necessary, as opposed to a standard MLP requiring a sometimes lengthy learning procedure. On the other hand, the estimation itself is undoubtedly more costly to perform. In order to overcome this problem, a clustering version of the GRNN is proposed. This clustering is similar to the binned version of the kernel smoothing, together with the weighted average of rounded points, also known as WARP. Furthermore, the leave-one-out cross-validation scheme is proposed under the name "holdout method" in order to tune the smoothing parameter.

The interesting aspect of the GRNN is that it introduced the neural network community to classical non-parametric estimators in a relatively statistics-free manner. It seems, though, that one might as well go back to the sources of the technique. The non-parametric literature contains far more than the GRNN can offer. Even by limiting our scope to kernel smoothing, many techniques allow for interesting insight into aspects such as model convergence, confidence intervals on the estimation or smoothing parameter selection, among others.

COMMENTS

B.1 Application of non-parametric methods, namely k-NN and kernel smoothers, are discussed and compared to parametric regression in e.g. chapter 4 and Goutte (1996, 1997).

B.2 K nearest neighbours estimators are discussed in many forms in many sources. Härdle (1990) gives a good introduction and provides an interesting discussion of computational aspects, including update formulas in the sorted, one-dimensional case.

B.3 The bias and variance decomposition for the k nearest neighbours estimation is due to Lai (1977), cited by Härdle (1990).

B.4 The K nearest neighbours algorithm is implemented in the KTools package discussed in the appendix.

B.5 The Parzen kernel density estimator is due to Parzen (1962). A thorough presentation of kernel smoothers is given by Härdle (1990).

B.6 Proposition B.2 can be found in Mack (1981). The formulation given there encompasses all kernel shapes, but we felt compelled to limit ourselves to one kernel type for the sake of clarity.

B.4 The kernel smoothing algorithm is implemented in the KTools package discussed in the appendix.

B.8 Selection of the smoothing parameter and cross-validation have been pioneered by Grace Wahba. The leave-one-out and cross-validation methods are discussed here in sections 3.3 and 3.4. The estimation technique used in the context of this chapter is different, but the idea is entirely similar.

B.9 The leave-one-out error estimates are implemented for k-NN and kernel smoothers (with various kernels) in the KTools package discussed in the appendix. Both basic and improved versions are available.

B.11 A variable kernel method based on a parameterised metric was proposed by Lowe (1995), who uses a conjugate gradient method to optimise the parameters.

B.12 Tuning by minimisation uses the optimisation techniques reviewed in chapter 1, both for one-dimensional, and multi-dimensional, first-order methods.

B.13 The term GRNN was coined by Specht (1991). It is interesting to note that the same author was involved with non-parametric estimation in the sixties before applying the method to neural models. The Parzen kernel density estimator is due to Parzen (1962).

References

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. (1997). Lag space estimation in time series modelling. In *Proceedings of ICASSP'97*. IEEE.

Härdle, W. (1990). *Applied nonparametric regression*. Number 19 in Econometric Society Monographs. Cambridge University Press.

Lai, S. L. (1977). *Large sample properties of k-nearest neighbor procedures.* PhD thesis, Department of Mathematics, UCLA, Los Angeles.

Lowe, D. (1995). Similarity metric learning for a variable-kernel classifier. *NC*, 7(1):72–85.

Mack, Y. P. (1981). Local properties of k-nn regression estimates. *SIAM Journal on Algorithm and Discrete Methods*, 2:311–323.

Parzen, E. (1962). On estimation of a probability density and mode. *Annals of Mathematical Statistics*, 33:1065–76.

Specht, D. F. (1991). A general regression neural network. *IEEE Transactions on Neural Networks*, 2(6):568–576.

Bibliography

Akaike, H. (1969). Fitting autoregressive models for prediction. *Annals of the Institute of Statistical Mathematics*, 21:243–247.

Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723.

Aleksić, Z. (1991). Estimating the embedding dimension. *Physica D*, 52:362–368.

Amari, S. I. (1967). A theory of adaptive pattern classifiers. *IEEE Transactions on Electronic Computers*, 16:279–307.

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

Battiti, R. (1992). First- and second-order methods for learning: Between steepest descent and newton's method. *Neural Computation*, 4(2):141–166.

Bishop, C. M. (1995a). *Neural Networks for pattern recognition*. Clarendon Press, Oxford.

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

Bottou, L. (1991). *Une approche théorique de l'apprentissage connexionniste : application à la reconnaissance de la parole*. Thèse de doctorat, Université Paris XI, Orsay.

Bottou, L. and Le Cun, Y. (1988). SN: A simulator for connectionist models. In *NeuroNîmes 88*, pages 371–382, Nîmes, France.

Bryson, A., Denham, W., and Dreyfuss, S. (1963). Optimal programming problem with inequality constraints. I: Necessary conditions for extremal solutions. *AIAA journal*, 1:25–44.

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

Cibas, T., Fogelman Soulié, F., Gallinari, P., and Raudys, S. (1994). Variable selection with Optimal Cell Damage. In *Proceedings of ICANN'94*, pages 727–730.

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.

Efron, B. (1986). Why isn't everyone a bayesian? *The American Statistician*, 40(1):1–11. with comments.

Efron, B. E. (1982). *The Jacknife, the Bootstrap and Other Resampling plans*, volume 38 of *CBMS-NSF Regional Conference Series in Applied Mathematics.* SIAM.

Fletcher, R. (1987). *Practical Methods of Optimization.* Wiley.

Friedman, J. H. (1996). On bias, variance, 0/1 - loss, and the curse-of-dimensionality. Technical report, Department of Statistics, Stanford University. ftp://playfair.stanford.edu/pub/friedman/curse.ps.Z.

Geman, S., Bienenstock, E., and Doursat, R. (1992). Neural networks and the bias/variance dilemna. *Neural Computation*, 4(1):1–58.

Goutte, C. (1996). On the use of a pruning prior for neural networks. In NNSP96 (1996), pages 52–61.

Goutte, C. (1997a). Extracting the relevant delays in time series modelling. In *Neural Networks for Signal Processing VII – Proceedings of the 1997 IEEE Workshop*, number VII in NNSP, Piscataway, New Jersey. IEEE.

Goutte, C. (1997b). Lag space estimation in time series modelling. In *Proceedings of ICASSP'97.* IEEE.

Goutte, C. (1997c). Note on free lunches and cross-validation. *Neural Computation*, 9(6). to appear.

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

Goutte, C. and Ledoux, C. (1995). Synthèse des techniques de commande connexionniste. Technical Report 95/02, LAFORIA.

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, L. K. and Larsen, J. (1996). Linear unlearning for cross-validation. *Advances in Computational Mathematics*, 5(2,3):269–280.

Hansen, L. K. and Rasmussen, C. E. (1994). Pruning from adaptive regularization. *Neural Computation*, 6:1223–1232.

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

Härdle, W. (1990). *Applied nonparametric regression.* Number 19 in Econometric Society Monographs. Cambridge University Press.

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.

He, X. and Asada, H. (1993). A new method for identifying orders of input-output models for nonlinear dynamic systems. In *American Conference on Control*, San Francisco, California.

Hertz, J., Krogh, A., and Palmer, R. G. (1991). *Introduction to the theory of neural computation*. Addison-Wesley.

Hocking, R. R. (1976). The analysis and selection of variables in linear regression. *Biometrics*, 32:1–49.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–368.

Jaynes, E. T. (1985). Bayesian methods: general background. In Justice, J., editor, *Maximum Entropy and Bayesian Methods in Applied Statistics*, pages 1–25. Cambridge University Press.

Kouam, A. (1993). *Approches connexionnistes pour la prévision des séries temporelles*. PhD thesis, Université de Paris Sud.

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*.

Lai, S. L. (1977). *Large sample properties of k-nearest neighbor procedures*. PhD thesis, Department of Mathematics, UCLA, Los Angeles.

Larsen, J. and Hansen, L. K. (1994). Generalized performance of regularized neural networks models. In Vlontzos, J., Hwang, J. N., and Wilson, E., editors, *Neural Networks for Signal Processing IV – Proceedings of the 1994 IEEE Workshop*, number IV in NNSP, pages 42–51, Piscataway, New Jersey. IEEE.

Larsen, J. and Hansen, L. K. (1995). Empirical generalization assessment of neural network models. In Girosi, F., editor, *Neural Networks for Signal Processing V – Proceedings of the 1995 IEEE Workshop*, number V in NNSP, pages 42–51, Piscataway, New Jersey. IEEE.

Larsen, J., Hansen, L. K., Svarer, C., and Ohlsson, M. (1996). Design and regularization of neural networks: the optimal use of a validation set. In NNSP96 (1996), pages 62–71.

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.

Leray, P. (1996). La sélection de variables. Technical report, Laforia.

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.

Lowe, D. (1995). Similarity metric learning for a variable-kernel classifier. *NC*, 7(1):72–85.

Mack, Y. P. (1981). Local properties of k-nn regression estimates. *SIAM Journal on Algorithm and Discrete Methods*, 2:311–323.

MacKay, D. (1992a). Bayesian interpolation. *Neural Computation*, 4:415–447.

MacKay, D. (1992b). A practical bayesian framework for backprop networks. *Neural Computation*, 4:448–472.

MacKay, D. (1993). Hyperparameters: Optimize or integrate out? In Heidbreder, G., editor, *Maximum entropy and Bayesian Methods*. Kluwer, Dordrecht.

Mallows, C. (1973). Some comments on c_p. *Technometrics*, 15:661–675.

McLeod, A. I. (1994). Diagnostic checking of periodic autoregression models with application. *Journal of Time Series Analysis*, 15(2):221–233.

Molina, C., Sampson, N., Fitzgerald, W. J., and Niranjan, M. (1996). Geometrical techniques for finding the embedding dimension of time series. In NNSP96 (1996), pages 161–169.

Møller, M. (1993a). *Efficient Training of Feed-Forward Neural Networks*. PhD thesis, Computer Science department, Aarhus University.

Møller, M. (1993b). A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, 6(4):525–533.

Møller, M. (1993c). Supervised learning on large redundant training sets. *International Journal of Neural Systems*, 4(1):15–25.

Moody, J. (1991). Note on generalization, regularization and architecture selection in nonlinear learning systems. In Juang, B. H., Kung, S. Y., and Kamm, C. A., editors, *Proceedings of the first IEEE Workshop on Neural Networks for Signal Processing*, number I in NNSP, pages 1–10, Piscataway, New Jersey. IEEE.

Neal, R. M. (1992). Bayesian training of backpropagation network by the hybrid monte carlo method. Technical Report CRG-TR-92-1, Connectionist Research Group, Department of Computer Science, University of Toronto.

NNSP96 (1996). *Neural Networks for Signal Processing VI – Proceedings of the 1996 IEEE Workshop*, number VI in NNSP, Piscataway, New Jersey. IEEE.

Nørgaard, P. M. (1996). *System identification and control with neural networks*. PhD thesis, Department of Automation, Technical University of Denmark.

Parzen, E. (1962). On estimation of a probability density and mode. *Annals of Mathematical Statistics*, 33:1065–76.

Pi, H. and Peterson, C. (1994). Finding the embedding dimension and variable dependences in time series. *Neural Computation*, 6(3):509–520.

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.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (1992). *Numerical Recipes in C.* Cambridge University Press, 2nd edition.

Rasmussen, C. E. (1993). Generalization in neural networks. Master's thesis, Electronics institute, Technical University of Denmark.

Ripley, B. D. (1996). *Pattern Recognition and Neural Networks.* Cambridge University Press.

Robbins, H. and Munro, S. (1951). A stochastic approximation method. *Annals of Math. Stat.*, 22:400–407.

Rumelhart, D., Hinton, G., and Williams, R. (1986). Learning internal representation by error propagation. In Rumelhart, D. and McClellan, J., editors, *Parallel Distributed Processing : exploring the microstructure of cognition*, volume 1, pages 318–362. MIT Press.

Savit, R. and Green, M. (1991). Time series and dependent variables. *Physica D*, 50(1):95–116.

Scales, J. A. and Smith, M. L. (1994). *Introductory Geophysical Inverse Theory.* Samizdat Press, available via FTP form hilbert.mines.colorado.edu.

Schwartz, G. (1978). Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464.

Sørensen, P. H., Nørgård, M., Hansen, L. K., and Larsen, J. (1996). Cross-validation with luloo. In *Proceedings of 1996 International Conference on Neural Information Processing, ICONIP'96.*

Specht, D. F. (1991). A general regression neural network. *IEEE Transactions on Neural Networks*, 2(6):568–576.

Svarer, C., Hansen, L., and Larsen, J. (1993). On design and evaluation of tapped-delay neural network architectures. In et al., H. B., editor, *IEEE International Conference on Neural Networks*, ICNN, pages 46–51, Piscataway, NJ. IEEE.

Thodberg, H. H. (1993). Ace of Bayes: application of neural network with pruning. Technical Report 1132-E, Danish meat research institute, Roskilde, Danmark.

Thodberg, H. H. (1996). A review of bayesian neural networks with an application to near infrared spectroscopy. *IEEE Transactions on Neural Networks*, 7(1):56–72.

Tibshirani, R. (1996). Bias, variance and prediction error for classification rules. Technical report, University of Toronto. http://utstat.toronto.edu/pub/tibs/biasvar.ps.

Tsypkin, Y. Z. and Nikolic, Z. J. (1971). *Adaptation and learning in automatic systems*, volume 73 of *Mathematics in science and engineering*. Academic Press, New York and London.

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

Weigend, A. S., Huberman, B. A., and Rumelhart, D. E. (1990). Predicting the future: a connectionist approach. *International Journal of Neural Systems*, 1(3):193–210.

Werbos, P. J. (1974). *Beyond regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, Harvard University, Cambridge, MA.

White, H. (1989). Learning in artificial neural networks: A statistical perspective. *Neural Computation*, 1:425–464.

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

Wolpert, D. H. (1993). On the use of evidence in neural networks. In Hanson, S. J., Cowan, J. D., and Giles, C. L., editors, *Advances in Neural Information Processing Systems*, number 5 in NIPS, pages 539–546. Morgan Kaufmann.

Wolpert, D. H. (1995). What Bayes has to say about the evidence procedure. In Heidbreder, G., editor, *1993 Maximum Entropy and Bayesian Methods Conference*. Kluwer.

Wolpert, D. H. (1996). The lack of a priori distinctions between learning algorithms. *Neural Computation*, 8(8):1341–1390.

Wolpert, D. H. and Macready, W. G. (1995). The mathematics of search. Technical Report SFI-TR-95-02-010, Santa Fe Institute.

Yule, G. U. (1927). On a method of investigating periodicities in disturbed series with special reference to Wolfer's sunspot numbers. *Philos. Trans. R. Soc. Lond. Ser. A*, 226:267.

Zhu, H. and Rohwer, R. (1996). No free lunch for cross validation. *Neural Computation*, 8(7):1421–1426.

Index