Boosting, bagging and bragging applied to nonparametric regression - an empirical approach

Lusilda Boshoff, Hons.B.Sc.

Dissertation submitted in partial fulfilment of the requirements for the degree Master of Science in Statistics at the North-West University (Potchefstroom Campus)

Supervisor: Prof. C.J. Swanepoel
Co-supervisor: Prof. J.W.H. Swanepoel

December 2009
Potchefstroom

Boosting, bagging and bragging applied to nonparametric regression – an empirical approach

Abstract

The purpose of this study is to determine the effect of improvement methods such as boosting, bagging, bragging (a variation of bagging), as well as combinations of these methods, on nonparametric kernel regression. The improvement methods are applied to the Nadaraya-Watson (N-W) kernel regression estimator, where the bandwidth is tuned by minimizing the cross-validation function. It is known that the N-W estimator is associated with variance related drawbacks.

Marzio and Taylor (2008), Hall and Robinson (2009) and Swanepoel (1988, 1990) introduced boosting, bagging and bragging methods to the field of kernel regression. In the current study combinations of boosting, bagging and bragging methods are explored to determine the effect of the methods on the variability of the N-W regression estimator. A variety of methods are utilized to determine the bandwidth, by minimizing the cross-validation function. The different resulting regression estimates are evaluated by minimizing the global MISE discrepancy measure.

Boosting is a general method for improving the accuracy of any given learning algorithm and has its roots in machine learning. However, due to various authors' contributions to the development of the methodology and theory of boosting, its applications expanded to a wide range of fields. For example, boosting has been shown in the literature to improve the Nadaraya-Watson learning algorithm.

Bagging, an acronym for **b**ootstrap **agg**regat**ing**, is a method involving the generation of multiple versions of a predictor. These replicates are used to get an aggregated estimator. In the regression setting, the aggregation calculates an average over multiple versions which are obtained by applying the bootstrap principle, i.e. by drawing bootstrap samples from the original training set and using these bootstrap samples as new training sets (Swanepoel 1988, 1990, Breiman 1996a). We also apply some modifications of the method such as bragging where, instead of the average, a robust estimator is calculated by using the bootstrap samples.

Boosting, bagging and bragging methods can be seen as ensemble methods. Ensemble methods train multiple component learners and then combine their predictions. The generalization ability of an ensemble is often significantly better than that of a single learner. Results and conclusions verifying existing literature are provided, as well as new results for

the new methods.

REFERENCES

Breiman, L. (1996a). Bagging predictors, *Machine Learning* **24**: 123–140.

Hall, P. and Robinson, A. P. (2009). Reducing variability of crossvalidation for smoothing-parameter choice, *Biometrika* **96**(1): 175–186.

Marzio, M. D. and Taylor, C. C. (2008). On boosting kernel regression, *Journal of statistical planning and inference* **138**: 2483–2498.

Swanepoel, J. W. H. (1988). Point estimation based on approximating functionals and the bootstrap, *Technical report*, Dept. of Statistics, Potchefstroom University, South Africa.

Swanepoel, J. W. H. (1990). A review of bootstrap methods, *South African Statistical Journal* **24**: 1–34.

"Boosting", "bagging" en "bragging" toegepas op nieparametriese regressie – 'n empiriese benadering

Uittreksel

Hierdie studie se doel is om te bepaal wat die invloed van sogenaamde versterkingsmetodes soos "boosting", "bagging", "bragging" ('n variasie van "bagging"), asook kombinasies van hierdie metodes, op nieparametriese kernregressie is. Die versterkingsmetodes word toegepas op die Nadaraya-Watson (N-W) kernregressieberamer, waar die bandwydte bepaal word deur minimisering van die kruisgeldigheidsbepalingsfunksie. Dit is bekend dat die N-W beramer geassosieer word met variansie-verwante probleme.

Marzio en Taylor (2008), Hall en Robinson (2009) en Swanepoel (1988, 1990) het "boosting", "bagging" en "bragging" metodes in die gebied van kernregressie ontwikkel en aangewend. In die huidige studie word kombinasies van "boosting", "bagging" en "bragging" metodes ondersoek om vas te stel wat die effek van die metodes op die varieerbaarheid van die N-W regressieberamer is. 'n Verskeidenheid metodes word ondersoek om die bandwydte vas te stel, deur minimisering van die kruisgeldigheidsbepalingsfunksie. Die verskillende resulterende regressieberamers word geëvalueer deur minimisering van die globale "MISE" afstandsmaat.

"Boosting" is 'n algemene metode om die akkuraatheid van enige leeralgoritme te verbeter en het sy wortels in masjienleringsprosedures. Verskeie navorsers het bydraes gelewer in die ontwikkeling van die metodologie en teorie van "boosting" en die toepassingsveld is uitgebrei na 'n wye reeks gebiede. In die literatuur is byvoorbeeld aangetoon dat "boosting" die Nadaraya-Watson leeralgoritme versterk.

"Bagging" is 'n akroniem vir "**b**ootstrap **agg**regat**ing**". Dit is 'n metode wat die generering van veelvuldige weergawes van 'n voorspeller behels. Hierdie veelvuldige weergawes word gebruik om 'n gemiddelde beramer te skep. In die regressie-opset word 'n gemiddelde oor veelvuldige weergawes bereken. Die veelvuldige weergawes word verkry deur die skoenlusmetode toe te pas, d.w.s. deur skoenlussteekproewe uit die oorspronklike oefensteekproef te trek en hierdie skoenlussteekproewe as nuwe oefensteekproewe te gebruik (Swanepoel 1988, 1990, Breiman 1996a). Wysigings van die metode, soos die "bragging"-metode waar 'n robuuste beramer in plaas van die gemiddelde bereken word, word ook toegepas.

"Boosting", "bagging" en "bragging" metodes kan gesien word as geheelmetodes ("ensemble methods"). Hierdie metodes oefen meervoudige-komponent leerders en kombineer dan hulle voorspellings. Die veralgemeningsvermoë van geheelmetodes is dikwels beduidend beter as dié van 'n enkelleerder.

Resultate en gevolgtrekkings wat bestaande literatuur bevestig, word getoon, asook nuwe resultate vir die nuwe metodes.

BRONNELYS

Breiman, L. (1996a). Bagging predictors, *Machine Learning* **24**: 123–140.

Hall, P. and Robinson, A. P. (2009). Reducing variability of crossvalidation for smoothing-parameter choice, *Biometrika* **96**(1): 175–186.

Marzio, M. D. and Taylor, C. C. (2008). On boosting kernel regression, *Journal of statistical planning and inference* **138**: 2483–2498.

Swanepoel, J. W. H. (1988). Point estimation based on approximating functionals and the bootstrap, *Technical report*, Dept. of Statistics, Potchefstroom University, South Africa.

Swanepoel, J. W. H. (1990). A review of bootstrap methods, *South African Statistical Journal* **24**: 1–34.

Preface

Overview

In regression analysis the relationship between two or more quantitative variables is studied. For example, suppose n data points $\{(X_i, Y_i)\}_{i=1}^n$ have been collected, where both the predictor and response variables are one-dimensional. The regression relationship can be formulated as

$$Y_i = m(X_i) + \varepsilon_i, \ i = 1, \ldots, n,$$

where

$$m(x) = E(Y|X = x)$$

is the unknown regression function and the ε_i's are independent random variables with mean 0 and variance σ^2, usually referred to as *errors*. The aim is to estimate the unknown regression function $m(x)$, using available data (Kutner, Li, Nachtsheim and Neter 2005, Härdle 1990).

The estimation of $m(x)$ can be done in two ways: parametrically or nonparametrically. The *parametric* approach assumes that $m(x) \equiv m(x; \boldsymbol{\beta})$ has a known functional form that depends on unknown parameters $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)$, $0 < p < \infty$. Using an appropriate estimation method, it is possible to utilize the data to estimate the parameters $\beta_1, \ldots, \beta_p$ and thereby obtain an estimate of m.

Nonparametric regression estimation on the other hand does not assume a functional form, but is rather concerned with qualitative properties of m. These methods focus on deriving useful trends from the data, rather than on the reduction of parameters (Eubank 1988, p. 2,3). Kernel regression estimation and related methods such as k-nearest neighbour regression estimation (see for example Härdle (1990)) are popular nonparametric regression techniques. In particular, we consider in this study the well-known Nadaraya-Watson kernel

regression estimator (Nadaraya 1964, Watson 1964),

$$\hat{m}_{NW}(x) = \frac{\sum_{i=1}^{n} K_h(x - X_i)Y_i}{\sum_{i=1}^{n} K_h(x - X_i)},$$

where K_h denotes the a kernel function which is dependent on some bandwidth h. The choice of h is a matter of concern and various data-driven approaches to select h exist. In this study, the cross-validation methods of bandwidth selection are used.

Lately, three improvement methods are "hot topics" in statistical research, namely the boosting, bagging and bragging methods. Boosting (Schapire 1990, Freund 1995) and bagging (Breiman 1996a) are computationally intensive ensemble methods developed in the machine learning context, while bragging (Swanepoel 1990, Bühlmann 2003) is a robust form of bagging. In the machine learning context, ensemble methods involve that multiple component learners are trained and their predictions are combined to provide learners with significantly better generalization ability than the basic learners (Zhou and Yang 2005, p. 48). These methods have been extended to applications in statistics, such as the improvement of regression estimates. For example, new results of Marzio and Taylor (2008) improve the Nadaraya-Watson kernel regression estimate by using the boosting method, while Hall and Robinson (2009) apply the bagging method to cross-validation bandwidth selection in kernel regression estimation to produce better regression estimates in terms of reducing global discrepancy measures.

In the present study, the main concern is to explore the applicability of the boosting, bagging and bragging methods, as well as combinations of these methods, to the Nadaraya-Watson kernel regression estimator. The aim is to determine if boosting, bagging, bragging and combination methods will improve the Nadaraya-Watson kernel regression estimator and to quantify this improvement by means of simulation studies.

Objectives

The main objectives of this dissertation are as follows:

- to present a brief overview of basic literature on nonparametric regression methods.
- to introduce the main features and applications of the bootstrap method which underlies the bagging and bragging methods.
- to present an overview of important aspects of the boosting, bagging and bragging

methods, such as developmental aspects, limitations and previous applications of the methods in the regression context.

- to empirically determine and compare the performance of boosted, bagged and bragged Nadaraya-Watson estimation for a variety of simulation setup scenarios.
- to develop combination methods where the bagging and bragging methods are applied to cross-validation smoothing parameter selection in the boosted Nadaraya-Watson estimator and to study the performance of these combination methods empirically.
- to present algorithms for various ways in which the boosting, bagging, bragging and combination methods can be applied to the Nadaraya-Watson estimator.
- to illustrate the application of the new methods to real life examples.

Outline

A basic outline of this dissertation is now presented.

Chapter 1 provides a brief introduction to the main aspects of nonparametric smoothing methods.

In Chapter 2 the boosting methodology in general is explored and existing literature on the application of the boosting method to the regression context by means of L_2-boosting is considered.

The bagging and bragging improvement methods are based on bootstrap principles. Chapter 3 provides a brief introduction to some of the main features and applications of the bootstrap method.

Existing literature on aspects of the bagging and bragging methods that are relevant for this study are summarized in Chapter 4.

Chapter 5 presents definitions and algorithms of all methods applied in the simulation studies (i.e. methods from the existing literature and new methods).

In Chapter 6 results of the conducted Monte Carlo studies are discussed and conclusions are drawn.

Finally, two practical examples based on real-life data is presented in Chapter 7.

Tables and graphs with the results of the conducted simulation studies are presented in Appendices A, B and C.

Voorwoord

Oorsig

In regressie-analise word die verwantskap tussen twee of meer kwantitatiewe veranderlikes bestudeer. Byvoorbeeld, gestel n datapunte $\{(X_i, Y_i)\}_{i=1}^n$ is versamel, waar beide die voorspeller- en responsveranderlikes eendimensioneel is. Die regressie verwantskap kan geformuleer word as

$$Y_i = m(X_i) + \varepsilon_i, \ i = 1, \ldots, n,$$

waar

$$m(x) = E(Y|X = x)$$

die onbekende regressiefunksie is en die ε_i's, wat gewoonlik *foute* genoem word, onbekende stogastiese veranderlikes met gemiddeld 0 en variansie σ^2 is. Die doel is om die onbekende regressiefunksie $m(x)$ met behulp van beskikbare data te beraam (Kutner et al. 2005, Härdle 1990).

Die beraming van $m(x)$ kan op twee maniere gedoen word: parametries of nieparametries. Die *parametriese* benadering neem aan dat $m(x) \equiv m(x; \boldsymbol{\beta})$ 'n bekende funksionele vorm het wat van onbekende parameters $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)$, $0 < p < \infty$ afhang. Deur van 'n gepaste beramingsmetode gebruik te maak, kan die data gebruik word om die parameters $\beta_1, \ldots, \beta_p$ te beraam om sodoende 'n beramer vir m te verkry.

Nieparametriese regressie beraming aan die ander kant aanvaar nie 'n funksionele vorm vir m nie, maar fokus eerder op sy kwalitatiewe eienskappe. Hierdie metodes fokus daarop om nuttige tendense in die data vas te stel, eerder as op die afleiding van parameters (Eubank 1988, p. 2,3). Kernregressieberaming en verwante metodes soos k-naaste-punte regressieberaming (sien byvoorbeeld Härdle (1990)) is populêre nieparametriese regressie tegnieke. In die besonder bestudeer ons in hierdie studie die bekende Nadaraya-Watson

kernregressieberamer (Nadaraya 1964, Watson 1964),

$$\hat{m}_{NW}(x) = \frac{\sum_{i=1}^{n} K_h(x - X_i)Y_i}{\sum_{i=1}^{n} K_h(x - X_i)},$$

waar K_h 'n kernfunksie aantoon wat afhanklik is van 'n bandwydte h. Die keuse van h is die saak van belang en verskeie data-gedrewe metodes om h te kies, bestaan. In hierdie studie word kruisgeldigheidsmetodes gebruik om die bandwydte te kies.

Deesdae is drie versterkingsmetodes gewilde onderwerpe in statistiese navorsing, naamlik die "boosting", "bagging" en "bragging" metodes. "Boosting" (Schapire 1990, Freund 1995) en "bagging" (Breiman 1996a) is berekeningsintensiewe geheelmetodes ("ensemble methods") wat in die masjienleringskonteks ontwikkel is, terwyl "bragging" (Swanepoel 1990, Bühlmann 2003) 'n robuuste vorm van "bagging" is. In die masjienleringskonteks behels geheelmetodes dat meervoudige-komponentleerders geoefen word en hulle voorspellings dan gekombineer word om leerders met beter veralgemeningsvermoë as die oorspronklike leerders te vorm (Zhou and Yang 2005, p. 48). Hierdie metodes is uitgebrei na toepassings in statistiek, soos die versterking van regressieberamers. Byvoorbeeld, nuwe resultate van Marzio and Taylor (2008) verbeter die Nadaraya-Watson regressieberamer met behulp van die "boosting" metode, terwyl Hall and Robinson (2009) die "bagging" metode op kruisgeldigheidsmetodes om die bandwydte in kernregressie te bepaal, toepas, om beter regressieberamers in terme van die reduksie van globale afstandsmate te verkry.

In die huidige studie is die hoofsaak om die toepasbaarheid van die "boosting", "bagging" en "bragging" metodes, sowel as kombinasies van hierdie metodes, op die Nadaraya-Watson kernregressieberamer te bepaal. Die doel is om te bepaal of "boosting", "bagging", "bragging" en kombinasiemetodes die Nadaraya-Watson kernregressieberamer sal verbeter en om hierdie verbetering deur middel van simulasiestudies te kwantifiseer.

Doelwitte

Die belangrikste doelwitte van hierdie verhandeling is soos volg:

- om 'n breë oorsig oor die basiese literatuur aangaande nie-parametriese regressiemetodes te gee.
- om die belangrikste kenmerke en toepassings van die skoenlusmetode wat "bagging" en "bragging" onderlê, te bespreek.

- om 'n oorsig oor belangrike aspekte van die "boosting", "bagging" en "bragging" metodes te gee, soos aspekte rakende die ontwikkelings, beperkings en vorige toepassings van die metodes in die regressiekonteks.
- om die prestasie van die Nadaraya-Watson beramer na "boosting", "bagging" en "bragging" daarop toegepas is empiries te bepaal en te vergelyk, in verskeie simulasie-opset scenarios.
- om kombinasiemetodes te ontwikkel waar die "bagging" en "bragging" metodes toegepas word op die kruisgeldigheidsmetode van bandwydte seleksie in die "gebooste" Nadaraya-Watson beramer en om die prestasie van hierdie kominasiemetodes empiries te bestudeer.
- om algoritmes voor te stel vir die verskeie maniere waarop "boosting", "bagging" en "bragging" en kombinasiemetodes op die Nadaraya-Watson beramer toegepas kan word.
- om die toepassing van die nuwe metodes op werklike data te illustreer.

Uitleg

Die basiese uitleg van hierdie verhandeling word nou gegee.

Hoofstuk 1 verskaf 'n breë inleiding oor die belangrikste aspek van nieparametriese regressiemetodes.

In Hoofstuk 2 word die "boosting" metodologie in die algemeen verken en bestaande literatuur oor die toepassing van die "boosting" metode in die regressiekonteks deur middel van "L_2-boosting" word beskou.

Die "bagging" en "bragging" versterkingsmetodes is gebaseer op die beginsels van die skoenlusmetode. Hoofstuk 3 verskaf 'n breë inleiding tot die belangrike kenmerke en toepassings van die skoenlusmetode.

Bestaande literatuur oor aspekte van die "bagging" en "bragging" metodes wat relevant is vir hierdie studie word in Hoofstuk 4 bespreek.

Hoofstuk 5 gee die definisies en algoritmes van al die metodes wat in die simulasiestudies gebruik word (d.w.s. metodes uit die bestaande literatuur en die nuwe metodes).

In Hoofstuk 6 word die uitslae van die Monte Carlo studies bespreek en gevolgtrekkings word gemaak.

Ten slotte word twee praktiese voorbeelde wat op werklike data gebaseer is in Hoofstuk 7 getoon.

Tabelle en grafieke met die uitslae van die simulasiestudies word in Bylae A, B en C voorgestel.

Bedankings

Die outeur wil hiermee graag die volgende persone bedank:

- Prof. C.J. Swanepoel vir haar leiding, insig, entoesiasme en volgehoue motivering wat noodsaaklik was vir die voltooiing van hierdie studie.
- Prof. J.W.H. Swanepoel vir sy kundigheid, voorstelle en raad wat regdeur hierdie studie insig gegee het.
- My kollegas, Stefan Jansen van Vuuren, Leonard Santana en Gerhard Koekemoer, vir waardevolle gesprekke en hulp met programmering.
- My ouers, Willem en Elfriede Boshoff, vir opvoeding en soveel liefde, asook vir hulle volgehoue ondersteuning, motivering en belangstelling in hierdie projek.
- My broer en suster, Carel en Anneke, vir ondersteuning, liefde en vriendskap.

"Dit is nie aan die mens self te danke dat hy kan eet en drink en onder al sy arbeid nog die goeie kan geniet nie. Ek het ingesien dat dit 'n gawe uit die hand van God is." Vir die voorreg om hierdie taak te kon uitvoer, dank ek Hom wat wysheid en insig gee.

Contents

Chapter 1

Nonparametric regression estimators

1.1 Introduction

The aim of the present study is twofold. The first goal is to gain more insight into an improvement method that has its roots in the machine learning context, referred to as boosting. Specifically, the effect of boosting on the Nadaraya-Watson (N-W) estimator will be of interest. Secondly, the influence and effect of another nonparametric tool, i.e. the bootstrap method, are studied. In particular, the effect of bootstrap improvement methods on cross-validation smoothing parameter selection for the N-W estimator and the boosted N-W estimator is studied. The aim here is to evaluate the effect of two bootstrap-based methods, i.e. bagging and bragging, on the cross-validation choice of bandwidth in the N-W estimator, and of bandwidth, together with the choice of the number of boosting iterations, in the boosted N-W estimator. Before the boosting, bagging and bragging methods are discussed, the reader should be introduced to the regression estimator under consideration, the N-W estimator. The N-W estimator belongs to the class of nonparametric regression estimators. Härdle (1990) discusses a wide range of nonparametric regressions estimators. These include kernel, k-nearest neighbour, orthogonal series and spline smoothers. In particular, the N-W estimator, relevant for this study, is a kernel smoother.

The goal of this chapter is to present the reader with a summary of the main aspects of nonparametric smoothing methods.

The discussion starts in Section 1.2 with a brief review of regression methods, considering fundamental issues involved in parametric and nonparametric regression. Regression data could originate from either fixed or random design settings. Fixed and random data generation schemes are discussed in Section 1.3. Section 1.4 introduces kernel and nearest neighbour regression estimates. The performance of an estimator can be assessed via various

loss functions. Loss functions, also referred to as discrepancy measures, are discussed in Section 1.5, where estimators' mean squared error (MSE) and mean integrated squared error (MISE) properties are considered, with specific reference to the asymptotic properties of kernel and nearest neighbour estimators, as it appear in the literature. The importance of bandwidth selection methods receives attention in Section 1.6, where methods to choose the bandwidth in practice are presented from the literature. Summarizing remarks about the choice of the kernel function in kernel regression follow in Section 1.7, as well as brief discussions regarding boundary problems in Section 1.8. Remarks about derivative estimation will be made in Section 1.9, as well as extensions to the multivariate case in Section 1.10.

1.2 Regression methods: a brief review

Strict theories and precise methodologies, depending rigidly on fixed assumptions and rules, played a prominent role in the development of science in general and statistical science in particular. People tended to think that real data should be analysed using these methodologies. However, real data often do not fit into fixed frameworks and many such problems were left alone due to lack of ability to be moulded in a specific form. This tendency to avoid analysis of these cases hindered innovative developments in flexible thinking in such problems and the evolution of new methods for practical data analysis. However, the field of nonparametric regression continued to develop new techniques to meet the challenges of the forthcoming era. A variety of mathematically reliable techniques that are not governed by rigid forms such as linear curves or a normal distribution were developed. Nonparametric regression is a prominent tool in various settings such as the progress of computers and neural networks, data mining, modelling, pattern recognition and related subjects. Literature has expanded, as well as the number of software programs available for carrying out nonparametric regression on computers. Nonparametric regression is therefore no longer available to only a few specialists, but has become an indispensable tool for dealing with diverse problems concerning every day life regarding human beings, nature and society (Takewaza 2006).

In this chapter we consider basic aspects of regression analysis. The books of Kutner et al. (2005) and Härdle (1990) will be mainly used as references for this discussion.

Regression analysis in general is the statistical methodology that studies the relation between two or more quantitative variables. A regression curve, also referred to as a regression function or regression equation, describes a general relationship between a vector of explanatory variables $\mathbf{X}$ and possible response variables $\mathbf{Y}$. For simplicity we restrict our discussion

to the one-dimensional case, where one explanatory variable and one response variable is considered. Extensions to multivariate situations are possible for all methods discussed below. Knowledge of the relationship between X and Y reveals important information regarding monotonicity and location of special features such as extreme values. It may indicate a tendency of the response variable to vary with the predictor variable in a systematic fashion, or it may reveal whether the variables have a special dependence structure.

Suppose that n data points $\{(X_i, Y_i)\}_{i=1}^n$ have been collected. The regression relationship (function/equation) can be formulated as

$$Y_i = m(X_i) + \varepsilon_i, \ i = 1, \ldots, n, \tag{1.1}$$

where

$$m(x) = E[Y|X = x] \tag{1.2}$$

is the unknown regression function, i.e. the average value of Y given the observed value of X. The ε_i's are independent random variables (referred to as "errors") with mean 0 and variance σ^2. The sample of size n is used to obtain a useful estimate of the regression equation. The estimate $\hat{m}(x)$ is regarded useful if it tends to $m(x)$ as $n \to \infty$ when $m(x)$ is a smooth function and if $|Y_i - \hat{m}(X_i)| = |\hat{\varepsilon}_i|$, $i = 1, 2, \ldots, n$, are small values. The $\hat{\varepsilon}_i$'s are referred to as residuals. (Watson 1964, Takewaza 2006, p. 359).

We can approach the estimation of $m(x)$ in one of two ways: parametrically or nonparametrically. A parametric regression model assumes that the form of m is known except for finitely many unknown parameters. If, for example, it is assumed that $Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i$, a linear relationship is assumed with unknown intercept and slope. The intercept β_0 and slope β_1 are unknown regression coefficients, also called parameters. More generally, for parametric regression we have $m(x) = m(x; \boldsymbol{\beta})$, with $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)$, $0 < p < \infty$. Using an appropriate estimation methodology, it is possible to utilize the data to estimate the parameters $\beta_1, \ldots, \beta_p$ to obtain an estimate of m. The resulting estimate is a curve that has been selected from the family of curves allowed under the parametric model that conforms to the data in some fashion (Eubank 1988, p. 2). Furthermore, parametric regression favours expressions of m with as small a number of parameters as possible and selects a regression equation with a large number of parameters only when good representation of the data demands it.

However, it is often true that a scatterplot of the data does not show a simple functional form. Rather, a flexible functional form of the regression curve is suggested, without the

restrictions imposed by a parametric model.

The rigidity of parametric regression can be overcome by removing the restriction that m belongs to a parametric family. That is, one could use nonparametric regression (also called smoothing) techniques, where data-dependent methods are applied to estimate the regression function (Wand and Jones 1995, p. 3). A nonparametric regression model generally only assumes that m belongs to some infinite dimensional collection of functions, for example m may be assumed to be differentiable. For nonparametric modelling, assumptions are concerned only with qualitative properties of m, in contrast to the quite specific assumptions of parametric modelling (Eubank 1988, p. 3). Nonparametric regression focuses mainly on deriving useful trends from the data, rather than on the reduction of parameters.

The nonparametric regression approach has several goals. Firstly, it provides a versatile method of exploring the general relationship between two variables. Also, it gives predictions of future observations without referring to a fixed parametric model. Prediction of new observations is of particular interest in time series analysis. Nonparametric autoregression methods are applied to obtain such predictions. Nonparametric methods furthermore provide tools for finding and studying the influence of outliers and isolated observations. Literature shows that, in certain applications, classical parametric models are too restrictive to give acceptable explanations of the observed phenomena. Detection and treatment of outliers are important steps in featuring some aspects of a dataset. Extreme points affect the scale of plots, causing the main body of the data to become unnoticeable. Diagnostic methods in parametric models handle most outlier problems well, but some outliers and their influence remain hidden. However, nonparametric smoothing methods provide versatile screening methods for detecting outliers and diminishing their influence. Moreover, nonparametric regression methods provide flexible ways of substituting missing values by interpolating between adjacent X-values (Härdle 1990).

Various methods to obtain a nonparametric regression estimate of m exist. The simplest regression estimators are local versions of location estimators. In this study we consider smoothers consisting of local averages.

The guiding principle for these methods is that large weight is given to observations in a small neighbourhood around x and small or no weight is given to points far away from x. This procedure can be formulated by

$$\hat{m}(x) = \frac{1}{n}\sum_{i=1}^{n} W_{ni}(x)Y_i, \tag{1.3}$$

where $\{W_{ni}(x)\}_{i=1}^{n}$ denotes a sequence of weights which may depend on the whole vector $\{X_i\}_{i=1}^{n}$ (Härdle 1990, p. 16). In particular, we employ kernel functions to determine appropriate weights in this study.

Smoothers, by definition, average over observations. The averaging is controlled by the weight sequence $\{W_{ni}(x)\}_{i=1}^{n}$, which is tuned by a smoothing parameter. The smoothing parameter regulates the size of the neighbourhood around x. Too large a neighbourhood causes oversmooth curves and biased estimates of m, whereas too small a neighbourhood contributes to very rough, undersmoothed estimates for $\hat{m}$, with inflated variability. The resulting smoothing parameter selection problem that arises, stands central in nonparametric regression estimation (Härdle 1990, p. 18).

Outliers in the Y-values affect results obtained from the small number of observations in the neighbourhoods and new developed procedures strive towards ensuring that less weight is given to outliers. These methods are referred to as robust smoothers (Härdle 1990, p. 20).

1.3 The stochastic nature of the observations

Two possible scenarios for the origin of the data are now discussed (Härdle 1990, Chu and Marron 1991, p. 407), i.e. the fixed and random design settings. The data $\{(X_i, Y_i)\}_{i=1}^{n}$ can be generated from one of these two schemes.

1.3.1 Fixed design setting

Firstly, the fixed design model is concerned with controlled, nonstochastic X-variables. The model is given by

$$Y_i = m(x_i) + \varepsilon_i, \ i = 1, \ldots, n,$$

where the x_i's are nonrandom design points with $a \leq x_1 \leq \ldots \leq x_n \leq b$ and the ε_i's are independent random variables with mean 0 and variance σ^2. The x-values are usually chosen by the experimenter, as in a designed experiment. In many experiments the points are taken to be equidistributed on an interval $[a, b]$. Without loss of generality it can be assumed that $[a, b] = [0, 1]$ with $x_i = \frac{i}{n}$. Härdle (1990, p. 21) mentions as an example of a fixed design case a study of human growth curves, where a team of pediatricians determined the X-values well in advance. Experimental studies often result in fixed design settings.

1.3.2 Random design setting

Alternatively, the data can be generated by the random design model. In this setting the data points are thought of as being realizations from a bivariate probability distribution, where $\{(X_i, Y_i)\}_{i=1}^n$ are independent, identically distributed random variables. The model stated in (1.2) is well defined if $E(|Y|) < \infty$. If the joint density $f(x, y)$ exists, then $m(x)$ can be calculated as

$$m(x) = \frac{\int y f(x, y) dy}{f(x)}, \tag{1.4}$$

where $f(x) = \int f(x, y) dy$ denotes the marginal density of X. The error terms, i.e. the ε_i's, are defined by $\varepsilon_i = Y_i - m(X_i)$ and assumed to have mean 0 and variance σ^2. In this model the X-values are usually not chosen by the experimenter. Härdle (1990, p. 21) gives as an example of a random design case a sample of women drawn randomly from the population, where their heights and ages are observed and studied. Observational studies and sample surveys often result in random design settings.

One might think that there is little practical difference between the fixed and random design settings, because the regression function only depends on the conditional distribution, where the X-values are given. Even though this is true, the nature of the X-values greatly influences the performance of the estimators as we will see later (Chu and Marron 1991, p. 407).

1.4 Smoothing techniques

As far as regression methods are concerned, smoothing of a dataset $\{(X_i, Y_i)\}_{i=1}^n$ involves a specific way of approximation of the mean response curve m in the regression relationship (1.1). The topic of smoothing and smoothing techniques in general will be discussed in this section by referring to three well-known textbooks, namely Härdle (1990), Wand and Jones (1995) and Fan and Gijbels (1996). The functions to be smoothed may include the regression curve itself, derivatives of the regression curve, or functions of these derivatives. The basic idea suggests that, if m is believed to be smooth, the information of X_i near a point x should contain usable information about the value of m at the point x and should therefore be used to estimate $m(x)$ (Eubank 1988).

There exist several approaches to the nonparametric regression problem, for example those based on kernel functions, nearest neighbour functions, spline functions and orthogonal series. Within each of these broad classes there are a variety of approaches. This brief

literature overview will be limited to the Nadaraya-Watson (Nadaraya 1964, Watson 1964), Priestley-Chao (Priestley and Chao 1972) and Gasser-Müller (Gasser and Müller 1979) kernel regression estimators, nearest neighbour regression estimates and spline smoothing. Kernel and nearest neighbour estimates have the advantage of being mathematically and intuitively easy to understand and to implement. In our simulation studies we focus on the Nadaraya-Watson estimator in particular. We conclude this section with remarks on the treatment of outliers and an overview of local polynomial fitting.

1.4.1 Kernel regression smoothing

If one assumes the principle that data points close to a point x (in the case of a one-dimensional covariate space) carry more information about the value of $m(x)$ than data points located more remotely from x, then it makes sense to estimate the regression function by utilizing a method involving "locally weighted averages", as defined in (1.3). The shape and size of the weights $\{W_{ni}(x)\}_{i=1}^{n}$ in (1.3) are defined by a density function with a scale parameter that adjusts the form and size of the weights near x. Kernel regression smoothing utilizes a "kernel" function K as shape function. The size of the weights in kernel regression is parameterized by a scale parameter called the "bandwidth", which we denote by h.

More formally, denote the weight sequence for kernel smoothers by

$$W_{hi}(x) = \frac{K_h(x - X_i)}{\hat{f}_h(x)}, \tag{1.5}$$

where

$$K_h(u) = h^{-1}K(u/h)$$

is the kernel with scale factor h and $\hat{f}_h(x)$ denotes some estimate of the marginal density $f(x)$ of X in (1.4).

Wand and Jones (1995, p. 12) show how the kernel estimate is constructed for a small set of points by centering a scaled kernel at each observation. The value of the kernel estimate at the point x is simply the weighted average of the n kernel ordinates at that point.

Regarding kernel smoothing, the following three subsections will formalize the main concepts.

a) The kernel function

Kernel functions are usually symmetric, real-valued probability density functions that are continuous and bounded, with $\int K(u)du = 1$. A variety of different kernel functions could be used, such as the popular examples defined in Table 1.1.

Kernel	K(u)
Uniform	$\frac{1}{2}I(\lvert u\rvert \leq 1)$
Gaussian	$\frac{1}{\sqrt{2\pi}}e^{-u^2/2}$
Triangular	$(1-\lvert u\rvert)I(\lvert u\rvert \leq 1)$
Epanechnikov	$\frac{3}{4}(1-u^2)I(\lvert u\rvert \leq 1)$
Biweight	$\frac{15}{16}(1-u^2)^2I(\lvert u\rvert \leq 1)$
Triweight	$\frac{35}{32}(1-u^2)^3I(\lvert u\rvert \leq 1)$

Table 1.1: Examples of popular kernel functions

More generally, the symmetric Beta family of densities

$$K(t) = \frac{1}{\text{Beta}(1/2, \gamma+1)}(1-t^2)_+^\gamma, \qquad \gamma = 0, 1, \ldots,$$

where the subscript + denotes the positive part, leads to well-known kernel functions. The choices $\gamma = 0$, 1, 2 and 3 represent the uniform, Epanechnikov and the so-called "biweight" and "triweight" kernel functions respectively (Fan and Gijbels 1996, p. 15). In fact, this family includes most of the widely used kernel functions, also the Gaussian kernel in the limit as $\gamma \to \infty$ (Marron and Nolan 1988).

b) The bandwidth

The bandwidth h, also called a smoothing parameter, is a nonnegative number controlling the size of the local neighbourhood in the sense that it determines the size of the weights (Härdle 1990, p. 24). The shape of the smooth will greatly depend on the choice of h. If the bandwidth is chosen too small, the estimate will follow the data very closely, because the local average calculated in each point makes use of too few observations. The resulting

curve shows too much variability. On the other extreme, a bandwidth chosen too large will result in too smooth a curve, because observations situated far from x will also contribute to the local average. Important features of the underlying curve may not be represented by the smooth, which has low variance, but is highly biased. A trade-off between reducing the variance by increasing h and keeping the bias low by decreasing h, exists (Chu and Marron 1991, p. 405). This results in the so-called smoothing parameter selection problem, which is discussed in Section 1.6. Keep in mind that h depends on the sample size n and is often denoted by h_n. However, we keep to the notation present, i.e. h, for simplicity.

c) Three popular examples of kernel regression estimators

We now define three main kernel estimators for (1.4), where the shape of the kernel weights in each instance is determined by a kernel K and the size of the weights is parameterized by a bandwidth h.

- **The Nadaraya-Watson estimator**

  Nadaraya (1964) and Watson (1964) proposed what is known as the Nadaraya-Watson estimator,

  $$\hat{m}_{NW}(x) = \frac{\sum_{i=1}^{n} K_h(x - X_i)Y_i}{\sum_{i=1}^{n} K_h(x - X_i)}. \tag{1.6}$$

  For the Nadaraya-Watson estimator

  $$\hat{f}_h(x) = n^{-1} \sum_{i=1}^{n} K_h(x - X_i)$$

  in (1.5). This choice of $\hat{f}_h(x)$ is known as the Rosenblatt-Parzen kernel density estimator and was introduced by Rosenblatt (1956) and Parzen (1962) to estimate the marginal distribution of the X-values, $f(x)$, which is the denominator in (1.4). The numerator of $\hat{m}_{NW}(\cdot)$ is the analogous estimate of $\int y f(x, y)dy$, which is the numerator in (1.4). These choices for the denominator and numerator ensure that Nadaraya-Watson weights add up to one. The above definition is stated for the random design setting, but also holds for the fixed design setting where the X_i-values, $i = 1, 2, \ldots, n$, are fixed, controlled and nonstochastic.

- **The Priestley-Chao estimator**

  For the fixed design setting with equidistant x-values chosen within the interval $[0, 1]$, Priestley and Chao (1972) defined the following estimator:

$$\hat{m}_{PC}(x) = \frac{1}{n}\sum_{i=1}^{n} K_h(x - X_i)Y_i. \tag{1.7}$$

  For the random design model, with X-values chosen between 0 and 1, their estimator is given by:

$$\hat{m}_{PC}(x) = \sum_{i=1}^{n}(X_i - X_{i-1})K_h(x - X_i)Y_i, \tag{1.8}$$

  where $\{(X_i, Y_i)\}_{i=1}^n$ is assumed to be ordered by the X-values. For the Priestley-Chao estimator the weights do not necessarily add up to one.

- **The Gasser-Müller estimator**

  Gasser and Müller (1979) suggested a related estimator that modifies the Priestley-Chao estimate and which is similar to the estimator of Cheng and Lin (1981):

$$\hat{m}_{GM}(x) = \sum_{i=1}^{n} Y_i \int_{S_{i-1}}^{S_i} K_h(x - u)du, \tag{1.9}$$

  where $\{(X_i, Y_i)\}_{i=1}^n$ is assumed to be ordered by the X-values. The S_i's are interpolating the sequence of X_i's, i.e. $S_0 = -\infty$, $S_n = \infty$ and $X_i \leq S_i \leq X_{i+1}$, $i = 1, \ldots, n-1$ in the random design setting. A possible choice is $S_i = \dfrac{X_i + X_{i+1}}{2}$. The process is easily adapted for the fixed design setting.

  The choice $S_0 = -\infty$ and $S_n = \infty$ above ensure that the sum of the weights is one. This will create a strong boundary effect (discussed in more detail later), because near either end the observation at the end will receive a very large weight. Other choices for S_0 and S_n are $S_0 = 0$ and $S_n = 1$. This may however cause even greater boundary effects, because the weights near the edges do not sum to one, so instead of giving large weight to the outermost data point, the large weight is essentially given to the arbitrary value of zero (Chu and Marron 1991, p. 408).

  Literature reveals that S_i can be expressed as $S_i = \beta X_i + (1-\beta)X_{i+1}$, where $\beta \in [0, 1]$. Cheng and Lin (1981) investigate the choice of $\beta = 1$. Another popular choice is $\beta = 1/2$, which corresponds to the choice of S_i in the previous paragraph. Chu and Marron (1991, 408) argue that the practical difference between these two choices

is negligible for the fixed design and essentially equally spaced case (which includes designs that satisfy the asymptotic condition $x_i = i/n + o(n^{-1})$). However, for the random design setting the difference is quite large, with $\beta = 1/2$ being the better choice.

The Gasser-Müller and Priestley-Chao estimators are conveniently defined without the random denominator (see the definition of the Nadaraya-Watson estimator above). These definitions make both estimators easier to handle, for example when deriving asymptotic properties.

1.4.2 Nearest neighbour regression smoothing

All of the kernel estimators defined in the previous section are based on weight functions defined on strips of constant width (the bandwidth) which are referred to as the neighbourhood. For a fixed bandwidth the number of data points varies from strip to strip (Altman 1992, p. 177). In other words, the kernel estimator $\hat{m}_h(x)$ is defined as a weighted average of the response variable in a fixed neighbourhood around x, determined in shape and size by the kernel K and the bandwidth h.

The construction of nearest neighbour estimates differs from that of kernel estimators. The k—nearest neighbour ($k - NN$) estimate is a weighted average in a varying neighbourhood. The neighbourhood is defined through those X-variables that are among the k nearest neighbours of x in Euclidean distance. The $k-NN$ estimate in the point x is then calculated as the weighted average of the response variables whose corresponding X-values fall in the neighbourhood (Härdle 1990, p. 42). In other words, the weights in (1.3) are now calculated as weighted averages within strips of data points, where the width of the strips vary, but the number of data points in each strip stays constant. The $k - NN$ weight sequence was introduced by Loftsgaarden and Quesenberry (1965).

More formally, Härdle (1990, p. 42) defines the $k - NN$ smoother as

$$\hat{m}_k(x) = \frac{1}{n}\sum_{i=1}^{n} W_{ki}(x)Y_i, \tag{1.10}$$

where $\{W_{ki}(x)\}_{i=1}^{n}$ is a weight sequence defined through the set of indices

$$J_x = \{i : X_i \text{ is one of the } k \text{ nearest observations to } x\}.$$

For the $k - NN$ estimate with uniform weights the weight sequence is defined as

$$W_{ki}(x) = \begin{cases} \frac{n}{k}, & \text{if } i \in J_x \\ 0, & \text{otherwise.} \end{cases} \tag{1.11}$$

The smoothing parameter k regulates the degree of smoothness of the estimated curve. It plays a role similar to the bandwidth in kernel smoothers. According to Härdle (1990), the influence of varying k in $k-NN$ smoothers on features of the estimated curve is the same as the influence of varying h in kernel regression estimation with a uniform kernel. The smoothing parameter selection problem is also present in $k-NN$ smoothing: k has to be chosen as a function of n or even of the data. Two goals are identified: firstly the noise (variance) should be reduced by letting $k = k_n$ tend to infinity as a function of the sample size. Secondly the approximation error (bias) should be kept low by shrinking the neighbourhood around x asymptotically to zero, i.e. $k = k_n$ should be defined such that $k_n/n \to 0$. These two aims are clearly conflicting. Once again a trade-off situation between the reduction of the observational noise and a good approximation of the regression function arises. Härdle (1990, p. 43) provides expressions for the asymptotic bias and variance of the $k-NN$ estimate with uniform weights defined in (1.11). The trade-off is achieved asymptotically by choosing $k \sim n^{-4/5}$.

1.4.3 Spline smoothing

To motivate the smoothing spline, we first consider the problem of finding a function m that minimizes $\sum_{i=1}^{n}\{Y_i - m(X_i)\}^2$. For most statistical applications, not all solutions are desirable, since not all solutions produce a model as complex as the original data. Most models over-parametrize the model, resulting in estimated parameters with large variability (Fan and Gijbels 1996). In the literature, a penalty for over-parametrization was introduced via the roughness, measured by $\int\{m''(x)\}^2dx$. The resulting penalizing least squares regression goal is to find $\hat{m}_\lambda$ that minimizes

$$\sum_{i=1}^{n}\{Y_i - m(X_i)\}^2 + \lambda\int\{m''(x)\}^2dx \tag{1.12}$$

for a nonnegative real number $\lambda \geq 0$, called the smoothing parameter. The first part of the expression penalizes the lack of fit which represents the basic modelling goal. The second part puts a penalty on the roughness, which relates to the over-parametrization. The choice of λ can range from $\lambda = 0$ to $\lambda = +\infty$. The resulting estimate correspondingly ranges from the complex interpolation model to a simple linear model, indicating that the model complexity of the smoothing spline approach is effectively controlled by the smoothing parameter λ. The estimator $\hat{m}_\lambda$ is called the smoothing spline estimator (Fan and Gijbels 1996, p. 43).

The problem of minimizing (1.12) over the class of all twice differentiable functions on the interval $[a, b] = [X_{(1)}, X_{(n)}]$ has a unique solution $\hat{m}_\lambda(x)$ which is called the cubic spline. Härdle (1990) refers to leading articles in this regard. One of the main properties of the cubic spline $\hat{m}_\lambda(x)$ is that it is a cubic polynomial between two successive X-values. Also, at the observation points X_i the curve $\hat{m}_\lambda(X_i)$ and its first two derivatives are continuous, but there may be discontinuity in the third derivative. Moreover, at the boundary points $X_{(1)}$ and $X_{(n)}$ the second derivatives of $\hat{m}_\lambda(x)$ are zero. It should be noted that these properties follow from the specific choice of the roughness penalty involved in cubic spline.

A difficult aspect of spline smoothing is that $\hat{m}_\lambda$ is defined implicitly as the solution to a functional minimization problem. To judge the behaviour of the estimator and to see how the data react to the estimator are not possible in a straightforward manner. However, Härdle (1990) points out that $\hat{m}_\lambda$ is in fact a weighted average of the Y-observations, i.e.

$$\hat{m}_\lambda(x) = n^{-1} \sum_{i=1}^{n} W_{\lambda i}(x) Y_i.$$

Silverman (1984, Theorem A) shows that the effective weight function $W_{\lambda i}(x)$ looks like a kernel K_s, where the kernel function K_s is given by

$$K_s(u) = 1/2 \exp(-|u|/\sqrt{2}) \sin(|u|/\sqrt{2} + \pi/4).$$

Härdle (1990) states that for large n, small λ and X_i not too close to the boundary, the weight function of interest is

$$W_{\lambda i}(x) \approx f(X_i)^{-1} h(X_i)^{-1} K_s \left(\frac{x - X_i}{h(X_i)} \right)$$

with local bandwidth $h(X_i) = \lambda^{1/4} n^{-1/4} f(X_i)^{-1/4}$ (Härdle 1990, Theorem 3.4.1). K_s is a symmetric kernel function with negative lobes and vanishing second moment, i.e.

$$\int u^2 K_s(u) du = 0.$$

Furthermore, Huber (1979) shows that, under periodicity assumptions about m, the spline smoother is exactly equivalent to a weighted kernel-type average of the response values.

A survey of the literature on the question of how much to smooth in spline smoothing, where mean squared error properties are of concern, can be found in Eubank (1988). The smoothing parameter selection problem (optimizing λ) is related to the convergence rates of the splines. Related issues have been studied by Grace Wahba in several publications (Härdle 1990, Chapter 3).

Statistical software packages that compute the spline coefficients of the local cubic polynomials often require a bound Λ on the sum of squares $\sum_{i=1}^{n}\{Y_i - m(X_i)\}^2$. These programs solve $\int\{m''(x)\}^2 dx = \min$ under the constraint $\sum_{i=1}^{n}\{Y_i - m(X_i)\}^2 \leq \Lambda$. A connection between the two parameters λ and Λ is provided in Härdle (1990, Proposition 3.4.2).

The smoothing parameter λ can be chosen by using the data (Fan and Gijbels 1996). One way of selecting λ is to minimize the cross-validation (CV) criterion

$$CV(\lambda) = n^{-1}\sum_{j=1}^{n}\{Y_j - \hat{m}_{\lambda,(j)}(X_j)\}^2,$$

where $\hat{m}_{\lambda,(j)}$ is the estimator satisfying (1.12) without using the j^{th} observation (Allen 1974, Stone 1974). Cross-validation methods are computer intensive, especially the improved generalized cross-validation method (GCV) proposed by Wahba (1977).

Spline smoothing is explained and discussed by Fan and Gijbels (1996, Section 2.6) and Härdle (1990, Section 3.4). Fast computation issues of smoothing splines can be found in Eubank (1988) and Wahba (1990).

1.4.4 Treatment of outliers

The treatment of outliers or extreme points needs consideration when exploring features of a dataset. Extreme points often dominate the rest of the observations and affect the structures of the data as well as the resulting analyses. It is important to keep in mind that outliers are part of the joint distribution of the data and contain information for estimating the regression curve. However, a small fraction of the data should not be allowed to dominate the small-sample behaviour of the statistics to be calculated. Outliers are powerful in the sense that any smoother based on local averages that is applied to the data will tend to follow the outlying observations. The methods listed below cope with outliers in some way and are referred to as "robust" or "resistant" methods. The discussion is based on Härdle (1990) unless otherwise specified.

a) Median smoothing

Median smoothing is a straightforward resistant technique. For this estimation technique, the conditional median curve $\text{med}(Y|X = x)$ is of importance, rather than the conditional mean curve $E(Y|X = x)$. The median smoother is defined by a sequence of local medians

of the response observations, instead of local averages. More formally, consider

$$\hat{m}_{med}(x) = \text{med}\{Y_i : i \in J_x\}, \tag{1.13}$$

$$J_x = \{i : X_i \text{ is one of the } k \text{ nearest observations to } x\}.$$

The local means of the responses are not robust against outliers. Moving a response value to infinity would drag the smooth to infinity as well. Downweighting large residuals should be attempted. Since only the median response value is used in median smoothing, responses leading to large residuals will carry no weight. Median smoothing is a highly robust technique, where the extreme response observations do not affect the local medians of the responses. The downside of median smoothing is that it produces rough, wiggly curves. Velleman (1980) and Mallows (1980) developed remedial resmoothing and twicing techniques to improve the process of median smoothing. An iterative method involving the median of residuals, obtained after an initial fit, is referred to as the "locally weighted scatterplot smoothing" (LOWESS) method and is discussed in Härdle (1990, p. 192).

b) L-smoothing

L-smoothing uses local trimmed averages of the response observations, instead of local means or local medians. If $Z_{(1)}, Z_{(2)}, \ldots, Z_{(n)}$ denote the order statistics from n observations $\{Z_j\}_{j=1}^n$, a trimmed mean is defined as

$$\bar{Z}_\alpha = (n - 2[\alpha n])^{-1} \sum_{j=[\alpha n]}^{n-[\alpha n]} Z_{(j)}, \quad 0 < \alpha < 1/2,$$

i.e. the mean of the inner $100(1-2\alpha)\%$ of the data. The local trimmed average at the point x is defined as the trimmed mean of the response observations whose corresponding X-values lie in a neighbourhood of x, where the neighbourhood is linked to the choice of a bandwidth sequence $h = h_n$. This type of smoothing is called L-smoothing. It is robust in the sense that extreme response values producing large residuals do not enter the local averaging procedure.

More generally, consider the conditional L-functional

$$l(x) = \int_0^1 J(v) F^{-1}(v|x) dv, \tag{1.14}$$

where $F(\cdot|x)$ is the conditional distribution function of Y given $X = x$ and $F^{-1}(v|x) = \inf\{y : F(y|x) \geq v\}, 0 < v < 1$, is the conditional quantile function associated with $F(\cdot|x)$. For example, consider the following choices of $J(v)$:

- For $J(v) \equiv 1$

$$l(x) = \int_0^1 F^{-1}(v|x)dv = \int_{F^{-1}(0|x)}^{F^{-1}(1|x)} ydF(y|x) = E(Y|X) = m(x),$$

where the substitution $y = F^{-1}(v|x)$ was used.

- For $J(v) \equiv I(\alpha \leq v \leq 1-\alpha)/(1-2\alpha), 0 < \alpha < 1/2$, with symmetric conditional distribution function,

$$l(x) = \frac{1}{1-2\alpha}\int_\alpha^{1-\alpha} F^{-1}(v|x)dv = \int_{F^{-1}(\alpha|x)}^{F^{-1}(1-\alpha|x)} ydF(y|x),$$

where the substitution $y = F^{-1}(v|x)$ was used.

In practice $F(\cdot|x)$ is unknown and needs to be estimated. Let $\hat{F}(\cdot|x)$ denote an estimator of $F(\cdot|x)$. If $\hat{F}(\cdot|x)$ is used in (1.14), L-smoothers are obtained. For example, consider the following choices of $\hat{F}(\cdot|x)$:

- Take $\hat{F}(\cdot|x) = F_n(\cdot|x)$, the empirical conditional distribution function. Then

$$\hat{l}(x) = \int_{F_n^{-1}(\alpha|x)}^{F_n^{-1}(1-\alpha|x)} ydF_n(y|x) = (n - 2[\alpha n])^{-1} \sum_{j=[\alpha n]}^{n-[\alpha n]} Y_{(j)} = \bar{Y}_\alpha,$$

i.e. the trimmed average of the response observations such that the corresponding X_i's are in a neighbourhood of x.

- Estimate $F(\cdot|x)$ by the kernel technique:

$$\hat{F}_h(t|x) = \frac{n^{-1}\sum_{i=1}^n K_h(x - X_i)I(Y_i \leq t)}{\hat{f}_h(x)}$$

to obtain

$$\hat{m}_h^L(x) = \int_0^1 J(v)\hat{F}_h^{-1}(v|x)dv.$$

Asymptotic results for L-smoothers were derived by Stute (1984), Owen (1987) and Härdle, Janssen and Serfling (1988b).

c) R-smoothing

The R-smoothing procedure is derived from R-estimates of location and motivated by rank tests. Suppose that $F(\cdot|x)$ is symmetric around $m(x)$ and J is a nondecreasing function defined on $(0,1)$ such that $J(1-s) = -J(s)$. The score

$$T(\theta, F(\cdot|x)) = \int_{-\infty}^{\infty} J\Big(\frac{1}{2}(F(v|x) + 1 - F(2\theta - v|x)\Big)dF(v|x) \tag{1.15}$$

is then zero for the choice $\theta = m(x)$. Since $F(\cdot|x)$ is unknown it needs to be estimated. Let $\hat{F}(\cdot|x)$ denote an estimator of $F(\cdot|x)$. If $\hat{F}(\cdot|x)$ is used in (1.15), the score would be roughly equal to zero for a good estimate of $m(x)$.

In general, the solution of $T(\theta, \hat{F}(\cdot|x))$ is not unique or has irregular behaviour. In attempt to solve this, Cheng and Cheng (1987) suggested

$$\hat{m}^R(x) = \frac{1}{2}\Big(\sup\{\theta : T(\theta, \hat{F}(\cdot|x)) > 0\} + \inf\{\theta : T(\theta, \hat{F}(\cdot|x)) < 0\}\Big).$$

In particular, if $F(\cdot|x)$ is estimated by the kernel conditional distribution function $\hat{F}_h(\cdot|x)$, the estimate of $m(x)$ is given by

$$\hat{m}_h^R(x) = \frac{1}{2}\Big(\sup\{\theta : T(\theta, \hat{F}_h(\cdot|x)) > 0\} + \inf\{\theta : T(\theta, \hat{F}_h(\cdot|x)) < 0\}\Big).$$

Cheng and Cheng (1990) derived asymptotic results for R-smoothing methods.

d) M-smoothing

M-smoothing is an outlier resistant smoothing technique based on M-estimates of location. In (1.3) a local averaging procedure was proposed to define an estimator for $m(x)$:

$$\hat{m}(x) = \frac{1}{n}\sum_{i=1}^{n} W_{ni}(x)Y_i.$$

As discussed in the next section, smoothers of this form could be seen as local constant polynomial fits, i.e. solutions to local least squares problems

$$\hat{m}(x) = \arg\min_{\theta} \frac{1}{n}\sum_{i=1}^{n} W_{ni}(x)(Y_i - \theta)^2. \tag{1.16}$$

The idea of M-smoothing is to use nonquadratic loss functions to reduce the influence of outliers, instead of the quadratic loss function used in (1.16).

More formally, assume that the conditional distribution $F(\cdot|x)$ is symmetric. Define the kernel M-smoother as

$$\hat{m}_h^M(x) = \arg\min_{\theta} \frac{1}{n}\sum_{i=1}^{n} W_{hi}(x)\rho(Y_i - \theta), \tag{1.17}$$

where $\rho(\cdot)$ denotes some loss function. Huber (1981) mentions such a loss function with lighter tails:

$$\rho(u) = \begin{cases} \frac{1}{2}u^2, \text{ if } |u| \le c; \\ c|u| - \frac{1}{2}c^2, \text{ if } |u| > c, \end{cases} \tag{1.18}$$

where c is a constant that regulates the degree of robustness. Large values of c produce the quadratic loss function. For small values of c (for example if c is one or two times the standard deviation of the residuals) the smoother is more resistant against outliers. To solve (1.17), the expression must be differentiated to θ and set equal to zero,

$$\frac{1}{n}\sum_{i=1}^{n} W_{hi}(x)\psi(Y_i - \theta) = 0,$$

where $\psi = \rho'$. A variety of possible choices of ψ result in consistent estimators $\hat{m}_h^M(x)$. The choice $\psi(u) = u$ yield the ordinary kernel smoother $\hat{m}_h(x)$. Note that $\hat{m}_h^M(x)$ is implicitly defined and should be calculated by using iterative numerical methods. Härdle (1987) showed a fast algorithm based on the Fast Fourier Transform and a one-step approximation to $\hat{m}_h^M(x)$.

The question is how to determine how much is gained or lost in asymptotic accuracy when using M-smoothers. It was found that the bias is the same as for kernel smoothers and therefore the ratio of asymptotic variances of (outlier) resistant and nonresistant estimators should be studied, which is a problematic issue (see Härdle (1990)). However, literature reveals that the influence of outliers is indeed reduced by using M-smoothers.

In the context of spline smoothing, Cox (1983) defined an M-type spline as

$$\arg\min_m \left\{ n^{-1}\sum_{i=1}^{n} \rho(Y_i - m(X_i)) + \lambda \int [m''(x)]^2 dx \right\},$$

where ρ is a loss function with lighter tails than the usual quadratic form, for example the loss function in (1.18).

The theory of M-smoothing has been developed well by various authors (see Härdle (1990, p. 199-200)).

1.4.5 Local polynomial fitting

Another way to look at kernel regression smoothing presented in Section 1.4.1 is to see it as a special case of the broader class of estimators: local polynomial estimators. Fan and Gijbels (1996, Chapter 2) provide a brief summary of the basic literature on local polynomial fitting. The issue of derivative estimation is closely linked to this topic.

The local polynomial fitting procedure can be summarized as follows (Fan and Gijbels 1996): denote the $\gamma - th$ derivative of $m(x)$ by $m^{(\gamma)}(x)$. Suppose that the regression function

can locally be approximated by

$$m(z) \approx \sum_{j=0}^{p} \frac{m^{(j)}(x)}{j!}(z-x)^j \equiv \sum_{j=0}^{p} \beta_j (z-x)^j$$

for z in a neighbourhood of x, by using Taylor expansions. This is a simple local polynomial model, which suggests that the following locally weighted regression problem should be solved:

$$\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} \{Y_i - \sum_{j=0}^{p} \beta_j (X_i - x)^j\}^2 K_h(X_i - x), \tag{1.19}$$

where $\boldsymbol{\beta} = (\beta_1, \beta_2, \ldots, \beta_p)'$, $K_h(\cdot)$ denotes a kernel function and h is a bandwidth. If $\hat{\boldsymbol{\beta}} = (\hat{\beta}_1, \hat{\beta}_2, \ldots, \hat{\beta}_p)'$ minimizes the equation above, then an estimator for the $\gamma - th$ derivative of $m(x)$, $m^{(\gamma)}(x)$, is $\hat{m}_\gamma(x) = \gamma! \hat{\beta}_\gamma$. The whole curve $\hat{m}^{(\gamma)}(\cdot)$ is obtained by running the above local polynomial regression with x varying in an appropriate estimation domain.

Härdle (1990, p. 30-31) claims that kernel estimators are local polynomial fits and verifies it by applying local constant and local parabolic fits. In the first case, let θ be a constant value. The expression to be minimized is

$$\sum_{i=1}^{n} \{Y_i - \theta\}^2 K_h(X_i - x).$$

The solution to the minimization problem is just the Priestley-Chao regression estimate given in (1.7), which can thus be seen as a local constant polynomial fit, i.e. a degree 0 polynomial. In the latter case, Härdle (1990) chooses the kernel $K_h(u) = 1/2 I(|u| \leq 1)$ and states that the expression to be minimized with respect to a and b is then given as

$$\sum_{i=1}^{n} \{Y_i - a - b(X_i - x)^2\}^2 K_h(X_i - x).$$

After some calculations, $\hat{a}$ can be written as $\hat{m}(x)$, but in terms of a different kernel.

Important issues such as the choice of the bandwidth parameter h, the order of the local polynomials and the choice of the kernel function K are discussed in Fan and Gijbels (1996, Chapter 3). Advantages of local polynomial fitting procedures include that it is attractive both from practical and theoretical view points and that the method adapts to various types of designs. Also, the boundary effects are absent: the bias at the boundary points stay at the same order as the inner-point bias, without using boundary kernels – no boundary modifications are required. This is a great advantage, since boundary modifications in higher dimensions are difficult to achieve. The method is easily applied to censored data.

1.5 Measuring the discrepancy

This section focuses on concepts and methods developed for assessing the performance of a regression estimator. We define discrepancy measures between the estimator and the true regression function and discuss ways to evaluate these expressions, as suggested in the literature. The measures can be evaluated pointwise (Section 1.5.1), or globally (Section 1.5.2).

Note that, if the smoothing parameter is chosen as a suitable function of the sample size n, all the smoothers mentioned in the previous section converge to the true curve as the number of observations increases, i.e., pointwise discrepancy measures tend to zero as n increases. Although knowledge about the convergence of the estimator is important, it is not enough. The speed at which the convergence happens needs consideration as well. Härdle (1990, Section 4.1) shed some light on the dependence of the convergence rate on various factors, such as the smoothness of m and the dimension of X. Various other contributions regarding the speed at which smoothers converge were made by many more authors, see Härdle (1990, Section 4.1).

1.5.1 Pointwise measures

A popular pointwise discrepancy measure is the mean squared error (MSE) (Wand and Jones 1995). The MSE is "pointwise" in the sense that it is calculated for a fixed value of x. In this section the MSE is defined and asymptotic MSE results for kernel and $k-NN$ regression estimates are presented.

The mean squared error

The MSE of $\hat{m}(x)$ at some point $x \in \mathbb{R}$ is defined as:

$$\text{MSE}[\hat{m}(x)] = E[\hat{m}(x) - m(x)]^2. \tag{1.20}$$

Although other pointwise criteria such as the mean absolute error

$$\text{MAE}[\hat{m}(x)] = E[|\hat{m}(x) - m(x)|\,]$$

exist, Wand and Jones (1995, p. 14) argue that the MSE is preferred, due to its mathematical simplicity. The MSE will be used as pointwise criterium in this study.

An appealing feature of the MSE is that it can be written as the sum of the variance and the squared bias of the estimator:

$$\text{MSE}[\hat{m}(x)] = \text{Var}[\hat{m}(x)] + [E[\hat{m}(x)] - m(x)]^2. \tag{1.21}$$

Estimators for $\text{Var}[\hat{m}(x)]$ and $\text{Bias}^2[\hat{m}(x)] = [E[\hat{m}(x)] - m(x)]^2$ are available so that this decomposition is very useful in the analysis and interpretation of the estimator. Important questions regarding the rate at which the MSE tends to zero and the relevance of this pointwise measure of accuracy in comparison to more global measures (discussed in the next section), are answered in standard textbooks (see for example Härdle (1990)).

Asymptotic mean squared error results

The aim is to choose an estimator $\hat{m}(x)$ that minimizes the discrepancy measure of interest. This paragraph is concerned with asymptotic properties of pointwise discrepancy measures, in particular the MSE.

The MSE relies on the bandwidth h (or k for the $k-NN$ estimator) in a complex way. As Wand and Jones (1995, p. 19) point out, large sample approximations for the leading bias and variance terms help with interpretation. These approximations have relatively simple expressions that allow better understanding of the role of the bandwidth on the performance of the estimator.

The asymptotic bias and variance of the estimators under consideration have been studied extensively in the literature. Tables 1.2 and 1.3 provide theoretical approximation results for the fixed and random design cases respectively, given that the following list of assumptions hold, where c_K and d_K are defined as:

$$c_K = \int K^2(u)du \tag{1.22}$$

and

$$d_K = \int u^2 K(u)du. \tag{1.23}$$

For the fixed design setting with a one-dimensional predictor variable X, the following assumptions must hold (Härdle 1990):

(A1) K has support $[-1, 1]$ with $K(-1) = K(1) = 0$.

(A2) m is twice continuously differentiable on a neighbourhood of the point x.

(A3) $max_i|X_i - X_{i-1}| = O(n^{-1})$.

(A4) $Var[\varepsilon_i] = \sigma^2, i = 1, \ldots, n$.

(A5) For kernel smoothers: $n \to \infty, h \to 0, nh \to \infty$.

For $k-NN$ smoother: $n \to \infty, k \to 0, nk \to \infty$.

For the random design setting, the following additional assumptions must hold (Chu and Marron 1991, p. 411):

(A6) The marginal density f of X_j has a bounded and continuous first derivative and is bounded above by 0 on a neigbourhood of x.

(A7) X_j and ε_j are uncorrelated.

The theoretical approximation results under these assumptions are given in Table 1.2 for the fixed design case and in Table 1.3 for the random design case.

Estimator	Bias	Variance	Reference
Nadaraya-Watson	$\frac{h^2}{2} d_K m''(x)$	$\frac{\sigma^2}{nh} c_K$	Gasser and Müller (1984, p. 175)
Priestley-Chao	$\frac{h^2}{2} d_K m''(x)$	$\frac{\sigma^2}{nh} c_K$	Bias: Priestley and Chao (1972, p. 389) Variance: Chu and Marron (1991, p. 410)
Gasser-Müller	$\frac{h^2}{2} d_K m''(x)$	$\frac{\sigma^2}{nh} c_K$	Mack and Müller (1989, p. 230)
$k-NN$	$\frac{1}{8}\left(\frac{k}{n}\right)^2 d_K m''(x)$	$\frac{2\sigma^2}{k} c_K$	Härdle (1990, p. 76)

Table 1.2: Asymptotic MSE properties: Fixed design model

It is clear from Table 1.2 that the Nadaraya-Watson, the Priestley-Chao and the Gasser-Müller estimators have the same mean squared error properties in the fixed design scenario. Furthermore, computations for equispaced uniform $X_i = i/n$ values show that the $k-NN$ estimator $\hat{m}_k$ and the kernel estimator $\hat{m}_h$ coincide for k roughly equal to $2nh$ (Härdle 1990). The $k-NN$ estimator behaves like a kernel estimator for the choice $h = \frac{k}{2n}$. However, from Table 1.3, the bias and variance are complicated functionals of m and of the density f for the random design case.

Recall that the aim is to choose an estimator that minimizes the discrepancy measure, i.e. the MSE in this case. In all of the bias-variance combinations in Tables 1.2 and 1.3 we notice that a choice of h (or k) that reduces the bias will cause an increase in variance and

Estimator	Bias	Variance	Reference
Nadaraya-Watson	$\frac{h^2}{2}d_K\left(m''(x)+\frac{2m'(x)f'(x)}{f(x)}\right)$	$\frac{\sigma^2}{f(x)nh}c_K$	Härdle (1990, p. 77)
Gasser-Müller	$\frac{h^2}{2}d_K m''(x)$	$\frac{3\sigma^2}{2f(x)nh}c_K$	Härdle (1990, p. 77)
$k-NN$	$\frac{1}{24}\left(\frac{k}{n}\right)^2\frac{m''(x)f(x)+2m'(x)f'(x)}{f^3(x)}$	$\frac{\sigma^2}{k}$	Härdle (1990, p. 43)

Table 1.3: Asymptotic MSE properties: Random design model

vice versa. In order to find an optimal bandwidth that solves this trade-off problem we need to solve what is called the "smoothing parameter problem". This topic is further discussed in Section 1.6.

1.5.2 Global measures

The MSE evaluates the discrepancy between the estimator and the true curve in a fixed point x. The question arises whether "global" measures that represent the discrepancy over the entire range of x-values cannot be defined. Some global measures of discrepancy are proposed by Härdle (1990, p. 90) and are listed below. We keep the notation presented in Härdle (1990). Let w denote a weight function.

1. The integrated absolute deviation:

$$d_L(\hat{m},m)=\int|\hat{m}(x)-m(x)|f(x)w(x)dx. \tag{1.24}$$

2. The integrated squared error (ISE):

$$d_I(\hat{m},m)=\int(\hat{m}(x)-m(x))^2f(x)w(x)dx. \tag{1.25}$$

3. The averaged squared error (ASE):

$$d_A(\hat{m},m)=\frac{1}{n}\sum_{i=1}^{n}(\hat{m}(X_i)-m(X_i))^2w(X_i). \tag{1.26}$$

4. The maximal absolute deviation:

$$d_{L_\infty}(\hat{m}, m) = \sup_{x \in \mathcal{X}} |\hat{m}(x) - m(x)|, \tag{1.27}$$

where the $\sup_x$ ranges over a set $\mathcal{X} \in \mathbb{R}^d$ of interest.

5. A conditional version of d_A:

$$d_C(\hat{m}, m) = E[d_A(\hat{m}, m)|X_1, \ldots, X_n]. \tag{1.28}$$

6. The mean integrated squared error (MISE):

$$\text{MISE} = E[d_I(\hat{m}, m)] = \int \text{MSE}[\hat{m}(x)]dx. \tag{1.29}$$

The measures in (1.24) to (1.28) are global in the sense that they cover the whole range of x-values. However, these measures are based on only the dataset at hand. Each dataset is merely one realization of a random sample from the population. To recover the whole curve over the entire population, the expected value of the integrated squared error (ISE) could be taken. The resulting measure is the mean integrated squared error (MISE), given in (1.29) (Fan and Gijbels 1996, p. 14).

It could be shown that, in most cases, kernel smoothers converge to the true curve in the sense of the global distance measures that were just defined (Härdle 1990, p. 90). Apart from the convergence of the smoothers, the exact speed of convergence over a class of functions could be quantified. This is important from both practical and theoretical viewpoints. See Härdle (1990, Section 4.1) for more on this topic.

Wand and Jones (1995) state that the MSE and MISE expressions depend on the bandwidth in a complicated way, which makes the interpretation of the influence of the bandwidth on the performance of the estimators difficult. Large sample approximations of the variance and bias terms play an important role in solving this dilemma. Wand and Jones (1995) derive expressions for the asymptotic MISE under conditions on K, h and m.

1.6 Choosing the smoothing parameter

This paragraph focuses on methods developed for choosing the smoothing parameter. The choice of the smoothing parameter is critical to the performance of nonparametric regression estimators. The following underlying points should be kept in mind.

As mentioned in Section 1.5.1, there exists a trade-off between minimizing the bias and the variance of the estimate. This boils down to the selection of a neigbourhood size that avoids overfitting, but at the same time prevents too much variance.

To understand this clearly, consider the two extreme scenarios: on the one hand, a small bandwidth will cause the estimator to follow the data very closely (small bias), but with much wigglyness (large variance). This scenario is called overfitting. On the other hand, a large bandwidth will result in a very smooth curve (small variance) that lies near the mean of the original data (large bias). This is referred to as underfitting (Altman 1992, p. 180).

Since the MSE is the sum of the squared bias and the variance, the choice of smoothing parameter that minimizes the MSE will be a good local choice. Choosing a smoothing parameter that will be optimal over the entire curve also seems feasible. To achieve this, a smoothing parameter that minimizes the ASE, ISE or MISE is demanded. Such choices of h (or k) will be theoretically optimal. A vast literature exist regarding the choice of smoothing parameter. Brief discussions of the basic ideas follow below.

1.6.1 Methods for choosing the smoothing parameter

Härdle (1990) states that the accuracy of kernel smoothers as estimators for m or estimators for the derivatives of m is a function of the kernel K and, to a great extent, the bandwidth h. In real life situations, the underlying regression curve m is unknown. This implies that measures such as the ASE, ISE and MISE defined in Section 1.5.2 cannot be calculated. We need to estimate these theoretical measures and hope that the bandwidths that minimize these estimates would also minimize their corresponding theoretical measures.

We will discuss this matter along the lines of Härdle, Hall and Marron (1988a, p. 87) and Härdle (1990). Throughout this discussion h denotes the bandwidth, but can be replaced by k for the $k-NN$ estimate of m.

Define the optimal bandwidth in this case to be the minimizer of the ASE d_A and denote it by h_0. Note that h_0 for this choice of distance measure (the ASE) is optimal in the sense that it chooses $\hat{m}$ as close to m as possible for the dataset at hand, not for the average over all possible datasets. Other possibilities for h_0 exist, for example the minimizer of the mean average squared error (MASE), $E[d_A(h)]$.

Suppose the ASE $d_A(\hat{m}, m)$ must be minimized (Härdle 1990). A naive approach would be to choose the value of h that minimizes the so-called "resubstitution estimate" of the

prediction error:

$$p(h) = \frac{1}{n}\sum_{j=1}^{n}[Y_j - \hat{m}_h(X_j)]^2 w(X_j).$$

The problem is that $p(h)$ is an increasing function in h, resulting in a situation where the best bandwidth will always be the smallest bandwidth. This is due to the fact that $p(h)$ is a biased estimate of the ASE, because it uses the same dataset to construct and assess the estimate. To see this, Härdle (1990) shows that expanding $p(h)$ leads to a last term

$$C_{1n}(h) = -2n^{-1}\sum_{j=1}^{n}\varepsilon_j\left[n^{-1}\sum_{i=1}^{n}W_{hi}(X_j)Y_i - m(X_j)\right]w(X_j), \tag{1.30}$$

where W_{hi} denotes the Nadaraya-Watson weight sequence defined in (1.5). $C_{1n}(h)$ has expectation (given $\{X_1, X_2, \ldots, X_n\}$)

$$-2n^{-1}\sum_{j=1}^{n}\left[n^{-1}W_{hj}(X_j)\sigma^2(X_j)\right]w(X_j).$$

This quantity tends to zero at the same rate as the variance component of the ASE, which explains why $p(h)$ is a biased estimate for the ASE.

Ways to obtain an unbiased estimate of the ASE should be pursued. We henceforth briefly discuss three ways to approach this.

a) Cross-validation

The leave-one-out cross-validation method utilizes the regression smoother in which the j^{th} observation is deleted when the estimate $\hat{m}$ is calculated in the point X_j:

$$\hat{m}_{h,(j)}(X_j) = \frac{1}{n}\sum_{i=1, i\neq j}^{n}W_{hi}(X_j)Y_i. \tag{1.31}$$

The resulting function is given by the (weighted) sum of the squared deleted residuals

$$CV(h) = \frac{1}{n}\sum_{j=1}^{n}[Y_j - \hat{m}_{h,(j)}(X_j)]^2 w(X_j). \tag{1.32}$$

The function $CV(h)$ is referred to as the "cross-validation function".

The cross-validation method chooses the bandwidth that minimizes $CV(h)$. $CV(h)$ is an unbiased estimate of $p(h)$. Härdle (1990) shows that the cross-product term of $CV(h)$ is

$$C_{2n}(h) = -2n^{-1}\sum_{j=1}^{n}\varepsilon_j\left[n^{-1}\sum_{i\neq j, i=1}^{n}W_{hi}(X_j)Y_i - m(X_j)\right]w(X_j),$$

which has, contrary to the cross-product term of $p(h)$ in (1.30), an expectation of zero. Note that this does not guarantee that $\hat{h}$ will minimize d_A. It is also required that $C_{2n}(h)$ converges uniformly over h to zero. This matter is further discussed in Härdle (1990, p. 154).

b) Penalizing functions

Another solution to the problem of having a biased estimate of the ASE is to multiply $p(h)$ by a correction factor that will cause asymptotic cancellation of the bias. A penalizing function $\Xi(u)$ with first order Taylor expansion $\Xi(u) = 1 + 2u + O(u^2)$, $u \to 0$, is introduced. The prediction error is adjusted by $\Xi(n^{-1}W_{hj}(X_j))$, i.e. it is modified to

$$G(h) = n^{-1}\sum_{j=1}^{n}(Y_j - \hat{m}_h(X_j))^2\Xi(n^{-1}W_{hj}(X_j))w(X_j).$$

By multiplying this expression out, it can be seen that $G(h)$ is roughly equal to $d_A(\hat{m}_h, m)$.

Some examples of penalizing functions include:

- Generalized cross-validation (Craven and Wahba 1979):

$$\Xi_{GCV}(n^{-1}h^{-1}) = (1 - n^{-1}h^{-1}K(0))^{-2}.$$

- Akaike's information criterion (Akaike 1970):

$$\Xi_{AIC}(n^{-1}h^{-1}) = \exp(2n^{-1}h^{-1}K(0)).$$

- Finite prediction error (Akaike 1974):

$$\Xi_{FPE}(n^{-1}h^{-1}) = (1 + n^{-1}h^{-1}K(0))/(1 - n^{-1}h^{-1}K(0)).$$

- A model selector of Shibata (1981):

$$\Xi_S(n^{-1}h^{-1}) = 1 + 2n^{-1}h^{-1}K(0).$$

- The bandwidth selector T of Rice (1984a):

$$\Xi_T(n^{-1}h^{-1}) = (1 - 2n^{-1}h^{-1}K(0))^{-1}.$$

The main difference among the penalizing functions occurs in the left tail, where small bandwidths are penalized differently. Convergence rates of these methods are disappointing. See Härdle (1990, Chapter 5) for more details.

c) The plug-in method

The plug-in method to find a data-driven optimal bandwidth involves finding the theoretical expression of h that will minimize the asymptotic MSE, discussed in Section 1.5.1, and then estimating the unknown parameters in this expression. The plug-in procedure in the random design setting, for example, is based on the asymptotic expansion of the mean squared error for kernel smoothers:

$$MSE = n^{-1}h^{-1}\sigma^2(x)c_K/f(x) + h^4[d_K(m''(x) + 2m'(x)(f'/f)(x))/2]^2.$$

The unknown parameters in this expression, such as $m''(x)$, $m'(x)$ and $\sigma^2(x)$, should be estimated by using the data. This implies a preliminary smoothing process, which in turn implies that bandwidth choice is a question in the preliminary process as well. The substantial uncertainty about bandwidth choice in the first step, as well as the restriction to a certain smoothness class, make this method less popular, even though it achieves the same efficiency as the cross-validation and penalizing function methods.

Data-driven methods to choose the bandwidth, such as the methods discussed in a), b) and c) above, will be mathematically justified if the data-driven bandwidth $\hat{h}$ is asymptotically optimal:

$$\frac{\text{ASE}(\hat{h})}{\inf_{h\in H_n}\text{ASE}(h)} \xrightarrow{\text{a.s.}} 1, \tag{1.33}$$

where $\text{ASE}(h)$ denotes the ASE calculated when a bandwidth h was used, $\hat{h}$ denotes the selected bandwidth and H_n is a set of reasonable bandwidths (Härdle 1990, p. 160). In other words, the score (e.g. CV or G) must approximate the ASE uniformly over h. This definition can be extended to other discrepancy measures as well. According to Härdle and Marron (1985a) and Härdle and Marron (1985b) it holds that $\hat{h}$ is asymptotically optimal if it was chosen to minimize $CV(h)$ or $G(h)$, given that a set of assumptions hold. In other words, cross-validation bandwidth selection and bandwidth selection using penalizing functions yield asymptotically optimal bandwidths under certain assumptions. Härdle (1990, Chapter 5) provides a valuable discussion on this topic. He also illustrates that bandwidth selection in derivative estimation is similar to finding a bandwidth for estimating m itself.

1.7 Choosing the Kernel

In nonparametric regression estimation, the aim is to find a suitable estimate $\hat{m}$ for the unknown m by using the dataset at hand. In the previous section, a "good" choice for $\hat{m}$

(i.e. a choice for which the discrepancy measure is small) was associated with finding the optimal bandwidth. Discrepancy measures, such as the MSE and MISE, are however not only dependent on the bandwidth. Dependence on the kernel function through c_K and d_K, defined in (1.22) and (1.23) respectively, is evident. Härdle (1990, p. 134) shows that if the bandwidth that minimizes the MSE is used, the minimal MSE depends on the kernel K through the factor

$$V(K)B(K) = c_K^2 d_K, \tag{1.34}$$

which should be minimized as a function of K. More generally, consider the case when the aim is to estimate the υ^{th} derivative of m, $m^{(\upsilon)}$ (see Section 1.9 for more on the estimation of derivatives). The functional in (1.34) can then be expressed in terms of derivative kernels, $K^{(\upsilon)}$ (Härdle 1990, p. 134).

It turns out that the kernel must be standardized in some way. There are several approaches to this standardization problem. Gasser, Müller and Mammitzsch (1985) standardized the kernel and minimized $V(K)B(K)$ with respect to K to obtain polynomials of degree p.

In the field of density estimation, Epanechnikov (1969) derives an optimal kernel with respect to the MSE. The Epanechnikov kernel was formulated in Table 1.1. Benedetti (1977) shows that the Epanechnikov kernel is optimal with respect to the MSE in the regression setting as well. Gasser et al. (1985) extend the idea and derive optimal kernels with respect to the MSE for the estimation of the υ^{th} derivative of m where m is assumed to be p-times differentiable. Table 1.4 contains some of these optimal kernels. See Härdle (1990, p. 135) for more detail on the order of these kernels.

How important is the choice of the kernel? In other words, what is the influence of choosing a suboptimal kernel? To answer this question, Härdle (1990, p. 136) considers the efficiency ratio

$$D(K_{opt}, K) = \frac{c_K^2 d_K}{c_{K_{opt}}^2 d_{K_{opt}}},$$

where K_{opt} denotes the optimal kernel and K is the suboptimal kernel in question. He obtains the values of the efficiency ratio in the $k = 0$, $p = 2$ setting, where K is chosen to be different non-optimal kernels. In each case the efficiency ratio does not differ significantly from 1, which implies that the choice of kernel function on the basis of the MSE is not very important. Other properties, such as computational efficiency, deserve more consideration than hunting for the optimal kernel. The optimal bandwidth is of greater importance.

υ	p	Kernel
0	2	$(3/4)(-u^2+1)I(\|u\| \leq 1)$
0	4	$(15/32)(7u^4-10u^2+3)I(\|u\| \leq 1)$
1	3	$(15/4)(u^3-u)I(\|u\| \leq 1)$
1	5	$(105/32)(-9u^5+14u^3-5u)I(\|u\| \leq 1)$
2	4	$(105/16)(-5u^4+6u^2-1)I(\|u\| \leq 1)$
2	6	$(315/64)(77u^6-135u^4+63u^2-5)I(\|u\| \leq 1)$

Table 1.4: Optimal kernels with respect to the MSE

1.8 Behaviour at the boundary

For both kernel and nearest neighbour regression estimators, there is a considerable increase in bias near the boundaries of the data interval. The explanation is intuitively simple, especially for small sample sizes: at the boundary, fewer observations which are grouped asymmetrically around x, can be averaged and thus the variance and bias can be affected. To see this, recall that the estimate in a point x is a local weighted average of the response values of data points in a small neighbourhood around x, where the size of the neigbourhood is determined through a smoothing parameter h. For example, suppose that $x \in [0,1]$. Whenever $x \in [0,h) \cup (1-h,1]$, the window over which the local average is to be computed becomes asymmetric, since the effective window $[x-h, x+h]$ is not fully contained in $[0,1]$. In practice, when a finite sample is used there is an increase in bias in the boundary region $[0,h) \cup (1-h,1]$.

Asymptotically, the MSE reflects this problem only in the end points 0 and 1. This is due to the fact that for $x \in (0,1)$ one has that as $n \to \infty$, $[x-h, x+h] \subset [0,1]$ since $h \to 0$. The boundary effects vanish asymptotically for all interior points, $x \in (0,1)$ (Müller 1988,

p. 32).

However, for global measures such as the MISE, the influence of the increased bias in the limit points 0 and 1 dominates the measure. To see this, first define the order of a kernel. Suppose that a kernel, K, has compact support on $[-\tau, \tau]$, $\int K(x)dx = 1$ and that K fulfills a Lipschitz condition of order λ_K, $0 < \lambda_K \leq 1$. K is of order t if

$$\int_{-\tau}^{\tau} K(x)x^j dx = 0, \; j = 1, \ldots, t-1$$

and

$$\int_{-\tau}^{\tau} K(x)x^t dx \neq 0.$$

Gasser and Müller (1979, p. 58) show that for an estimate that uses a kernel of order t and when $v = 0$ (i.e. we are interested in estimating m itself, not some of its derivatives), the integrated squared bias over $[0, 1]$ is of the order of magnitude $O(h^3)$, while the integrated squared bias over $(0, 1)$ is of the order of magnitude $O(h^{2t})$. The conclusion is that the bias in the boundary points $x = 0$ and $x = 1$ dominate global asymptotic behaviour even for $t = 2$ and becomes worse for higher order kernels.

From the above discussion it is clear that boundary bias is a problem, whether looked at from a finite sample or an asymptotic point of view. One way to deal with this problem is to use "boundary kernels". See Gasser and Müller (1979), Gasser and Müller (1984) and Rice (1984b) for further discussions of this topic. As mentioned before, local polynomial fitting offers a partial solution to this problem.

1.9 Estimating derivatives of m

Kernel estimation can be used to estimate derivatives of m as well. This is done by replacing the weights in (1.3) by their derivatives, which implies using derivatives of the kernel function. For example, for the case where the Priestley-Chao weights are applied in the equidistant design setting, the v^{th} derivative with respect to x is given by

$$\hat{m}_h^{(v)}(x) = n^{-1}h^{-(v+1)} \sum_{i=1}^{n} K^{(v)}\left(\frac{x - X_i}{h}\right) Y_i.$$

This estimate is thus a local average of the response variables in which the v^{th} derivative of the kernel function has been used to calculate the weights. As for kernel estimation of m, the aim is to find an estimate that converges to the derivative m^v under consideration. If the weights adhere to certain smoothness conditions and the smoothing parameter is suitable, convergence is achieved. See Härdle (1990, p. 32) for a more detailed discussion.

1.10 Multidimensional predictor variables

The discussion thus far was confined to the case of univariate design points. The ideas in this chapter can be extended to the case where multidimensional fixed predictor variables, $\mathbf{x}_i = (x_{i1}, \ldots, x_{id})$, or multidimensional random predictor variables, $\mathbf{X}_i = (X_{i1}, \ldots, X_{id})$, are present. The multivariate fixed and random design models are notationally analogous to the models in Section 1.4, except that K, x, x_i and X_i are replaced by their multivariate extensions, where the multidimensional product kernel function is defined by

$$K(u_1, \ldots, u_d) = \prod_{j=1}^{d} K(u_j)$$

(Härdle 1990, p. 27).

Chapter 2

Boosting

2.1 Introduction

Boosting is a general method for improving the accuracy of any given learning algorithm (Schapire 1999, p. 1). Boosting has its roots in the machine learning context (see for example Schapire (1990) and Freund (1995)), but various authors contributed to the development of the methodology and theory of boosting and its applications expanded to a wide range of fields. In this regard, literature reveals that boosting is an efficient improvement method in the regression context (see for example Freund and Schapire (1997), Friedman (2001) and Bühlmann and Yu (2003)). In particular, boosting proved to improve the Nadaraya-Watson learning algorithm (Marzio and Taylor 2008), which is the learner (estimator) of concern in this study. In this chapter we explore the boosting methodology in general and investigate the application of the method in the regression context by using L_2-boosting.

The reader should realize that the term "machine learning" refers to "computer learning" methods. Also, it is not the computer (or any other machine) that is "learning". As will become clear, we are simply dealing with algorithms or estimators and the reader should not be confused.

Furthermore, the term "learner" is used by the "machine learning" community. In the text below some sections will refer specifically to machine learning literature and the term "learner" will be used in such cases. However, the term "estimator" is well established in statistics and almost universally recognised and used. In statistical context we will therefore use the term "estimator" instead of "learner" throughout the chapter.

In Section 2.2 boosting is briefly introduced, as it originated in the machine learning context. Section 2.3 discusses initial attempts to apply boosting in the regression context and the gradient descent view of boosting is introduced in Section 2.4. Section 2.5 considers

L_2-boosting and the method is applied to the Nadaraya-Watson regression estimator in Section 2.6.

Schapire (1990) and Freund (1995) developed algorithms for improving the accuracy of methods for learning binary concepts in machine learning. Freund and Schapire's AdaBoost algorithm (1997) improved on the early algorithms. The boosting method was originally considered as an iterative procedure which, given the training data, would at each step
(i) select a learner from a given class of weak learners,
(ii) evaluate a weight for this learner and
(iii) output the weighted majority vote of the selected learner to the next step
(Lugosi and Vayatis 2004, p. 30). This method attracted much attention in the machine learning community, mainly because of its good empirical performance (Lutz, Kalisch and Bühlmann 2008, p. 3331). In general, boosting algorithms performed well on most of the benchmark datasets in the sense that the test error (i.e. the prediction error of the final hypothesis on new test data) decreases as successive iterations are run (Lugosi and Vayatis 2004, p. 30).

In the regression context some boosting algorithms were proposed, but it was not until Breiman (1999) showed that boosting can be viewed as a functional gradient descent algorithm, that practical boosting algorithms for regression was possible (Lutz et al. 2008, p. 3331). Boosting, when seen as a functional gradient descent technique, tries to optimize in the direction of the negative gradient of a loss function. From this point of view, boosting connects statistics and numerical minimization (Lutz and Bühlmann 2006). In particular, Friedman (2001) developed boosting methods for regression which involves minimizing the squared error loss function, known as L_2-boosting (Bühlmann 2006). Like other boosting methods, L_2-boosting uses a pre-chosen fitting method, called the learner (estimator), iteratively. This boils down to iterative refitting of the residuals (Bühlmann and Yu 2003). Various choices of weak learners (estimators) can be used. In this study we are specifically interested in using nonparametric regression estimates of the unknown regression curve as base learners (estimators).

We now proceed to describe aspects of the boosting method.

2.2 Boosting for classification

The origin of boosting goes back to the development of machine learning, where it provided algorithms for classification problems. Boosting combines simple "rules" to form an ensem-

ble such that the performance of the single ensemble member is improved, i.e. "boosted". Intuitively, boosting could be seen as an iterative procedure where examples that are misclassified in one iteration receive higher weights in the next iteration. For instance, examples near the decision boundary are harder to classify and therefore get higher weights. The idea of iterative reweighting of the training sample is fundamental to boosting (Meir and Rätsch 2003, p. 123).

We now present some of the main ideas of the boosting methodology as it was originally understood in the classification setting.

2.2.1 Historical development of boosting

Boosting has its roots in PAC (Probably Approximately Correct) learning, a theoretical framework for studying machine learning, as proposed by Valiant (1984). Kearns and Valiant (1994) showed that weak learners, each performing just slightly better than random, can be combined to form an arbitrarily accurate learning algorithm when enough data is available. We briefly discuss the PAC (also called the "distribution free") learning model and initial attempts to boosting in this context.

In the PAC model, the learner tries to identify an unknown concept based on randomly chosen examples of the concept (Schapire 1990, p. 197). A concept is a binary-valued mapping over some domain X. Let c denote a concept and $c(x)$ the label of the instance x according to the concept c. A concept class $\mathcal{C}$ is a collection of concepts (Freund 1995, p. 264). The learner has access to a set of examples $(x, c(x))$, where x is drawn randomly from some fixed, but unknown distribution $\mathcal{D}$ on the domain X and $c \in \mathcal{C}$ is the target concept. After some amount of time, the learner must output a hypothesis $h : X \to [0, 1]$, where $h(x)$ can be interpreted as a randomized prediction of $c(x)$: $c(x)$ is 1 with probability $h(x)$ and 0 with probability $1 - h(x)$. The error of the hypothesis h is the expectation $E_{x\sim\mathcal{D}}[|h(x) - c(x)|]$ (Freund and Schapire 1997, p. 124-125).

Freund and Schapire (1997, p. 125) defined a strong PAC-learning algorithm as "an algorithm that, given $\epsilon, \delta > 0$ and access to random samples, outputs with probability $1 - \delta$ a hypothesis with error at most ϵ". A weak PAC-learning algorithm on the other hand "satisfies the same conditions, but only for $\epsilon \geq 1/2 - \gamma$ where γ is either a constant, or decreases as $1/p$ where p is a polynomial in the relevant parameters".

Kearns and Valiant (1994) proved that monotone boolean functions can be learned weakly, but not strongly, with respect to the uniform distribution. This seemed to suggest that weak

and strong distribution-free learning should be separated (Freund 1995, p. 257). However, Schapire (1990) showed that any weak learning algorithm can be boosted into a strong learning algorithm, i.e. weak and strong PAC learning are equivalent in the distribution-free case. Freund (1995) developed an algorithm which improved in efficiency over the work of Schapire (1990), but still suffered from some practical drawbacks (Schapire 1999). The AdaBoost algorithm of Freund and Schapire (1997), discussed in Section 2.2.2, tried to solve these deficiencies.

The algorithms of both Schapire (1990) and Freund (1995) call the given weak learning algorithm multiple times, each time presenting it with a different distribution (set of weights) over the domain X. Finally, all the generated hypotheses are combined into a single hypothesis. These procedures alter the distribution over the domain X in such a way that the probability of the harder parts of the space is increased, to force the weak learner to generate new hypotheses that make less mistakes in these parts (Freund and Schapire 1997, p. 125).

The first experiments with these early boosting algorithms were done by Drucker, Schapire and Simard (1993), who applied the boosting idea to a real-world problem, using neural networks as base learners (Meir and Rätsch 2003).

2.2.2 AdaBoost

The AdaBoost (Adaptive Boosting) algorithm, due to Freund and Schapire (1997), solved a lot of the practical difficulties of the earlier boosting algorithms and is generally considered as the first practical boosting algorithm (Rudin, Schapire and Daubechies 2007, p. 2723). This section provides a brief discussion of the AdaBoost procedure. The discussion is along the lines of Schapire (1999, p. 1,2), unless otherwise referenced.

In the subsections below we first give an algorithm to the AdaBoost procedure. Then a discussion of the algorithm and some features of AdaBoost follow the algorithm. The algorithm presents the AdaBoost procedure in the notation of Schapire (1999).

a) The AdaBoost algorithm

The AdaBoost algorithm is now presented.

Algorithm 2.1. ***AdaBoost***

1. Suppose the data $S = (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ is given, where $x_i \in X$ and $y_i \in Y = \{-1, 1\}$.

2. Initialize $D_1(i) = \frac{1}{n}$, $i = 1, \ldots, n$.

3. For $t = 1, \ldots, T$:

   (a) Train the weak learner using the set of weights D_t to obtain the hypothesis h_t : $X \to \{-1, 1\}$.

   (b) Calculate the weighted training error ϵ_t of h_t:

$$\epsilon_t = P_{i \sim D_t}[h_t(x_i) \neq y_i] = \sum_{i: h_t(x_i) \neq y_i} D_t(i). \tag{2.1}$$

   (c) Choose $\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right)$.

   (d) Update the weights:

$$\begin{aligned} D_{t+1}(i) &= \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } h_t(x_i) = y_i \\ e^{\alpha_t} & \text{if } h_t(x_i) \neq y_i \end{cases} \\ &= \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}, \end{aligned} \tag{2.2}$$

   where Z_t is a normalization factor chosen such that D_{t+1} will be a distribution, i.e. $\sum_{i=1}^{n} D_{t+1}(i) = 1$. In particular

$$Z_t = \sum_{j=1}^{n} D_t(j) \exp(-\alpha_t y_j h_t(x_j))$$

   (Bühlmann and Hothorn 2007, p. 479).

4. Output the final hypothesis:

$$H(x) = sign \left(\sum_{t=1}^{T} \alpha_t h_t(x) \right).$$

b) Discussion of the AdaBoost algorithm

Although AdaBoost has its roots in the PAC model, we present the procedure using a more general learning framework. Suppose that a training set $S = (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ is given as input, where $x_i \in X$, some domain or instance space, and $y_i \in Y$, some discrete label set. In other words, S is a random sample from some fixed but unknown distribution $\mathcal{P}$ on $X \times Y$. Each pair (x_i, y_i), $i = 1, 2, \ldots, n$, is called a training example. The classification goal is to predict the lable y given an instance x (Freund and Schapire 1997, p. 125). For

simplicity, assume a binary classifier $Y = \{-1, 1\}$ in this discussion. Several methods to extend AdaBoost to the multiclass setting exist. See Schapire (1999) for various references in this regard.

The aim of the AdaBoost procedure is to find a final hypothesis H which is consistent with most of the training sample (i.e. $H(x_i) = y_i$ for most $1 \leq i \leq n$) (Freund and Schapire 1997, p. 125). To achieve this, the AdaBoost procedure iteratively calls a given weak learning algorithm T times (i.e. T boosting iterations are used).

Note that the boosting algorithm makes use of "weak" learners. Meir and Rätsch (2003, p. 126) describes a weak learner for a sample S as a learner that is able to achieve a weighted training error given in (2.1) that is strictly smaller than $1/2$, i.e. the learner is only required to perform better than random.

One of the main ideas of the AdaBoost algorithm is to maintain a distribution (a set of weights) over the training set. Denote the weight of this distribution on the training example i for the t^{th} boosting iteration by $D_t(i)$. Initially, the weights are set to be uniform, $D_1(i) = 1/n$. After each iteration the weights of the misclassified examples are increased so that the weak learner is forced to concentrate on the hard examples.

It is the job of the weak learner to find a weak hypothesis $h_t : X \to \{-1, 1\}$ that is appropriate for the set of weights D_t, i.e. a hypothesis that has a small error with respect to the distribution D_t. The goodness of the weak hypothesis is measured by its error, given in (2.1).

Once the weak hypothesis h_t has been determined, a parameter α_t is calculated. α_t measures the importance of h_t. Note that α_t increases as ϵ_t decreases.

Next, the set of weights D_t is updated according to the rule in (2.2). This rule increases the weights of misclassified examples and decreases the weights of examples that were correctly classified. In this way, the weak learner is forced to focus on the hard examples.

After T boosting iterations, the booster combines the weak hypotheses into a single prediction rule (Freund and Schapire 1997, p. 120): the final hypothesis H is the weighted majority vote of the T weak hypotheses h_t, where α_t denotes the weight assigned to h_t. The end result is a final combined classifier, given by a thresholded linear combination of the weak classifiers (Rudin et al. 2007, p. 2724).

c) Features of AdaBoost

Remarks on advantages and disadvantages of the AdaBoost procedure and its resistance to overfitting are now considered.

As stated by Schapire (1999, p. 4) AdaBoost has many advantages. It is fast, simple and easy to program. The only parameter that needs to be tuned is T, the number of iterations. No prior knowledge about the weak learner is required. Lastly, with AdaBoost it is not necessary to design a learning algorithm that is accurate over the entire space. The aim is only to find a weak learner that performs better than random.

On the negative side, the dependence of AdaBoost on the given data and the weak learner can counter accurate results: insufficient data, overly complex weak learners or weak hypotheses that are too weak can cause bad performance (Schapire 1999, p. 4).

It was observed in early experiments that boosting, including AdaBoost, seemed to be resistant to overfitting, in the sense that the test error does not increase, even after a large number of iterations (Rudin et al. 2007, p. 2724). In this regard, Bühlmann and Hothorn (2007, p. 479) comment that it had been debated until about the year 2000 whether AdaBoost is resistant to overfitting and no stopping is necessary. In the years to follow it was commonly acknowledged that boosting algorithms, including AdaBoost, are eventually overfitting and need early stopping. Still, the overfitting behaviour is very slow (AdaBoost is quite resistant to overfitting), which was later ascribed to the fact that boosted learners' variance increase exponentially slow, while their bias decrease exponentially fast. This phenomenon will be discussed later in the regression context.

Much more can be said about boosting methods applied in the machine learning context. However, the interest of this study is nonparametric regression and we move over to the development of boosting in the field of regression.

2.3 Boosting for regression

For univariate regression the learning problem is based on a dataset $S = (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, where $x_i \in X$, the range of possible values the predictor variable can attain, and $y_i \in \mathbb{R}$, implying that y_i can take on real values, rather than being restricted to a finite set as in the classification context (Meir and Rätsch 2003, p. 165).

Several boosting methods for regression have been proposed. Initially these methods involved the reduction of the regression problem to a classification problem. The AdaBoost.R

algorithm, proposed by Freund and Schapire (1997), is one example of this approach and will be briefly discussed. These methods are however subject to several drawbacks. Practical regression boosting algorithms could only emerge after Breiman (1999) introduced the functional gradient descent view of boosting (Lutz et al. 2008, p. 3331), discussed in Section 2.4.

2.3.1 AdaBoost.R

Suppose that the aim is to estimate a regression function m that takes on values in $[0, 1]$. To derive a boosting algorithm in this context, Freund and Schapire (1997, p. 135) reduce the regression problem to a binary classification problem: for each example (x_i, y_i) in the training set, a continuum of examples indexed by pairs (i, y) are defined for all $y \in [0, 1]$. This is done by defining the instances of the converted sample as $\tilde{x}_{i,y} = (x_i, y)$ with associated labels $\tilde{y}_{i,y} = I[y \geq y_i]$. Intuitively, this means that each example (x_i, y_i) in the training set is mapped to an infinite set of binary questions: for each $y \in Y$ the question "is the label y_i larger or smaller than y?", is asked. In this way the regression problem is converted to a classification problem, to which AdaBoost can be applied. See Freund and Schapire (1997, p. 135) for more details on the AdaBoost.R algorithm.

Duffy and Helmbold (2000, p. 210) states that, although AdaBoost.R can be effective, it suffers from two drawbacks. First, AdaBoost.R expands each data point in the sample to many classification tasks. The number of branch points in the piece-wise linear integral increases exponentially with the number of iterations. This makes the algorithm computationally intractable (Rätsch, Demiriz and Bennett 2002, p. 193). Second, the "error" function that the base regressor should be minimizing is not (except for the first iteration) a standard loss function. Furthermore, the loss function changes in each iteration and even differs between examples in the same iteration. This makes it difficult to determine if a certain weak learner is appropriate for the algorithm. However, new developments paved the way to more successful application of boosting in the regression context.

2.4 Functional gradient descent view of boosting

In the regression context some more boosting algorithms were proposed which will not be discussed here. However, it was not until Breiman (1999) secured the ground for a functional gradient descent (FGD) view of boosting that practical boosting algorithms for regression were possible (Lutz et al. 2008, p. 3331). To explain the FGD view of boosting and place it

into context, it is necessary to consider further developments of boosting in the classification context.

2.4.1 Alternative views of AdaBoost

Reasons for the success of boosting in general, and in particular of AdaBoost, were not yet clear. An important attempt to explain the success was in terms of the large margin theory, which is due to Schapire, Freund, Bartlett and Lee (1998). In this view boosting methods tend to maximize the margin of correctly classified examples (Lugosi and Vayatis 2004, p. 31). However, large margin theory alone does not completely explain the efficiency of boosting, as shown by Breiman (1999). Breiman (1999) was the first to interpret AdaBoost as the minimization of an exponential criterion. Even though Breiman's approach was game-theoretical rather than statistical, his work can be seen as the ancestor of the "statistical view of boosting".

The statistical view of boosting was first formally described by Friedman, Hastie and Tibshirani (2000). Buja, Mease and Wyner (2007, p. 507) explain how the work of Friedman et al. (2000) established the meaning of boosting as model fitting. In broad terms, boosting creates linear combinations of weak learners of half the logit of the underlying conditional class probabilities, $P(Y = 1|x)$. In this way, boosting could be seen as class probability estimation in the conditional Bernoulli mode. This led Friedman et al. (2000) to replace exponential loss with a loss function that statisticians are more familiar with, namely the negative log-likelihood of the Bernoulli model. They also replaced boosting's reweighting with iteratively reweighted least squares, a reweighting method commonly used in statistics, to implement Newton descent/Fisher scoring. These attempts resulted in their LogitBoost algorithm. In this view, AdaBoost estimates half the logit and LogitBoost estimates the logit, both using stagewise fitting, but by different approaches to the functional gradient that produces the additive terms. Friedman (2001) discarded weighting altogether by approximating gradients with plain least squares. This resulted in what we will call the "FGD view of boosting", discussed next.

2.4.2 The FGD view of boosting

Consider the random design model, i.e. a set of realizations of independent, identically distributed (i.i.d.) random variables $S = (X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)$, where $X_1, X_2, \ldots, X_n$ are one-dimensional predictor variables and $Y_1, Y_2, \ldots, Y_n$ are one-dimensional response vari-

ables, which can be either continuous (regression problem) or discrete (classification problem). The setting can be easily extended to the multivariate case, where $X_1, X_2, \ldots, X_n$ are p-dimensional predictor variables (Schmid and Hothorn 2008, p. 299). The FGD view of boosting can also be applied in the fixed design context, where a set of fixed, ordered points $x_1, x_2, \ldots, x_n$ that are often assumed to be equispaced is given, resulting in the sample space $S = (x_1, Y_1), (x_2, Y_2), \ldots, (x_n, Y_n)$ (Marzio and Taylor 2008, p. 2484).

According to Bühlmann and Yu (2003, p. 325), the aim is to estimate the function $m : \mathbb{R} \to \mathbb{R}$ that minimizes the expectation of some cost function:

$$E[C(Y, m(X))], \; C(\cdot, \cdot) : \mathbb{R} \times \mathbb{R} \to \mathbb{R}^+, \tag{2.3}$$

where (X, Y) follows the same distribution as each of the data points $\{(X_i, Y_i)\}_{i=1}^n$.

The cost function C is assumed to be differentiable and convex in the second argument, to ensure that the gradient method works well. Bühlmann and Yu (2003, p. 325) present the most prominent cost functions:

1. $C(y, f) = \exp(yf)$ with $y \in \{-1, 1\}$, which is the cost function for AdaBoost;
2. $C(y, f) = \log_2(1 + \exp(-2yf))$ with $y \in \{-1, 1\}$, which is the cost function for LogitBoost; and
3. $C(y, f) = (y-f)^2/2$ with $y \in \mathbb{R}$ or $y \in \{-1, 1\}$, which is the cost function for L_2Boost. The L_2Boost algorithm will be discussed in Section 2.5.

The population minimizers of (2.3) are then

$$m(x) = \frac{1}{2} \log \left(\frac{P[Y = 1|X = x]}{P[Y = -1|X = x]} \right)$$

for AdaBoost and LogitBoost and

$$m(x) = E[Y|X = x]$$

for L_2Boost, i.e. the regression relation as defined in Chapter 1 (Bühlmann and Yu 2003, p. 325).

Since the population distribution is usually unknown, the function $m(\cdot)$ that minimizes (2.3) should be estimated by using real data. The estimation from data can be done by considering the empirical risk

$$\frac{1}{n} \sum_{i=1}^{n} C(Y_i, m(X_i)) \tag{2.4}$$

and applying iterative steepest descent (Bühlmann and Yu 2003, p. 325).

Bühlmann and Hothorn (2007, p. 480) present the generic FGD algorithm to this estimation problem as proposed by Friedman (2001) as follows:

Algorithm 2.2. ***Generic FGD Boosting***

1. Suppose the data $S_n = \{(X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)\}$ is given.

2. Fit an initial learner, $\hat{m}_0(\cdot)$. Possible choices are $\hat{m}_0(\cdot) \equiv \arg\min_c \sum_{i=1}^n C(Y_i, c)$, or $\hat{m}_0(\cdot) \equiv 0$. Set $\boldsymbol{t = 0}$.

3. Let $\boldsymbol{t = t + 1}$. Compute the negative gradient vector $-\dfrac{\delta C(Y_i, m)}{\delta m}$ and evaluate at $m = \hat{m}_{t-1}(X_i)$:
$$U_i = -\frac{\delta C(Y_i, m)}{\delta m}\bigg|_{m=\hat{m}_{t-1}(X_i)} \quad i = 1, \ldots, n.$$

4. Fit the negative gradient vector $U_1, U_2, \ldots, U_n$ to $X_1, X_2, \ldots, X_n$ by the base procedure. Call this fit $\tilde{m}_t(\cdot)$. Thus $\tilde{m}_t(x)$ can be viewed as an approximation of the negative gradient vector.

5. Update
$$\hat{m}_t(\cdot) = \hat{m}_{t-1}(\cdot) + \nu \tilde{m}_t(\cdot),$$
where $0 < \nu \leq 1$ is a real-valued step-length factor, i.e. proceed along an estimate of the negative gradient vector.

6. Repeat steps 3 to 5 until $t = T$, where T is some stopping value for the number of iterations.

The phenomenon of "regularization" is embedded in the process above: in the algorithm, two values need specification, namely the number of iterations T and the step-length factor ν. These serve as regularization parameters, ensuring that the data do not fit too closely. Friedman (2001, p. 1203) notes that reducing the expected loss on the training data (i.e. the sample data given to the learner in step 1 of the algorithm) beyond some point causes the expected loss for the population to stop decreasing and often start increasing.

Yao, Rosasco and Caponetto (2007, p. 289) explain this phenomenon as follows: in gradient descent algorithms two iteration paths are relevant, namely the gradient flow for expected loss minimization which depends on the unknown (population) probability measure, called the population iteration, and the gradient flow for empirical loss minimization based on the

sample, named the sample iteration. Both paths start from the origin, but they leave each other as the iterations proceed. The population iteration converges to the regression function, while the sample iteration converges to an overfitting function. Regularization is needed to keep the two paths close and prevent the sample iteration to become an overfitting function. This results in a bias-variance trade-off: call the distance between the population iteration and the regression function the bias and the gap between the population iteration and the sample iteration the variance. Stopping too early may reduce variance, but enlarge bias, while stopping too late may reduce the bias, while enlarging the variance. Regularization parameters are needed to solve this trade-off.

The stopping iteration T is the main tuning parameter which should be chosen such that the data is not overfitted. In the literature T is determined via cross-validation or using some information criterion such as AIC-type stopping criteria (Bühlmann and Hothorn 2007). In this study we apply the cross-validation method to choose T.

Next, consider the step-length factor ν. Friedman (2001) suggests that $\nu = \nu(m)$ should be estimated in each boosting iteration by performing a line search. Bühlmann and Hothorn (2007, p. 480) argue that this procedure is rather time-consuming and of relative low importance when it comes to the predictive ability of FGD. They suggest choosing a constant ν, as long as ν is small, such as $\nu = 0.1$ in order to achieve a stagewise adaptation of the true predictor function m. In their application of boosting to the N-W estimator, Marzio and Taylor (2008) choose $\nu = 1$. We therefore choose $\nu = 1$ in this study as well. Future research should investigate other choices of ν.

Schmid and Hothorn (2008, p. 300) note that there exists a dependence between the number of boosting iterations T and the step-length factor ν: T usually increases with a decreasing value of ν. They show that when the corrected AIC-criterion is used for stopping, this dependence is approximately linear, so that T is approximately inversely proportional to ν. The choice of ν is of rather low importance if the corrected AIC criterion is used for stopping: the stopping iteration T automatically adapts to the value of ν.

We now consider L_2-boosting, the boosting algorithm that, in the FGD view, uses the squared error loss function.

2.5 L_2-boosting

Bühlmann and Hothorn (2007, p. 483) present L_2-boosting as the "simplest and perhaps most instructive boosting algorithm" that is "very useful for regression, in particular in

presence of very many predictor variables". The negative gradient in the second step of the FGD algorithm is now the classical residual vector. Basically this algorithm starts off by fitting some initial learner, like fitting the base procedure to the original data, then uses the residuals from the previous iteration as new response vector, refits the base procedure and continues to iteratively repeat the process (Bühlmann and Yu 2006, p. 1003).

Bühlmann and Hothorn (2007, p. 483) present the L_2-boosting algorithm of Friedman (2001) as follows:

Algorithm 2.3. *L_2-boosting*

1. Suppose the data $S_n = \{(X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)\}$ is given.
2. Fit an initial learner, $\hat{m}_0(\cdot)$. The default value is $\hat{m}_0(\cdot) \equiv \bar{Y}$. Set $\boldsymbol{t} = 0$.
3. Let $\boldsymbol{t} = \boldsymbol{t} + 1$. Compute the residuals,
$$U_i = Y_i - \hat{m}_{t-1}(X_i), i = 1, \ldots, n.$$
4. Fit the residual vector $U_1, U_2, \ldots, U_n$ to $X_1, X_2, \ldots, X_n$ by the base procedure. Call this fit $\tilde{m}_t(\cdot)$.
5. Update
$$\hat{m}_t(\cdot) = \hat{m}_{t-1}(\cdot) + \nu \tilde{m}_t(\cdot),$$
where $0 < \nu \leq 1$ is a real-valued step-length factor, as in the general FGD algorithm.
6. Repeat steps 3 to 5 until $t = T$, where T is some stopping value for the number of iterations.

Tukey (1977) has already proposed this method for one iteration, i.e. $T = 1$ and $\nu = 1$ under the name "twicing".

L_2-boosting can be used for both regression and classification. For the continuous case when $Y \in \mathbb{R}$, a regression estimate for $E[Y|X = x]$ is directly given by $\hat{m}_t(\cdot)$. For the discrete binary case when $Y \in \{-1, 1\}$, a classifier under equal misclassification costs is given by $sign(\hat{m}_t(\cdot))$, since $E[Y|X = x] = P[Y = 1|X = x] - P[Y = -1|X = x]$. Then, since AdaBoost and LogitBoost aim to estimate

$$m(x) = \frac{1}{2} \log \left(\frac{P[Y = 1|X = x]}{P[Y = -1|X = x]} \right),$$

$sign(\hat{m}_t(\cdot))$ is an appropriate classifier (Bühlmann and Yu 2003, p. 326).

2.5.1 Regularization in L_2-boosting

As described for the general FGD boosting algorithm, boosting until $T = \infty$ is typically not a good idea. The MSE with $T = \infty$ is equal to the noise level σ^2. This is due to the fact that repeating the boosting process infinitely many times yield the fully saturated model that exactly fits the data - that is, the bias is zero, but the variance is σ^2. Regularization is needed by controlling the number of boosting iterations, T. When working with real data, this implies monitoring an estimate of the MSE, for example by using cross-validation (Bühlmann and Yu 2003, p. 328).

Bühlmann and Yu (2003, p. 327) discuss the regularization problem for L_2-boosting in the case of symmetric linear learners, such as smoothing splines. In their article, they prove a theorem that shows an interesting bias-variance trade-off, not seen in literature before: as the number of boosting iterations T varies, both the bias and variance terms change exponentially, with the bias decreasing exponentially fast and the variance increasing exponentially slow. This is in contrast with the standard algebraic trade-off commonly seen in nonparametric estimation. The advantage of this new exponential trade-off is a flatter near-optimal region for the optimal T: the trade-off not only gets very close to the optimal MSE, but also stays quite flat afterward.

The need for regularization provides insight in the requirement that a "weak" learner, that is a learner with sufficiently low variance, should be used for boosting. If the learner is too strong (follows the data too closely), then even after one boosting iteration the MSE will increase, rather than decrease (Bühlmann and Yu 2003, p. 336).

Bühlmann and Yu (2003, p. 333) show that the resistance of boosting against overfitting is stronger for classification than for regression.

2.6 L_2-boosting for the Nadaraya-Watson regression estimator

Suppose that the aim is to estimate $m(x) = E[Y|X = x]$, given a sample S. Kernel regression estimators, such as those described in Chapter 1, can be implemented. In boosting context, Marzio and Taylor (2008) propose using the Nadaraya-Watson (N-W) kernel regression estimator defined in (1.6) as base learner, which could be boosted for better performance. In the subsections below we present an algorithm for the L_2BoostNW-method, discuss features

of the method and make some remarks on the extension of the method to the multivariate setting.

a) The L_2BoostNW algorithm

The L_2BoostNW-method can be described by the following pseudocode (Marzio and Taylor 2008):

Algorithm 2.4. ***L_2BoostNW***

1. Suppose the data $S_n = \{(X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)\}$ is given.

2. Fit an initial learner, $\hat{m}_0(\cdot)$ by fitting the N-W regression estimator to the original data: $\hat{m}_0(\cdot) \equiv \hat{m}_{NW}(\cdot)$, using some bandwidth, h. The bandwidth can be chosen in different ways, for example by cross-validation or using penalizing functions. See Chapter 1, Section 1.6 for a discussion of bandwidth selection methods. Set $t = 0$.

3. Let $t = t + 1$. Compute the residuals,

$$U_i = Y_i - \hat{m}_{t-1}(X_i), i = 1, \ldots, n.$$

4. Fit the residual vector $U_1, U_2, \ldots, U_n$ to $X_1, X_2, \ldots, X_n$ by the N-W estimator. Call this fit $\tilde{m}_t(\cdot)$.

5. Update

$$\hat{m}_t(\cdot) = \hat{m}_{t-1}(\cdot) + \nu \tilde{m}_t(\cdot),$$

where $0 < \nu \leq 1$ is a real-valued step-length factor. Marzio and Taylor (2008) choose $\nu = 1$. This choice ought to influence the number of boosting iterations.

6. Repeat steps 3 to 5 until $t = T$, where T is some stopping value for the number of iterations.

b) Features of L_2BoostNW

Some features of the L_2BoostNW algorithm are now considered.

1. Choice of the kernel

   In order to describe the properties of L_2BoostNW, Marzio and Taylor (2008) compactly denotes the N-W estimator as

$$\hat{\mathbf{m}}_0 = \mathbf{NKY},$$

where

$$\hat{\mathbf{m}}_0^T \equiv (\hat{m}_0(X_1), \hat{m}_0(X_2), \ldots, \hat{m}_0(X_n)),$$

$$\mathbf{N}^{-1} \equiv diag\left(\left\{\sum_{i=1}^{n} K_h(X_1 - X_i)\right\}, \left\{\sum_{i=1}^{n} K_h(X_2 - X_i)\right\}, \ldots, \left\{\sum_{i=1}^{n} K_h(X_n - X_i)\right\}\right),$$

$$(\mathbf{K})_{ij} \equiv K_h(X_i - X_j),$$

$$\mathbf{Y}^T \equiv (Y_1, Y_2, \ldots, Y_n) \text{ and}$$

$$i, j = 1, 2, \ldots, n$$

Although the N-W fit is linear, the hat matrix ($\mathbf{NK}$) is not symmetric. Therefore the preceding theory established by Bühlmann and Yu (2003) for L_2-boosting of symmetric linear learners is not applicable. New theory had to be developed.

Marzio and Taylor (2008) state that L_2BoostNW works properly only if the set of characteristics roots of the hat matrix $\mathbf{NK}$ is a subset of $(0, 1]$. In Theorem 1 of their article, they identify a set of kernels that satisfies this property. The theorem states that a continuous, second-order kernel that is a Fourier-Stieltjes transform of a finite measure and that is symmetric and unimodal, will qualify for the property. The property does not hold for many popular kernel functions, such as the Epanechnikov, biweight and triweight kernels. However, Gaussian and triangula kernels yield strictly positive characteristic roots and can be used. This phenomenon is surprising, since the choice of kernel as described in Section 1.7 is often done merely by computational convenience.

2. Bias-variance trade-off

Marzio and Taylor (2008, p. 2487) show that, similar to the case where L_2-boosting was applied to symmetric learners, an exponential bias-variance trade-off occurs when L_2-boosting is applied to the N-W estimator: the bias decreases exponentially fast towards zero, while the variance increases exponentially slow towards σ^2. Once again, the sub-optimal area is quite flat, resulting in a resistance to overfitting. They note that band-width selection should take into account the number of boosting iterations. Therefore, rather than choosing the optimal number of boosting iterations T and the bandwidth h separately, the optimal pair (h, T) should be chosen. This can be done, for example, by cross-validation when working with real data. However, boosted kernel estima-tors seem to be less sensitive to the bandwidth choice than standard kernel regression

estimators. Regularizing through oversmoothing in conjunction with many boosting iterations increases the combinations of (h, T) for which boosting works. Thus, the potential of reducing the need of an accurate bandwidth and stopping rule, emerges (Marzio and Taylor 2008, p. 2492).

3. Asymptotic properties after one boosting iteration

   In Section 1.5.1 the asymptotic bias and variance for the N-W regression estimate were provided, given that a set of conditions hold. Note that the bias is $O(h^2)$. Marzio and Taylor (2008, p. 2487) show that the asymptotic MSE properties of a once boosted N-W estimator, $\hat{m}_1(x)$, are as follows:

   $$\text{Bias}[\hat{m}_1(x)] = o(h^2)$$

   and

   $$\text{Var}[\hat{m}_1(x)] \leq 4\text{Var}[\hat{m}_0(x)],$$

   under the usual set of assumptions. Thus, bias is reduced from $O(h^2)$ to $o(h^2)$. Since at the first step the magnitude order of the variance is preserved, the MSE is reduced if n is large enough.

4. Comparison with boosted splines

   By using simulation studies, Marzio and Taylor (2008, p. 2491-2492) show that L_2-BoostNW outperforms boosted splines, suggested by Bühlmann and Yu (2003), for all sample sizes used in the study. This is the case even though the N-W base learner is asymptotically inferior to splines. The best results are obtained when the N-W estimator is weaker than the spline. This phenomenon confirms the need for a weak learner when boosting is applied.

c) Extension to the multivariate case

As stated before, boosting is especially effective in the case of multiple predictor variables. Marzio and Taylor (2008, p. 2493) extend their method to the multivariate case, as discussed below.

Suppose that d-dimensional data, $S = (\mathbf{X}_1, Y_1), (\mathbf{X}_2, Y_2), \ldots, (\mathbf{X}_n, Y_n)$ is given, where each predictor variable has d dimensions, $\mathbf{X}_i = X_{i1}, X_{i2}, \ldots, X_{id}$, $i = 1, 2, \ldots, n$. The smoother is extended in the usual way, by building multiplicative kernels with a diagonal bandwidth

matrix

$$\prod_{j=1}^{d} K_h(x_j - X_{ij}).$$

In a real data scenario, the bandwidth h and number of boosting iterations T can be chosen by cross-validation as the pair (h_{CV}, T_{CV}) that solves

$$\min_{h,t} = \sum_{i=1}^{n} (Y_i = \hat{m}_t^{(-i)}(\mathbf{X}_i))^2,$$

where $m_t^{(-i)}(\mathbf{X}_i)$ is the L_2BoostNW estimate of $m(\mathbf{X}_i)$ when the i-th observation is omitted.

Chapter 3

The bootstrap

3.1 Introduction

Statisticians often intend to estimate parameters of the true underlying distribution of a population by using only the sample data at hand. The accuracy of these estimates come to question: how close are the estimates in expectation to the true parameters and how much do they vary from sample to sample? These matters have been studied abundantly in the literature. In some cases theoretical approaches, which rely on assumptions about the underlying distribution of the population, are feasible. However, these assumptions greatly influence the conclusions that are drawn. Furthermore, mathematical complexity and computational difficulties of theoretical approaches can place restrictions on the inference.

Efron (1979) introduced a very general resampling procedure to combat these problems, i.e. the bootstrap method.

In the current study we apply bootstrap principles by means of the "bagging" and "bragging" methods to improve the Nadaraya-Watson regression estimator. The goal of this chapter is to briefly introduce the reader to some of the main features and applications of the bootstrap method. This should serve as background for the next chapter, where the bagging and bragging methods are discussed.

Apart from what is mentioned in the rest of this paragraph, the content of the chapter is as follows: in Section 3.2 the basic formulation of the bootstrap methodology is presented. Section 3.3 considers some examples of applications of the bootstrap. Particularly, the bootstrap estimation of the standard error (Section 3.3.1), the bootstrap estimation of the bias (Section 3.3.2) and the double bootstrap (Section 3.3.3) are featured. In Section 3.4 we show how the bootstrap method is applied to the statistical problem considered in this study, namely regression.

The discussion will be guided by Efron and Tibshirani (1993) and Swanepoel (1990), unless stated otherwise. In this discussion attention is given to the nonparametric bootstrap.

One handy application of the bootstrap is to estimate the distribution of statistics based on independent observations. With minimal assumptions made and little theoretical analysis, the bootstrap can be applied in an automatic way to a wide variety of other contexts, regardless of the theoretical complexity. The availability of high speed computers set the ground for resampling methods, such as the bootstrap, where problems that are too complex for traditional statistical methods can be solved.

For example, bagging (Breiman 1996a), or bootstrap aggregating, is a procedure developed to reduce the variance of some statistical estimate. It uses bootstrap samples to provide multiple versions of the statistic of interest, which are then aggregated to provide a less variable estimate of the original parameter. Bragging (Swanepoel (1990), Bühlmann (2003)) is similar to bagging, except that the median of the bootstrap versions of the statistic of interest is obtained, rather than the arithmetic mean as in bagging. In this study we apply bagging and bragging to the smoothing parameter selection problem in kernel regression. The bagging and bragging methodology is discussed in Chapter 4. In order to comprehend the bagging and bragging methods, one needs to be familiar with the mechanism underlying these methods – the bootstrap.

We now proceed by formulating the bootstrap method.

3.2 Bootstrap methodology

Let $\mathbf{X}_n = (X_1, X_2, \ldots, X_n)$ be a finite random sample of size n, drawn independently from a common unknown probability distribution F. Suppose $T_n(\mathbf{X}_n, F)$ is some specified random variable of interest, possibly depending on F. Let $\hat{F}$ denote any suitable estimator for F. A popular choice for $\hat{F}$ is F_n, the empirical distribution function (EDF) of $\mathbf{X}_n$, i.e. the distribution with mass n^{-1} at each of the data points $X_1, X_2, \ldots, X_n$ (F_n is defined more formally in (3.1) below). The bootstrap method consists of approximating the sampling distribution of $T_n(\mathbf{X}_n, F)$ under F by the bootstrap distribution of $T_n(\mathbf{X}_n^*, \hat{F})$ under $\hat{F}$, where $\mathbf{X}_n^* = (X_1^*, X_2^*, \ldots, X_n^*)$ denotes a random sample of size n from $\hat{F}$. To perform the actual calculations of the bootstrap, Efron (1979) suggests a Monte Carlo approximation. Repeated realizations of $\mathbf{X}_n^*$ are generated by drawing random samples of size n from $\hat{F}$, say $\mathbf{X}_n^*(1), \mathbf{X}_n^*(2), \ldots, \mathbf{X}_n^*(B)$. The sampling distribution of the corresponding values $T_n(\mathbf{X}_n^*(1), \hat{F}), T_n(\mathbf{X}_n^*(2), \hat{F}), \ldots, T_n(\mathbf{X}_n^*(B), \hat{F})$ is taken as an approximation of the

actual bootstrap distribution of $T_n(\mathbf{X}_n^*, \hat{F})$. By taking B sufficiently large, this approximation can be made arbitrarily accurate. In situations where explicit expressions for the bootstrap estimate exist, analytical, direct calculation methods are feasible and Monte Carlo approximation is not necessary.

We now present different choices of $\hat{F}$.

Estimators for F

Different estimators $\hat{F}$ for F can be used. Some of the possibilities are listed below.

1. The classical estimator of F is $\hat{F} = F_n$, the empirical distribution function (EDF) of $\mathbf{X}_n$. The EDF is defined as

$$F_n(x) = \frac{1}{n}\sum_{i=1}^{n} I(X_i \leq x), \tag{3.1}$$

where I is the indicator function, defined as

$$I(A) = \begin{cases} 1 & \text{if } A \text{ occurs} \\ 0 & \text{if } A^C \text{ occurs.} \end{cases}$$

If $\hat{F} = F_n$, the bootstrap sample $(X_1^*, X_2^*, \ldots, X_n^*)$ is simply a random sample of size n drawn with replacement from the original sample $(X_1, X_2, \ldots, X_n)$, i.e. each data point in the original sample has probability 1/n to be drawn as the next data point in the bootstrap sample. The Glivenko-Cantelli theorem states that F_n approximates F uniformly for large sample sizes, i.e. $\lim_{n\to\infty} \sup_x |F_n(x) - F(x)| = 0$ almost surely (Jacod and Protter 2004, p. 185). The case when $\hat{F} = F_n$ is referred to as the nonparametric bootstrap.

2. In some situations the sample data are drawn from a known distribution, say $F_{\boldsymbol{\theta}}$, where $\boldsymbol{\theta}$ denotes the vector of the distribution's parameters, which is unknown. The sample data are then used to estimate the unknown parameters $\boldsymbol{\theta}$ and the estimated distribution F_{par} is obtained. Bootstrap samples are generated by drawing B random samples of size n from F_{par}. The case when $\hat{F} = F_{par}$ is called the parametric bootstrap.

3. The EDF F_n is a discrete approximation of an often continuous distribution F. Davison and Hinkley (1997, p. 79) propose that, if it is reasonable to suppose that the underlying probability distribution F is continuous, kernel distribution estimation can improve on the EDF. For a scalar x the kernel density estimator is given by

$$\hat{f}_h(x) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - X_i}{h}\right),$$

where $K(\cdot)$ is the kernel function, a continuous and symmetric density function with mean zero and unit variance and h is the choice of bandwidth. Bootstrap samples and calculations are then based on the corresponding kernel distribution function F_h. The case when $\hat{F} = F_h$ is called the smoothed bootstrap.

In this study we use the nonparametric bootstrap with $\hat{F} = F_n$ throughout. For this choice of $\hat{F}$, the so-called "plug-in" principle applies. According to the plug-in principle, some aspect of F, like its mean or median, can be estimated by the corresponding aspect of F_n. The bootstrap method is a direct application of the plug-in principle.

3.3 Bootstrap estimation of the standard error and bias

Let θ denote a parameter of an unknown population distribution F that needs to be estimated by using a random sample from F, i.e. $X_1, X_2, \ldots, X_n$. Denote the estimator by $\hat{\theta} = \hat{\theta}(X_1, X_2, \ldots, X_n)$. The aim of the bootstrap is to estimate the distributional properties of the estimator $\hat{\theta}$. In this section bootstrap algorithms to estimate the statistic's standard error, bias and the standard error of the statistic's estimated standard error are presented.

3.3.1 The bootstrap estimate of the standard error

Suppose that the standard error of the estimator $\hat{\theta} = \hat{\theta}(X_1, X_2, \ldots, X_n)$ is denoted by

$$\sigma_F(\hat{\theta}) = \sqrt{\mathrm{Var}_F(\hat{\theta})} \tag{3.2}$$

and needs to be estimated. By applying the plug-in principle, the ideal bootstrap estimate of the standard error is given by

$$\sigma_{F_n}(\hat{\theta}^*) = \sqrt{\mathrm{Var}_{F_n}(\hat{\theta}^*)}, \tag{3.3}$$

where $\hat{\theta}^* = \hat{\theta}(X_1^*, X_2^*, \ldots, X_n^*)$ and $X_1^*, X_2^*, \ldots, X_n^*$ are independent, identically distributed (i.i.d.) random variables from the EDF F_n.

The following algorithm describes a Monte Carlo algorithm to obtain an approximation of the ideal bootstrap estimate:

Algorithm 3.1. ***Bootstrap estimate of the standard error***

1. Having observed $\mathbf{X}_n = (X_1, X_2, \ldots, X_n)$, construct the EDF F_n.

2. Draw a bootstrap random sample $\mathbf{X}_n^* = (X_1^*, X_2^*, \ldots, X_n^*)$ from F_n. In other words, sample with replacement from $\mathbf{X}_n = (X_1, X_2, \ldots, X_n)$ with equal probability n^{-1}.

3. Calculate the bootstrap replication of the statistic $\hat{\theta}$, $\hat{\theta}^* = \hat{\theta}(X_1^*, X_2^*, \ldots, X_n^*)$, by using the bootstrap random sample.

4. Independently repeat steps 2 and 3 B times (B is a large number) to obtain bootstrap replications $\hat{\theta}^*(1), \hat{\theta}^*(2), \ldots, \hat{\theta}^*(B)$.

5. Calculate the bootstrap estimation of the standard error:

$$\hat{\sigma}_B = \sqrt{\frac{1}{B-1}\sum_{b=1}^{B}(\hat{\theta}^*(b) - \hat{\theta}^*(\cdot))^2}, \tag{3.4}$$

where

$$\hat{\theta}^*(\cdot) = \frac{1}{B}\sum_{b=1}^{B}\hat{\theta}^*(b).$$

Note that $\sigma_{F_n}(\hat{\theta}^*)$ in (3.3) denotes the ideal bootstrap estimate of $\sigma_F(\hat{\theta})$ in (3.2), while $\hat{\sigma}_B$ in (3.4) denotes the Monte Carlo approximation of $\sigma_{F_n}(\hat{\theta}^*)$. It can be shown that $\hat{\sigma}_B \to \sigma_{F_n}(\hat{\theta}^*)$ as $B \to \infty$. For estimation of the standard error a choice of B between 50 and 200 is usually sufficient (Efron and Tibshirani 1993, p. 52).

3.3.2 The bootstrap estimate of the bias

Define the bias of the estimator $\hat{\theta} = \hat{\theta}(X_1, X_2, \ldots, X_n)$ by

$$\beta_F(\hat{\theta}) = \mathrm{E}_F(\hat{\theta}) - \theta. \tag{3.5}$$

Suppose $\beta_F(\hat{\theta})$ needs to be estimated. By applying the plug-in principle, the ideal bootstrap estimate of the bias is given by

$$\beta_{F_n}(\hat{\theta}^*) = \mathrm{E}_{F_n}(\hat{\theta}^*) - \hat{\theta}, \tag{3.6}$$

where $\hat{\theta}^* = \hat{\theta}(X_1^*, X_2^*, \ldots, X_n^*)$ and $\hat{\theta} = \hat{\theta}(X_1, X_2, \ldots, X_n)$.

The following algorithm describes a Monte Carlo algorithm to obtain an approximation of the ideal bootstrap estimate:

Algorithm 3.2. ***Bootstrap estimate of the bias***

1. Having observed $\mathbf{X}_n = (X_1, X_2, \ldots, X_n)$, construct the EDF F_n.

2. Calculate $\hat{\theta} = \hat{\theta}(X_1, X_2, \ldots, X_n)$.

3. Draw a bootstrap random sample $\mathbf{X}_n^* = (X_1^*, X_2^*, \ldots, X_n^*)$ from F_n. In other words, sample with replacement from $\mathbf{X}_n = (X_1, X_2, \ldots, X_n)$ with equal probability n^{-1}.

4. Calculate the bootstrap replication of the statistic $\hat{\theta}$, i.e. $\hat{\theta}^* = \hat{\theta}(X_1^*, X_2^*, \ldots, X_n^*)$.

5. Independently repeat steps 3 and 4 B times (B is a large number) to obtain bootstrap replications $\hat{\theta}^*(1), \hat{\theta}^*(2), \ldots, \hat{\theta}^*(B)$.

6. Calculate the bootstrap estimation of the bias:

$$\hat{\beta}_B = \hat{\theta}^*(\cdot) - \hat{\theta}, \tag{3.7}$$

where

$$\hat{\theta}^*(\cdot) = \frac{1}{B}\sum_{b=1}^{B} \hat{\theta}^*(b).$$

In this case $\beta_{F_n}(\hat{\theta}^*)$ in (3.6) denotes the ideal bootstrap estimate of $\beta_F(\hat{\theta})$ in (3.5), while $\hat{\beta}_B$ in (3.7) denotes the Monte Carlo approximation of $\beta_{F_n}(\hat{\theta}^*)$. It holds that $\hat{\beta}_B \to \beta_{F_n}(\hat{\theta}^*)$ as $B \to \infty$.

Efron and Tibshirani (1993, p. 130) show that B should be chosen quite large if one wants to trust $\hat{\beta}_B$ as an estimate for $\beta_F(\hat{\theta})$. To overcome this, they propose an improved bootstrap estimate of the bias. See Efron and Tibshirani (1993, Section 10.4) for more details on the improved method.

3.3.3 The double bootstrap

When calculating bootstrap approximations of the distributional properties of an estimator $\hat{\theta}$ the question, 'how accurate and precise are these bootstrap estimates?' arises. In other words, one is interested in determining the standard error and bias of the bootstrap standard error and bias. In order to obtain such estimations the 'double bootstrap' can be applied – a method that involves resampling the resampled data and bootstrapping the bootstrap. To illustrate this idea we discuss an algorithm for the estimation of the standard error of the bootstrap standard error.

Recall from (3.3) that the ideal bootstrap estimate of the standard error of a statistic $\hat{\theta}$ is given by

$$\sigma_{F_n}(\hat{\theta}^*) = \sqrt{\mathrm{Var}_{F_n}(\hat{\theta}^*)}.$$

Then the ideal bootstrap estimate of the standard error of $\sigma_{F_n}(\hat{\theta}^*)$ is given by

$$\sigma_{F_n}\left(\sigma_{F_n^*}(\hat{\theta}^{**})\right) = \sqrt{\mathrm{Var}_{F_n}(\hat{\sigma}^*)}, \tag{3.8}$$

where

$$\hat{\sigma}^* = \sigma_{F_n^*}(\hat{\theta}^{**}) = \sqrt{\mathrm{Var}_{F_n^*}(\hat{\theta}^{**})}.$$

Here $\sigma_{F_n^*}(\hat{\theta}^{**})$ denotes the bootstrap estimate of the standard error based on a second phase bootstrap sample $\mathbf{X}_n^{**} = (X_1^{**}, X_2^{**}, \ldots, X_n^{**})$, which is drawn with replacement from a first phase bootstrap sample $\mathbf{X}_n^* = (X_1^*, X_2^*, \ldots, X_n^*)$ and used to determine $\hat{\theta}^{**} = \hat{\theta}(X_1^{**}, X_2^{**}, \ldots, X_n^{**})$.

The Monte Carlo algorithm for this double bootstrap procedure is as follows:

Algorithm 3.3. ***The double bootstrap***

1. Having observed $\mathbf{X}_n = (X_1, X_2, \ldots, X_n)$, construct the EDF F_n.
2. Draw a bootstrap random sample $\mathbf{X}_n^* = (X_1^*, X_2^*, \ldots, X_n^*)$ from F_n. In other words, sample with replacement from $\mathbf{X}_n = (X_1, X_2, \ldots, X_n)$ with equal probability n^{-1}. For the rest of step 2, consider the bootstrap sample $\mathbf{X}_n^*$ as the original sample.
   (a) Having observed $\mathbf{X}_n^* = (X_1^*, X_2^*, \ldots, X_n^*)$, construct the bootstrap EDF F_n^*.
   (b) Draw a double bootstrap random sample $\mathbf{X}_n^{**} = (X_1^{**}, X_2^{**}, \ldots, X_n^{**})$ from F_n^*. In other words, sample with replacement from $\mathbf{X}_n^* = (X_1^*, X_2^*, \ldots, X_n^*)$ with equal probability n^{-1}.
   (c) Calculate the double bootstrap replication of the statistic $\hat{\theta}^*$, $\hat{\theta}^{**} = \hat{\theta}(X_1^{**}, X_2^{**}, \ldots, X_n^{**})$, i.e. by using the double bootstrap random sample.
   (d) Independently repeat steps (b) and (c) R times, obtaining double bootstrap replications, $\hat{\theta}^{**}(1), \hat{\theta}^{**}(2), \ldots, \hat{\theta}^{**}(R)$.
   (e) Calculate:

$$\hat{\sigma}_R^* = \sqrt{\frac{1}{R-1}\sum_{r=1}^{R}(\hat{\theta}^{**}(r) - \hat{\theta}^{**}(\cdot))^2},$$

   where

$$\hat{\theta}^{**}(\cdot) = \frac{1}{R}\sum_{r=1}^{R}\hat{\theta}^{**}(r).$$

3. Independently repeat step 2 B times, obtaining bootstrap replications $\hat{\sigma}_R^*(1), \hat{\sigma}_R^*(2), \ldots, \hat{\sigma}_R^*(B)$.

4. Calculate:

$$\hat{\hat{\sigma}}_B = \sqrt{\frac{1}{B-1}\sum_{b=1}^{B}(\hat{\sigma}_R^*(b) - \hat{\sigma}_R^*(\cdot))^2},$$

where

$$\hat{\sigma}_R^*(\cdot) = \frac{1}{B}\sum_{b=1}^{B}\hat{\sigma}_R^*(b).$$

For other applications of the double bootstrap, such as bias correction, see Davison and Hinkley (1997, p. 103).

3.4 Bootstrap in regression

Regression analysis is the statistical methodology that studies the relationship between two or more quantitative variables. A regression curve describes the relationship between a vector of explanatory variables $\mathbf{X}$ and a response variable Y. The regression function is the average value of Y given the observed values of $\mathbf{X}$ (Härdle 1990, p. 3,4). We now provide brief excerpts from Chapter 1 to place the application of the bootstrap method in regression procedures in context.

Formally, suppose n data points $\{(\mathbf{X}_i, Y_i)\}_{i=1}^n$ have been collected. The regression relationship can be formulated as

$$Y_i = m(\mathbf{X}_i) + \varepsilon_i, \ i = 1, \ldots, n, \tag{3.9}$$

where

$$m(\mathbf{x}) = E(Y|\mathbf{X} = \mathbf{x})$$

is the unknown regression function, i.e. the average value of Y given the observed values of $\mathbf{X}$. The ε_i's are independent random variables with mean 0 and variance σ^2. The sample of size n is used to obtain a useful estimate of the regression equation, i.e. to determine $\hat{m}(\mathbf{x})$ (Kutner et al. 2005).

This estimation problem can be approached in one of two ways: parametrically or non-parametrically. A parametric regression model assumes that the form of m is known except for finitely many unknown parameters. More specifically it is assumed that $m(\mathbf{x}) = m(\mathbf{x}; \boldsymbol{\beta})$,

with $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)$, $0 < p < \infty$. By applying an appropriate estimation methodology, such as least squares,

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} [Y_i - m(\mathbf{X}_i; \boldsymbol{\beta})]^2,$$

it is possible to utilize the data to estimate the parameters $\boldsymbol{\beta}$ by $\hat{\boldsymbol{\beta}}$ and thereby obtain an estimate of m. The bootstrap can be used to get insight into the sampling distribution of $\hat{\boldsymbol{\beta}}$ (Efron and Tibshirani 1993, Chapter 9).

A nonparametric regression model generally only assumes that m belongs to some infinite dimensional collection of functions. There are various methods to obtain a nonparametric regression estimate of m. One possibility is local versions of location estimators, where large weights are given to observations in a small neighbourhood around $\mathbf{x}$ and small or no weight to points far away from $\mathbf{x}$. For a univariate predictor variable x, this procedure can be defined as

$$\hat{m}(x) = \frac{1}{n} \sum_{i=1}^{n} W_{ni}(x) Y_i,$$

where $\{W_{ni}(x)\}_{i=1}^{n}$ denotes a sequence of weights. The weight sequence $\{W_{ni}(x)\}_{i=1}^{n}$ is tuned by a smoothing parameter which regulates the size of the neighbourhood around x. The smoothing parameter is often estimated by using sample data (Härdle 1990). Through the bagging and bragging procedures, bootstrap methods are used to improve the choice of the smoothing parameter (see for example Hall and Robinson (2009)).

Bootstrap in regression can be done in one of two ways: bootstrapping residuals or bootstrapping pairs. We will illustrate these methods by applying it to the estimated parametric model $m(\mathbf{x}; \hat{\boldsymbol{\beta}})$ where the distributional properties of $\hat{\boldsymbol{\beta}}$ are of interest. Specifically, we will consider the standard errors of $\hat{\boldsymbol{\beta}}$. The discussion is based on Efron and Tibshirani (1993, Chapters 8 and 9) and Davison and Hinkley (1997, Chapters 6 and 7).

3.4.1 Bootstrapping residuals

Suppose that the errors in (3.9) has an underlying distribution G. The idea of the bootstrap residuals (or model-based) approach is to calculate the sample residuals

$$\hat{\varepsilon}_i = Y_i - m(\mathbf{X}_i, \hat{\boldsymbol{\beta}}), i = 1, \ldots, n \tag{3.10}$$

where $\hat{\boldsymbol{\beta}}$ was determined by some estimation technique, to center these residuals and then to construct the empirical distribution G_n of the centered residuals. Bootstrap residuals are obtained by resampling from G_n, i.e. by sampling with replacement from the centered

residuals. These resampled residuals are used to construct the bootstrap sample from which bootstrap estimates of the regression coefficients $\hat{\boldsymbol{\beta}}^*$ are obtained. The distributions of these bootstrap estimates of the regression coefficients serve as approximations of the sampling distributions of the estimated regression coefficients $\hat{\boldsymbol{\beta}}$.

The following Monte Carlo algorithm captures the process above, using bootstrapped residuals to estimate the standard errors of $\hat{\boldsymbol{\beta}}$.

Algorithm 3.4. ***Bootstrapping residuals***

1. Suppose that the sample $\mathbf{S}_n = ((\mathbf{X}_1, Y_1), (\mathbf{X}_2, Y_2), \ldots, (\mathbf{X}_n, Y_n))$ is given.

2. Use the sample to estimate $\boldsymbol{\beta}$ with some estimation technique. For example, if the method of least squares is used, determine

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} [Y_i - m(\mathbf{X}_i; \boldsymbol{\beta})]^2.$$

3. Calculate the sample residuals

$$\hat{\varepsilon}_i = Y_i - m(\mathbf{X}_i, \hat{\boldsymbol{\beta}}), i = 1, \ldots, n.$$

4. Center the residuals

$$e_i = \hat{\varepsilon}_i - \bar{\varepsilon},\ i = 1, 2, \ldots, n$$

where $\bar{\varepsilon} = \dfrac{1}{n} \sum_{i=1}^{n} \hat{\varepsilon}_i$ to obtain $\mathbf{e}_n = (e_1, e_2, \ldots, e_n)$ and construct the EDF G_n of the centered residuals.

5. Draw a bootstrap random sample $\mathbf{e}_n^* = (e_1^*, e_2^*, \ldots, e_n^*)$ from G_n. In other words, sample with replacement from $\mathbf{e}_n = (e_1, e_2, \ldots, e_n)$ with equal probability n^{-1}.

6. Obtain the bootstrap sample $\mathbf{S}_n^* = ((\mathbf{X}_1, Y_1^*), (\mathbf{X}_2, Y_2^*), \ldots, (\mathbf{X}_n, Y_n^*))$ where

$$Y_i^* = m(\mathbf{X}_i, \hat{\boldsymbol{\beta}}) + e_i^*,\ i = 1, \ldots, n.$$

7. Calculate the bootstrap estimate of the coefficient vector $\hat{\boldsymbol{\beta}}$ by using the bootstrap random sample $\mathbf{S}_n^*$. In the case of least squares estimation this implies

$$\hat{\boldsymbol{\beta}}^* = \arg\min_{\hat{\boldsymbol{\beta}}} \sum_{i=1}^{n} [Y_i^* - m(\mathbf{X}_i; \hat{\boldsymbol{\beta}})]^2.$$

8. Independently repeat steps 5 to 7 B times, obtaining bootstrap replications $\hat{\boldsymbol{\beta}}^*(1)$, $\hat{\boldsymbol{\beta}}^*(2), \ldots, \hat{\boldsymbol{\beta}}^*(B)$.

9. Calculate the bootstrap estimation of the standard error of the j^{th} estimated regression coefficient $\hat{\beta}_j$ as

$$\hat{\sigma}_{j,B} = \sqrt{\frac{1}{B-1}\sum_{b=1}^{B}(\hat{\beta}_j^*(b) - \hat{\beta}_j^*(\cdot))^2}, j = 0, 1, \ldots, p, \tag{3.11}$$

where

$$\hat{\beta}_j^*(\cdot) = \frac{1}{B}\sum_{b=1}^{B}\hat{\beta}_j^*(b).$$

Note that $\hat{\sigma}_{j,B} \to \hat{\sigma}_j$ as $B \to \infty$ where $\hat{\sigma}_j$ denotes the ideal bootstrap estimate of the standard error of the j^{th} estimated regression coefficient.

3.4.2 Bootstrapping pairs

The resampling of residuals in the previous section has the disadvantage that it relies greatly on the assumption that the error terms ε_i have equal variances. That is, the error between Y_i and its conditional expectation $m(\mathbf{X}_i; \hat{\boldsymbol{\beta}})$ has the same distribution G, regardless what the values of $\mathbf{X}_i$ are. This is a serious assumption and can fail even if the model for the conditional expectation is correct. To counter problems that may arise from this assumption, pairs of data can be resampled, rather than the residuals.

The bootstrapping pairs (or bootstrapping cases) approach assumes that the data is a random sample from some bivariate distribution F of $(\mathbf{X}, Y)$. No assumptions about the random errors in (3.9) are made, except that they are independent. Note especially that no assumptions are made about the equality of the error variances - heteroscedasticity can be accommodated. However, if homoscedasticity holds, bootstrapping pairs are not as efficient as bootstrapping residuals (Davison and Hinkley 1997, p. 264).

Take F to be the bivariate distribution of $(\mathbf{X}, Y)$. Denote the EDF of the data pairs by F_n. Resampling will be done from this EDF, i.e. resampling involves sampling with replacement from $\mathbf{S}_n = ((\mathbf{X}_1, Y_1), (\mathbf{X}_2, Y_2), \ldots, (\mathbf{X}_n, Y_n))$ to obtain bootstrap samples $\mathbf{S}_n^* = ((\mathbf{X}_1^*, Y_1^*), (\mathbf{X}_2^*, Y_2^*), \ldots, (\mathbf{X}_n^*, Y_n^*))$. Bootstrap versions of the regression coefficient estimates $\hat{\boldsymbol{\beta}}^*$ are calculated by applying the same least squares algorithm to the bootstrap sample $\mathbf{S}_n^*$ as was used to obtain the original estimates $\hat{\boldsymbol{\beta}}$ when it was applied to $\mathbf{S}_n$.

The Monte Carlo algorithm that describes the process of using bootstrapping pairs to estimate the standard errors of $\hat{\boldsymbol{\beta}}$ is as follows:

Algorithm 3.5. ***Bootstrapping pairs***

1. Suppose that the sample $\mathbf{S}_n = ((\mathbf{X}_1, Y_1), (\mathbf{X}_2, Y_2), \ldots, (\mathbf{X}_n, Y_n))$ is given.

2. Draw a bootstrap random sample $\mathbf{S}_n^* = ((\mathbf{X}_1^*, Y_1^*), (\mathbf{X}_2^*, Y_2^*), \ldots, (\mathbf{X}_n^*, Y_n^*))$ from F_n. In other words, sample pairs of data with replacement from $\mathbf{S}_n = ((\mathbf{X}_1, Y_1), (\mathbf{X}_2, Y_2), \ldots, (\mathbf{X}_n, Y_n))$.

3. Calculate the bootstrap estimate of the coefficient vector $\hat{\boldsymbol{\beta}}$ by using the bootstrap random sample $\mathbf{S}_n^*$. In the case of least squares estimation, this implies

$$\hat{\boldsymbol{\beta}}^* = \arg\min_{\hat{\boldsymbol{\beta}}} \sum_{i=1}^{n} [Y_i^* - m(\mathbf{X}_i^*; \hat{\boldsymbol{\beta}})]^2.$$

4. Independently repeat steps 2 and 3 B times, obtaining bootstrap replications $\hat{\boldsymbol{\beta}}^*(1), \hat{\boldsymbol{\beta}}^*(2), \ldots, \hat{\boldsymbol{\beta}}^*(B)$.

5. Calculate the bootstrap estimation of the standard error of the j^{th} estimated regression coefficient $\hat{\beta}_j$ as

$$\hat{\sigma}_{j,B} = \sqrt{\frac{1}{B-1} \sum_{b=1}^{B} (\hat{\beta}_j^*(b) - \hat{\beta}_j^*(\cdot))^2}, j = 0, 1, \ldots, p, \tag{3.12}$$

where

$$\hat{\beta}_j^*(\cdot) = \frac{1}{B} \sum_{b=1}^{B} \hat{\beta}_j^*(b).$$

Note that once again $\hat{\sigma}_{j,B} \rightarrow \hat{\sigma}_j$ as $B \rightarrow \infty$ where $\hat{\sigma}_j$ denotes the ideal bootstrap estimate of the standard error of the j^{th} estimated regression coefficient.

The bootstrap method is applicable in many fields. In this chapter some very basic ideas of the bootstrap methodology were defined and illustrated, to serve as introduction for the discussion of the bagging and bragging methods in the next chapter. Bagging and bragging (Swanepoel (1990), Breiman (1996a)) are bootstrap procedures developed to reduce the variance of statistical estimates and utilize bootstrap samples generated by resampling data pairs. We will henceforth use the pairs bootstrap approach.

Chapter 4

Bagging

4.1 Introduction

In this study the aim is to show how to improve the Nadaraya-Watson regression estimator by applying the boosting, bagging and bragging improvement methods. The boosting method was discussed in Chapter 2 and in Chapter 3 the bootstrap method, which underlie bagging and bragging, was introduced. The goal of this chapter is to present the bagging and bragging methodology.

We start our discussion by introducing fundamental ideas regarding the bagging and bragging methods in the rest of this introductory paragraph. The boosting (Schapire 1990 and Freund 1995), bagging and bragging (Breiman 1996a and Swanepoel 1988, 1990) methods can be seen as ensemble methods. Some thoughts on ensemble methods are mentioned in Section 4.2. Experimentally, bagging seems to be effective in improving unstable predictors. Theoretical analysis confirms that bagging can reduce the variability of unstable predictors. The bagging methodology and modifications such as bragging are presented in Section 4.3. In this study we are particularly interested in applying bagging and bragging to nonparametric regression methods, such as kernel regression. For kernel methods, the choice of smoothing parameter is a matter of concern. This step can be performed by using the method of cross-validation or other methods such as the plug-in method. In this study we attempt smoothing parameter selection via cross-validation and in Section 4.4 literature on the application of bagging to cross-validation bandwidth selection in kernel estimation is reviewed.

We now introduce the basic idea of bagging and bragging along the lines of Swanepoel (1988, 1990).

Swanepoel (1988, 1990) discusses the problem of constructing estimates for a parameter

θ, which can be written in the form

$$\theta = \psi(F)$$

for some suitable functional ψ. From this functional approach it is shown that if $\psi(F)$ can be approximated by a sequence of functionals, namely

$$\psi_m(F) \approx \psi(F),$$

with the approximation becoming increasingly accurate as $m \to \infty$, then $\psi_m(F_n)$ can be taken as an estimator of θ, with $m = m(n)$ suitably chosen (F_n denotes the EDF defined in (3.1)).

Suppose $T_m(X_1, X_2, \ldots, X_m)$ is some known estimator of θ, such as the sample mean. Then two possible choices of $\psi_m(F)$ are

$$\psi_{m,1}(F) = E[T_m(X_1, X_2, \ldots, X_m)] \tag{4.1}$$

and

$$\psi_{m,2}(F) = \text{med}[T_m(X_1, X_2, \ldots, X_m)], \tag{4.2}$$

where "med" indicates the median of a random variable. In this case we have that

$$\psi_{m,1}(F_n) = E_*[T_m(X_1^*, X_2^*, \ldots, X_m^*)] \tag{4.3}$$

and

$$\psi_{m,2}(F_n) = \text{med}_*[T_m(X_1^*, X_2^*, \ldots, X_m^*)] \tag{4.4}$$

respectively, where $(X_1^*, X_2^*, \ldots, X_m^*)$ is a bootstrap random sample of size m taken from F_n. Usually the ideal bootstrap estimates $\psi_{m,1}(F_n)$ and $\psi_{m,2}(F_n)$ are approximated by Monte Carlo algorithms, as introduced by Efron (1979).

Note that the choice of $\psi_{m,1}(F_n)$ in (4.3) is currently known as bagging in statistical literature and $\psi_{m,2}(F_n)$ in (4.4) is known as bragging, both of which will be discussed later in this chapter.

It is also proved that $\psi_m(F_n)$ is strongly consistent and asymptotically normally distributed as $n \to \infty$. The estimators have other robust properties which will not be pursued further in this discussion. See Swanepoel (1988, 1990) for more on this topic.

It should be emphasized that the above functional approach by Swanepoel (1988, 1990) is very general and can be applied to almost all areas in statistics, for example point estimation, interval estimation, regression and machine learning.

It is suggested that $m = n/3$ is a good choice from a mean squared error point of view (Swanepoel 1988, 1990).

In the next section we shall proceed with presenting the bagging and bragging procedures as it developed from the theory of ensemble methods.

4.2 Ensemble methods

Advances in data collection and computer technologies have led to a rapid increase in the number of large datasets and methods to analyze these datasets. Computationally intensive procedures have been developed that are especially useful for high-dimensional dataset problems (Bühlmann and Yu 2002, p. 927). Under these developments are "ensemble learning algorithms". Ensemble methods train multiple component learners and then combine their predictions. The generalization ability of an ensemble is often significantly better than that of a single learner (Zhou and Yang 2005, p. 48).

The principle of ensemble methods is to construct a linear combination of some model fitting method, instead of a single fit of the method (Bühlmann 2004, p. 878). Consider a set of realizations of independent, identically distributed (i.i.d.) random variables $S = (X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)$, where $X_1, X_2, \ldots, X_n$ are one-dimensional or p-dimensional predictor variables and $Y_1, Y_2, \ldots, Y_n$ are one-dimensional response variables, which can be either continuous (regression problem) or discrete (classification problem). This is the random design model. In the fixed design model a set of fixed, ordered points $x_1, x_2, \ldots, x_n$ is given, resulting in the sample space $S = (x_1, Y_1), (x_2, Y_2), \ldots, (x_n, Y_n)$.

Given a new explanatory covariate x, a predictor for the corresponding response variable is denoted by

$$\hat{\theta}_n(x) = l_n\{(X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)\}(x), \tag{4.5}$$

where $l_n\{(X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)\}(x)$ denotes some base procedure which yields the estimated function $\hat{\theta}_n(x)$, given the data. Examples of base procedures in the regression context is nonparametric kernel estimators when p is small, or nonparametric statistical methods with some structural restrictions, such as regression trees, when $p \geq 2$. For a classification problem, the base procedure can for example be class-probability estimates from a classification tree (Bühlmann 2004, p. 878).

The original idea of ensemble methods is to run the base procedure multiple times by using reweighted original data to obtain different estimates $\hat{\theta}_{n;1}(x), \hat{\theta}_{n;2}(x), \ldots, \hat{\theta}_{n;T}(x)$, where

$\hat{\theta}_{n;t}(x)$ is the obtained estimator when applying the base procedure on the t^{th} reweighted dataset, $t = 1, 2, \ldots, T$. An ensemble-based function estimate is constructed by taking linear combinations of the different estimates:

$$\hat{\theta}_{n,ens}(x) = \sum_{t=1}^{T} \alpha_t \hat{\theta}_{n;t}(x)$$

(Bühlmann 2004, p. 878).

Bagging, introduced by Swanepoel (1988, 1990) in functional context and by Breiman (1996a) from a machine learning viewpoint, and boosting, which is due to Schapire (1990) and Freund (1995), are examples of ensemble methods. For bagging, the linear combination coefficients $\alpha_t = 1/T, t = 1, 2, \ldots, T$ are averaging weights, whereas for boosting, $\sum_{t=1}^{T} \alpha_t$ increases as T increases (Bühlmann 2004, p. 878).

Ensemble methods are relatively simple methods to improve the predictive performance of a base procedure. The reasons for improvement differ, for example in bagging the improvement is mainly due to a reduction in the variance, while boosting chiefly reduces the bias. This indicates that bagging and boosting are quite different ensemble methods. Boosting can also be viewed from another perspective, namely the functional gradient descent view of boosting (see Chapter 2.4). This non-ensemble perspective on boosting has huge interpretational advantages over seeing it as an ensemble method (Bühlmann 2004, p. 878).

As mentioned before, bagging is discussed in this chapter and the boosting method was discussed in Chapter 2.

4.3 The bagging methodology

Bagging, an acronym for **b**ootstrap **agg**regat**ing**, is a method involving the generation of multiple versions of a predictor. These replicates are used to get an aggregated predictor. When a numerical outcome is predicted, as in the regression setting, the aggregation calculates an average over the multiple versions. The multiple versions are obtained by applying the bootstrap principle, i.e. by drawing bootstrap samples from the original training set and using these bootstrap samples as new training sets. The bootstrap methodology was discussed briefly in Chapter 3.

The goal of bagging is to reduce the variance of a predictor and in this way improve the accuracy. For bagging to succeed in this aim, the procedure that constructs each predictor must be unstable, i.e. changes in the training set should significantly change the predictor that is constructed (Peterson, Molinaro, Sinisi and Van der Laan 2007, p. 1696). In this

section we consider the bagging methodology and how it is applied to reduce the variability of unstable predictors. We also discuss some modifications of the method, such as bragging.

First, we present an algorithm for bagging. We keep to general notation in the algorithm, in order to include applicability of the algorithm to both classification and regression problems. Application of the algorithm to the classification and regression contexts respectively, are discussed at the end of Section 4.3.1.

4.3.1 The bagging algorithm

Assume the situation described in Section 4.2, i.e. having a predictor for the response variable denoted by

$$\hat{\theta}_n(x) = l_n\{(X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)\}(x), \tag{4.6}$$

where $l_n\{(X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)\}(x)$ denotes some base procedure which yields the estimated function $\hat{\theta}_n(x)$, given the data. Theoretically, the bagging algorithm can be defined as follows (Bühlmann and Yu 2002, p. 928):

Algorithm 4.1. ***Bagging***

1. Construct a bootstrap sample $S^* = \{(X_1^*, Y_1^*), (X_2^*, Y_2^*), \ldots, (X_n^*, Y_n^*)\}$, according to the empirical distribution of the pairs $\{(X_i, Y_i)\}_{i=1}^n$ in S.

2. Fit $\hat{\theta}_n(x)$ to S^* to obtain the bootstrap model $\hat{\theta}_n^*(x)$ by using the plug-in principle: $\hat{\theta}_n^*(x) = l_n\{(X_1^*, Y_1^*), (X_2^*, Y_2^*), \ldots, (X_n^*, Y_n^*)\}(x)$, where $\hat{\theta}_n$ is defined in (4.6).

3. The bagged predictor is $\hat{\theta}_{n,Bag}(x) = E^*[\hat{\theta}_n^*(x)] = E[\hat{\theta}_n^*(x)|S]$.

This is exactly the procedure of Swanepoel (1988,1990), except for the bootstrap sample size, which Swanepoel takes as $m = n/3$, although in Swanepoel (1986) a rule of thumb recommends that $m = 2n/3$.

Generally, the exact bootstrap expectation of the predictor cannot be expressed analytically and a Monte Carlo algorithm is used to approximate the exact bootstrap expectation. Thus, in practice, the bootstrap expectation in the third step is implemented by a Monte Carlo procedure (Bühlmann and Yu 2002, p. 928): fix the number of bootstrap samples that will be drawn and denote it by B. The choice of B depends on the sample size and the computational cost involved in evaluating the predictor. Breiman (1996a, p. 135) suggests

that less replicates are required for bagging in the regression setting than in the classification setting. He uses 25 replications for regression and 50 for classification. This number should be adapted to suit the specific problem at hand. Repeat step 1 in the bagging algorithm above B times to obtain B bootstrap replicates of the sample, each denoted by $\{(X_1^*, Y_1^*), (X_2^*, Y_2^*), \ldots, (X_n^*, Y_n^*)\}$. For every bootstrap simulation $b \in 1, \ldots, B$, calculate the bootstrap predictor in step 2, i.e. $\hat{\theta}_{n;b}^*(x)$.

Next, the multiple versions of the predictor should be aggregated. This aggregation depends on the nature of the problem:

1. Classification:

   In the classification set-up the response variable Y_i contains the label of the i^{th} case, which could be an element of $\{1, 2, \ldots, K\}$ for a K-class problem. The aim is to find an approximation to the classification rule defined by the probability function

   $$\{p_1^*(x), p_2^*(x), \ldots, p_K^*(x)\},$$

   where

   $$p_k^*(x) = P[\hat{\theta}_n^*(x) = k], k = 1, 2, \ldots, K.$$

   For this problem,

   $$\hat{\theta}_{n,Bag_{MC}}(x) \approx \arg\max_k d_k(x),$$

   where

   $$d_k(x) = \frac{1}{B} \sum_{b=1}^{B} I[\hat{\theta}_{n;b}^*(x) = k]$$

   (Pino-Mejías, Jiménez-Gamero, Cubiles-de-la Vega and Pascual-Acosta 2008, p. 266).

2. Regression:

   For the regression problem, the response variables Y_i, $i = 1, 2, \ldots, n$ are realizations of a real valued random variable (Pino-Mejías et al. 2008, p. 266). The aim is to obtain a predictor for the conditional mean response

   $$E[Y|X = x] = m(x).$$

   In this scenario the B computed models are averaged,

   $$\hat{\theta}_{n,Bag_{MC}}(x) \approx \frac{1}{B} \sum_{b=1}^{B} \hat{\theta}_{n;b}^*(x) \tag{4.7}$$

   (Bühlmann and Yu 2002, p. 928).

In this study we are particularly interested in the regression scenario, which will be addressed in the rest of this chapter.

Sometimes, the bagging algorithm is not very useful. Modifications of the bagging method can solve problems that arise when applying bagging.

4.3.2 Modifications

In the bagging algorithm described in Section 4.3.1, the bootstrap samples from the training set have empirical distribution functions which are in a "neigbourhood" of the empirical distribution function corresponding to the training sample (Pino-Mejías et al. 2008, p. 266). The resulting procedure does not work in all situations (Bühlmann and Yu 2002, p. 929). A more general bagging methodology can be defined, where other classes of neighbourhoods of the empirical distribution of the training sample are considered, or where aggregation is done by another rule than averaging (Pino-Mejías et al. 2008, p. 266). These modifications include sampling without replacement, using resample sizes that differ from the original training set's size and aggregating using the median over the multiple versions of the predictor, rather than the mean. We now discuss such modifications of the bagging method.

a) Subagging and moon-bagging

Conventional bootstrap obtains bootstrap replicates by drawing n data points with replacement from the original training dataset of size n. According to Bühlmann and Yu (2002), this procedure can be altered by resampling without replacement, rather than with replacement. Another modification is to choose a resample size m that differs from the original sample size n, i.e. the m-out-of-n way of bootstrapping (Swanepoel 1986).

- For subagging (**subsample aggregating**) a sample size $m = [an]$ is used, with $0 < a < 1$ ($[y]$ denotes the largest integer value smaller than or equal to y). Resampling is done without replacement. Besides for theoretical advantages, subagging is also computationally more effective than bagging, because the original predictor is evaluated many times for m instead of n data points and $m < n$ (Bühlmann and Yu 2002, p. 929).

  Note that when subagging is applied and $m = n$ is chosen, no averaging takes place, since the empirical distribution function of the bootstrap replicates is exactly the same as the empirical distribution function of the original training sample (Buja and Stuetzle 2006, p. 327).

- Moon-bagging provides an alternative to subagging and is an acronym for **m-out-of-n boostrap aggregating**. It involves replacing the bootstrap step in Algorithm 4.1 (i.e. step 1) by the m-out-of-n bootstrap, as discussed by Swanepoel (1986). For this procedure a sample size $m = [an]$ is used, with $0 < a < 1$, where resampling is done with replacement.

  The difference between subagging and moon-bagging essentially disappears if m is small relative to n (Bühlmann and Yu 2002, p. 946).

  Barutçuoğlu and Alpaydin (2003, p. 76) explain that the particular bootstrap sample size m being used has an effect on the performance of bagging. The optimal ratio of m to n depends on the data. One way to find this ratio, is to manually finetune it for a specific dataset. The process can also be automated. They propose a procedure called "best-ratio bagging" that removes a fraction of the training set for validation, performs bagging multiple times with different ratios on the remaining training data and then chooses the bagging model with the lowest error on the validation set as final model. This method is computationally quite expensive.

  An interesting and often good choice in practice is $m = [n/2]$. This method is generally known as "half subagging". It saves more than half of the computing time, because the computational order of an estimator $\hat{\theta}_n$ is usually at least linear in n. In the case when $\hat{\theta}_n$ is a U-statistic, half subagging is exactly equivalent to bagging. When $\hat{\theta}_n$ is a decision tree, half subagging yields empirical results that are very similar to bagging (Bühlmann 2004, p. 884).

  Finally, the resample size m may also be chosen to be larger than n. In fact, m can attain any value between 1 and ∞. In the case of $m > n$, resampling must be done with replacement. Computationally, choosing $m > n$ is not desirable. However, $m = \infty$ is the conceptually plausible upper bound on m: in this case no averaging takes place, since the empirical distribution function of the bootstrap replicates is then exactly the same as the empirical distribution function of the original training sample (Buja and Stuetzle 2006, p. 327).

b) Bragging, trimmed bagging and nice bagging

Another way to extend bagging is to calculate a robust location estimator instead of the average over the bootstrap predictors as in (4.7). Bragging, trimmed bagging and nice bagging are examples of robust methods.

- Bühlmann (2003) suggests that the median could be computed and called the procedure bragging:

$$\hat{\theta}_{n,Brag_{MC}}(x) \approx \text{median}_{1\leq b\leq B}\hat{\theta}^*_{n;b}(x).$$

Swanepoel (1988, 1990) also suggested this approach.

- Croux, Joossens and Lemmens (2007) propose trimmed bagging, a procedure where aggregation is done over only the "most accurate" predictors. Their focus falls on the classification problem, where the out-of-bag or out-of-bootstrap error rate is used to find the most accurate predictors. Out-of-bag error measures involve using out-of-bag samples and corresponding out-of-bag predictions. Breiman (1996c) introduces these out-of-bag methods: suppose that $S = (X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)$ is the training sample that should be used to determine the prediction $\hat{\theta}_n(x)$. In bagging, B bootstrap training sets $S^*_1, S^*_2, \ldots, S^*_B$ are used to determine B bootstrap predictions $\hat{\theta}^*_{n;1}(x), \hat{\theta}^*_{n;2}(x), \ldots, \hat{\theta}^*_{n;B}(x)$. In each bootstrap sample, some data points will appear multiple times, while other observations will be missing. The observations that are missing form the out-of-bag sample for that particular bootstrap sample. Denote the array of out-of-bag samples by $S^{-*}_1, S^{-*}_2, \ldots, S^{-*}_B$. The out-of-bag error rate $\text{ER}(\hat{\theta}^*_{n;b}(x))$ is defined as the proportion of data points in the out-of-bag sample that are misclassified (Croux et al. 2007, p. 364).

Trimmed bagging involves the following (Croux et al. 2007): first, order the bootstrap versions of the classifier by increasing out-of-bag error rate,

$$\text{ER}(\hat{\theta}^*_{n;(1)}(x)) \leq \text{ER}(\hat{\theta}^*_{n;(2)}(x)) \leq \ldots \leq \text{ER}(\hat{\theta}^*_{n;(B)}(x)).$$

Trim off the α% bootstrap estimates that perform worst, i.e. that have the largest error rates, and aggregate over only the best performing estimates. Then calculate the trimmed bagged estimator

$$\hat{\theta}_{n,Trimbag_{MC}}(x) \approx \text{average}_{1\leq b\leq [(1-\alpha)B]}\hat{\theta}^*_{n;(b)}(x), \tag{4.8}$$

where $[a]$ denotes the largest integer smaller than or equal to a.

For the regression case, the out-of-bag mean squared prediction error can be used (Croux et al. 2007, p. 367).

The trimming portion α should be chosen small enough, such that averaging is still done over a substantial number of learners. Otherwise the variance will not be reduced,

which is the primary aim of bagging. However, choosing α too small will cause "bad" learners with large error rates to be included. Croux et al. (2007, p. 364) choose $\alpha = 0.25$ as a compromise.

The trimmed bagging estimator in (4.8) gives equal weight to all of the "best performing" bootstrap estimates when calculating the trimmed average $\hat{\theta}_{n,Trimbag_{MC}}(x)$. Alternatively, other weighting schemes can be applied in trimmed bagging, where the weights of learners are taken to be inversely proportional to their error rates. It seems as if these complex weighted trimmed averages do not perform significantly better than simple trimmed bagging (Croux et al. 2007, p. 364).

- Finally, consider an alteration of trimmed bagging, where the choice of α is data driven. This method averages over only the predictors that perform better than the initial learner $\hat{\theta}_n(x)$. This method corresponds to the "nice bagging" procedure, proposed by Skurichina and Duin (1998): let B' denote the largest b for which

$$\mathrm{ER}[\hat{\theta}^*_{n;(b)}(x)] < \mathrm{ER}[\hat{\theta}_n(x)].$$

The new estimator is then given by

$$\hat{\theta}_{n,Nicebag_{MC}}(x) \approx \mathrm{average}_{1 \leq b \leq B'} \hat{\theta}^*_{n;(b)}(x).$$

4.3.3 Why and when does bagging work?

Bagging was introduced by Swanepoel (1988, 1990) and Breiman (1996a) as a method to reduce the mean squared error (MSE) of learning algorithms. Breiman (1996a) showed empirically that bagging can indeed be successful in lowering the MSE and provided the following heuristic explanation: for learning methods that are "unstable" functions of the data, averaging over the bootstrap samples reduces the variance component of the MSE. In other words, the variance of the bagged estimator $\hat{\theta}_{n;Bag}(x)$ is smaller than or equal to the variance of the base learner $\hat{\theta}_n(x)$. If the original predictor is "unstable", this reduction is drastic. On the other hand, the bias component of the MSE of the bagged learner stays roughly the same as the bias of the initial learner. This leads to a lower MSE for "unstable" learners, whereas the MSE stays roughly the same for "stable" predictors (Bühlmann and Yu 2002, p. 928).

As mentioned above, bagging is effective when learners are "unstable". Breiman (1996b, p. 2354) heuristically defines "instability" as follows: a predictor is unstable if small changes

in the training sample can cause large changes in the predicted value(s). Bühlmann and Yu (2002, p. 928) formulate this definition more formally: a statistic $\hat{\theta}_n(x) = l_n\{(X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)\}(x)$ is called stable at x if $\hat{\theta}_n(x) = \theta(x) + o_p(1)$ as $n \to \infty$ for some fixed value $\theta(x)$. Note that this definition differs from the definition of consistency in that the value θ here is only a stable limit and not necessarily the parameter of interest. Instability occurs, according to this definition, if a predictor does not converge to a fixed value: another realization from the distribution that generated the data would result in a different predictor with positive probability. Neural networks, classification and regression trees and subset selection methods are examples of unstable predictors, while k nearest neighbour methods are stable (Peterson et al. 2007, p. 1696).

Hall and Robinson (2009, p. 176) point out that several investigations on the statistical properties of bagging have been done: Borra and Di Ciaccio (2001) studied bagging of nonparametric regression estimates, Bühlmann and Yu (2002) investigated theoretical arguments on why bagging and subagging work, Buja and Stuetzle (2006) studied the statistical properties when $\hat{\theta}_n(x)$ is a U-statistic and Friedman and Hall (2007) studied the effect of bagging on nonlinear aspects of statistical estimators.

4.4 Bagging of the cross-validation method

4.4.1 Introduction

As mentioned before, the bagging method is used to reduce the variability of unstable estimators. From Chapter 1.6.1 we know that cross-validation is a very useful tool for choosing the smoothing parameter in a wide variety of contexts. Cross-validated parameters, however, have relatively high variance, which is widely seen as an impediment to good performance. Hall and Robinson (2009) argue that the application of bagging to cross-validated parameters can reduce their variability in a simple, significant and reliable way. In this section we present their arguments with specific reference to the regression context.

4.4.2 Reducing of variability

According to Hall and Robinson (2009, p. 175), problems caused by the great variability of cross-validated parameters are evident from theoretical analysis. Consider for instance empirical 'plug-in' bandwidth selectors and denote them by $\hat{h}_{plug}$. Recall from Chapter 1.6.1 that a plug-in estimator is obtained by finding the theoretical expression of the bandwidth

that will minimize an asymptotic discrepancy measure such as the MSE and then estimating the unknown parameters in this expression by using the data at hand. There exist a variety of these selectors that are 'relatively root-n consistent' for the optimal bandwidth h_0 in the sense that

$$\frac{\hat{h}_{plug} - h_0}{h_0} \to 0$$

at a rate of $n^{-1/2}$. In comparison, for the cross-validation bandwidth $\hat{h}_{CV}$ it holds that

$$\frac{\hat{h}_{CV} - h_0}{h_0} \to 0$$

at much slower rate of $n^{-1/10}$, due to its high variability. Bagging can reduce this stochastic variability significantly, while keeping the simple, flexible character of cross-validation. For example, bagging can be used in practically all settings where cross-validation is used for choosing the bandwidth with the aim of minimizing the L_2-distance measure.

In their bagging algorithm Hall and Robinson (2009) apply sampling without replacement, since sampling with replacement could result in ties which might cause the cross-validation procedure to break down. Each bootstrap replicate contains m data points, with $m < n$. They provide a heuristic explanation of the procedure's performance: consider the case where kernel methods are used to estimate a function, such as a probability density or regression function. It is generally known that the theoretically optimal bandwidth h_0 that minimizes the MISE (mean integrated squared error) typically satisfies

$$h_0 \sim Cn^{-b}, \tag{4.9}$$

where $b > 0$ depends on the 'regularity' of the problem and C is a functional of the target function. The value of b is generally known. Both plug-in and cross-validation estimators of h_0 attempt to estimate C: plug-in estimators in an explicit way and cross-validation implicitly. Let $\hat{C}$ denote the cross-validation estimator of C. Then the cross-validation bandwidth is given by $\hat{h}_{CV} = \hat{C}n^{-b}$. Bagging reduces the variability of $\hat{h}_{CV}$ by substituting $\hat{C}$ by $E^*[\hat{C}^*] = E[\hat{C}^*|S]$, where S denotes the original training sample and $\hat{C}^*$ is the bootstrap version of $\hat{C}$. When applied in this way, bagging will mostly reduce the variability of the bandwidth selector, provided that $\hat{C}$ is a highly nonlinear function of the data, which is the case in this setting.

The empirical success of bagging in the context of smoothing parameter selection utilizing cross-validation methods leads one to ask about the theoretical reasoning behind its success. Hall and Robinson (2009) provide a general theoretical explanation of why bagging reduces

the variance in the context of smoothing parameter selection. For example, in the case of kernel density estimation, suppose that cross-validation is constructed to optimize an L_2 distance measure. Take the resampling ratio to be fixed, $a = m/n$. Then the following holds: bagging reduces the asymptotic variance and mean squared error of a cross-validation bandwidth selector by a factor of at least $a^{4/5}$ and the reduction can be as much as $a^{9/5}$, unless a is very small (Hall and Robinson 2009, p. 181, Remark 1). For the case of subagging when $m = n/2$ is chosen, i.e. $a = 0.5$, this implies that the reduction in variability is approximately 50% or greater: it lies between $0.5^{4/5} = 57\%$ and $0.5^{9/5} = 29\%$. See Hall and Robinson (2009) for further theoretical explanations.

Besides its applicability in the setting of kernel methods, bagged cross-validation can be used in a variety of contexts, for instance in conjunction with orthogonal-series and smoothing spline methods. In the next paragraph, we consider the application of bagging to cross-validation smoothing parameter selection in kernel regression.

4.4.3 Methods to estimate the smoothing parameter

The following discussions are based on Hall and Robinson (2009) who dealt mainly with density estimation, but also referred to the regression case. In this section, the ideas in their paper are applied to the regression problem.

Consider the regression problem where the conditional mean response $E[Y|X = x] = m(x)$ is to be estimated using the data $S = (X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)$. Let $\hat{m}_h$ denote a kernel estimator of m with bandwidth h. Let $\hat{m}_{h,(j)}(X_j)$ be the leave-one-out estimator in the point X_j where the j^{th} observation is deleted when determining the estimator in that point. Define the cross-validation function

$$CV(h) = \frac{1}{n}\sum_{j=1}^{n}[Y_j - \hat{m}_{h,(j)}(X_j)]^2$$

as in (5.2), where the weight function is now taken to be 1 for simplicity. The cross-validation method chooses the bandwidth that minimizes $CV(h)$ to construct the curve estimator. Denote the value by $\hat{h}_{CV}$.

$CV(h)$ can be seen as an empirical approximation to the L_2 error criterion $M_n(h) = \int E[\hat{m}_h(x) - m(x)]^2 f(x)dx$ (f denotes the density of the explanatory variables), up to terms that do not depend on h. In this sense, $\hat{h}_{CV}$ can be interpreted as an approximation to $h_0 = h_{0n} = \arg\min_h M_n(h)$.

There are at least two ways in which the smoothing parameter can be estimated when

using bagged cross-validation. For both methods, independent bootstrap replicates of size $m < n$ are drawn without replacement from the original sample. Calculate $CV^*(h)$, the cross-validation function, for each of the bootstrap samples:

$$CV^*(h) = \frac{1}{m}\sum_{j=1}^{n}[Y_j^* - \hat{m}^*_{h,(j)}(X_j^*)]^2.$$

Here m refers to the sample size and $\hat{m}^*_{h,(j)}(X_j^*)$ to the bootstrap leave-one-out estimator in the point X_j^*, where the j^{th} observation is deleted in that point. Now, bagged cross-validation can be applied in one of the following ways:

1. Bagging $CV(h)$:

   Compute the average of the function $CV^*(h)$ over all the bootstrap replicates:

   $$CV_{bagg}(h) = E^*[CV^*(h)] = E[CV^*(h)|S].$$

   Choose the estimated bandwidth $\hat{h}_{bagg}$ to be the value of h that minimizes $CV_{bagg}(h)$ after rescaling for sample size. See the comment on rescaling in the following paragraph.

2. Bagging $\hat{h}_{CV}$:

   Calculate the value $\hat{h}^*_{CV}$ that minimizes $CV^*(h)$ in each bootstrap sample separately. Define the estimated bandwidth $\hat{h}_{bagg}$ as the average value of $\hat{h}^*_{CV}$ over all the bootstrap samples after rescaling for sample size.

The first-order properties of these two methods are generally the same. In their study, Hall and Robinson (2009) found that bagging $CV(h)$ and bagging $\hat{h}_{CV}$ both led to reductions in the variance and MSE compared to standard cross-validation. Generally, bagging $\hat{h}_{CV}$ led to greater reductions than bagging $CV(h)$. We therefore pursue the second method with $\hat{h}_{CV}$ in our studies. Future investigations should explore bagging $CV(h)$ as well.

See more about bandwidth selection algorithms that utilize bagging in Chapter 5.4.

4.4.4 Rescaling of the bandwidth

Hall and Robinson (2009) suggest that rescaling of the bandwidth should be done. Rescaling of the bandwidth should be done to account for the smaller sample size m that is used in the bootstrap replicates. It is based on standard formulae for bandwidth selectors. For instance, in the case of kernel methods where the theoretically optimal bandwidth is given by (4.9), the bagged bandwidth based on resample size m should be multiplied by the factor $(m/n)^b$.

For the case where the aim is to give optimal performance for functions with two derivatives (i.e. the regression estimator is of second order), $b = 1/5$ is chosen. In our simulation study, rescaling is applied.

Chapter 5

The new methods and simulation studies

5.1 Introduction

In Chapters 2 to 4, the fundamental elements of the improvement methods, i.e. boosting, bagging and bragging, as well as the historical development of each, were discussed. Marzio and Taylor (2008) and Hall and Robinson (2009) respectively introduced boosting and bagging to the field of kernel regression. In the current study these ideas are extended empirically and some new methods, namely bragging and combinations of boosting, bagging and bragging, are explored. In particular, we explore the effect of all methods on the N-W kernel regression estimator.

In short, the aim of this study is to gain more information regarding boosting methods in kernel regression and to investigate whether improvement methods that utilize bootstrap methodology, i.e. the bagging and bragging methods, can improve the variability aspect of cross-validation smoothing parameter selection in using the N-W and the boosted N-W estimators.

Marzio and Taylor (2008) showed that boosting can be applied to improve the N-W estimator. We extend the boosting method for a wider range of simulation settings. Furthermore, Hall and Robinson (2009) claim that bagging can be applied effectively to combat variability existing in cross-validation bandwidth selection in kernel regression estimation. We propose three different algorithms to investigate this statement. Bragging (Swanepoel 1990, Bühlmann 2003), a robust form of bagging, has as far as we know not yet been applied to the N-W estimator in the literature. We present three different algorithms for the application of bragging to the N-W estimator. Finally, we propose new procedures that combine the boosting methodology with bagging and bragging respectively.

In this chapter, previous definitions, algorithms and methodologies of the methods mentioned above will be summarized and new methodologies will be formulated. In Chapter 6 results of the conducted Monte Carlo studies are reported and discussed. Tables and graphs of the results are presented in Appendices A, B and C.

Before discussing the improvement methods, we shall revise the mechanisms of some basic procedures which will be used in the improvement algorithms and provide an outline of the new simulation study in Section 5.2. The remainder of Chapter 5 will consist of detailed descriptions of the methods and algorithms applied in this study. In particular, the boosting method is considered in Section 5.3, bagging in Section 5.4, bragging in Section 5.5 and combinations of boosting with bagging and boosting with bragging in Section 5.6.

Before proceeding to the discussion of the basic procedures in Section 5.2, we now present an outline of the contributions made in this study.

5.1.1 New contributions resulting from this study

In this study we refer to Marzio and Taylor (2008), Hall and Robinson (2009), Swanepoel (1990) and Bühlmann (2003) regarding the so-called improvement methods. Remarks on each method are given below.

- **Boosting:**

  Marzio and Taylor (2008) apply boosting to the N-W kernel regression estimator, but for a restricted range of settings. We explore how the boosted N-W estimator performs under different data schemes (by varying the underlying distribution of the covariate data and for different choices of error variance). We also use two different kernels in an attempt to understand the mechanism and effect of boosting better. In Section 5.3 the boosting process for the N-W estimator is described.

- **Bagging:**

  Hall and Robinson (2009) argue that bagging can be applied to the cross-validation method of bandwidth selection in kernel methods. Since the cross-validation bandwidth choice has great variability, bagging should improve its performance. They suggest that the method is applicable in the kernel regression setting and provide empirical results for the application of bagging to the local linear estimator (i.e. an estimator

in the family of local polynomial estimators, see Chapter 1, Section 1.4.5 for a general discussion of local polynomial fitting) using the biweight kernel. As mentioned in Chapter 4, Section 4.4.3, Hall and Robinson suggest that bagging can be applied to cross-validation bandwidth selection in at least two ways, namely bagging $CV(h)$ and bagging $\hat{h}_{CV}$. We consider only bagging $\hat{h}_{CV}$. We further propose three ways in which the bagging of $\hat{h}_{CV}$ can be approached and apply the three bagging procedures to the cross-validation bandwidth selection process in N-W kernel regression estimation for various simulation setup settings. These procedures are discussed more informatively in Section 5.4.

- **Bragging:**

  Bragging entails that the median of bootstrap versions of the required estimator is calculated, rather than the average as in bagging (Swanepoel 1990, Bühlmann 2003). In the literature, bragging has not yet been applied in the kernel regression setting as far as we know. We apply bragging to the cross-validation method of bandwidth selection in the N-W estimator in three different ways and compare the results with the other improvement methods. These new procedures are discussed in Section 5.5.

- **Combinations of boosting with bagging and bragging respectively:**

  Finally we propose "bagged boosting" and "bragged boosting". In these new methods we combine the improving effects of respectively bagging and bragging in cross-validation bandwidth selection with the positive effect of boosting to obtain better regression estimates. In particular, we explore if, and for which scenarios, cross-validation in boosting, where the optimal number of boosting iterations and the bandwidth are chosen by cross-validation, can be improved upon by applying bagging or bragging. Section 5.6 deals with three procedures that combine bagging and boosting and three procedures that combine bragging and boosting.

5.2 Important definitions, formulas and remarks

Before discussing the improvement methods, we revise in Section 5.2 the mechanisms of some basic procedures which will be used in the improvement algorithms and provide an overview of our simulation study. The regression functions $m(x)$ that will be used throughout this

study will be introduced in Section 5.2.1. Some basic procedures feature repeatedly in our simulation studies and these methods are revised in Section 5.2.2. In Section 5.2.3 we provide the main algorithm which forms the basic framework for all conducted simulation studies. In Section 5.2.4 an outline of our simulation studies is provided.

5.2.1 The regression function $m(x)$

For the sake of completeness of this paragraph we introduce the regression functions $m(x)$ that will be estimated throughout this study. The three functions are displayed in Table 5.1.

Model	$m(x)$
1	$x + 2\exp(-16x^2)$
2	$\sin(2x) + 2\exp(-16x^2)$
3	$0.3\exp\{-4(x+1)^2\} + 0.7\exp\{-16(x-1)^2\}$

Table 5.1: Regression functions $m(x)$

A more detailed discussion and graphs of these functions will be provided in Chapter 6, Section 6.2.

5.2.2 Revision of basic procedures

This section focuses on a brief discussion of the calculations involved in the N-W kernel regression estimator, the cross-validation method of bandwidth selection and the MISE as measure of performance of a fitted curve. These are the basic procedures which will be used in the simulation studies. For detailed theoretical discussions, previous chapters as well as applicable references should be consulted.

- **The N-W kernel regression estimator**

  Various kernel regression smoothers exist, as was discussed in Chapter 1, Section 1.4.1. Nadaraya (1964) and Watson (1964) proposed what is known as the Nadaraya-Watson

(N-W) estimator

$$\hat{m}_{NW}(x) = \frac{\sum_{i=1}^{n} K_h(x - X_i)Y_i}{\sum_{i=1}^{n} K_h(x - X_i)}, \tag{5.1}$$

where

$$K_h(u) = h^{-1}K(u/h)$$

is the kernel with scale factor h as before. We shall focus on the N-W regression estimator in this study.

Note that the denominator of the estimator in (5.1) can become very small, especially for small sample sizes, when leave-one-out-estimates have to be calculated in cross-validation procedures. To overcome problems that arise due to division by very small values, it is customary to build a correction into calculations (Hall and Robinson 2009), i.e. in all cases where the value of the denominator is less than 0.001, a value of 0.001 is added to the denominator. This mechanism is applied throughout this study.

Another way to deal with this problem is as follows: Put $l_i = \log(K_h(x - X_i))$ and $l_I = \max_i l_i$. Then

$$\begin{aligned}
&\frac{\sum_{i=1}^{n} \exp(l_i - l_I)Y_i}{\sum_{i=1}^{n} \exp(l_i - l_I)} \\
= &\frac{\sum_{i=1}^{n} \exp(\log(K_h(x - X_i)) - l_I)Y_i}{\sum_{i=1}^{n} \exp(\log(K_h(x - X_i)) - l_I)} \\
= &\frac{\sum_{i=1}^{n} (K_h(x - X_i) \cdot \exp^{-l_I})Y_i}{\sum_{i=1}^{n} K_h(x - X_i) \cdot \exp^{-l_I}} \\
= &\hat{m}_{NW}(x).
\end{aligned}$$

This denominator is at least 1. Therefore the mechanism above is not necessary. However, practical application of this method reveals that in case of small values of h, the value of the kernel function is close or equal to zero, which, when taking the logs in the $l_i's$, produces more problems than when applying the original mechanism of adding a small constant. We will therefore not proceed with this method.

- **Cross-validation bandwidth selection**

The choice of bandwidth is critical in kernel regression estimation. In Section 1.6.1, various ways to choose the bandwidth were discussed. In this study we use the leave-one-out cross-validation method. The cross-validation method will be revised briefly.

Let the sample $S_n = \{(X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)\}$ be given. Leave-one-out cross-validation uses the regression smoother in which the j^{th} observation is deleted when the estimate $\hat{m}$ is calculated in the point X_j. For the N-W smoother, the leave-one-out estimate is defined by

$$\hat{m}_{NW,(j)}(X_j) = \frac{\sum_{i=1, i\neq j}^{n} K_h(X_j - X_i)Y_i}{\sum_{i=1, i\neq j}^{n} K_h(X_j - X_i)}. \tag{5.2}$$

The cross-validation function is then calculated as the sum of the squared deleted residuals

$$CV(h) = \frac{1}{n}\sum_{j=1}^{n}[Y_j - \hat{m}_{NW,(j)}(X_j)]^2. \tag{5.3}$$

Note that when we introduced the cross-validation function in (1.32), weights were included in the expression. In our studies we follow the approach of Hall and Robinson (2009) and choose the weights to be equal to 1 for $i = 1, 2, \ldots, n$. Therefore the weights are omitted in (5.3).

The cross-validation method chooses the bandwidth that minimizes $CV(h)$. Denote this bandwidth by $\hat{h}_{CV}$.

We now present an algorithm for determining the standard N-W estimator where the bandwidth is selected by means of cross-validation.

Algorithm 5.0. ***The standard N-W estimator using cross-validation for bandwidth selection (NW)***

1. Let $S_n = \{(X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)\}$ be the given sample.
2. Calculate the leave-one-out estimate for a given bandwidth h

$$\hat{m}_{NW,(j)}(X_j) = \frac{\sum_{i=1, i\neq j}^{n} K_h(X_j - X_i)Y_i}{\sum_{i=1, i\neq j}^{n} K_h(X_j - X_i)},$$

   where K_h is the kernel with scale factor h.
3. Determine the cross-validation function for the particular bandwidth h

$$CV(h) = \frac{1}{n}\sum_{j=1}^{n}[Y_j - \hat{m}_{NW,(j)}(X_j)]^2.$$

4. Repeat steps 2 and 3 for a grid of bandwidths to obtain $CV(h)$ for each h value.

5. Select $\hat{h}_{CV}$ as the bandwidth that minimizes $CV(h)$ in step 4.

6. Calculate the N-W estimator using the selected bandwidth $\hat{h}_{CV}$

$$\hat{m}_{NW}(x) = \frac{\sum_{i=1}^{n} K_{\hat{h}_{CV}}(x - X_i)Y_i}{\sum_{i=1}^{n} K_{\hat{h}_{CV}}(x - X_i)},$$

where $K_{\hat{h}_{CV}}$ is the kernel with scale factor $\hat{h}_{CV}$.

- **The MISE as global measure of performance**

When different regression estimation methods are explored, one would like to evaluate the performance of each method to assess how effective the method is, i.e. how close the smooth is to the true curve. In Chapter 1, Section 1.5, different discrepancy measures were discussed. In this study we use the mean integrated squared error (MISE) as measure of assessment.

The MISE is defined as

$$\text{MISE} = E[d_I(\hat{m}, m)], \tag{5.4}$$

where $d_I(\hat{m}, m)$ is the integrated squared error (ISE) defined in (1.25). Due to Fubini's law, the integrals implied in (5.4) can be interchanged, resulting in an equivalent expression for the MISE,

$$\text{MISE} = \int \text{MSE}[\hat{m}(x)]dx,$$

where

$$\text{MSE}[\hat{m}(x)] = E[\hat{m}(x) - m(x)]^2.$$

Note that the MISE is a global measure of discrepancy: it covers the whole population (not merely one realization of a random sample from the population) by taking the expected squared differences between the estimated values and their corresponding true values for each fixed point x by means of the MSE. Furthermore, the MISE covers the entire range of x-values by integrating the MSE over the range of x-values.

Recall from Chapter 1, Section 1.5.1, that a useful feature of the MSE is that it can be written as the sum of the variance and the squared bias of the estimator:

$$\text{MSE}[\hat{m}(x)] = E[\hat{m}(x) - m(x)]^2$$

$$\begin{aligned} &= E[(\hat{m}(x) - E[\hat{m}(x)])^2] + [E[\hat{m}(x)] - m(x)]^2 \\ &= \text{Var}[\hat{m}(x)] + \text{Bias}^2[\hat{m}(x)]. \end{aligned}$$

Integrating over all x-values gives

$$\text{MISE} = \int \text{Var}[\hat{m}(x)]dx + \int \text{Bias}^2[\hat{m}(x)]dx.$$

The MISE, integrated variance and integrated squared bias values will be reported and compared for the various estimation procedures that are investigated in this study.

In regression problems where the MISE is used as discrepancy measure, the aim is to choose the parameters in $\hat{m}(x)$ such that the MISE is minimized. In some instances, this minimization problem can be solved analytically, resulting in theoretically optimal choices of the parameters. In this study we do not follow a theoretical approach, but rather focus on an empirical approach. The expected values in the expressions for the MSE, variance and squared bias are approximated by averaging over Monte Carlo number of finite samples drawn randomly from the underlying distribution. These calculations are repeated for a fixed grid of x-values. Numerical integration is then used to integrate over the entire range of x-values to obtain an approximation to the global MISE measure.

5.2.3 Main algorithm

In this section the main algorithm that is applied in this study is presented. In short, the main algorithm involves calculating the approximated MISE value for each of the specific smoothing procedures used to calculate the estimate $\hat{m}(x)$. Keep in mind that in this study 14 different procedures to obtain $\hat{m}(x)$ are investigated. The main algorithm is applied to each of the different smoothing procedures and the approximated MISE, integrated variance and integrated squared bias values resulting from the main algorithm are then compared.

a) The main algorithm

The main algorithm is now presented.

Algorithm 5.1. *Main algorithm*

1. Draw MC number of random samples of size n from the underlying distribution and denote each by $S_n = \{(X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)\}$, with MC a large number.

2. For the specific estimation procedure (see Section 5.2.4 for an outline of the different procedures), calculate $\hat{m}(x)$, the estimated value of the smoother in the point x, for each of the MC samples in step 1 and denote the estimates by $\hat{m}_i(x)$, $i = 1, 2, \ldots, MC$. This involves the various ways of selecting bandwidths, as is clear from Algorithm 5.0 and Algorithms 5.2 – 5.14.

3. Calculate Monte Carlo approximations to the MSE, variance and squared bias in the point x:

$$\widehat{\text{MSE}}[\hat{m}(x)] = \frac{1}{MC}\sum_{i=1}^{MC}[\hat{m}_i(x) - m(x)]^2,$$

$$\widehat{\text{Var}}[\hat{m}(x)] = \frac{1}{MC}\sum_{i=1}^{MC}\left[\hat{m}_i(x) - \frac{1}{MC}\sum_{i=1}^{MC}[\hat{m}_i(x)]\right]^2$$

and

$$\widehat{\text{Bias}}^2[\hat{m}(x)] = \left[\frac{1}{MC}\sum_{i=1}^{MC}[\hat{m}_i(x)] - m(x)\right]^2.$$

4. Repeat steps 2 and 3 for a fixed grid of x-values. Denote the minimum grid value by x_{min} and the maximum grid value by x_{max} (the choice of x_{min} and x_{max} depends on the underlying distribution of the covariate data).

5. Use numerical integration to integrate over the entire range of x-values to obtain approximated global measures:

$$\widehat{\text{MISE}} = \int_{x_{min}}^{x_{max}} \widehat{\text{MSE}}[\hat{m}(x)]dx, \tag{5.5}$$

$$\text{Approximate Integrated Variance} = \int_{x_{min}}^{x_{max}} \widehat{\text{Var}}[\hat{m}(x)]dx \tag{5.6}$$

and

$$\text{Approximate Integrated Bias}^2 = \int_{x_{min}}^{x_{max}} \widehat{\text{Bias}}^2[\hat{m}(x)]dx. \tag{5.7}$$

6. Repeat the process for all the procedures used to estimate $\hat{m}(x)$ as listed in Section 5.2.4.

b) Remarks regarding Algorithm 5.1

- The MISE, integrated variance and squared bias values that are obtained in step 5 of the main algorithm are approximations to the corresponding true values in the sense that integration and aggregation are done over a finite number of grid points for a finite (MC) number of random samples S_n. For simplicity we often refer to these values as the "MISE", "integrated variance" and "integrated squared bias" in the rest of this study, but the reader should keep in mind that it is actually approximations to these measures that are reported.

- In performing Algorithm 5.1, steps 1, 3, 4 and 5 are relatively straight forward to execute. However, for step 2, i.e. to calculate $\hat{m}(x)$, various possibilities exist, which involve the choice of kernel functions, bandwidths and utilizing methods such as cross-validation, boosting and the bootstrap. In this study, the N-W estimate is of concern. The aim is to compare cases where $\hat{m}(x)$ is chosen to be the standard N-W estimate with bandwidth h determined via the cross-validation method, with cases where $\hat{m}(x)$ involves "improved" N-W estimates, i.e. cases where the boosting, bagging and bragging methods are applied to the standard N-W estimate.

  Algorithms 5.2 to 5.14 present different ways to determine "improved" N-W estimates in step 2. All of these alternatives play a role in the execution of step 2.

To clarify the execution of step 2 in our simulation studies, a more detailed sketch of the simulation studies is presented in Section 5.2.4, which will be elaborated on in Chapter 6, Section 6.2.

5.2.4 Simulation studies: a brief overview

The simulation is twofold: Simulation Study I and Simulation Study II, which are briefly introduced in this section. More detailed descriptions of the procedures and parameter choices will be provided throughout the rest of this chapter and in Chapter 6.

As in previous chapters, let h denote the bandwidth and T the stopping value for the number of boosting iterations.

1. Simulation Study I

   Simulation Study I is the main study of this dissertation. The purpose of Simulation Study I is to investigate whether

- boosting enhances the performance of the N-W estimator,
- improvement methods that utilize bootstrap methodology, i.e. the bagging and bragging methods, can improve data-driven smoothing parameter selection in the N-W and the boosted N-W estimators and if
- the data-driven cross-validation methods used to select all smoothing parameters in Simulation Study I, decrease variability that is inherent to cross-validation methods.

The effect of bagging and bragging on other data-driven smoothing parameter selection methods, such as the plug-in method or methods utilizing penalizing functions, will be explored in future studies.

The aim of Simulation Study I is to answer the following specific questions:

(a) Does boosting improve the standard N-W estimator when h and T are selected via cross-validation methods?

(b) Does bagging improve cross-validation selection of h in the standard N-W estimator?

(c) Does bragging improve cross-validation selection of h in the standard N-W estimator?

(d) Does bagging improve cross-validation selection of h and T in the boosted N-W estimator?

(e) Does bragging improve cross-validation selection of h and T in the boosted N-W estimator?

(f) How do all of the above methods compare regarding reduction of variability and minimizing of MISE?

To answer these questions, we apply each of the methods (boosting, bagging and bragging) to determine $\hat{m}(x)$ in step 2 of the main algorithm, Algorithm 5.1. In total, Simulation Study I involves 14 different procedures.

Table 5.2 lists the 14 procedures, presenting a short description of each procedure, the number of the algorithm that describes the particular procedure and the code that will be used in the remainder of this dissertation to refer to the procedure.

The procedures described in Table 5.2 are used to estimate the mean response function $m(x)$. As mentioned above, this is all done in order to execute step 2 in the main algorithm. Implementing the rest of the main algorithm will reveal the behaviour of the MISE components of the various procedures. The resulting MISE, integrated variance and integrated squared bias values can then be compared and used to draw conclusions about the procedures.

Procedure	Algorithm	Procedure code
Cross-validation is applied to select h in the standard N-W estimator	5.0	NW
Cross-validation is applied to select h and T in the boosted N-W estimator	5.2	Boost_1^*
Cross-validation is applied to select h and T in the boosted N-W estimator		Boost_6^*
Bagging is applied to cross-validation choice of h in the standard N-W estimator, approach 1	5.3	Bag1
Bagging is applied to cross-validation choice of h in the standard N-W estimator, approach 2	5.4	Bag2
Bagging is applied to cross-validation choice of h in the standard N-W estimator, approach 3	5.5	Bag3
Bragging is applied to cross-validation choice of h in the standard N-W estimator, approach 1	5.6	Brag1
Bragging is applied to cross-validation choice of h in the standard N-W estimator, approach 2	5.7	Brag2
Bragging is applied to cross-validation choice of h in the standard N-W estimator, approach 3	5.8	Brag3
Bagging is applied to cross-validation choice of h and T in the boosted N-W estimator, approach 1	5.9	Bagboost1
Bagging is applied to cross-validation choice of h and T in the boosted N-W estimator, approach 2	5.10	Bagboost2
Bagging is applied to cross-validation choice of h and T in the boosted N-W estimator, approach 3	5.11	Bagboost3
Bragging is applied to cross-validation choice of h and T in the boosted N-W estimator, approach 1	5.12	Bragboost1
Bragging is applied to cross-validation choice of h and T in the boosted N-W estimator, approach 2	5.13	Bragboost2
Bragging is applied to cross-validation choice of h and T in the boosted N-W estimator, approach 3	5.14	Bragboost3

Table 5.2: Procedures used in Simulation Study I (* Boost_1 and Boost_6 respectively indicate the application of Algorithm 5.2 with $T \leq 1$ and $T \leq 6$)

Remarks

- For the procedure NW in Table 5.2 the appropriate smoothing parameter is selected as follows: for each MC sample, apply Algorithm 5.0 to obtain $\hat{m}_{NW}$ in step 2 of the main algorithm, i.e. Algorithm 5.1, and complete the algorithm.
- For the procedures Boost_1^* and Boost_6^* in Table 5.2, Algorithm 5.2 is applied, where $T \leq 1$ and $T \leq 6$ for the two procedures respectively. In the boosting algorithm $\hat{m}_0$ is determined as for procedure NW described in the previous remark.
- Finally, note that for each of the bagging and bragging procedures in Table 5.2, three different approaches are applied. The basic ideas behind approaches 1, 2 and 3, in short, are as follows:
  - ► The first approach involves determining the cross-validation smoothing parameter for each bootstrap sample, calculating the mean or median of the bootstrap smoothing parameters and then applying the resulting mean or median smoothing parameter value to the original sample to obtain $\hat{m}(x)$.
  - ► The second approach is similar to the first, except that the mean or median smoothing parameter value over the bootstrap samples is applied to each of the bootstrap samples (and not to the original sample) to obtain bootstrap regression estimates. The mean or median of the bootstrap regression estimates are then calculated to obtain $\hat{m}(x)$.
  - ► The third approach involves determining the cross-validation smoothing parameter for each bootstrap sample and then applying each bootstrap sample's smoothing parameter to that particular bootstrap sample to obtain bootstrap regression estimates. The mean or median of the bootstrap regression estimates are then calculated to obtain $\hat{m}(x)$.

These descriptions of the three approaches are very brief and introductory and detailed accounts of how the approaches are applied in the various procedures, are provided in the algorithms.

Results and interpretations of Simulation Study I will be discussed in Section 6.3. Tables and graphs containing the results of Simulation Study I are presented in Appendices A and C respectively.

2. Simulation Study II

   Simulation Study II served as a "preliminary investigation" and focuses specifically on boosting as improvement method.

   The goal of Simulation Study II is to explore the behaviour of boosted N-W estimation in various simulation setup settings. Furthermore, Simulation Study II aims to investigate which "optimal" values of the smoothing parameters minimize the approximated MISE in the sense described below.

   To attain these goals the selection of the smoothing parameters is deferred in Simulation Study II in the sense that parameter values are chosen "optimally" with regard to minimizing the approximated MISE and not by means of data-driven selection methods as it was done in Simulation Study I. More specifically, the performance of the boosted N-W estimator is compared to the performance of the standard N-W estimator in terms of the MISE when smoothing parameters are selected "optimally".

   The smoothing parameter of concern for the standard N-W estimator is the bandwidth h. Furthermore, jointly the pair consisting of the bandwidth h and the number of boosting iterations T is of interest for the boosted N-W estimator. In Simulation Study II Algorithm 5.1 is executed repeatedly for grids of possible parameter values, i.e. for a grid of h-values for the N-W estimator and a joint grid of h and T values for the boosted N-W estimator. For each parameter value h in the N-W estimator and parameter pair (h, T) in the boosted N-W estimator, the approximated MISE is recorded. The h and (h, T) pair that minimize the approximated MISE, are the selected "optimal" smoothing parameters for the standard and boosted N-W estimators respectively and should be used to estimate $m(x)$ for the given simulation setup.

   Simulation Study II is a follow-up of the work done by Marzio and Taylor (2008) over a larger parameter platform, i.e. the performance of the boosted N-W estimator is studied under different data schemes (by varying the underlying distribution of the covariate data and for different choices of error variance), using two different kernels.

   Results and interpretations of Simulation Study II will be discussed in Section 6.4. Tables containing the results of Simulation Study II are displayed in Appendix B.

In order to implement the procedures in Simulation Study I and II, Algorithms 5.2 to 5.14 are applied, which are based on boosting, bagging and bragging methods. All procedures and algorithms are now discussed.

5.3 Boosting

Marzio and Taylor (2008) introduced boosting as a very effective method to improve the performance of the N-W kernel regression estimator, with regard to the minimized MISE. In Chapter 2 the boosting method, i.e. the development, applications and features of the method, was discussed. Section 2.6 of Chapter 2 considered the application of boosting to the N-W regression estimator. For the sake of completeness we repeat the algorithm given in Algorithm 2.4 here, specifying our choice of parameters in the algorithm where relevant.

Algorithm 5.2. ***Determine*** $\hat{\mathrm{m}}(\mathrm{x})$ ***by boosting (Boost)***

1. Let $S_n = \{(X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)\}$ be the given sample.

2. Fit an initial learner, $\hat{m}_0(x)$ by fitting the N-W regression estimator to the original data: $\hat{m}_0(x) \equiv \hat{m}_{NW}(x)$, using some appropriate bandwidth, h. See Section 5.3.1 for a discussion of the allowed bandwidths. Let $t = 0$.

3. For $t = t + 1$, compute the residuals,

$$U_i = Y_i - \hat{m}_{t-1}(X_i), i = 1, \ldots, n.$$

4. Fit the residual vector $U_1, U_2, \ldots, U_n$ to $X_1, X_2, \ldots, X_n$ by means of the N-W estimator using an appropriate bandwidth. Call this fit $\tilde{m}_t(x)$.

5. Update the previous estimator $\hat{m}_{t-1}(x)$:

$$\hat{m}_t(x) = \hat{m}_{t-1}(x) + \tilde{m}_t(x).$$

   (Note that we choose the real-valued step-length factor ν in Algorithm 2.4 to be 1. This choice was made by Marzio and Taylor (2008). Future studies should investigate different choices for ν.)

6. Repeat steps 3 to 5 until $t = T$, where T is some stopping value for the number of boosting iterations.

Note that in existing literature and in this study, the same h-value is used in steps 2 and 4 above. This may be wrong, especially in the random design model. The regression in step 4 may need another bandwidth, more appropriate for the U-values. This is, however, part of another ongoing study and results will be reported in another publication.

5.3.1 The choice of bandwidth and number of boosting iterations

In Algorithm 5.2 there are two parameters that need specification, i.e. the bandwidth h and the number of boosting iterations T. The chosen values for h and T have a great effect on the performance of the boosted N-W estimator. We now describe how we approach these choices in our simulation studies.

1. In Simulation Study I the bandwidth and number of boosting iterations are selected via *cross-validation*. For this process, the estimator $\hat{m}(x)$ in step 2 of the main algorithm (Algorithm 5.1) is the boosted N-W estimator in which the cross-validation choice of optimal number of boosting iterations and the corresponding bandwidth are used. Thus, step 2 of the main algorithm involves determining the optimal cross-validation values of h and T, via the algorithms, applied together with the original data to determine $\hat{m}(x)$. The rest of the main algorithm is then executed to obtain the MISE value. This MISE value can be used for comparison purposes with other procedures.

   More specifically, the application of cross-validation in step 2 involves that the cross-validation function $CV(h,T)$ has to be minimized. This cross-validation function is expressed by

$$CV(h,T) = \frac{1}{n}\sum_{j=1}^{n}[Y_j - \hat{m}_{T,(j)}(X_j)]^2, \tag{5.8}$$

   where $\hat{m}_{T,(j)}(X_j)$ is the leave-one-out N-W estimate in the point X_j after T boosting iterations. The (h,T) combination that minimizes $CV(h,T)$ is chosen as the estimated parameter values for the setup at hand. Denote these by $\hat{h}_{CV}$ and $\hat{T}_{CV}$. $\hat{m}(x)$ in step 2 of the main algorithm is then taken to be the boosted N-W estimator after $\hat{T}_{CV}$ boosting iterations using $\hat{h}_{CV}$ as bandwidth. This process is applied in the Boost, Bagboost and Bragboost procedures in Table 5.2.

2. In Simulation Study II the bandwidth and number of boosting iterations are selected *"optimally" in terms of the MISE*. In this approach, grids of allowed h and T values are determined. These grids are combined to form a 2-dimensional grid of possible (h,T) combinations. The main algorithm given in Algorithm 5.1 is executed for each of the (h,T) pairs to obtain the MISE value for each pair. The (h,T) combination that results in the minimum MISE is regarded the "optimal" choice of parameters for the setup at hand. Denote these by h_{opt} and T_{opt}.

In a similar manner, the "optimal" bandwidth for the standard N-W estimator is calculated. The MISE value resulting from the standard N-W estimator using its optimal bandwidth is compared to the MISE value resulting from the boosted N-W estimator after T_{opt} boosting iterations, where the bandwidth h_{opt} is used. By means of this comparison, the effectiveness of boosting in terms of the MISE can be established. This method serves as a typical image of the boosting methodology when applied to the N-W estimator and is used to verify the results obtained in Simulation Study I by comparing the behaviour of the boosting process in both studies.

5.4 Bagging

Only Simulation Study I is involved in investigations regarding bagging, bragging, bagged boosting and bragged boosting which are covered in the remaining part of Chapter 5.

All methods in the rest of this chapter aim to improve the *cross-validation* choice of smoothing parameters. In particular, the bagging and bragging methods aim to improve the selection of h in the standard N-W estimator, whereas the goal of bagged boosting and bragged boosting is to improve the selection of h and T in the boosted N-W estimator. We now proceed to discuss the first improvement method for cross-validation bandwidth selection in the standard N-W estimator, namely bagging.

As mentioned in Chapter 4, the principles of bootstrap aggregating (bagging) were stated by Swanepoel (1988, 1990) in functional context and by Breiman (1996a) from a machine learning viewpoint. Hall and Robinson (2009) present bagging as an effective method to improve cross-validation as tool for smoothing parameter choice, for instance in kernel density or kernel regression estimation. The relatively high variance of cross-validation smoothing parameters influence the performance of the estimators in which the chosen parameters are used. Hall and Robinson have shown that bagging can reduce the variability simply, significantly and reliably. In Chapter 4, a discussion of the bagging methodology was provided. In Section 4.4, the application of bagging to cross-validation was considered. In this section, we provide algorithms to clarify how to implement the bagging method in the simulation studies.

5.4.1 Remarks regarding bagging algorithms

The following points are of importance regarding the simulation studies:

- In Section 4.4.3, we stated that bagging can be applied to cross-validation bandwidth selection methods. Recall that Hall and Robinson (2009) proposed two ways in which bagging can be applied to cross-validation bandwidth selection. In the first, bootstrap versions of the cross-validation function are aggregated to obtain $CV_{bag}(h)$ and consequently the estimated bandwidth $\hat{h}_{bag}$ is chosen to be the value of h that minimizes $CV_{bag}(h)$.

  Alternatively, it was suggested that the value $\hat{h}^*_{CV}$ that minimizes $CV^*(h)$ in each bootstrap sample is chosen, whereafter the estimated bandwidth $\hat{h}_{bag}$ is defined as the average value of $\hat{h}^*_{CV}$ over all the bootstrap samples.

  Hall and Robinson (2009) found that both methods reduce the variance and mean squared error when compared to standard cross-validation. Furthermore, they found that generally, although not always, the second method leads to greater improvements than the first method. Our preliminary simulation studies confirmed this and we report only on the second approach in this study.

- When applying the second approach, where the value $\hat{h}^*_{CV}$ that minimizes $CV^*(h)$ in each bootstrap sample is determined and the estimated bandwidth $\hat{h}_{bag}$ is consequently taken to be the average value of $\hat{h}^*_{CV}$ over all the bootstrap samples, there are at least three different ways to perform the bootstrap aggregation. We shall refer to these methods as:

  - ► Bagging I
  - ► Bagging II
  - ► Bagging III

  The algorithms to determine the final N-W estimators when these three methods are applied, are described in Algorithms 5.3, 5.4 and 5.5 respectively. Results and conclusions about the methods are presented in the next chapter.

- In all algorithms, resampling is done without replacement. Hall and Robinson (2009) noted that resampling with replacement can result in ties, which can cause the cross-validation method to break down. We empirically experimented with the different resampling mechanisms. In preliminary studies, resampling with replacement resulted

in much higher MISE values than resampling without replacement or standard cross-validation. We continued resampling without replacement and report only on this approach.

- In all algorithms, resamples of size $m = [an]$, $0 < a < 1$, are drawn, where $[y]$ denotes the largest integer value smaller than or equal to y. Bühlmann and Yu (2002) referred to this modified form of bagging as "subagging", a term which was defined in Section 4.3.2. We shall continue to use the term "bagging", however.

- The rescaling step in Algorithm 5.3 below is necessary to account for the fact that the bagged bandwidth is based on a smaller sample size, i.e. m rather than n, and is then applied to determine $\hat{m}_{NW}(x)$, using the original n data points. See Section 4.4.4 for a discussion of the rescaling step in Algorithm 5.3. In Algorithms 5.4 and 5.5 the rescaling step is not necessary, since the bootstrap bandwidths are used to obtain bootstrap estimates $\hat{m}^*_{NW}(x)$ when using the bootstrap resamples with m data points.

5.4.2 Bagging algorithms

We now present the three bagging algorithms.

Algorithm 5.3. ***Determine*** $\hat{\mathrm{m}}(\mathrm{x})$ ***by bagging 1 (Bag1)***

1. Let $S_n = \{(X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)\}$ be the given sample.

2. Randomly sample m data pairs without replacement from S_n to obtain a bootstrap sample $S^*_m = \{(X^*_1, Y^*_1), (X^*_2, Y^*_2), \ldots, (X^*_m, Y^*_m)\}$.

3. Calculate a bootstrap version of the leave-one-out cross-validation function in (5.3) by using the bootstrap sample S^*_m to obtain $CV^*(h)$ over a grid of h-values. Obtain the value $\hat{h}^*_{CV}$ that minimizes $CV^*(h)$.

4. Repeat steps 2 and 3 B times to obtain bootstrap replications of the cross-validation bandwidth $\hat{h}^*_{CV}(1), \hat{h}^*_{CV}(2), \ldots, \hat{h}^*_{CV}(B)$.

5. Calculate the average of the bootstrap replications of the bandwidth

$$\hat{h}_{bag_m} = \frac{1}{B}\sum_{b=1}^{B} \hat{h}^*_{CV}(b).$$

6. Rescale the bandwidth in step 5: $\hat{h}_{bag} = (\hat{h}_{bag_m}) \left(\frac{m}{n}\right)^{\frac{1}{5}}$.

7. Fit the N-W estimator $\hat{m}_{NW}(x)$ in (5.1), using the original sample S_n and the bandwidth $\hat{h}_{bag}$.

Algorithm 5.4. ***Determine*** $\hat{\mathrm{m}}(\mathrm{x})$ ***by bagging 2 (Bag2)***

1. Repeat steps 1 to 5 of Algorithm 5.3.

2. Use each bootstrap sample S_m^* and the aggregated bandwidth $\hat{h}_{bag_m}$ and fit the N-W estimator given in (5.1) to obtain bootstrap versions of the regression estimate $\hat{m}^*_{NW}(x)^{(1)}, \hat{m}^*_{NW}(x)^{(2)}, \ldots, \hat{m}^*_{NW}(x)^{(B)}$.

3. Calculate the average of the bootstrap replications of the regression estimate

$$\hat{m}^*_{NW}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{m}^*_{NW}(x)^{(b)}.$$

Algorithm 5.5. ***Determine*** $\hat{\mathrm{m}}(\mathrm{x})$ ***by bagging 3 (Bag3)***

1. Let $S_n = \{(X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)\}$ be the given sample.

2. Randomly sample m data pairs without replacement from S_n to obtain a bootstrap sample $S_m^* = \{(X_1^*, Y_1^*), (X_2^*, Y_2^*), \ldots, (X_m^*, Y_m^*)\}$.

3. Calculate a bootstrap version of the leave-one-out cross-validation function in (5.3) by using the bootstrap sample S_m^* to obtain $CV^*(h)$ over a grid of h-values. Obtain the value $\hat{h}^*_{CV}$ that minimizes $CV^*(h)$.

4. Use the bootstrap sample S_m^* and corresponding bandwidth $\hat{h}^*_{CV}$ and fit the N-W estimator in (5.1) to obtain a bootstrap version of the regression estimate $\hat{m}^*_{NW}(x)$.

5. Repeat steps 2, 3 and 4 B times to obtain bootstrap replications of the regression estimate $\hat{m}^*_{NW}(x)^{(1)}, \hat{m}^*_{NW}(x)^{(2)}, \ldots, \hat{m}^*_{NW}(x)^{(B)}$.

6. Calculate the average of the bootstrap replications of the regression estimate

$$\hat{m}^*_{NW}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{m}^*_{NW}(x)^{(b)}.$$

5.5 Bragging

Swanepoel (1990) and Bühlmann (2003) suggested that the median of bootstrap versions of an estimator can be computed instead of the arithmetic mean and called the procedure bragging. In this way, a robust estimator is obtained.

As far as we know, bragging has not yet been applied in the literature to bandwidth selection in kernel regression. We apply bragging to cross-validation as tool for bandwidth selection in the N-W kernel regression estimator and compare the results of this new method to those of standard cross-validation and bagging.

5.5.1 Remarks regarding bragging algorithms

The following points are of importance regarding our simulation studies:

- Similar to bagging, there are two ways in which bragging can be applied to cross-validation bandwidth selection. In the first, the median of bootstrap versions of the cross-validation function is calculated to obtain $CV_{brag}(h)$ and consequently the estimated bandwidth $\hat{h}_{brag}$ is chosen to be the value of h that minimizes $CV_{brag}(h)$. Alternatively, the value $\hat{h}^*_{CV}$ that minimizes $CV^*(h)$ in each bootstrap sample is chosen, whereafter the estimated bandwidth $\hat{h}_{brag}$ is defined as the median value of $\hat{h}^*_{CV}$ over all the bootstrap samples. As for bagging, we report only on the second approach in this study.

- As for bagging, there are at least three different ways to perform bragging when the second approach mentioned above is used. We shall refer to these methods as:

  - ► Bragging I
  - ► Bragging II
  - ► Bragging III

  The algorithms to determine the final N-W estimators when these three methods are applied, are described in Algorithms 5.6, 5.7 and 5.8 respectively. Results and conclusions about the methods are presented in the next chapter.

- Resampling is done without replacement.

- Resamples of size $m = [an]$, $0 < a < 1$, are drawn in all algorithms.

- A rescaling step is required in Algorithm 5.6 below.

5.5.2 Bragging algorithms

We now present the three bragging algorithms.

Algorithm 5.6. ***Determine*** $\hat{\mathrm{m}}(\mathrm{x})$ ***by bragging 1 (Brag1)***

1. Repeat steps 1 to 4 of Algorithm 5.3.
2. Calculate the median of the bootstrap replications of the bandwidth

$$\hat{h}_{brag_m} = \text{median}_{1\leq b\leq B}\hat{h}^*_{CV}(b).$$

3. Rescale the bandwidth in step 2: $\hat{h}_{brag} = (\hat{h}_{brag_m})\left(\frac{m}{n}\right)^{\frac{1}{5}}$.
4. Fit the N-W estimator $\hat{m}_{NW}(x)$ in (5.1), using the original sample S_n and the bandwidth $\hat{h}_{brag}$.

Algorithm 5.7. ***Determine*** $\hat{\mathrm{m}}(\mathrm{x})$ ***by bragging 2 (Brag2)***

1. Repeat steps 1 to 4 of Algorithm 5.3.
2. Calculate the median of the bootstrap replications of the bandwidth

$$\hat{h}_{brag_m} = \text{median}_{1\leq b\leq B}\hat{h}^*_{CV}(b).$$

3. Use each bootstrap sample S^*_m and the median bandwidth $\hat{h}_{brag_m}$ and fit the N-W estimator given in (5.1) to obtain bootstrap versions of the regression estimate $\hat{m}^*_{NW}(x)^{(1)}, \hat{m}^*_{NW}(x)^{(2)}, \ldots, \hat{m}^*_{NW}(x)^{(B)}$.
4. Calculate the median of the bootstrap replications of the regression estimate

$$\hat{m}^*_{NW}(x) = \text{median}_{1\leq b\leq B}\hat{m}^*_{NW}(x)^{(b)}.$$

Algorithm 5.8. ***Determine*** $\hat{\mathrm{m}}(\mathrm{x})$ ***by bragging 3 (Brag3)***

1. Let $S_n = \{(X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)\}$ be the given sample.
2. Randomly sample m data pairs without replacement from S_n to obtain a bootstrap sample $S^*_m = \{(X^*_1, Y^*_1), (X^*_2, Y^*_2), \ldots, (X^*_m, Y^*_m)\}$.

3. Calculate a bootstrap version of the leave-one-out cross-validation function in (5.3) by using the bootstrap sample S_m^* to obtain $CV^*(h)$ over a grid of h-values. Obtain the value $\hat{h}_{CV}^*$ that minimizes $CV^*(h)$.

4. Use the bootstrap sample S_m^* and corresponding bandwidth $\hat{h}_{CV}^*$ and fit the N-W estimator in (5.1) to obtain a bootstrap version of the regression estimate $\hat{m}_{NW}^*(x)$.

5. Repeat steps 2, 3 and 4 B times to obtain bootstrap replications of the regression estimate $\hat{m}_{NW}^*(x)^{(1)}, \hat{m}_{NW}^*(x)^{(2)}, \ldots, \hat{m}_{NW}^*(x)^{(B)}$.

6. Calculate the median of the bootstrap replications of the regression estimate

$$\hat{m}_{NW}^*(x) = \text{median}_{1 \leq b \leq B} \hat{m}_{NW}^*(x)^{(b)}.$$

5.6 Bagged and bragged boosting

Finally, we introduce a set of methods that combines the improving effect of boosting with that of bagging or bragging. We presume that these new methods will result in even better regression estimates than the methods discussed in this chapter thus far, although the possibility of overfitting should be guarded against. Our presumption will be investigated empirically in Chapter 6.

As mentioned in Section 5.3, boosting can improve the performance of the standard N-W estimator. In the boosted N-W estimator, two parameters need to be tuned, i.e. the number of boosting iterations T and the bandwidth h. One way to select T and h is by means of the cross-validation method. The goal of our new procedures is to apply bagging and bragging to the cross-validation selection of parameters T and h. We refer to these procedures as "bagged boosting" and "bragged boosting" respectively.

5.6.1 Remarks regarding bagged and bragged boosting algorithms

The following points are of importance regarding our simulation studies:

- The bagged and bragged boosting methods can be outlined as follows:

  (i) For both bagged and bragged boosting, the boosting cross-validation function in (5.8) is calculated over all bootstrap samples to obtain bootstrap versions, $CV^*(h, T)$.

  (ii) The number of boosting iterations $\hat{T}_{opt}^*$ that minimizes $CV^*(h, T)$ is determined for each bootstrap sample.

(iii) The optimal number of boosting iterations is determined:

For bagged boosting, the average value of the $\hat{T}^*_{opt}$ values is calculated and rounded to the nearest integer to obtain the bagging best number of boosting iterations $\hat{T}_{bagboost}$.

For bragged boosting, the median value of the $\hat{T}^*_{opt}$ values is determined and rounded to the nearest integer to obtain the bragging best number of boosting iterations $\hat{T}_{bragboost}$.

(iv) The corresponding optimal bandwidth is determined:

For bagged boosting, the bandwidth that has minimized $CV^*(h, \hat{T}_{bagboost})$, i.e. the cross-validation function for the $\hat{T}_{bagboost}$-th boosting iteration, is retained for each bootstrap sample. The average of these bandwidths is chosen as $\hat{h}_{bagboost}$.

For bragged boosting, the bandwidth that has minimized $CV^*(h, \hat{T}_{bragboost})$, i.e. the cross-validation function for the $\hat{T}_{bragboost}$-th boosting iteration, is retained for each bootstrap sample. The median value of these bandwidths is chosen as $\hat{h}_{bragboost}$.

(v) The selected values for T and h are used to obtain the regression estimates.

- More specifically, bagged and bragged boosting can each be approached in three different ways. We shall refer to these methods as:
  - ► Bagged boosting I and bragged boosting I
  - ► Bagged boosting II and bragged boosting II
  - ► Bagged boosting III and bragged boosting III

  The algorithms to determine the final boosted N-W estimators when these methods are applied, are described in Algorithms 5.9 to 5.14. Results and conclusions about the methods are presented in the next chapter.
- Resampling is done without replacement.
- Resamples of size $m = [an]$, $0 < a < 1$, are drawn in all algorithms.
- A rescaling step is required in Algorithms 5.9 and 5.12 below.

5.6.2 Bagged boosting algorithms

We now present the following algorithms to enable bagged boosting.

Algorithm 5.9. ***Determine*** $\hat{\mathrm{m}}(\mathrm{x})$ ***by bagged boosting 1 (Bagboost1)***

1. Let $S_n = \{(X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)\}$ be the given sample.

2. Randomly sample m data pairs without replacement from S_n to obtain a bootstrap sample $S_m^* = \{(X_1^*, Y_1^*), (X_2^*, Y_2^*), \ldots, (X_m^*, Y_m^*)\}$.

3. (a) Calculate a bootstrap version of the boosting leave-one-out cross-validation function in (5.8) by using the bootstrap sample S_m^* to obtain $CV^*(h, T)$ over a grid of h-values for a range of boosting iterations $T = 0, 1, \ldots, T_{max}$, where T_{max} is the maximum number of boosting iterations allowed.

   (b) Determine the minimum value of $CV^*(h, T)$ over all (h, T) combinations. Denote the number of boosting iterations that results in this minimum value by $\hat{T}_{opt}^*$.

   (c) For each boosting iteration (i.e. for $T = 0, 1, \ldots, T_{max}$), determine the minimum value of $CV^*(h, T)$. Denote the bandwidth that minimizes $CV^*(h, T)$ for each fixed value of T by $\hat{h}_T^*$, $T = 0, 1, \ldots, T_{max}$.

4. Repeat steps 2 and 3 B times to obtain bootstrap replications of the optimal number of boosting iterations $\hat{T}_{opt}^*(1), \hat{T}_{opt}^*(2), \ldots, \hat{T}_{opt}^*(B)$ and, for each boosting iteration $T = 0, 1, \ldots, T_{max}$, the optimal bandwidths $\hat{h}_T^*(1), \hat{h}_T^*(2), \ldots, \hat{h}_T^*(B)$.

5. (a) Calculate the average optimal number of boosting iterations and round it off to the nearest integer

$$\hat{T}_{bagboost} = \text{ROUND}\Big\{\frac{1}{B}\sum_{b=1}^{B}\hat{T}_{opt}^*(b)\Big\}.$$

   (b) Calculate that average optimal bandwidth for $\hat{T}_{bagboost}$ boosting iterations over all bootstrap samples

$$\hat{h}_{bagboost_m} = \frac{1}{B}\sum_{b=1}^{B}\hat{h}^*_{\hat{T}_{bagboost}}(b).$$

   (c) Rescale the bandwidth: $\hat{h}_{bagboost} = (\hat{h}_{bagboost_m})\left(\frac{m}{n}\right)^{\frac{1}{5}}$.

   (d) Fit the boosted N-W estimator in Algorithm 5.2 to the original sample S_n, using $\hat{T}_{bagboost}$ boosting iterations and bandwidth $\hat{h}_{bagboost}$.

Algorithm 5.10. ***Determine*** $\hat{\mathrm{m}}(\mathrm{x})$ ***by bagged boosting 2 (Bagboost2)***

1. Repeat steps 1 to 4 of Algorithm 5.9.

2. (a) Calculate the average optimal number of boosting iterations and round it off to the nearest integer

$$\hat{T}_{bagboost} = \text{ROUND}\Big\{\frac{1}{B}\sum_{b=1}^{B}\hat{T}^*_{opt}(b)\Big\}.$$

(b) Calculate that average optimal bandwidth for $\hat{T}_{bagboost}$ boosting iterations over all bootstrap samples

$$\hat{h}_{bagboost_m} = \frac{1}{B}\sum_{b=1}^{B}\hat{h}^*_{\hat{T}_{bagboost}}(b).$$

(c) Fit the boosted N-W estimator in Algorithm 5.2 to each of the bootstrap samples S^*_m, using $\hat{T}_{bagboost}$ boosting iterations and bandwidth $\hat{h}_{bagboost_m}$. Denote the resulting bootstrap versions of the boosted regression estimate by

$$\hat{m}^*_{\hat{T}_{bagboost}}(x)^{(1)}, \hat{m}^*_{\hat{T}_{bagboost}}(x)^{(2)}, \ldots, \hat{m}^*_{\hat{T}_{bagboost}}(x)^{(B)}.$$

(d) Calculate the average of the bootstrap replications of the regression estimate

$$\hat{m}^*_{\hat{T}_{bagboost}}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{m}^*_{\hat{T}_{bagboost}}(x)^{(b)}.$$

Algorithm 5.11. ***Determine*** $\hat{\text{m}}(\text{x})$ ***by bagged boosting 3 (Bagboost3)***

1. Let $S_n = \{(X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)\}$ be the given sample.

2. Randomly sample m data pairs without replacement from S_n to obtain a bootstrap sample $S^*_m = \{(X^*_1, Y^*_1), (X^*_2, Y^*_2), \ldots, (X^*_m, Y^*_m)\}$.

3. (a) Calculate a bootstrap version of the boosting leave-one-out cross-validation function in (5.8) using the bootstrap sample S^*_m to obtain $CV^*(h, T)$ over a grid of h-values for a range of boosting iterations $T = 0, 1, \ldots, T_{max}$, where T_{max} is the maximum number of boosting iterations allowed.

(b) Determine the minimum value of $CV^*(h, T)$ over the range of all (h, T) combinations. Denote the (h, T) combination that results in this minimum value by $(\hat{T}^*_{opt}, \hat{h}^*_{opt})$.

4. Fit the boosted N-W estimator in Algorithm 5.2 to the bootstrap sample S^*_m, using $\hat{T}^*_{opt}$ boosting iterations and bandwidth $\hat{h}^*_{opt}$. Denote the resulting bootstrap version of the boosted regression estimate by $\hat{m}^*_{\hat{T}^*_{opt}}(x)$.

5. Repeat steps 2, 3 and 4 B times to obtain bootstrap replications of the boosted regression estimates, $\hat{m}^*_{\hat{T}^*_{opt}}(x)^{(1)}, \hat{m}^*_{\hat{T}^*_{opt}}(x)^{(2)}, \ldots, \hat{m}^*_{\hat{T}^*_{opt}}(x)^{(B)}$.

6. Calculate the average of the bootstrap replications of the regression estimate

$$\hat{m}^*_{\hat{T}^*_{opt}}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{m}^*_{\hat{T}^*_{opt}}(x)^{(b)}.$$

5.6.3 Bragged boosting algorithms

We now present the following algorithms to enable bragged boosting.

Algorithm 5.12. *Determine* $\hat{\mathrm{m}}(\mathrm{x})$ *by bragged boosting 1 (Bragboost1)*

1. Repeat steps 1 to 4 of Algorithm 5.9.

2. (a) Calculate the median optimal number of boosting iterations and round it off to the nearest integer

$$\hat{T}_{bragboost} = \mathrm{ROUND}\Big\{\mathrm{median}_{1\leq b\leq B}\hat{T}^*_{opt}(b)\Big\}.$$

   (b) Calculate that median optimal bandwidth for $\hat{T}_{bragboost}$ boosting iterations over all bootstrap samples

$$\hat{h}_{bragboost_m} = \mathrm{median}_{1\leq b\leq B}\hat{h}^*_{\hat{T}_{bragboost}}(b).$$

   (c) Rescale the bandwidth: $\hat{h}_{bragboost} = (\hat{h}_{bragboost_m})\left(\frac{m}{n}\right)^{\frac{1}{5}}$.

   (d) Fit the boosted N-W estimator in Algorithm 5.2 to the original sample S_n, using $\hat{T}_{bragboost}$ boosting iterations and bandwidth $\hat{h}_{bragboost}$.

Algorithm 5.13. *Determine* $\hat{\mathrm{m}}(\mathrm{x})$ *by bragged boosting 2 (Bragboost2)*

1. Repeat steps 1 to 4 of Algorithm 5.9.

2. (a) Calculate the median optimal number of boosting iterations and round it off to the nearest integer

$$\hat{T}_{bragboost} = \mathrm{ROUND}\Big\{\mathrm{median}_{1\leq b\leq B}\hat{T}^*_{opt}(b)\Big\}.$$

(b) Calculate that median optimal bandwidth for $\hat{T}_{bragboost}$ boosting iterations over all bootstrap samples

$$\hat{h}_{bragboost_m} = \text{median}_{1 \leq b \leq B} \hat{h}^*_{\hat{T}_{bragboost}}(b).$$

(c) Fit the boosted N-W estimator in Algorithm 5.2 to each of the bootstrap samples S^*_m, using $\hat{T}_{bragboost}$ boosting iterations and bandwidth $\hat{h}_{bragboost_m}$. Denote the resulting bootstrap versions of the boosted regression estimate by

$$\hat{m}^*_{\hat{T}_{bragboost}}(x)^{(1)}, \hat{m}^*_{\hat{T}_{bragboost}}(x)^{(2)}, \ldots, \hat{m}^*_{\hat{T}_{bragboost}}(x)^{(B)}.$$

(d) Calculate the median of the bootstrap replications of the regression estimate

$$\hat{m}^*_{\hat{T}_{bragboost}}(x) = \text{median}_{1 \leq b \leq B} \hat{m}^*_{\hat{T}_{bragboost}}(x)^{(b)}.$$

Algorithm 5.14. ***Determine*** $\hat{\text{m}}(\text{x})$ ***by bragged boosting 3 (Bragboost3)***

1. Repeat steps 1 to 5 of Algorithm 5.11.

2. Calculate the median of the bootstrap replications of the regression estimate

$$\hat{m}^*_{\hat{T}^*_{opt}}(x) = \text{median}_{1 \leq b \leq B} \hat{m}^*_{\hat{T}^*_{opt}}(x)^{(b)}.$$

Chapter 6

Results and conclusions

6.1 Introduction

The choice of the kernel function and bandwidth are important for the nonparametric estimation of regression functions. The aim is usually to find the kernel function and bandwidth that minimize some discrepancy measure, such as the mean integrated squared error (MISE). Although the Nadaraya-Watson (N-W) estimator is generally regarded as a trustworthy way to obtain answers regarding regression problems, it may be true that the N-W estimator can provide even better answers if improvement methods are applied to the process. In this chapter we present the results of Monte Carlo studies conducted to assess and compare the improvement procedures of which the definitions and algorithms were discussed in Chapter 5, as applied to the N-W kernel regression estimator. The specific methods considered are:

- boosting,
- bagging,
- bragging,
- bagged boosting and
- bragged boosting.

In order to draw useful conclusions about the different methods, it is necessary to evaluate the performance of each method in various simulation settings. In particular, we assume three different underlying regression functions $m(x)$ (see Section 6.2.1).

In our studies we vary

- the underlying distribution of the covariate data,
- the variance of the error term used to generate response data,
- the sample size and
- the kernel function used in the formulation of the N-W estimator

with the aim of making meaningful new recommendations and conclusions. A discussion of how we varied each of the aspects in the list above and other aspects of the simulation setup, is provided in Section 6.2.

As mentioned in Chapter 5, our simulation study consists of two studies, i.e. Simulation Studies I and II described in Section 5.2.4, with results and conclusions reported in Sections 6.3, 6.4 and 6.5, and tables of results and graphs listed in Appendices A, B and C.

Overall conclusions are listed in Section 6.6.

6.2 Setup of the simulation studies

In this section, we point out aspects of the simulation setup that apply for both studies. Many possible different selections could have been applied in the topics mentioned below. In some cases we followed the choices made by Marzio and Taylor (2008). However, we explain the choices we made below.

6.2.1 The underlying regression function m(x)

Three different regression models $m(x)$ are considered, which will be referred to as "Model 1", "Model 2" and "Model 3" respectively. These models are displayed in Table 6.1.

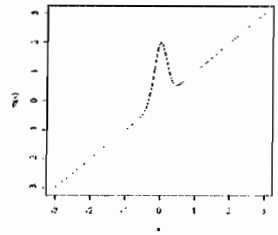
To motivate the use of these three models, we refer to Marzio and Taylor (2008), who studied Models 1, 2 and 3, as well as a fourth model

$$m(x) = 0.4x + 1. \tag{6.1}$$

These four models, proposed by Fan and Gijbels (1996, p. 111) represent so-called "difficult estimation problems due to the level of the noise-to-signal ratios" inherent in the forms (Marzio and Taylor 2008, p. 2489).

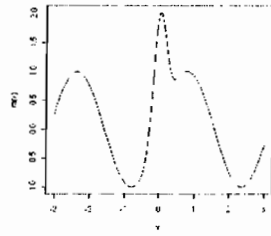
For the purpose of this study, we do not attempt the estimation of the fourth model of Fan and Gijbels (1996) presented in (6.1). The reason why it is omitted, is related to the

Model 1



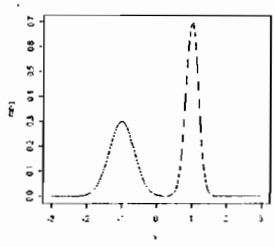
$$m(x) = x + 2\exp(-16x^2)$$

Model 2



$$m(x) = \sin(2x) + 2\exp(-16x^2)$$

Model 3



$$m(x) = 0.3\exp\{-4(x+1)^2\} + 0.7\exp\{-16(x-1)^2\}$$

Figure 6.1: Simulation models

results of Müller (1993, p. 139), who points out that one of the main problems of the N-W estimator is the difficulty with which it estimates straight regression lines when X is not uniformly distributed. The estimation of the model in (6.1) when the data is drawn from an $N(0,1)$-distribution illustrates this problem. In their study, Marzio and Taylor (2008) show that boosting provides a solution to this problem. However, this is a research topic in own

right and will be pursued further in another investigation project.

Note that Model 1 is almost a straight line. It was mentioned above that the N-W estimator has difficulty in estimating straight lines. In other words, the N-W estimator is a weak learner for straight line regression relationships. We therefore expect that boosting will effectively improve the N-W estimation of Model 1. Marzio and Taylor (2008) found this presumption to be true and we will confirm their finding for a wider range of settings.

We extend the limited study of Marzio and Taylor (2008) by applying boosting to the first three of the models of Fan and Gijbels (1996), but for a wider range of simulation settings as discussed in the points below.

6.2.2 Construction of the data

The following aspects are of interest:

a) Underlying distribution of the covariate data:

Two distributions are considered for the covariate data, i.e. X-values are drawn from the $U(-2, 2)$ and $N(0, 1)$ distributions respectively.

b) Construction of the response data:

Response data Y are then constructed according to the rule

$$Y = m(X) + \sigma\varepsilon, \tag{6.2}$$

where:

- the n ε-values are random errors drawn from the $N(0, 1)$ distribution,
- σ is chosen to assume the values 0.2, 0.6 and 1.0 for Models 1 and 2 and 0.1, 0.3 or 0.5 for Model 3 and
- m is Model 1, 2 or 3 in Table 6.1.

c) Sample size:

Marzio and Taylor (2008) used samples of sizes $n = 50$, $n = 100$ and $n = 200$ in their study. We apply these sample sizes in our study as well.

6.2.3 More simulation aspects that apply to both studies

Apart from the choices regarding the simulation setup, other simulation aspects are of concern as well, of which some are listed below. The discussion below is applicable to both Simulation Study I and II.

a) Kernel function:

The standard normal and triangular kernels, presented in Figure 6.2, are used in the kernel regression estimation in this study.

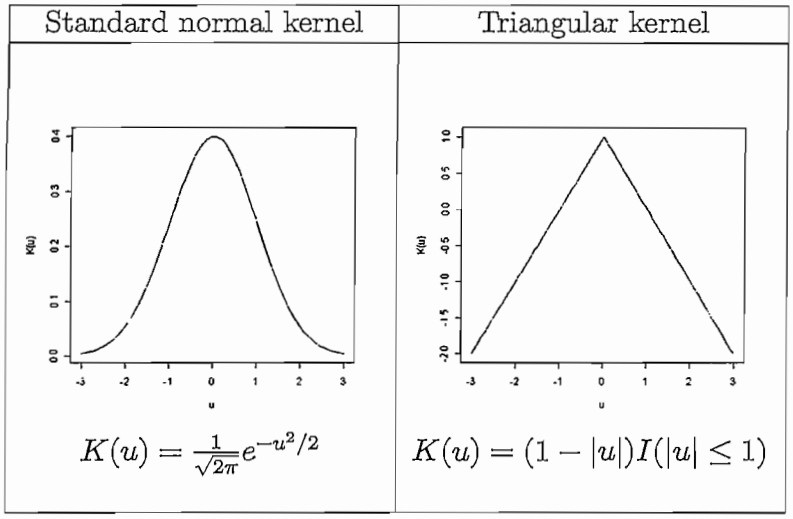


Figure 6.2: Kernels

To motivate this choice of kernel functions, we refer to the discussion in Section 2.6: Marzio and Taylor (2008) showed that boosting of the N-W estimator works properly only if the kernel yields strictly positive characteristic roots. This is the case for the $N(0,1)$ and triangular kernels, but not for many other popular kernels, such as the Epanechnikov, biweight and triweight kernels. We therefore restrict our study to the findings of Marzio and Taylor (2008), i.e. the $N(0,1)$ and triangular kernels.

Note, however, that the Epanechnikov and other kernels mentioned above can be applied to other estimators than the boosted N-W estimator. For example, these kernels can be applied in investigations concerning the effect of the bragging and bragging methods on the standard N-W estimator. Investigations on the application of other kernels than the $N(0,1)$ and triangular kernels will be considered in another study.

b) Grid of h-values:

The grids of allowed bandwidths differed for Simulation Study I and Simulation Study II. This matter will be addressed in Sections 6.2.4 and 6.2.5.

c) Maximum number of boosting iterations:

Marzio and Taylor (2008) allowed a maximum of 6 boosting iterations in their application of boosting to Models 1, 2 and 3. In a small scale study we found that there were cases for which the optimal number of boosting iterations exceeded 6. However, the improvement regarding the minimization of the MISE for these additional boosting iterations was very small. Because the boosting algorithms are very computer-time intensive, allowing for extra boosting iterations has great time implications. Considering the extra computation time that would be needed for extra boosting iterations and the relatively small reduction in the corresponding MISE-values, we limit the number of boosting iterations to 6. Small scale investigations with up to 20 boosting iterations, confirmed the remarks above.

d) Number of Monte Carlo iterations:

Both Marzio and Taylor (2008) and Hall and Robinson (2009) used $MC = 200$ in their simulation studies. We therefore generate 200 Monte Carlo samples ($MC = 200$) for sample sizes $n = 50$, $n = 100$ and $n = 200$ respectively.

e) Grid of x-values:

To determine the approximated MISE, as described in Algorithm 5.1, the pointwise MSE has to be calculated for a grid of x-values. A fixed grid is constructed between the values -2 and $+2$ whenever covariate data are drawn from the $U(-2, 2)$ distribution and between -3 and $+3$ for covariate data from the $N(0, 1)$ distribution. 100 grid points are used in each case.

f) Correction factor:

Initially, we experienced problems in Simulation Study I with extremely small values in the denominators during the execution of the algorithms. These small values in the denominators caused the cross-validation function to break down. Fan (1993) proposed adding a factor of n^{-2} to the denominator of the nonparametric regression estimator defined in formula (2.2) of his article due to "a technical reason (to avoid zero in the denominator)". Similarly, we

add a factor of 0.001 to the denominator, whenever the value of the denominator is less than 0.001. Hall and Robinson (2009, p. 178) followed a similar approach in their regression simulation studies and called the process "ridge correction".

Although the addition of a correction factor is not necessary for Simulation Study II (the denominator never reaches small enough values to cause the procedure to break down), we add the correction factor to small denominators in this study as well, in order to make comparisons between results of Simulation Study I (the MISE resulting from cross-validation and improved cross-validation choices for h and T) and Simulation Study II (the MISE resulting from "optimal" choices for h and T) possible.

g) Univariate simulation setup:

For simplicity, simulation studies are conducted for the univariate case, where the relationship between a single predictor variable and a single response variable is studied. The studies can easily be extended to include multivariate scenarios.

h) Computer package:

All calculations are done using double precision arithmetic in Compaq Visual Fortran 6 and routines from the MSIMSL library. Graphs are drawn with R 2.8.1. The Fortran code can be found on the CD-ROM attached to this dissertation.

6.2.4 More simulation aspects that apply to Simulation Study I

In addition to the aspects of the simulation setup applicable to both Simulation Study I and II that were discussed above, there are some aspects of Simulation Study I that need special consideration. These matters are now raised.

a) Sample sizes and number of iterations:

Simulation Study I consists of two parts, which will be explained in Section 6.3. The first part is referred to as SimIA and the second part is referred to as SimIB. The following sample sizes and number of iterations are of interest:

- Number of boosting iterations:

  SimIA allows for a maximum of 1 boosting iteration, while a maximum of 6 boosting iterations are allowed in SimIB.

- Number of bootstrap samples:

  Hall and Robinson (2009) use 50 bootstrap samples in their study. We follow this choice as guideline and draw 50 bootstrap samples for the application of the bagging and bragging methods throughout.

- Number of points in the bootstrap resamples:

  In the bagging and bragging algorithms presented in Chapter 5 it is indicated that bootstrap resamples of size $m = [an]$, $0 < a < 1$, are drawn, where $[y]$ denotes the largest integer value smaller than or equal to y. Bühlmann (2004, p. 884) suggests that $m = [n/2]$, i.e. $a = 0.5$, is often a good choice in practice, when one does not want to optimize over the tuning parameter m. We use $a = 0.5$ throughout. Swanepoel (1988,1990) takes $m = n/3$, although it was suggested by Swanepoel (1986) that m should be $2n/3$.

  Note that Hall and Robinson (2009) investigate six different values of a in their study, namely 0.0625, 0.125, 0.25, 0.375, 0.5 and 0.75. In future investigations, we shall extend our study to include a range of values of a in order to explore the proposed new improvement methods for a wider range of resample sizes. Preliminary small studies indicated than $m = 2n/3$ performs better than $m = n/2$ or $m = n/3$.

b) **h-Grid issues:**

In Simulation Study II we find that the optimal bandwidths differ greatly for the different simulation setup scenarios. A challenge in Simulation Study I is to determine an h-grid for each simulation setup scenario. With regard to this challenge, we make a few remarks.

- The role of the h-grids in Simulation Study I and II:

  Note the difference between Simulation Study I and II with regard to the h-grids:

  Simulation Study II

  Issues on the h-grid in Simulation Study II are discussed in Section 6.2.5. Note that, in Simulation Study II, the h-grid is applied only once, i.e. the mean integrated squared error over the 200 Monte Carlo samples is calculated only once for each h-grid value. These MISE values are then minimized to find the optimal h in terms of the MISE. Since the h-grid is applied only once, it is easy to widen the grid if a boundary value of the grid is chosen as the optimal h-value.

Simulation Study I

In Simulation Study I, the h-grid is applied repeatedly. To see this, consider the following cases:

1. When cross-validation (i.e. cross-validation without improvement methods bagging or bragging) is used to choose the bandwidth for the standard or boosted N-W estimator (procedures NW, Boost_1 and Boost_6), the h-grid is applied individually in the cross-validation algorithm for each Monte Carlo sample. For each Monte Carlo sample, the bandwidth that minimizes the cross-validation function is chosen as the cross-validation optimal bandwidth for that Monte Carlo sample and used in the estimation of $m(x)$ in step 2 of the main algorithm, Algorithm 5.1.

   The number of times that the h-grid is applied when ordinary cross-validation is used to select the bandwidth, is as follows:

   - For the standard N-W estimator (procedure: NW), the h-grid is applied $MC = 200$ times, where $MC = 200$ is the number of Monte Carlo samples, since for each MC sample, $\min_h CV(h)$ must be found over a grid of h-values.
   - For the boosted N-W estimator (procedures: Boost_1 and Boost_6) the h-grid is applied an additional $MC \times (T + 1)$ times, where $MC = 200$ is the number of Monte Carlo samples and T is the maximum number of boosting iterations allowed, since the above process is repeated $(T + 1)$ times.
     Thus, for $T = 1$ (procedure: Boost_1) the h-grid is applied 400 times and for $T = 6$ (procedure: Boost_6) 1 400 times.

2. When the bagging or bragging method is used to improve cross-validation bandwidth selection in the standard or boosted N-W estimator (procedures Bag1, Bag2, Bag3, Brag1, Brag2, Brag3, Bagboost1, Bagboost2, Bagboost3, Bragboost1, Bragboost2 and Bragboost3), the h-grid is applied individually in the cross-validation algorithm for each bootstrap sample of each Monte Carlo sample. For each bootstrap sample of each Monte Carlo sample, the bandwidth that minimizes the cross-validation function is chosen as the bootstrap cross-validation optimal bandwidth for that bootstrap sample. These bootstrap bandwidths can then be combined in various ways, as described in the bagging, bragging, bagged boosting and bragged boosting algorithms in Chapter 5, for the estimation of $m(x)$ in step 2 of the main algorithm.

   The number of times that the h-grid is applied when bagged or bragged cross-validation

is used to select the bandwidth, is as follows:

- For the bagged or bragged standard N-W estimator (procedures: Bag1, Bag2, Bag3, Brag1, Brag2 and Brag3), the h-grid is applied $MC \times BagNr = 10000$ times, where $MC = 200$ is the number of Monte Carlo samples and $BagNr = 50$ is the number of bootstrap samples.
- For the bagged or bragged boosted N-W estimator (procedures: Bagboost1, Bagboost2, Bagboost3, Bragboost1, Bragboost2 and Bragboost3), the h-grid is applied an additional $MC \times BagNr \times (T+1) = 70000$ times, where $MC = 200$ is the number of Monte Carlo samples, $BagNr = 50$ is the number of bootstrap samples and $T = 6$ is the maximum number of boosting iterations allowed.

When considering the large number of times that the h-grid is called in Simulation Study I in comparison with Simulation Study II, it is clear that it will not always be computer-time economic in Simulation Study I to choose the h-grid wide enough to ensure that the boundary values of the h-grid are not chosen as optimal bandwidths, as was done in Simulation Study II.

The challenge is to find a grid for each simulation setup scenario that restricts the number of times that the boundary values of the grid is chosen as optimal cross-validation bandwidths to a minimum. The sub paragraph on h-grid boundaries below, contains more information.

- h-grid issues for boosting studies:

  In Simulation Study II it was seen that optimal bandwidths for boosted N-W estimation are generally larger than for standard N-W estimation. The h-grids in Simulation Study I should thus be wide enough to accommodate both the standard and boosted versions of the N-W estimator.

  Another approach would be to apply different h-grids to the standard and the boosted N-W estimators respectively. We do not follow this approach in our current study, but recommend that it should be considered in future studies. Such an approach would save computer computation time and could result in better estimators, since the grids can then be tuned specifically for standard and boosted N-W estimation respectively.

- h-grid issues for bootstrap studies:

  As mentioned before, we use bootstrap samples of size $m = n/2$ in the bagging, bragging, bagged boosting and bragged boosting algorithms. Optimal cross-validation bandwidths for samples of size m, where $m < n$, will be larger than for samples of size n, which is compensated for in our algorithms by rescaling the bagged or bragged bandwidths based on bootstrap samples of size m before it is applied to the initial samples of size n. For example, recall that the rescaling step in the first bagging algorithm involves the calculation

  $$\hat{h}_{bag} = (\hat{h}_{bag_m}) \left(\frac{m}{n}\right)^{\frac{1}{5}},$$

  where $\hat{h}_{bag_m}$ denotes the bagged bandwidth obtained from the bootstrap samples of size m and $\hat{h}_{bag}$ denotes the rescaled bandwidth that can be used to estimate $m(x)$ using the initial sample of n data points.

  The implication of the above comment on our choice of h-grids, is that, when the bagging, bragging, bagged boosting and bragged boosting methods, using bootstrap samples of size m, are applied, the h-grids should include larger h-values than the optimal bandwidths selected in Simulation Study II, which was based on samples of size n.

  Once again, one can consider applying different h-grids for usual and bagged or bragged cross-validation bandwidth selection respectively. We do not follow this approach in our current study, but recommend that it should be investigated in future studies, with the aim of saving computer computation time and obtaining better bagged or bragged estimates.

- h-Grid boundaries:

  By taking the above remarks, as well as optimal h-values from Simulation Study II into account, we have chosen preliminary h-grid boundaries for each of the simulation setup scenarios. The preliminary grids were then applied in a small scale study involving only 10 Monte Carlo samples and 10 bootstrap iterations. Cross-validation choices of h for these small scale studies were calculated and the grids were adapted accordingly when it seemed necessary to obtain final grids for each of the simulation setup scenarios. These grids are presented in the result tables in Appendix A. We display the grid-values in the tables in the form 0.02(0.01)1.00. This means that the h-grid for a specific procedure is allowed to range from 0.02 to 1.00 in increments of 0.01.

  In some cases the grids, although chosen with care in a small study, proved to be too small in the large scale study in the sense that the boundaries were selected too often. In this

regard, note the following:

- Cases where the selected bootstrap bandwidths in the bagging or bragging algorithms hit the upper or lower boundaries of the h-grids respectively for more than 200 out of the possible 1400 bootstrap samples, are indicated by $*$-signs next to the upper and/or lower boundary values in the result tables. h-Grids for these scenarios should be chosen over larger intervals in future investigations.
- For some large σ scenarios, especially for Model 3, it was already clear from the small scale study conducted to establish the h-grids, that cross-validation selection of h is very unstable, i.e. the selected h-values show large variation, so that determining a feasible h-grid seems impossible. In the Model 3 cases the h-grids are fixed to 0.01(0.01)1.00, even though we know that such grids are not wide enough. These cases correspond to cases in Simulation Study II where the optimal h-values in terms of the MISE turned out to be very large, for example scenarios where the maximum allowed h-values in Simulation Study II, i.e. $h = 5.0$, was selected as the optimal bandwidth in terms of the MISE.

Considering the two points above, it is clear that a detailed study on the behaviour of h needs to be done for the problematic scenarios. This forms part of a future study.

6.2.5 More simulation aspects that apply to Simulation Study II

In addition to the discussion of the simulation setup in Sections 6.2.1 to 6.2.3, bandwidth values allowed in Simulation Study II need to be explained.

h-Grid issues:

Due to the nature of Simulation Study II, it is easy to force the borders of the smoothing parameter grids in this part of the study to be wide enough so that the smoothing parameters that result in the minimum MISE values are never the boundary values of the grids. More specifically, recall from Chapter 5.2.4 that, in Simulation Study II, the main algorithm is executed repeatedly for a grid of possible parameter values. For each parameter value, i.e. for each h in the standard N-W estimator and for each (h, T) pair in the boosted N-W estimator, the approximated MISE value is recorded. The h or (h, T) pair that minimizes the approximated MISE, is the selected "optimal" smoothing parameter or smoothing parameter pair for the standard or boosted N-W estimator respectively. Whenever the "optimal" h or

(h, T) values for the allowed grids are given as boundary values of the grids, the grids can simply be expanded to prevent boundary values from being chosen.

With regard to the selection of h, the h-grid, calculated in increments of 0.01, is adapted for each setting to ensure that the selected optimal h-values in the N-W and boosted N-W estimators are not the lower or upper boundaries of the grid. In most settings, optimal h-values range between 0.05 and 2, as shown in the tables in Appendix B. Note, however, that for some Model 3, large σ settings (i.e. $\sigma = 0.5$), the allowed h-values should become quite large to prevent the upper boundary values of the grids from being chosen as the optimal bandwidths. In these cases, the grids are expanded to a maximum upper boundary value of $h = 5.0$. We indicate cases where the boundary value $h = 5.0$ is chosen as the optimal bandwidth with $*$'s in the result tables. These cases will be considered in the discussion in Section 6.4.2.

6.3 Observations and conclusions from Simulation Study I

As mentioned before, the aim of Simulation Study I is to investigate whether improvement methods that utilize bootstrap methodology, i.e. the bagging and bragging methods, can improve cross-validation smoothing parameter selection in the N-W and the boosted N-W estimators. In this section we explain and interpret the results of Simulation Study I. The result tables of Simulation Study I are displayed in Appendix A.

In Section 6.3.1 the layout of the tables with results of Simulation Study I are explained, to assist the reader in interpreting the tables. In Section 6.3.2 the results presented in the tables are interpreted formally and compared to those of Hall and Robinson (2009).

It should be kept in mind that Simulation Study I has the following goals:

1. To apply the boosting, bagging and bragging methods to the standard N-W estimator.

   Firstly, the effect of the boosting, bagging and bragging methods when applied to the standard N-W estimator, are studied. In the remainder of this paragraph we shall refer to the first part of Simulation Study I as SimIA. In particular, the following tasks are performed in SimIA:

   - Bagging as well as bragging methods are applied to the cross-validation selection of h, to be used in the standard N-W estimator.

- A further study that aims to investigate the potential parameter value and computer time issues of the boosting method to improve the N-W estimator, is conducted.

  The latter study was necessary, because it was observed in Simulation Study II (see Section 6.4) that boosting is not effective in all simulation setup scenarios. Furthermore, boosting is very computer-time intensive. To allow (in Simulation Study I) for 6 boosting iterations in scenarios where boosting was proved not to be effective (in Simulation Study II), or to apply improvement methods such as bagging and bragging to the boosting algorithm in these scenarios, is not time-economic and unnecessary. In SimIA, we allow for a maximum of only 1 boosting iteration and in this way we identify settings where boosting shows potential. By using these results together with the results from Simulation Study II, settings where boosting proofs to be practical and useful, are identified. In SimIB, which will be discussed below, boosting with 6 boosting iterations, as well as bagged and bragged boosting are applied to some of the identified settings.

Results of SimIA are displayed in Tables A.1 to A.12 in Appendix A.

2. To apply bagging and bragging to the boosted N-W estimator.

   In this part, which will be referred to as SimIB in the remainder of the study, a full scale boosting study is conducted in some of the settings which have been identified in Simulation Study II and SimIA as settings where boosting has potential to improve the N-W estimator. In particular, the following tasks are performed in SimIB:

   - The boosting algorithm, where the best (h, T) combination is selected by means of ordinary cross-validation, is executed to estimate $m(x)$, where a maximum of 6 boosting iterations are allowed.
   - Bagging and bragging methods are applied to the cross-validation selection of the combination (h, T) in the boosted N-W estimator. Once again, a maximum of 6 boosting iterations are allowed in the execution of the bagged and bragged boosting algorithms.

   Results of SimIB are displayed in Tables A.13 to A.30 in Appendix A.

6.3.1 Guidelines for reading Tables A.1 to A.30 of results

The results of SimIA and SimIB are displayed in Appendix A. Guidelines to read the respective tables are now provided.

1. The results of SimIA, where the effect of boosting, bagging and bragging on the standard N-W estimator are explored, are reported in Table A.1 to Table A.12. The following remarks can be used as guidelines when consulting these tables:

   - Each of the 12 tables displays the results of a particular combination of model choice, underlying covariate distribution of the predictor data X and kernel choice. The specific combination is indicated in the caption of the table.
   - Each table consists of nine 'blocks' of results, where each block is the result of a Monte Carlo study on its own and is not connected to the other blocks. The various blocks represent the different simulation setup scenarios included in SimIA.
   - In each block, the h-grid used for the particular simulation setup scenario is presented, in the form explained in Section 6.2.4. See Section 6.2.4 for a discussion of how the h-grids were determined. Recall from this discussion that a $*$-sign next to a minimum or maximum grid value indicates that the particular boundary value was hit more than 200 times during the bagging and bragging process. For these scenarios, wider grids should be considered in future investigations.
   - Each block consists of three columns. The three columns respectively display the MISE, the integrated squared bias and the integrated variance obtained when the main algorithm, Algorithm 5.1, is applied.
   - Each block consists of 8 rows. Each of the rows presents the results when a particular method is applied to obtain the estimate $\hat{m}(x)$ in step 2 of the main algorithm. The rows respectively display the following:
     - ► The first row contains the results when cross-validation is applied to select h for the standard N-W estimator. The label "NW" is used.
     - ► The second row contains the results when boosting with a maximum of one boosting iteration, as described in Algorithm 5.2, is applied to the standard N-W estimator, where the combination (h, T) is selected by ordinary cross-validation. The label "Boost" is used.
     - ► Rows 3 to 5 contain the results when bagging is applied to cross-validation bandwidth selection in the standard N-W estimator via Algorithms 5.3, 5.4 and 5.5 respectively. The labels "Bag1", "Bag2" and "Bag3" are used.
     - ► Rows 6 to 8 contain the results when bragging is applied to cross-validation bandwidth selection in the standard N-W estimator via Algorithms 5.6, 5.7 and 5.8

respectively. The labels "Brag1", "Brag2" and "Brag3" are used.

2. The results of SimIB, a boosting study conducted in some of the settings identified in Simulation Study II and SimIA as scenarios where boosting is effective, are reported in Table A.13 to Table A.30. The following remarks can be used as guidelines when consulting these tables:

   - Each of the tables represents a Monte Carlo study on its own and is not connected to the other tables. The various tables represent the different simulation setup scenarios that are included in SimIB. Each table's caption contains a description of the simulation setup scenario and the h-grid that is applied in the study. Note that the same h-grids are used as in SimIA.
   - Each table consists of four columns. The first column indicates the method that is applied to obtain the estimate $\hat{m}(x)$. The next three columns respectively display the MISE, the integrated squared bias and the integrated variance obtained when the main algorithm, Algorithm 5.1, is applied.
   - Each table consists of 15 rows. Each of the rows presents the results when a particular method is applied to obtain the estimate $\hat{m}(x)$ in step 2 of the main algorithm. Note that the first eight rows represent results from SimIA. We include these values again in the tables for SimIB to ease comparison and interpretation of the results in SimIB.

     The rows respectively display the following:

     ▶ The first seven rows respectively contains the NW, Bag1, Bag2, Bag3, Brag1, Brag2 and Brag3 values from SimIA. The eighth row contains the Boost-value from SimIA. Since Boost in SimIA allows for a maximum of 1 boosting iteration, this value is indicated by "Boost_1" in the SimIB tables.

     ▶ Row 9 contains the results when boosting with a maximum of six boosting iterations, as described in Algorithm 5.2, is applied to the standard N-W estimator, where the combination (h, T) is selected by ordinary cross-validation. The label "Boost_6" is used.

     ▶ Rows 10 to 12 contain the results when bagging is applied to cross-validation bandwidth selection in the boosted N-W estimator with a maximum of 6 boosting iterations via Algorithms 5.9, 5.10 and 5.11 respectively. The labels "Bagboost1", "Bagboost2" and "Bagboost3" are used.

- Rows 13 to 15 contain the results when bragging is applied to cross-validation bandwidth selection in the boosted N-W estimator with a maximum of 6 boosting iterations via Algorithms 5.9, 5.10 and 5.11 respectively. The labels "Bragboost1", "Bragboost2" and "Bragboost3" are used.

6.3.2 Interpretation of the observations and conclusions in Simulation Study I

The results of Simulations Study I are now discussed and conclusions are drawn.

A. Interpretation of the observations and conclusions of SimIA (Tables A.1 to A.12, Appendix A)

We now highlight observations made from the simulation results in SimIA. We interpret the results and draw conclusions.

Aims of the study: The aim of all the improvement methods is to provide better estimators for the mean regression curve than the standard N-W estimator, where cross-validation is used to select the bandwidth throughout.

We shall measure the improvement using the MISE as discrepancy measure. We now compare the achievement in terms of the MISE of the estimators where the boosting, bagging and bragging methods are applied, to the achievement of the standard N-W estimator with the usual cross-validation bandwidth. The abbreviations "NW", "Boost", "Bag1", "Bag2", "Bag3", "Brag1", "Brag2" and "Brag3", as displayed in the tables, will be used to refer to results obtained by bandwidth selection via the respective procedures listed in Table 5.2, in the remainder of this paragraph.

1. The effect of boosting:

   As mentioned before, the boosting investigation in SimIA is computer-time economic and a maximum of only one boosting iteration is allowed in this part of the study. The observations contained in Tables A.1 to A.12 enables the following remarks regarding the overall effect of the boosting method in the present design setting:

   - Procedure Boost reduces the MISE when compared to procedure NW for all small σ scenarios (i.e. $\sigma = 0.2$ for Models 1 and 2 and $\sigma = 0.1$ for Model 3), except for the Model 3 scenarios, where X-values were drawn from the $N(0,1)$ distribution.
   - For larger σ scenarios, procedure Boost does not significantly improve NW, except for the following scenarios:

- ► All Model 1, $X \sim N(0,1)$ scenarios.
- ► Model 2, Uniform X, $\sigma = 0.6$ scenarios.
- ► Some large n, Model 1 scenarios, for example for the $n = 100$ and $n = 200$, $\sigma = 0.6$ scenarios.

- The MISE reduction by Boost when compared to NW is the highest for small σ scenarios.
- Boost always lowers the bias when compared to NW. It is the increase in variance when compared to NW that causes increased MISE values in scenarios where Boost is not effective in terms of the MISE.
- These observations correspond with our findings in Simulation Study II.

Conclusion 1

In the light of the above remarks, all small σ scenarios, except for the Model 3, $X \sim N(0,1)$ scenarios are identified as scenarios for which more boosting iterations and bagged or bragged boosting can lead to further improvements in the sense of the minimized MISE. Some of these will be investigated in SimIB.

2. The effect of bagging and bragging:

   We now hope to find that bagging and bragging will provide better estimates in terms of the MISE than the standard N-W estimator, where the bandwidth was selected by means of cross-validation. We also hope that bagging and bragging will reduce variability in the application of the CV-based methods. However, the first part of the expectation is far from true. In fact, throughout the simulation studies it is clear that procedures Bag1, Bag2 and Bag3 and Brag1, Brag2 and Brag3 maintain MISE values in the order of the MISE associated with the standard N-W estimator, or in many cases, increase the MISE value. It is indeed true that the integrated variance decreases notably when procedures Bag1, Bag2 and Bag3 and Brag1, Brag2 and Brag3 are applied, but the squared bias component increases by suprizingly large amounts. Smaller variability therefore is achieved at a high cost of increased squared bias. The influence of the bootstrap and matters such as resampling with or without replacement will be studied in detail in a following project in this regard.

   For the purpose of comparison between NW and the bagging and bragging algorithms, we report that none of the procedures constantly delivers smaller MISE values than the standard NW estimator. On the positive side we remark that there is usually at least one

of the procedures that achieves smaller MISE than the N-W estimator. Observations from Tables A.1 to A.12 enable the following:

Conclusion 2

- From the result tables it is clear that, in most cases, at least one of the bagging and bragging methods result in lower MISE values than NW and the variability is reduced.
- Although the bias of the 'best' bagging and bragging estimators are mostly higher than the bias of NW, the variance of the improved estimators are considerably lower than the variance of NW. This corresponds to the findings of Hall and Robinson (2009), who show that the stochastic variability of cross-validation bandwidth selection in the kernel density and regression settings can be reduced significantly by bagging.
- The improvement induced by the application of the bagging and bragging algorithms is the largest in scenarios where the error variance σ is high. This is in contrast with the boosting method, which does not improve estimators when the data shows much variability. The observation that bagging and bragging are particularly effective in high error variance scenarios confirms the notion that the bagging and bragging methods are effective because it lowers the variance component of the discrepancy measure.

In the following points, the results of the various bagging and bragging methods are compared and discussed in more detail.

3. Comparison between bagging and bragging methods:

   In general, it seems that there is little to choose between the bagging and bragging methods in terms of the MISE. Note the following in this regard:

   - In general, bragging performs the same or slightly better than bagging for all $X \sim U(-2, 2)$, $\sigma = 0.2$ scenarios.
   - In contrast, bagging generally performs similar to, or better than bragging for large σ, $X \sim U(-2, 2)$ and all $X \sim N(0, 1)$ scenarios. In these cases we observe that:
     - ► Generally, Bag1 performs the same or slightly better than Brag1.
     - ► Generally, Bag2 and Bag3 perform significantly better than Brag2 and Brag3.
   - Generally, the integrated variances of Bag1, Bag2 and Bag3 are smaller than those of Brag1, Brag2 and Brag3 respectively, i.e. bagging produces estimated curves with lower variability than bragging.

- Regarding the integrated bias, we note that the bragging methods result in lower values than bagging methods for the $X \sim U(-2, 2)$, small σ scenarios. Comparison between the integrated bias of the bagging and bragging methods in other settings does not yield a clear pattern. For some cases, bagging results in lower integrated bias values than bragging and at times bragging performs better than bagging in terms of the integrated bias.

Conclusion 3

To summarize, it seems as if bagging performs at least as good or better than bragging in terms of the MISE in most cases, which can be ascribed to the lower variance component of estimators where bagging is applied. However, for very 'stable' scenarios, such as the $X \sim U(-2, 2)$, small σ scenarios, it seems as if bragging outperforms bagging due to a significantly lower bias and a similar or only slightly larger variance.

4. Comparison between the Bag1, Bag2 and Bag3 methods:

When comparing the three variations of the bagging method, we note the following:

- For Model 1, Bag1 performs better in terms of the MISE than Bag2 and Bag3 for all scenarios, except for the $X \sim N(0, 1)$, $\sigma = 1.0$, $N(0, 1)$ kernel scenario where Bag2 performs the best.
- For Model 2, Bag1 performs better in terms of the MISE than Bag2 and Bag3 in most cases, except for the $X \sim N(0, 1)$, $\sigma = 0.6$ and $\sigma = 1.0$ scenarios where Bag3 performs the best and Bag1 performs considerably worse than both Bag3 and Bag2.
- For the Model 3 we detect the following in terms of the MISE:
  - For $X \sim U(-2, 2)$, $\sigma = 0.1$ scenarios, Bag1 performs the best.
  - For $X \sim U(-2, 2)$, $\sigma = 0.3$ and $\sigma = 0.5$ scenarios there seems not to be a clear pattern of performance. In some cases Bag1 results in the lowest MISE, in other cases Bag2 performs the best and for other scenarios, Bag3 outperforms the other methods.
  - For all Model 3, $X \sim N(0, 1)$ scenarios, Bag3 performs the best and Bag1 the worst.
- In almost all cases, the bias of Bag1 is considerably smaller than the bias of Bag2 and Bag3, while the variance of Bag1 is considerably larger than the variance of Bag2 and Bag3.

- Bag1 differs from Bag2 and Bag3 in that Bag1 is an NW estimator using the full sample (with a bag based smoothing parameter), while Bag2 and Bag3 are averages of the NW estimators based on the bootstrap samples (rather than the whole sample).

  As an afterthought, it was suggested that a fourth form of bagging (say Bag4) would be to take the B individual bootstrap smoothing parameters, upscale them to full sample size, calculate B NW estimators from the full sample and then find the average of these estimators. The bootstrap in Bag4 is used only to get B possible choices of the smoothing parameter, but aggregation is done with estimators based on the full sample. This differs from Bag1 in that the regression estimators are averaged rather than the smoothing parameters, but the full sample is used for the regression estimators in both cases. A small scale investigation of this alternative shows that Bag4 is a feasible method which sometimes, but not always, improves Bag1, Bag2 and Bag3 in terms of the minimized MISE. Future studies will further explore this procedure.

Conclusion 4

It seems as if Bag1 provides the best results in all small error variance scenarios, except for the Model 3, $X \sim N(0,1)$ scenario, which was already identified as an 'unstable' scenario in Simulation Study II, where we noted that boosting does not improve N-W estimation in this scenario. For 'unstable' scenarios, Bag3 performs best in terms of the MISE, while Bag1 generally does not perform good in these cases. Furthermore, Bag1 results in estimators with lower bias, but higher variance values than Bag2 and Bag3.

5. Comparison between the Brag1, Brag2 and Brag3 methods:

   When comparing Brag1, Brag2 and Brag3, findings are similar to the findings above where Bag1, Bag2 and Bag3 were compared.

   A fourth bragging procedure (Brag4), analogous to the Bag4 procedure described above, shows to be more effective than Brag1, Brag2 and Brag3 in some, but not all, scenarios in a small scale study. Further comments on the Brag4 procedure will be reported in future publications.

Conclusion 5

Brag1 seems to provide the best results in all small error variance scenarios, except for the Model 3, $X \sim N(0,1)$ scenario. For 'unstable' scenarios, such as Model 2 and 3, $X \sim N(0,1)$ scenarios, Brag3 performs best in terms of the MISE. Furthermore, the bias

of Brag1 is considerably smaller than the bias of Brag2 and Brag3, while the variance of Brag1 is considerably larger than the variance of Brag2 and Brag3.

6. Discussion of the bagging and bragging results:

The observations in points 4 and 5 above can be verified from the bagging and bragging algorithms as it were introduced in Chapter 5. Consider the following with regards to the bagging algorithms:

- In the algorithm for Bag1, Algorithm 5.3, a bagged bandwidth is obtained by aggregating over the cross-validation bandwidths of the bootstrap samples. After rescaling, the bagged bandwidth is applied to the original Monte Carlo sample. For 'stable' scenarios, for example where σ is relatively small, each bootstrap sample will be representative of the original Monte Carlo sample from which the bootstrap sample was drawn and thus the aggregated bandwidth over all bootstrap samples is applicable on the original Monte Carlo sample.
- The algorithm for Bag3, Algorithm 5.5, uses the cross-validation bandwidth of each bootstrap sample to estimate $m(x)$ using the particular bootstrap sample. The estimates for $m(x)$ over the bootstrap samples are then aggregated. In 'unstable' cases, for example when σ is large and Model 3 is used, the bootstrap samples will vary considerably from bootstrap sample to bootstrap sample. Therefore, application of each bootstrap bandwidth to the particular bootstrap sample for which it was chosen, will be more efficient than aggregating the bootstrap bandwidths and applying the aggregated bandwidth to the original Monte Carlo sample.
- To understand why the variance components of Bag1 estimators are higher than the variance components of Bag2 and Bag3 estimators, consider the fact that the Bag2 and Bag3 algorithms obtain estimates of $m(x)$ for each bootstrap sample separately and then aggregate these estimates. In other words, aggregation is done over the bootstrap estimates to obtain final estimates $\hat{m}(x)$. In contrast, the Bag1 algorithm aggregates over the bootstrap bandwidths and uses the resulting bandwidth to determine $\hat{m}(x)$. In Bag2 and Bag3 the focus is thus on reducing the variability of $\hat{m}(x)$, whereas the focus in Bag1 is to reduce the variability of the cross-validation bandwidth used to estimate $\hat{m}(x)$. Therefore, Bag2 and Bag3 estimators will have smaller variance values then Bag1 estimators. However, accuracy is compromised and the Bag2 and Bag3 estimates will lie further from the true curve $m(x)$ resulting in higher bias values than Bag1 estimates.

The reasoning behind the observations made when Brag1, Brag2 and Brag3 are considered, are similar to the reasoning when Bag1, Bag2 and Bag3 are compared, as discussed above.

7. The influence of n, σ, the kernel and the covariate distribution:

   With regards to the sample size n, the error variance σ, the kernel and the covariate distribution we note the following:

   - The effect of the sample size:

     As n increases, the MISE, integrated bias and integrated variance values decrease. This can be expected, since large samples contain more information about the true underlying regression relationship $m(x)$ than small samples and will therefore result in estimators with less bias and variàbility. Furthermore, the improvement in terms of the MISE obtained by applying the improvement methods is generally less for large sample sizes than for small sample sizes. In other words, if boosting, bagging or bragging improved the N-W estimator for a small sample size, the improvement is less for a large sample size in the same simulation setup scenario. Once again, this is not surprising: since large samples contain more information about $m(x)$ than small samples, improvement methods will not have as great an effect on the N-W estimator as for small sample sizes.

   - The effect of the error variance:

     The MISE, integrated bias and integrated variance values increase as σ increases. This observation can be expected, since samples with larger error variances will result in worse estimators for $m(x)$, which will be reflected by higher bias and variance values. Also, whether boosting, bagging and bragging improve NW, seem to be greatly dependent on the error variance. Regarding boosting, it seems as if boosting can effectively improve NW for all small σ scenarios, except for the Model 3, $X \sim N(0,1)$ scenarios. In contrast, the improvement induced by the application of the bagging and bragging algorithms is the largest in scenarios where the error variance σ is high. These observations were discussed in the previous points.

   - The effect of the kernel:

     The $N(0,1)$ kernel result in smaller MISE, integrated squared bias and integrated variance values than the triangular kernel. This can be ascribed to the fact that the $N(0,1)$ kernel has longer tails and will therefore include more sample points in the estimation than the restricted triangular kernel. Furthermore, there seems not to be a dependence

between whether an improvement method improves the NW estimator and the kernel that is applied. In other words, if a certain improvement method improves NW in terms of the MISE for a particular simulation setup scenario using the $N(0,1)$ kernel, a similar trend is generally observed for the triangular kernel.

- The effect of the covariate distribution:

  The MISE, integrated bias and integrated variance values are generally larger when the covariate data is drawn from the $N(0,1)$ distribution than for scenarios where the data is drawn from the $U(-2,2)$ distribution. This is possibly because data from the $N(0,1)$ distribution contains less observations near the boundaries of the x-grid and therefore shows large variability and bias near the boundaries. The dependence of the improvement procedures on the covariate distribution seems unclear. Generally the improvement methods that improve NW in terms of the MISE for $U(-2,2)$, also improve NW for $N(0,1)$. However, this observation is not applicable throughout.

B. Interpretation of the observations and conclusions of SimIB (Tables A.13 to A.30, Appendix A)

We now highlight some observations made from the simulation results in SimIB. We interpret the results and draw conclusions.

Aims of the study: The aim of all the methods studied in SimIB is to improve the boosted N-W estimator. We now compare the achievement in terms of the MISE of the new estimators where the bagging and bragging methods are applied to the boosted N-W estimator, to the MISE associated with the MISE of the boosted N-W estimator with usual cross-validation smoothing parameters.

The abbreviations "Boost_1", "Boost_6", "Bag1", "Bag2", "Bag3", "Brag1", "Brag2" and "Brag3", "Bagboost1", "Bagboost2", "Bagboost3", "Bragboost1", "Bragboost2" and "Bragboost3", as displayed in the tables, will be used to refer to results obtained by bandwidth selection via the respective procedures listed in Table 5.2, in the remainder of this paragraph.

1. Comparison between Boost_6 and Boost_1:

   When comparing Boost_6 and Boost_1, we note the following:

   - In all scenarios included in this study, Boost_6 produces lower MISE-values than Boost_1. This can be expected, since only simulation setup scenarios where boosting showed potential for improving the N-W estimator in Simulation Study II and SimIA are considered in SimIB.

- Allowing for 6 boosting iterations lowers the integrated squared bias in all scenarios and a decrease or only slight increase in the integrated variance when compared to Boost_1. This observation confirms the notion of Marzio and Taylor (2008) that the bias of the boosted estimator decreases exponentially fast towards zero, while the variance increases exponentially slow towards σ^2.

2. Comparison between Boost_6 and the bagged and bragged boosting algorithms:

   When comparing Boost_6 and the Bagboost and Bragboost values, we note the following:

   - The best performing Bagboost or Bragboost algorithm for each particular simulation setup scenario (i.e. Bagboost1, Bagboost2, Bagboost3, Bragboost1, Bragboost2 or Bragboost3 – whichever results in the smallest MISE) improves Boost_6 in all considered scenarios in the sense that the MISE is reduced. Thus, Boost_6 can be effectively improved by bagging and/or bragging.
   - Almost all Bagboost and Bragboost variance values are lower than the variance of Boost_6. This observation corresponds to the notion of Hall and Robinson (2009) that bagging can reduce cross-validation variance simply, significantly and reliably. Note, however, that bagging and bragging often cause an increase in the bias when compared to the boosted N-W estimator. I.e. the variance reduction is compromised by an increased bias.

3. Comparison between the different bagged and bragged boosting algorithms:

   When comparing the different Bagboost and Bragboost values, we note the following:

   - Bagboost1 has considerably lower bias values, but higher variance values than Bagboost2 and Bagboost3. The same phenomenon is detected for the Bragboost methods. This corresponds to the effect of bagging and bragging on standard cross-validation and can be explained in the same way. See the discussion in Section 6.3.2.A.
   - When comparing the six Bagboost and Bragboost methods, it is clear that not all methods work well in all scenarios. In fact, some of the methods perform quite bad in terms of the MISE for certain scenarios (see for example the performance of Bragboost2 and Bragboost3 in Table A.23). However, it seems difficult to detect a fixed pattern in which these differences occur. Future investigations should explore these methods further to establish guidelines for the choice between the various Bagboost and Bragboost methods.

4. As mentioned before, the grids of h-values used in SimIB was chosen to be the same as the grids for SimIA. However, in the application of SimIB it became clear that these grids might not have been sufficient. The methods studied in SimIB may perform even better if larger bandwidths are allowed in the h-grids in future studies.

5. A negative aspect of the methods in SimIB, is that these methods are extremely computer-time intensive. Although these methods improve on existing methods, their feasibility comes to question when time constraints apply.

Conclusion 6

Bagboost and Bragboost show potential to improve the boosted N-W estimator. However, it is not yet clear which of the Bagboost and Bragboost algorithms should be applied in which circumstances. Future investigations should be conducted to further explore application of bagging and bragging to the boosted N-W estimator.

6. The influence of the kernel and the covariate distribution:

   For SimIB, observations regarding the influence of the kernel and covariate distribution are similar to observations made in SimIA. These observations was discussed in point 7 of the discussion of the observations and conclusions drawn from SimIA.

7. The influence of the bootstrap resample size m:

   In this study a bootstrap resample size of $m = \frac{n}{2}$ was used throughout. To explore this matter, we conducted a small-scale investigation study where we applied bootstrap resample sizes $m = \frac{2n}{3}$ and $m = \frac{n}{3}$. We found that estimators performed the best in terms of the minimized MISE when $m = \frac{2n}{3}$ is used and worst when $m = \frac{n}{3}$. This agrees with Swanepoel (1986)'s rule of thumb.

Conclusion 7

We conclude that the use of larger fractions of the original sample size in the bootstrap resamples may provide better regression estimators. However, making m too large, i.e. taking $m = n$, will result in no bagging or bragging at all, since we apply the sampling without replacement mechanism to prevent problems that may arise when ties cause the break-down of the cross-validation method. In other words, the ideal seems to increase the resample size m, but considering the mechanism of cross-validation, this can be done to a limited way. Furthermore, the use of the rescaling factor, as it is suggested by Hall

and Robinson (2009), where a bandwidth chosen by means of a bootstrap sample of size m should be rescaled before it is applied to the original sample of size n, needs careful consideration. It is a drastic application which may influence the results.

Rescaling is necessary in m-out-of-n situations. If plug-in methods were used instead of cross-validation methods, bootstrapping with replacement can be applied for all the above methods for $m = n$ and $m < n$. For $m = n$ rescaling is not needed. This work is in progress and will be reported in another publication.

We conclude that applying the bootstrap to the cross-validation method of bandwidth selection may not be effective. In future studies the bootstrap will be applied to plug-in bandwidth selection where sampling with replacement with $m = n$ can also be explored.

6.4 Observations and conclusions from Simulation Study II

As mentioned before, Simulation Study II is a preliminary study which focuses specifically on boosting, where smoothing parameter selection is deferred and "optimal" smoothing parameters in terms of the approximated MISE are selected to explore the potential of the boosting method. In this section we explain and interpret the results of Simulation Study II. The result tables of Simulation Study II are presented in Appendix B.

In Section 6.4.1 the layout of the result tables of Simulation Study II are explained, to guide the reader when interpreting the tables. In Section 6.4.2 the results displayed in the tables are presented and compared to those of Marzio and Taylor (2008).

It should be kept in mind that the aim of Simulation Study II is to:

1. Find the h-value responsible for minimizing the MISE in the ordinary N-W estimator. In the remainder of this paragraph we shall refer to this scenario as SimIIA.

2. Find the pair of values (h, T) responsible for minimizing the MISE in the boosted N-W estimator. This scenario will be referred to as SimIIB.

6.4.1 Guidelines for reading Tables B.1 to B.9 of results

The results of Simulation Study II are reported in Tables B.1 to Table B.9, which appear in Appendix B. The following remarks can be used as guidelines when consulting the tables:

- Each of the 9 tables displays the results of a particular combination of sample size and model choice. The specific combination is indicated in the caption of the table.
- In all tables, each row entry is the result of a Monte Carlo study on its own and is not connected to the next row entry. The various row entries represent the different simulation setup scenarios included in this study.
- In the first three columns of all tables, the value of σ, the underlying distribution of the covariate data and the kernel are presented respectively for the simulation study reported in the particular row.
- The next four columns respectively display the minimum MISE value, its integrated squared bias component, its integrated variance component and the h-value that resulted in the reported minimum MISE value of the standard N-W estimator. These are the results of SimIIA.
- The next five columns respectively display the minimum MISE value, its integrated squared bias component, its integrated variance component and the (h, T) combination that resulted in the reported minimum MISE value of the boosted N-W estimator. These are the results of SimIIB.
- The last column present the percentage gain that results from applying the boosting algorithm, i.e. the percentage improvement of the best boosted N-W estimate over the best N-W estimate in terms of the MISE.

6.4.2 Interpretation of the observations and conclusions in Simulation Study II

We now highlight some observations made from the simulation results in Simulation Study II. We interpret the results and draw conclusions when appropriate.

a) Effect of boosting on the MISE

From the tables in Appendix B, it is clear that for the cases where the boosting procedure was repeated, the standard N-W estimation of the underlying regression relationship was improved. The % gain from boosting, however, differs for the various simulation setup scenarios. In fact, for some simulation setup scenarios the optimal number of boosting iterations is zero (i.e. boosting does not improve standard N-W estimation), which implies

that the optimal boosted N-W estimator is simply the standard N-W estimator and the percentage gain by boosting is therefore zero. In such setups it seems that the N-W estimator is not a weak learner. In the following discussion, the roles of the various aspects of the simulation setup on the effect of boosting on the MISE are discussed.

1. Effect of σ:

   For both SimIIA and SimIIB, increasing values of σ result in increasing MISE-values. This is to be expected, since the underlying regression relationship $m(x)$ will be more difficult to estimate from data that were generated to deviate from $m(x)$ a lot by means of a large error variance, than from data that follows $m(x)$ closely, i.e. data constructed with a small error variance.

   Furthermore, when the % gain in the last column of the tables is considered, it is clear that increasing values of σ result in decreased improvement by boosting, i.e. the % gain by boosting in terms of the MISE is less for large σ scenarios. This trend is particularly evident for Models 2 and 3. In fact, for a lot of the large σ scenarios, the optimal number of boosting iterations is zero, which means that the best boosted N-W estimator is simply the standard N-W estimator on which no boosting was applied.

   Conclusion 8

   It seems that boosting is not effective when the variability of the observations is large, particularly for Models 2 and 3.

   To understand this phenomenon, consider the following: for large error variance scenarios, boosted versions of the estimator tend to follow the large random errors closely, resulting in worse estimators for the mean regression curve $m(x)$. In other words, boosting iterations may indeed reduce the squared bias component of the MISE because it follows the datapoints more closely, but the variance component increases so much in these scenarios that it results in an increased MISE value. Therefore the optimal number of boosting iterations in these scenarios turn out to be zero, as indicated in the tables.

2. Effect of kernel choice:

   For both SimIIA and SimIIB, the effect of the kernel choice with regard to minimizing the MISE is influenced by the underlying distribution of the covariate data. Note the following examples:

- For $X \sim U(-2, 2)$, the $N(0, 1)$ kernel performs slightly better than the triangular kernel throughout.
- For $X \sim N(0, 1)$, the $N(0, 1)$ kernel performs considerably better than the triangular kernel for Models 1 and 2. There is a smaller difference between the two kernels for Model 3.
- Throughout, the optimal bandwidths are smaller for the $N(0, 1)$ kernel than the triangular kernel.
- Optimal values of T seem to be somewhat larger for the $N(0, 1)$ kernel than the triangular kernel throughout, which imply that the boosting algorithm continues longer in order to minimize the MISE.

Conclusion 9

Apart from few exceptions, we conclude that in our study the $N(0, 1)$ kernel performs (in most cases) at least as good as the triangular kernel and often better with regard to the minimized MISE.

To understand this phenomenon, consider the influence range of the two kernels. The use of the $N(0, 1)$ kernel in the N-W estimator implies that more observation points are given some weight (due to the longer tails of the $N(0, 1)$ distribution), albeit sometimes small. The triangular kernel on the other hand gives zero weight to observation points far from the point x for which $\hat{m}(x)$ needs to be determined. This seems to have a decrementing effect on the smooth. The positive effect of the unbounded $N(0, 1)$ kernel is increased when the underlying covariate data are not uniformly distributed between two fixed points (i.e. when X is not $U(-2, 2)$ distributed), but are unevenly distributed with most observation points near the center and less observation points at the boundaries (i.e. X is $N(0, 1)$ distributed).

At this point it is important to mention that procedures using the unbounded $N(0, 1)$ kernel are computationally more intensive and take a lot longer than procedures based on the bounded triangular kernel. If time constraints are a matter of concern, the triangular kernel might be preferred, in spite of its slightly worse performance with regard to the minimized MISE.

3. Effect of the underlying distribution of the predictor variable:

   From the discussion above it is evident that the underlying distribution of the covariate data influences the MISE values for both SimIIA and SimIIB. Simulations where $X \sim U(-2, 2)$ produce smaller MISE values than simulations where $X \sim N(0, 1)$. This effect is seen for all models, but is most evident for Model 1 and least evident for Model 3. This phenomenon can be ascribed to an increased boundary effect when data are drawn from the unevenly distributed $N(0, 1)$ distribution. More specifically, when X is drawn from the $N(0, 1)$ distribution, less observations are drawn for values near the boundaries of the x-grid (-3 and $+3$ for $X \sim N(0, 1)$) than when X is drawn from the $U(-2, 2)$ distribution (where the x-grid ranges between -2 and $+2$), which increases the boundary effect explained in Section 1.8 for $N(0, 1)$ data. This could result in larger MISE values.

   The distribution from which the X-data are drawn also seems to influence whether boosting has a reducing effect on the MISE or not. This effect seems to be influenced by the choice of model. The few cases where boosting were repeated (i.e. $T > 0$ was chosen), contradictory behaviour for models 1-3 became evident. For Model 1, the percentage gain by boosting is more when $X \sim N(0, 1)$ than when $X \sim U(-2, 2)$. For Models 2 and 3, the effect is reversed: the percentage gain by boosting is less when $X \sim N(0, 1)$ than when $X \sim U(-2, 2)$. This observation accentuates the complex relationship between the various aspects of the data and the effect of boosting.

   Conclusion 10

   Model 1 is for most part a straight line and the N-W estimator is claimed to be a bad learner if a straight line is to be estimated. Therefore boosting "works" better here (the N-W estimator can be improved). However, the N-W estimator seems to be too strong a learner for the other models to benefit much from boosting. In other words, much of the information about m that is contained in the datasets of these models, is already reflected by the standard N-W estimator. Not much can be derived from models 2 and 3 in this regard.

4. Effect of the sample size:

   For both SimIIA and SimIIB and for all three models, the MISE decreases significantly as n increases. This is not surprising, since the underlying regression relationship $m(x)$ will be more evident from large samples carrying a lot of information, than smaller samples, which are used to estimate $m(x)$. The MISE, which gives an indication of the discrepancy

between the estimated curve and the true curve, will therefore decrease as n increases. Note that the effect is less clear for Model 3 than for the other two models.

The effect of sample size on whether boosting improves the N-W estimator or not, is influenced by the value of σ and the model of interest. For small values of σ, boosting seems to have an increasingly positive effect (larger % gain) as the sample size increases. However, for larger values of σ, the percentage gain by boosting is less for large samples than for small samples. These observations could be ascribed to the fact that large error variances distort the underlying regression relationship and larger samples seem to amplify the distortion.

Conclusion 11

In general, larger sample sizes will give a better indication of the underlying relationship between the X and Y data, but only if the error variances are small. For such instances, the boosted estimator will improve the N-W estimator, although the improvement effects are less for Model 3.

5. Effect of the model choice:

With regard to the effect of the shape of the model on the improvement imposed by boosting, we note the following:

- Boosting has a positive effect for most of the scenarios where Model 1 is used, although not to the same extent for all scenarios. This is expected, since the N-W estimator is indeed a weak learner for this curve which is mostly a straight line. The difficulty with which the N-W estimator estimates straight lines and the potential of boosting to improve the N-W estimation of such models, were discussed in Section 6.2.
- For Model 2, improving effects of boosting are limited to all $\sigma = 0.2$ scenarios and (to a small extent) to $\sigma = 0.6$ scenarios with $X \sim U(-2, 2)$.
- Boosting almost never improves on the N-W estimator for Model 3 scenarios, except for the (stable) $\sigma = 0.1$, $X \sim U(-2, 2)$ cases.
- When the triangular kernel is used, the % gain from boosting is considerably more for Model 1 than Model 2.
- When the $N(0, 1)$ kernel is used, the % gain from boosting is, in most cases, slightly more for Model 2 than for Model 1.
- Throughout, the % gain from boosting is more for Models 1 and 2 than for Model 3.

When considering the remarks above, one may be tempted to naively conclude that boosting generally improves more on the standard N-W estimator when the underlying regression relationship is almost a straight line, as is the case in Model 1 (i.e. a simple regression relationship), than when the underlying regression relationship is of curvy nature, as is the case in Model 2, or shows large, isolated peaks, as it is in Model 3 (i.e. more complex regression relationships). This is, however, an over simplified view.

To see this, compare the results of the present study to those of Marzio and Taylor (2008). They report the results of a similar study (however, without the correction factor of 0.001 mentioned earlier, making direct comparison of the MISE-values invalid) of the $X \sim U(-2,2)$, $N(0,1)$ kernel scenario. In contrast to our findings listed above, their results show that the % gain from boosting is least for Model 1, while N-W estimation of Models 2 and 3 gain more from boosting. Furthermore, Models 2 and 3 show an almost equal % gain from boosting. These differences are the direct consequence of different choices for the error variance σ used to construct the data. In their study, $\sigma = 0.4$ is used for Model 1, $\sigma = 0.3$ for Model 2 and $\sigma = 0.1$ for Model 3. Clearly, the choice of σ for each model influences the gain from boosting.

In the light of the above, it seems that the effect of boosting is not as much dependent on the model used when considered on its own, as on the combination of the model and the value of σ used to construct the response data. Note the following:

- In Model 1, where $m(x)$ has only a single relatively small bump, N-W estimation by means of data with large error variances will still be improved on by boosting. This is because even large error variances will not distort the data from showing the relatively simple underlying regression trend.
- For Model 2, where $m(x)$ is more curvy, smaller error variances should be used for boosting to be effective. When the error variance gets large, the data no longer reflect the complexity of $m(x)$.
- For Model 3, where $m(x)$ is zero except for two isolated, large peaks, only very small values of σ will result in data that shows the trend of the data.

Conclusion 12

To summarize: it seems that boosting is effective when the standard N-W estimator is a weak learner, i.e. a relatively weak estimator of the model at hand, given that the error

variance is not too large relative to the complexity of the model. Also, it seems that the effect of boosting is not as much dependent on the model used when considered on its own, as on the combination of the model and the value of σ used to construct the response data.

b) Further observations regarding the boosting process

Further observations regarding the boosting process are now discussed.

1. The effect of boosting on the bias and variance:

   Boosting reduces the integrated squared bias in all cases where boosting has an improving effect (i.e. $T_{opt} > 0$ is chosen), except for the case where Model 1 is used, covariate data are drawn from the $N(0,1)$ distribution, the $N(0,1)$ kernel is used and σ is taken as 1 ($n = 50, 100$ and 200). In this case the % loss in integrated squared bias is relatively small. We conclude that, generally, boosting has an improving effect on the bias component of the MISE.

   Boosting reduces the integrated variance in most cases where boosting has an improving effect (i.e. $T_{opt} > 0$ is chosen). For the cases where boosting causes an increase in the variance component of the MISE, the increase is small: the % loss in integrated variance is less than 12% for all cases, except for a 28.76% increase for the Model 1, $n = 200$, $\sigma = 1$, $X \sim N(0,1)$, triangular kernel scenario.

   We can say that whenever boosting reduces the MISE, it causes a decrease, or only slight increase in the variance component of the MISE.

   Note that, when T is allowed to be larger than T_{opt} (i.e. the stopping value of the number of boosting iterations resulting in a minimum MISE), the bias component of the MISE decreases quite fast, while the variance component of the MISE increases slowly. This is due to the fact that the estimator becomes increasingly complex as T increases and tends to closely reproduce the data, which results in overfitting after too many boosting iterations.

   Conclusion 13

   These observations confirm the main result of Marzio and Taylor (2008) that the bias of the boosted N-W estimator decreases quite fast towards zero, while the variance increases slow towards σ^2.

2. Optimal values of h and T:

   The tables in Appendix B suggest that, whenever $T_{opt} > 0$ is chosen, the optimal boosting bandwidth is larger than the optimal bandwidth for the standard N-W estimator. Marzio and Taylor (2008, p. 2492) confirm this observation when they remark that the bandwidth needs to increase with the number of boosting iterations. However, they show that the boosted estimator is less sensitive to bandwidth choice. After many boosting iterations, the combinations of (h, T) for which boosting works, increases. They conclude that the potential of reducing the need of an accurate bandwidth selection and stopping rule emerges.

6.5 Discussion of the graphs

In Appendix C a limited number of graphs are presented to graphically illustrate general aspects of the simulation studies, such as the bias and variance behaviour of the various methods. The presented graphs are now briefly explained.

1. In Figures C.1 and C.2 a single realization of a Monte Carlo sample for each of the data generation schemes in this study is illustrated where $n = 50$. This is in other words examples of the samples generated in step 1 of the main algorithm, Algorithm 5.1. Samples like these are used to estimate $m(x)$ in step 2 of the main algorithm. Models 1, 2 and 3 are presented by the solid lines in Figures C.1 and C.2.

2. The figures in the remainder of Appendix C graphically illustrate some of the results of SimIB.

   In particular, the $X \sim N(0, 1)$, triangular kernel, $\sigma = 0.2$ scenario is considered for $n = 50$, $n = 100$ and $n = 200$ respectively. This specific choice of scenario is arbitrary, any other scenario could have been chosen for this illustration.

   Note that the notation "Boost" is used in the graphs to illustrate "Boost_6", i.e. the boosting procedure referenced in Table 5.2 where 6 boosting iterations are allowed. For clarity we shall refer to "Boost" instead of "Boost_6" in this short discussion.

   The graphs are presented as follows:

   - Figures C.3 to C.7 illustrate the results for the $n = 50$ case. These are graphical representations of the results displayed in Appendix A, Table A.23.

- Figures C.8 to C.12 illustrate the results for the $n = 100$ case. These are graphical representations of the results displayed in Appendix A, Table A.24.
- Figures C.13 to C.17 illustrate the results for the $n = 200$ case. These are graphical representations of the results displayed in Appendix A, Table A.25.

For each of the above mentioned scenarios, two types of graphs are drawn. Firstly, each procedure is illustrated in a graph where the estimated curves for each of the 200 Monte Carlo samples are drawn on one figure. These graphs give insight in the variance involved for the particular method. Secondly, the procedures are illustrated in a graph where the average estimator over the 200 Monte Carlo samples is presented for various procedures. These graphs show the effect of the various methods on the bias.

The graphs in Figures C.3 to C.17 confirm the conclusions drawn from Simulation Study I in Section 6.3.2. More specifically, the following conclusions are particularly clear from the graphs:

- The boosting method produces curves that are closer to the underlying regression curve $m(x)$ than NW, i.e. the bias is reduced. The variability of the estimator obtain via the Boost procedure is not significantly more than the variability of the estimator obtained via the NW procedure.
- The Bag1, Bag2 and Bag3 and Brag1, Brag2 and Brag3 procedures produce smoother curves, i.e. curves with less variability, than the NW procedure. However, the curves obtained via the improvement methods often lie further from m than curves obtained via NW, i.e. the bias is increased.
- The Bagboost1, Bagboost2 and Bagboost3 and Bragboost1, Bragboost2 and Bragboost3 procedures produce smoother curves, i.e. curves with less variability, than the Boost procedure. However, the curves obtained via the improvement methods often lie further from m than curves obtained via Boost, i.e. the bias is slightly increased.
- In this scenario Bag1, Brag1, Bagboost1 and Bragboost1 produces estimates that are closer to $m(x)$, but more variable than Bag2 and Bag3, Brag 2 and Brag3, Bagboost2 and Bagboost3 and Bragboost2 and Bragboost3 respectively.
- As n increases, the estimates lie closer to $m(x)$ (i.e. the integrated bias decreases) and the variability becomes less (i.e. the integrated variance decreases).

6.6 Overall conclusions

1. Simulation Study II serves as a preliminary study to explore the performance of boosting in the simulation setup scenarios that are investigated in the main study, Simulation Study I. The findings in the two studies correspond.

2. Boosting improves the standard N-W estimator whenever the error variance is not too large. The bias of the boosted estimator in these scenarios is considerably lower than the bias of the standard N-W estimator, while the variance does not increase too much. Marzio and Taylor (2008) report similar results.

3. Bagging and bragging do not "improve" the estimation of $m(x)$ in the sense that bagged or bragged estimators are closer to the underlying regression curve than the standard or boosted N-W estimator on which the bagging or bragging methods are applied. In fact, the bias is almost always larger for the bagged and bragged estimators than for the standard or boosted N-W estimator on which it is applied.

4. Bagging is effective in reducing the variability of the cross-validation bandwidths and the resulting regression estimates, as was suggested by Hall and Robinson (2009). The findings for bragging are similar to the findings for bagging.

5. The limited number of graphs that are available are very disappointing. From these graphs it is clear that bagging and bragging methods reduce the variability of the N-W and boosted N-W estimators, but the increase in bias is large. Studies are conducted to draw graphs for other simulation setup scenarios, such as for large n and large σ settings and will be included in future publications.

   However, it should be noted that preliminary studies with MC small revealed that for $n = 200$ the NW as well as other methods estimate m more successfully, but the same tendencies prevail as was found for $n = 50$, with regard to bias and variance related issues.

6. Application of the bootstrap to the cross-validation method of bandwidth selection may not be effective. The fact that sampling must be done without replacement to prevent ties from causing the breakdown of the cross-validation method and the rescaling that is necessary for m-out-of-n without replacement sampling, may influence the results. In future studies the bootstrap will be applied to plug-in bandwidth selection where sampling with replacement with $m = n$ can also be explored.

7. Models 1, 2 and 3, considered in this study, are difficult estimation problems due to large signal-to-noise ratios. After all, the N-W estimator seems to perform quite well for these problems.

   When the error variance is reasonably small, boosting can improve on N-W estimation in the sense of a great reduction in the bias and only a small increase in the variance, resulting in lower MISE values. However, bagging and bragging result in curves that are, although more smooth, further from the regression curves than the N-W or boosted N-W estimators on which it was applied and therefore have larger MISE values.

   For large error variance scenarios, boosting does not improve the standard N-W estimator in the sense of the MISE. Bagging and bragging reduces the variability of the N-W and boosted N-W estimators in these scenarios enough to compensate for the increased bias, resulting in lower MISE values.

8. The sample sizes in this study were taken to be $n = 50$, $n = 100$ and $n = 200$ due to time constraints. Fan and Gijbels (1992) use samples of sizes $n = 200$ and $n = 400$ to obtain local linear kernel regression estimators to the models considered in this study that are a lot closer to the models than the estimators in our studies for $n = 50$ and $n = 100$ (see Fan and Gijbels (1992, p. 2020-2022) for graphs when larger sample sizes are used). Larger sample sizes should be applied in future studies.

Chapter 7

Applications to real data

In this chapter we graphically present the behaviour of the regression smoothers obtained via the various procedures presented in Chapter 5, Table 5.2, when applied to three real life datasets. Note that the second and third datasets are examples of time series data and therefore independent and homoscedastic error assumptions are not valid. In this sense, the procedures should be adapted for examples like these. However, we apply the procedures as it were described in previous chapters, without any alterations. Future research should be conducted to explore the applicability of the methods to time series data.

a) Application to the Production Time dataset

The first dataset is the "Production Time" dataset, taken from Kutner et al. (2005, p. 151). In a manufacturing study, the production times for 111 production runs were obtained. The data consists of the production lot size (x) and the corresponding production time in hours (y) for each run.

A scatterplot of the data is presented in Figure 7.1.

We determine the regression estimates via the various methods for a grid of 100 x-values ranging between the minimum predictor value in the dataset (0) and the maximum predictor value in the dataset (33). We use the triangular kernel to save computer time. The cross-validation method is used to select the smoothing parameters. The h-grid ranges between 0.5 and 10 with increments of 0.5 for all methods, a maximum of 6 boosting iterations is allowed in the boosting and combining methods and 50 bootstrap samples are drawn for the bagging, bragging and combination methods.

The resulting estimates are presented in Figure 7.4.

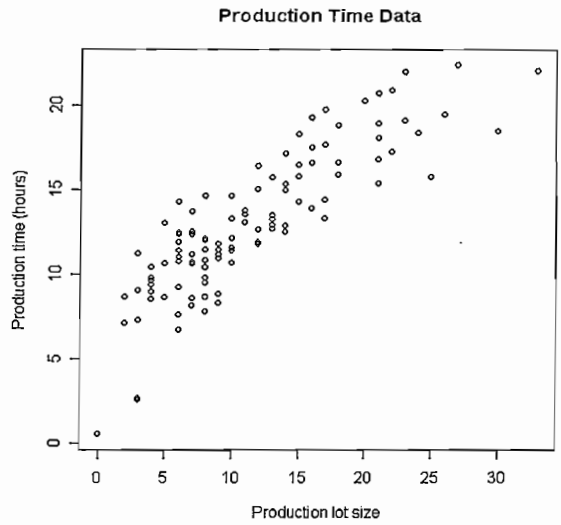


Figure 7.1: Production time dataset.

b) Application to the ENSO dataset

Consider the dataset "ENSO", taken from Kahaner, Moler and Nash (1989, p. 441-445). The data represent the monthly averaged atmospheric pressure differences between Easter Island and Darwin, Australia. The dataset consists of 168 observations of the response variable (y), atmospheric pressure, and the corresponding predictor variable (x), time.

A scatterplot of the data is presented in Figure 7.2. (The data is available on the internet at the url http://www.itl.nist.gov/div898/strd/nls/data/LINKS/DATA/ENSO.dat.)

We determine the regression estimates via the various methods for a grid of 100 x-values ranging between the minimum predictor value in the dataset (1) and the maximum predictor value in the dataset (168). We use the triangular kernel to save computer time. The cross-validation method is used to select the smoothing parameters. The h-grid ranges between 0.5 and 20 with increments of 0.5 for all methods, a maximum of 6 boosting iterations is allowed in the boosting and combining methods and 50 bootstrap samples are drawn for the bagging, bragging and combination methods. The resulting estimates are presented in Figure 7.5.

c) Application to the Motorcycle dataset

The third dataset is the "Motorcycle" dataset, taken from Silverman (1985, p. 8). The data consists of accelerometer readings taken through time in an experiment on the efficiency of

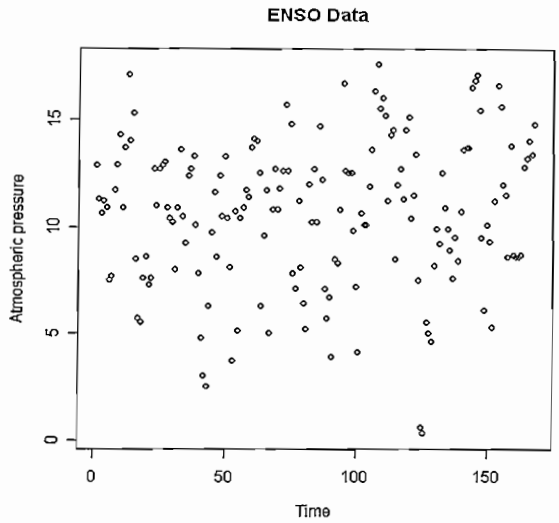


Figure 7.2: ENSO dataset.

crash helmets. Härdle (1990, Chapter 3) also considers this dataset. The dataset consists of 94 observations of the response variable (y), acceleration, and the corresponding predictor variable (x), time.

A scatterplot of the data is presented in Figure 7.3. (The data is available on the internet at the url http://www.stat.cmu.edu/ larry/all-of-statistics/=data/motor.dat.)

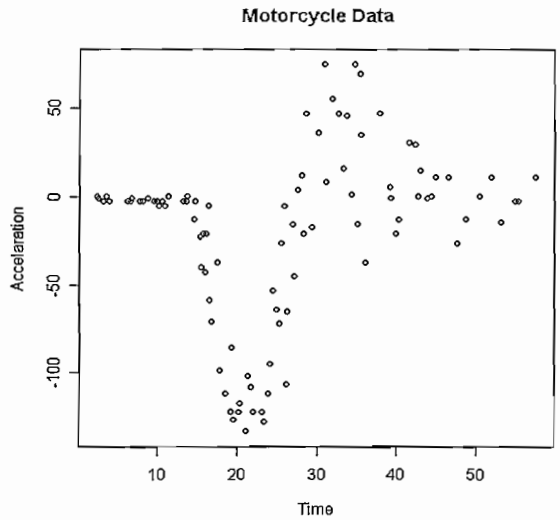


Figure 7.3: Motorcycle dataset.

We determine the regression estimates via the various methods for a grid of 100 equally spaced x-values in the range [0,60], which contains the domain of the dataset. We use the triangular kernel. The cross-validation method is used to select the smoothing parameters. The h-grid ranges between 0.5 and 20 with increments of 0.5 for all methods, a maximum of 6 boosting iterations is allowed in the boosting and combining methods and 50 bootstrap samples are drawn for the bagging, bragging and combination methods. The resulting estimates are presented in Figure 7.6.

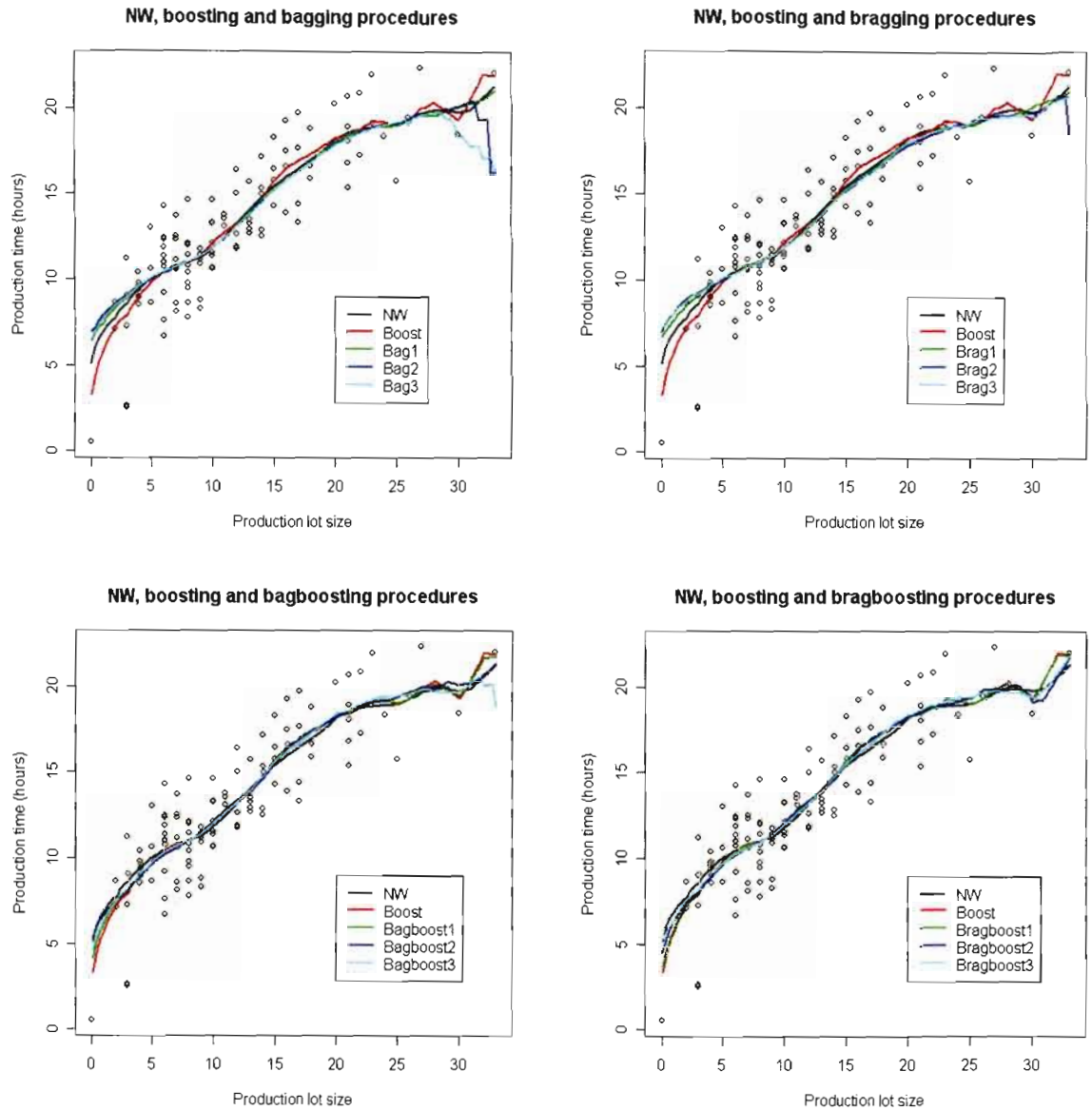


Figure 7.4: Smooth estimates of the Production Time dataset obtained via the various methods

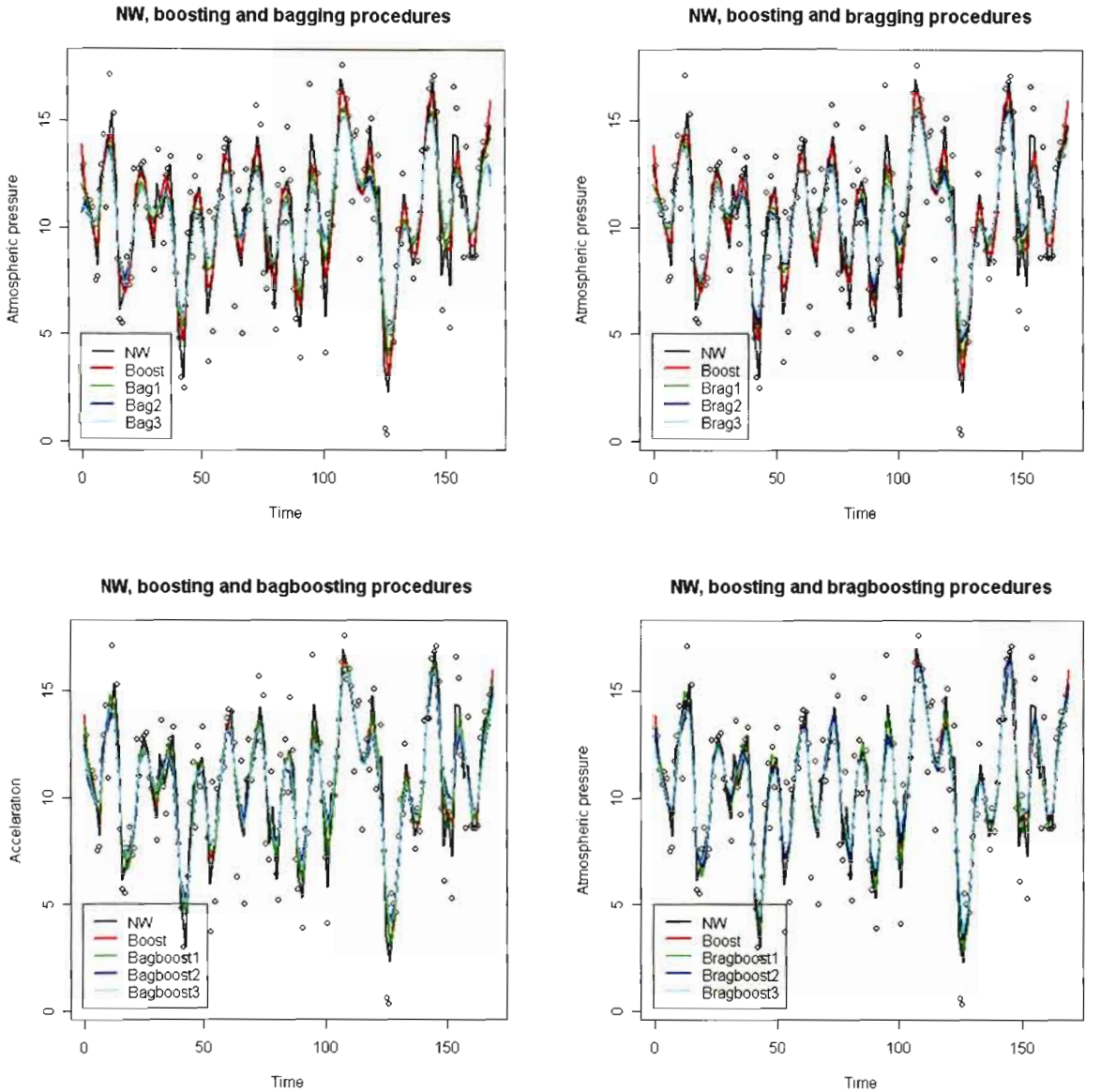


Figure 7.5: Smooth estimates of the ENSO dataset obtained via the various methods

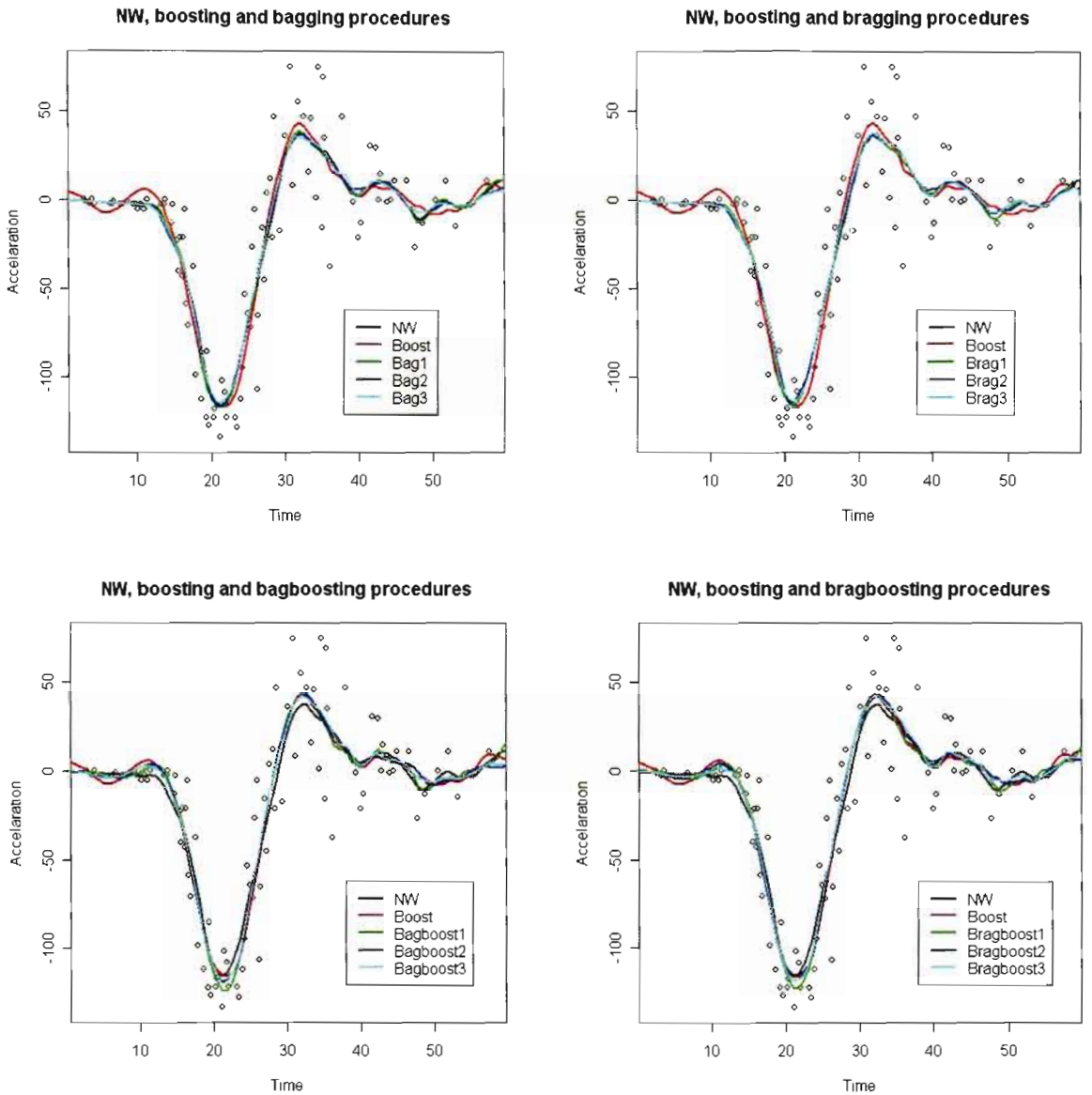


Figure 7.6: Smooth estimates of the Motorcycle dataset obtained via the various methods

Appendix A

Results of Simulation Study I

	$n = 50$			$n = 100$			$n = 200$		
	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$
$\sigma = 0.2$									
	$h = 0.05(0.01)0.40$			$h = 0.02(0.01)0.20$			$h = 0.02(0.01)0.15$		
NW	0.132777	0.020800	0.111977	0.049858	0.006607	0.043251	0.025184	0.003817	0.021367
Boost	0.107578	0.013564	0.094013	0.040775	0.004123	0.036652	0.020134	0.002289	0.017846
Bag1	0.123854	0.039302	0.084551	0.049238	0.009740	0.039497	0.024401	0.004341	0.020060
Bag2	0.163837	0.079290	0.084547	0.056758	0.019386	0.037372	0.026285	0.007807	0.018478
Bag3	0.168479	0.083822	0.084657	0.057567	0.020808	0.036759	0.026429	0.008128	0.018301
Brag1	0.120232	0.032620	0.087613	0.048846	0.008895	0.039950	0.024473	0.004327	0.020145
Brag2	0.144020	0.050111	0.093908	0.053299	0.014378	0.038921	0.025924	0.007029	0.018895
Brag3	0.151318	0.061679	0.089639	0.054113	0.016705	0.037408	0.026047	0.007432	0.018615
$\sigma = 0.6$									
	$h = 0.05(0.01)0.80$			$h = 0.05(0.01)0.80$			$h = 0.03(0.01)0.20^*$		
NW	0.443130	0.084020	0.359110	0.216386	0.040038	0.176348	0.118654	0.022721	0.095932
Boost	0.456190	0.074963	0.381228	0.214038	0.030025	0.184013	0.113647	0.016526	0.097120
Bag1	0.415492	0.154178	0.261314	0.217547	0.065380	0.152167	0.113931	0.028870	0.085061
Bag2	0.455719	0.221776	0.233943	0.238108	0.100654	0.137454	0.122242	0.045310	0.076932
Bag3	0.427257	0.195763	0.231494	0.231780	0.094180	0.137601	0.122912	0.045380	0.077532
Brag1	0.411153	0.124380	0.286774	0.214972	0.056528	0.158444	0.114736	0.029200	0.085537
Brag2	0.431879	0.164653	0.267226	0.229143	0.082290	0.146853	0.123260	0.043884	0.079376
Brag3	0.443736	0.192903	0.250833	0.234279	0.092791	0.141488	0.123826	0.045447	0.078379
$\sigma = 1.0$									
	$h = 0.03(0.01)1.00$			$h = 0.01(0.01)0.70$			$h = 0.10^*(0.01)0.30^*$		
NW	0.907643	0.170686	0.736957	0.472366	0.095861	0.376505	0.247677	0.055168	0.192509
Boost	0.955632	0.152618	0.803014	0.491739	0.087889	0.403850	0.484361	0.045605	0.205698
Bag1	0.769534	0.309369	0.460165	0.459514	0.184900	0.274614	0.244409	0.079970	0.164438
Bag2	0.818385	0.408293	0.410092	0.498784	0.249555	0.249230	0.265484	0.116112	0.149371
Bag3	0.775188	0.352248	0.422941	0.470699	0.211228	0.259471	0.262540	0.110757	0.151783
Brag1	0.790666	0.267925	0.522742	0.460822	0.155034	0.305788	0.247109	0.074911	0.172198
Brag2	0.820910	0.340164	0.480745	0.490588	0.204950	0.285638	0.267226	0.106665	0.160562
Brag3	0.815115	0.383504	0.431611	0.488560	0.226816	0.261744	0.267712	0.114119	0.153594

Table A.1: Model 1, $X \sim U(-2, 2)$, $N(0, 1)$ kernel

	$n = 50$			$n = 100$			$n = 200$		
	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$
$\sigma = 0.2$									
	$h = 0.05(0.01)1.00$			$h = 0.05(0.01)0.60$			$h = 0.05(0.01)0.40$		
NW	0.180075	0.028676	0.151399	0.063326	0.007418	0.055908	0.025603	0.003611	0.021992
Boost	0.138999	0.016251	0.122749	0.045241	0.004011	0.041230	0.020747	0.002070	0.018677
Bag1	0.150128	0.061837	0.088291	0.053311	0.013355	0.039955	0.024538	0.004476	0.020062
Bag2	0.207548	0.113063	0.094485	0.066431	0.026129	0.040303	0.026760	0.008034	0.018726
Bag3	0.207644	0.116629	0.091015	0.067456	0.028382	0.039075	0.026931	0.008577	0.018353
Brag1	0.144149	0.053383	0.090766	0.051930	0.011913	0.040017	0.024573	0.004261	0.020313
Brag2	0.178992	0.079628	0.099365	0.058111	0.019579	0.038531	0.025914	0.006916	0.018998
Brag3	0.179939	0.087388	0.092551	0.059765	0.022126	0.037639	0.025835	0.007580	0.018255
$\sigma = 0.6$									
	$h = 0.05(0.01)1.50$			$h = 0.15(0.01)1.30$			$h = 0.10(0.01)0.70$		
NW	0.468975	0.085373	0.383602	0.220009	0.036987	0.183022	0.119450	0.021307	0.098143
Boost	0.477782	0.071470	0.406312	0.221565	0.028154	0.193411	0.115672	0.015603	0.100069
Bag1	0.422404	0.160246	0.262158	0.217734	0.062884	0.154850	0.113498	0.027497	0.086002
Bag2	0.469508	0.233539	0.235969	0.241111	0.098234	0.142877	0.120646	0.043783	0.076863
Bag3	0.439193	0.207937	0.231256	0.233376	0.091615	0.141761	0.121132	0.043893	0.077239
Brag1	0.415285	0.126117	0.289167	0.215345	0.051912	0.163432	0.114326	0.026641	0.087685
Brag2	0.445125	0.171764	0.273361	0.232372	0.077963	0.154409	0.120452	0.040793	0.079659
Brag3	0.448143	0.198309	0.249834	0.236417	0.089888	0.146529	0.121062	0.043640	0.077422
$\sigma = 1.0$									
	$h = 0.05(0.01)2.00$			$h = 0.01(0.01)2.00$			$h = 0.08(0.01)0.80$		
NW	0.953917	0.161950	0.791967	0.489461	0.088588	0.400873	0.257892	0.050212	0.207680
Boost	0.988363	0.151477	0.836886	0.508738	0.078696	0.430042	0.257558	0.043375	0.214183
Bag1	0.780481	0.292482	0.487999	0.467629	0.184476	0.283153	0.247150	0.077113	0.170037
Bag2	0.830583	0.390965	0.439618	0.510167	0.251094	0.259073	0.266431	0.112863	0.153568
Bag3	0.794789	0.347998	0.446792	0.474964	0.204949	0.270015	0.262867	0.106771	0.156097
Brag1	0.804266	0.239109	0.565157	0.468524	0.143122	0.325402	0.250450	0.071273	0.179177
Brag2	0.836214	0.313059	0.523156	0.496313	0.193693	0.302620	0.266015	0.102449	0.163566
Brag3	0.827512	0.370027	0.457486	0.490793	0.220330	0.270463	0.265978	0.110920	0.155058

Table A.2: Model 1, $X \sim U(-2, 2)$, Triangular kernel

	$n = 50$			$n = 100$			$n = 200$		
	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$
$\sigma = 0.2$									
	$h = 0.05(0.01)0.40$			$h = 0.05(0.01)0.50$			$h = 0.05^*(0.01)0.25^*$		
NW	5.060624	2.662221	2.398403	3.615321	1.642151	1.973169	2.360403	0.828224	1.532179
Boost	4.405221	2.167701	2.237519	2.762688	1.108973	1.653716	1.453530	0.382455	1.071075
Bag1	4.697235	2.471375	2.225860	2.973445	1.273702	1.699743	1.827032	0.576652	1.250380
Bag2	5.462628	4.285872	1.176756	3.529311	2.519938	1.009374	2.220968	1.419661	0.801306
Bag3	5.447989	4.497159	0.950830	3.618433	2.790213	0.828220	2.268203	1.593609	0.674594
Brag1	4.892580	2.594423	2.298158	3.297780	1.462128	1.835652	2.074988	0.680525	1.394463
Brag2	5.960503	3.824263	2.136240	4.037371	2.264920	1.772450	2.688799	1.175715	1.513084
Brag3	6.165079	4.212036	1.953043	4.087809	2.459448	1.628361	2.595271	1.208004	1.387267
$\sigma = 0.6$									
	$h = 0.01(0.01)0.70$			$h = 0.01(0.01)0.40$			$h = 0.07^*(0.01)0.25^*$		
NW	4.719423	2.072441	2.646981	3.266717	1.189551	2.077167	1.891516	0.422551	1.468965
Boost	4.233363	1.567385	2.665977	2.729898	0.776387	1.953510	1.385859	0.174249	1.211610
Bag1	3.873085	1.599737	2.273348	2.769643	0.947800	1.821843	1.669213	0.341925	1.327288
Bag2	4.309304	3.034771	1.274533	3.078732	2.003673	1.075059	1.804149	0.960119	0.844030
Bag3	4.491509	3.451824	1.039685	3.100239	2.162531	0.937708	1.825581	1.059998	0.765583
Brag1	4.122432	1.746412	2.376020	2.925759	1.013314	1.912445	1.770606	0.369134	1.401472
Brag2	4.775495	2.650844	2.124651	3.511234	1.736900	1.774333	2.144389	0.758654	1.385736
Brag3	5.122345	3.236079	1.886266	3.442827	1.830225	1.612602	2.051355	0.763167	1.288187
$\sigma = 1.0$									
	$h = 0.10^*(0.01)0.70^*$			$h = 0.01(0.01)0.60$			$h = 0.10^*(0.01)0.30^*$		
NW	4.473139	1.402346	3.070793	3.359271	0.844033	2.515238	2.032171	0.254961	1.777210
Boost	4.256086	0.941460	3.314625	3.291609	0.594807	2.696801	1.941506	0.136716	1.804790
Bag1	3.582079	1.099990	2.482089	2.692271	0.674089	2.018182	1.759183	0.202017	1.557165
Bag2	3.548499	2.114164	1.434335	2.641460	1.358243	1.283217	1.578519	0.576033	1.002486
Bag3	4.063665	2.773985	1.289680	2.855672	1.688843	1.166829	1.626753	0.675074	0.951679
Brag1	3.956013	1.218178	2.737835	2.881172	0.718747	2.162425	1.829611	0.212334	1.617278
Brag2	4.252651	1.980744	2.271907	3.204588	1.230149	1.974439	1.948556	0.444708	1.503849
Brag3	4.561490	2.602306	1.959183	3.156168	1.454599	1.701569	1.884308	0.487613	1.396694

Table A.3: Model 1, $X \sim N(0,1)$, $N(0,1)$ kernel

	$n = 50$			$n = 100$			$n = 200$		
	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$
$\sigma = 0.2$									
	$h = 0.05(0.01)1.00$			$h = 0.05(0.01)0.80$			$h = 0.05(0.01)0.60$		
NW	6.422288	3.650138	2.772150	4.908290	2.496482	2.411808	3.310023	1.345379	1.964644
Boost	5.959722	3.196174	2.763547	4.207734	1.940218	2.267516	2.668581	0.944127	1.724454
Bag1	6.180954	3.535980	2.644973	4.484781	2.231793	2.252989	3.008433	1.156083	1.852349
Bag2	6.919521	5.643666	1.275855	5.044859	3.826342	1.218517	3.501413	2.471879	1.029534
Bag3	6.777449	5.788931	0.988518	4.969224	4.004503	0.964721	3.433769	2.602981	0.830788
Brag1	6.376428	3.659505	2.716923	4.708724	2.377372	2.331353	3.198816	1.265805	1.933012
Brag2	7.610161	5.231266	2.378895	5.613323	3.367845	2.245478	4.077780	2.107641	1.970139
Brag3	8.019158	5.911543	2.107615	5.942209	3.846219	2.095990	4.136150	2.278277	1.857874
$\sigma = 0.6$									
	$h = 0.01(0.01)1.60$			$h = 0.10(0.01)1.00$			$h = 0.10(0.01)0.70$		
NW	6.356441	3.234587	3.121854	4.782706	2.137722	2.644983	3.068348	0.956610	2.111738
Boost	5.685311	2.493250	3.192061	4.089943	1.511271	2.578671	2.552481	0.604452	1.948029
Bag1	5.447571	2.650162	2.797409	4.074682	1.675952	2.398730	2.762918	0.795036	1.967882
Bag2	5.955272	4.508321	1.446951	4.509291	3.157706	1.351585	3.014833	1.885433	1.129401
Bag3	5.960676	4.848782	1.111894	4.465287	3.352601	1.112686	2.931058	1.972716	0.958343
Brag1	5.855708	2.915282	2.940426	4.311110	1.822102	2.489007	2.911212	0.857309	2.053902
Brag2	6.923762	4.348197	2.575565	5.252760	2.946116	2.306644	3.532442	1.489859	2.042583
Brag3	7.202695	5.077462	2.125233	5.263300	3.200161	2.063139	3.549197	1.650980	1.898218
$\sigma = 1.0$									
	$h = 0.05(0.01)2.00$			$h = 0.05(0.01)1.50$			$h = 0.01(0.01)1.10$		
NW	6.309755	2.567769	3.741986	4.747311	1.652667	3.094644	3.207096	0.689869	2.517227
Boost	5.898350	1.822010	4.076340	4.652721	1.301902	3.350819	3.057391	0.452121	2.605269
Bag1	5.265470	2.017269	3.248201	3.962213	1.292669	2.669545	2.634483	0.491009	2.143474
Bag2	5.418022	3.608215	1.809808	4.093799	2.501458	1.592341	2.561582	1.250446	1.311136
Bag3	5.657356	4.304936	1.352420	4.092907	2.806795	1.286111	2.553575	1.430002	1.123573
Brag1	5.593759	2.146055	3.447704	4.237675	1.411746	2.825929	2.786165	0.520272	2.265892
Brag2	6.359242	3.475917	2.883324	4.813886	2.344651	2.469235	3.090723	1.024697	2.066026
Brag3	6.903378	4.649425	2.253953	4.915187	2.792879	2.122308	3.102899	1.219248	1.883651

Table A.4: Model 1, $X \sim N(0,1)$, Triangular kernel

	$n = 50$			$n = 100$			$n = 200$		
	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$
$\sigma = 0.2$									
	$h = 0.02(0.01)0.30$			$h = 0.02(0.01)0.20$			$h = 0.02(0.01)0.12$		
NW	0.130711	0.019579	0.111131	0.054613	0.006784	0.047829	0.026370	0.003623	0.022747
Boost	0.107177	0.012690	0.094487	0.042339	0.004146	0.038193	0.020477	0.002299	0.018178
Bag1	0.126284	0.024555	0.101729	0.052555	0.008672	0.043883	0.025730	0.004156	0.021575
Bag2	0.153453	0.059467	0.093986	0.059928	0.018530	0.041399	0.027558	0.007579	0.019979
Bag3	0.157421	0.066637	0.090784	0.060527	0.020245	0.040282	0.027530	0.007944	0.019586
Brag1	0.126545	0.021167	0.105378	0.052533	0.008104	0.044429	0.025876	0.004145	0.021731
Brag2	0.143036	0.036645	0.106391	0.057297	0.014039	0.043258	0.027270	0.006764	0.020506
Brag3	0.151462	0.047630	0.103831	0.058633	0.016355	0.042279	0.027213	0.007209	0.020004
$\sigma = 0.6$									
	$h = 0.01(0.01)0.60$			$h = 0.05(0.01)0.30$			$h = 0.05(0.01)0.35$		
NW	0.478722	0.072059	0.406663	0.227366	0.038676	0.188690	0.121844	0.022117	0.099727
Boost	0.466948	0.064097	0.402852	0.221706	0.028891	0.192815	0.114691	0.016216	0.098476
Bag1	0.413560	0.081834	0.331726	0.217230	0.051867	0.165363	0.116049	0.026799	0.089250
Bag2	0.428084	0.145220	0.282864	0.234252	0.085547	0.148705	0.123119	0.043120	0.079999
Bag3	0.436361	0.165767	0.270594	0.233413	0.085169	0.148244	0.124056	0.043553	0.080504
Brag1	0.440282	0.075182	0.365100	0.219284	0.050323	0.168961	0.116608	0.027056	0.089552
Brag2	0.446257	0.114460	0.331797	0.234729	0.076210	0.158519	0.123804	0.041589	0.082215
Brag3	0.452007	0.158276	0.293731	0.237178	0.084487	0.152692	0.125243	0.043933	0.081311
$\sigma = 1.0$									
	$h = 0.01^*(0.01)1.00^*$			$h = 0.05(0.01)0.65$			$h = 0.05(0.01)0.25^*$		
NW	0.978102	0.149350	0.828753	0.481635	0.090876	0.390758	0.255866	0.049357	0.206510
Boost	1.011530	0.131623	0.879907	0.490006	0.077391	0.412615	0.256930	0.040978	0.215952
Bag1	0.821927	0.186587	0.635339	0.457593	0.134238	0.323355	0.245237	0.067029	0.178207
Bag2	0.843563	0.286219	0.557344	0.495899	0.202047	0.293852	0.259584	0.100631	0.158953
Bag3	0.864198	0.334687	0.529511	0.491740	0.191984	0.299757	0.261241	0.099674	0.161568
Brag1	0.908009	0.148610	0.759399	0.464440	0.125933	0.338507	0.247571	0.067760	0.179811
Brag2	0.896058	0.215554	0.680503	0.495671	0.179213	0.316459	0.262441	0.098791	0.163650
Brag3	0.921862	0.365870	0.555992	0.501433	0.198524	0.302909	0.264452	0.103228	0.161224

Table A.5: Model 2, $X \sim U(-2, 2)$, $N(0, 1)$ kernel

	$n = 50$			$n = 100$			$n = 200$		
	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$
$\sigma = 0.2$									
	$h = 0.10(0.01)0.70$			$h = 0.10(0.01)0.60$			$h = 0.01(0.01)0.4$		
NW	0.147179	0.022984	0.124195	0.057544	0.006636	0.050908	0.027091	0.003447	0.023644
Boost	0.120425	0.013271	0.107154	0.045213	0.004179	0.041034	0.021692	0.002126	0.019566
Bag1	0.138131	0.037400	0.100731	0.055527	0.011017	0.044510	0.026127	0.004315	0.021812
Bag2	0.178268	0.077563	0.100704	0.065867	0.022300	0.043567	0.028304	0.007696	0.020608
Bag3	0.180575	0.087128	0.093448	0.065940	0.024704	0.041237	0.028126	0.008231	0.019895
Brag1	0.137742	0.033443	0.104299	0.054991	0.010175	0.044816	0.026279	0.004163	0.022116
Brag2	0.161534	0.055040	0.106495	0.061872	0.017338	0.044534	0.027684	0.006735	0.020949
Brag3	0.167096	0.065599	0.101496	0.062084	0.019851	0.042233	0.027548	0.007442	0.020106
$\sigma = 0.6$									
	$h = 0.05(0.01)1.00$			$h = 0.10(0.01)0.70$			$h = 0.10(0.01)0.50$		
NW	0.497728	0.071532	0.426196	0.235440	0.036613	0.198827	0.122993	0.020991	0.102002
Boost	0.476812	0.059955	0.416856	0.229890	0.025996	0.203893	0.117915	0.014698	0.103217
Bag1	0.413931	0.081136	0.332795	0.218232	0.047394	0.170838	0.115670	0.025159	0.090511
Bag2	0.430825	0.139574	0.291251	0.231748	0.078475	0.153274	0.122991	0.041014	0.081978
Bag3	0.430151	0.158673	0.271478	0.231622	0.080876	0.150746	0.123628	0.041730	0.081898
Brag1	0.430635	0.070501	0.360134	0.222044	0.046380	0.175663	0.116296	0.025580	0.090716
Brag2	0.443762	0.109056	0.334706	0.233797	0.071665	0.162132	0.124153	0.040126	0.084028
Brag3	0.447630	0.148156	0.299475	0.234826	0.080314	0.154511	0.124404	0.041832	0.082572
$\sigma = 1.0$									
	$h = 0.05(0.01)1.20$			$h = 0.05(0.01)1.00$			$h = 0.10(0.01)0.70$		
NW	0.991840	0.132643	0.859198	0.511039	0.086562	0.424477	0.262476	0.046340	0.216136
Boost	1.011360	0.123220	0.888140	0.503784	0.069637	0.434147	0.266254	0.038226	0.228028
Bag1	0.821404	0.141485	0.679919	0.455682	0.114970	0.340712	0.244219	0.062449	0.181770
Bag2	0.810220	0.222350	0.587870	0.485548	0.176123	0.309425	0.259502	0.094923	0.164579
Bag3	0.813409	0.272725	0.540684	0.484520	0.173407	0.311113	0.260379	0.093607	0.166772
Brag1	0.947379	0.120489	0.826891	0.469639	0.108516	0.361123	0.245788	0.061753	0.184035
Brag2	0.890792	0.176586	0.714207	0.494214	0.158637	0.335577	0.261954	0.092011	0.169943
Brag3	0.865744	0.296725	0.569019	0.497249	0.185932	0.311317	0.264533	0.098239	0.166293

Table A.6: Model 2, $X \sim U(-2, 2)$, Triangular kernel

	$n = 50$			$n = 100$			$n = 200$		
	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$
$\sigma = 0.2$									
	$h = 0.02(0.01)0.30$			$h = 0.02(0.01)0.20$			$h = 0.02(0.01)0.15$		
NW	0.862168	0.386647	0.475521	0.528522	0.156455	0.372067	0.291772	0.058437	0.233334
Boost	0.690410	0.223350	0.467060	0.372868	0.050431	0.322437	0.205885	0.013834	0.192051
Bag1	0.825814	0.361956	0.463858	0.516677	0.155361	0.361315	0.287423	0.055957	0.231465
Bag2	0.890312	0.656365	0.233947	0.557852	0.351367	0.206485	0.304560	0.159844	0.144716
Bag3	0.890944	0.688100	0.202844	0.550798	0.369301	0.181498	0.299300	0.168928	0.130372
Brag1	0.855549	0.390513	0.465036	0.528056	0.161916	0.366139	0.294437	0.059612	0.234825
Brag2	1.035639	0.635467	0.400172	0.668735	0.307404	0.361331	0.351896	0.115070	0.236826
Brag3	1.065652	0.714175	0.351477	0.679607	0.349663	0.329944	0.352469	0.129241	0.223228
$\sigma = 0.6$									
	$h = 0.02^*(0.01)0.4^*$			$h = 0.03^*(0.01)0.35$			$h = 0.05^*(0.01)0.17^*$		
NW	1.441310	0.270555	1.170755	0.980882	0.075119	0.905764	0.700669	0.042469	0.658200
Boost	1.471599	0.227901	1.243698	1.057736	0.035044	1.022693	0.738582	0.025239	0.713344
Bag1	1.488940	0.309733	1.179206	0.937449	0.071466	0.865983	0.688376	0.040691	0.647685
Bag2	1.197353	0.608120	0.589233	0.710456	0.204550	0.505906	0.476200	0.097067	0.379133
Bag3	1.169572	0.649215	0.520357	0.689151	0.232808	0.456343	0.464167	0.105393	0.358774
Brag1	1.463557	0.291567	1.171990	0.938467	0.071135	0.867332	0.689928	0.041220	0.648707
Brag2	1.421255	0.527948	0.893306	0.883845	0.160662	0.723183	0.624574	0.082672	0.541901
Brag3	1.456003	0.724611	0.731393	0.876210	0.241753	0.634456	0.602856	0.100462	0.502394
$\sigma = 1.0$									
	$h = 0.01^*(0.01)0.80$			$h = 0.01(0.01)0.45$			$h = 0.08^*(0.01)0.25^*$		
NW	2.801228	0.333267	2.467960	1.996087	0.092394	1.903692	1.493073	0.075271	1.417802
Boost	3.113076	0.252063	2.861013	2.256362	0.085314	2.171048	1.666588	0.060743	1.605844
Bag1	2.667547	0.281135	2.386413	1.868261	0.094599	1.773661	1.445576	0.079326	1.366250
Bag2	1.992309	0.621922	1.370388	1.242227	0.214954	1.027274	0.951727	0.128725	0.823002
Bag3	1.900690	0.871850	1.028841	1.177709	0.277766	0.899943	0.919319	0.137026	0.782294
Brag1	2.779193	0.278712	2.500481	1.890363	0.090965	1.799397	1.453005	0.079313	1.373692
Brag2	2.481523	0.512932	1.968591	1.580953	0.179665	1.401288	1.227970	0.123253	1.104717
Brag3	2.160469	1.000564	1.159905	1.430286	0.297735	1.132550	1.150051	0.139271	1.010780

Table A.7: Model 2, $X \sim N(0,1)$, $N(0,1)$ kernel

	$n = 50$			$n = 100$			$n = 200$		
	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$
$\sigma = 0.2$									
	$h = 0.10(0.01)0.60$			$h = 0.10(0.01)0.50$			$h = 0.01(0.01)0.40$		
NW	1.023647	0.542390	0.481257	0.707152	0.286807	0.420346	0.421142	0.128283	0.292859
Boost	0.881053	0.393971	0.487083	0.550203	0.160444	0.389759	0.298332	0.053374	0.244958
Bag1	0.964050	0.496216	0.467835	0.680231	0.274136	0.406096	0.413713	0.126872	0.286841
Bag2	1.023517	0.793908	0.229608	0.720249	0.507425	0.212824	0.454643	0.294014	0.160629
Bag3	1.012390	0.816974	0.195416	0.704261	0.523343	0.180918	0.444998	0.305830	0.139167
Brag1	0.987574	0.520754	0.466820	0.701646	0.289692	0.411955	0.423298	0.130783	0.292514
Brag2	1.148062	0.761921	0.386141	0.831159	0.454873	0.376286	0.547586	0.261365	0.286221
Brag3	1.186449	0.849471	0.336978	0.854335	0.510982	0.343353	0.538156	0.277352	0.260804
$\sigma = 0.6$									
	$h = 0.05(0.01)1.00$			$h = 0.10(0.01)0.80$			$h = 0.10(0.01)0.50$		
NW	1.554993	0.432180	1.122813	1.083605	0.174937	0.908667	0.768256	0.086485	0.681770
Boost	1.595960	0.323194	1.272766	1.092273	0.096247	0.996025	0.786703	0.050479	0.736224
Bag1	1.485540	0.397238	1.088302	1.037883	0.164465	0.873418	0.749295	0.085339	0.663956
Bag2	1.280068	0.713744	0.566324	0.855157	0.365177	0.489979	0.587842	0.203899	0.383943
Bag3	1.249023	0.787045	0.461978	0.810830	0.389286	0.421544	0.562834	0.215749	0.347085
Brag1	1.531254	0.416520	1.114733	1.054100	0.171975	0.882124	0.747374	0.085409	0.661965
Brag2	1.525863	0.712933	0.812930	1.045850	0.325736	0.720115	0.740035	0.184551	0.555484
Brag3	1.538463	0.932737	0.605726	1.038105	0.419474	0.618631	0.716744	0.226477	0.490267
$\sigma = 1.0$									
	$h = 0.01(0.01)1.60$			$h = 0.05(0.01)1.00$			$h = 0.05(0.01)0.80$		
NW	2.814963	0.456817	2.358146	2.000506	0.148786	1.851719	1.513612	0.101180	1.412432
Boost	3.095162	0.316898	2.778264	2.235607	0.080754	2.154852	1.701987	0.065465	1.636522
Bag1	2.611353	0.364908	2.246445	1.872173	0.137888	1.734285	1.473232	0.096990	1.376242
Bag2	1.927354	0.699518	1.227837	1.316400	0.308090	1.008310	0.998208	0.191372	0.806836
Bag3	1.828483	0.926271	0.902212	1.207398	0.364793	0.842605	0.929149	0.209875	0.719274
Brag1	2.766323	0.394083	2.372240	1.893622	0.141135	1.752487	1.479662	0.098501	1.381162
Brag2	2.430143	0.679365	1.750778	1.673805	0.275951	1.397854	1.256866	0.187424	1.069441
Brag3	2.169526	1.163393	1.006133	1.499447	0.457528	1.041919	1.134645	0.231849	0.902796

Table A.8: Model 2, $X \sim N(0,1)$, Triangular kernel

	$n = 50$			$n = 100$			$n = 200$		
	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$
$\sigma = 0.1$									
	$h = 0.01(0.01)0.60$			$h = 0.05^*(0.01)0.20$			$h = 0.01(0.01)0.15$		
NW	0.022010	0.003907	0.018103	0.009669	0.001717	0.007952	0.004716	0.000695	0.004021
Boost	0.021131	0.002847	0.018284	0.008592	0.001224	0.007368	0.004028	0.000439	0.003589
Bag1	0.020603	0.004640	0.015962	0.009680	0.002084	0.007596	0.004581	0.000784	0.003797
Bag2	0.022359	0.008726	0.013634	0.010714	0.003692	0.007021	0.004778	0.001393	0.003384
Bag3	0.022697	0.010159	0.012538	0.010683	0.003786	0.006897	0.004811	0.001435	0.003376
Brag1	0.020780	0.003774	0.017006	0.009733	0.002057	0.007676	0.004577	0.000812	0.003765
Brag2	0.021096	0.005533	0.015563	0.010568	0.003185	0.007383	0.004781	0.001333	0.003448
Brag3	0.022270	0.007944	0.014326	0.010650	0.003451	0.007199	0.004804	0.001374	0.003429
$\sigma = 0.3$									
	$h = 0.01^*(0.01)1.00^*$			$h = 0.01(0.01)0.90^*$			$h = 0.10(0.01)0.70$		
NW	0.103870	0.025975	0.077895	0.046159	0.010286	0.035874	0.025120	0.004291	0.020829
Boost	0.107745	0.023702	0.084042	0.047106	0.009158	0.037949	0.023272	0.006125	0.017147
Bag1	0.096396	0.041910	0.054486	0.049591	0.018478	0.031113	0.024761	0.009396	0.015365
Bag2	0.101053	0.052799	0.048254	0.053811	0.025387	0.028424	0.024710	0.009163	0.015547
Bag3	0.086717	0.042925	0.043792	0.048127	0.020525	0.027603	0.023277	0.005977	0.017300
Brag1	0.108411	0.026547	0.081865	0.045428	0.013720	0.031708	0.024729	0.008912	0.015818
Brag2	0.102939	0.032857	0.070082	0.047944	0.018775	0.029169	0.024977	0.009475	0.015502
Brag3	0.094845	0.048857	0.045988	0.049309	0.021233	0.028076	0.025278	0.003516	0.021762
$\sigma = 0.5$									
	$h = 0.01^*(0.01)1.00^*$			$h = 0.01^*(0.01)1.00^*$			$h = 0.1(0.01)0.8^*$		
NW	0.197675	0.051391	0.146284	0.099197	0.023131	0.076066	0.255866	0.049357	0.206510
Boost	0.208941	0.048736	0.160205	0.102449	0.021430	0.081018	0.256930	0.040978	0.215952
Bag1	0.155198	0.063400	0.091798	0.105596	0.055003	0.050592	0.244613	0.065458	0.179155
Bag2	0.156120	0.075733	0.080387	0.111487	0.066224	0.045263	0.260638	0.099110	0.161528
Bag3	0.141789	0.063386	0.078403	0.093156	0.044974	0.048182	0.261906	0.097921	0.163985
Brag1	0.211512	0.049030	0.162482	0.102791	0.039256	0.063535	0.247106	0.066095	0.181011
Brag2	0.189749	0.055722	0.134026	0.104732	0.047187	0.057545	0.263399	0.096663	0.166735
Brag3	0.144855	0.076327	0.068529	0.098798	0.051603	0.047195	0.265981	0.101966	0.164015

Table A.9: Model 3, $X \sim U(-2, 2)$, $N(0, 1)$ kernel

	$n = 50$			$n = 100$			$n = 200$		
	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$
$\sigma = 0.1$									
	$h = 0.01(0.01)1.00$			$h = 0.05(0.01)0.50$			$h = 0.10(0.01)0.40$		
NW	0.023455	0.003972	0.019483	0.009956	0.001543	0.008413	0.004738	0.000650	0.004088
Boost	0.021771	0.002664	0.019107	0.009122	0.001149	0.007973	0.004149	0.000378	0.003771
Bag1	0.021008	0.005076	0.015932	0.009802	0.001889	0.007913	0.004572	0.000734	0.003837
Bag2	0.022832	0.009030	0.013802	0.010688	0.003338	0.007351	0.004765	0.001298	0.003467
Bag3	0.023609	0.011291	0.012318	0.010648	0.003707	0.006940	0.004773	0.001363	0.003410
Brag1	0.020996	0.004352	0.016644	0.009860	0.001814	0.008046	0.004593	0.000731	0.003862
Brag2	0.021654	0.006220	0.015434	0.010503	0.002836	0.007667	0.004801	0.001219	0.003582
Brag3	0.022580	0.008388	0.014192	0.010442	0.003202	0.007240	0.004786	0.001313	0.003473
$\sigma = 0.3$									
	$h = 0.01(0.01)1.00$			$h = 0.05(0.01)0.80$*			$h = 0.10(0.01)0.80$		
NW	0.106905	0.022099	0.084806	0.046915	0.009786	0.037129	0.024626	0.004085	0.020541
Boost	0.112171	0.019834	0.092337	0.048097	0.008683	0.039414	0.025154	0.003286	0.021867
Bag1	0.092807	0.010869	0.081938	0.043670	0.011792	0.031878	0.023075	0.005763	0.017312
Bag2	0.081710	0.016614	0.065096	0.045728	0.017323	0.028404	0.024669	0.009044	0.015625
Bag3	0.086225	0.036677	0.049548	0.045178	0.016872	0.028306	0.024703	0.008923	0.015780
Brag1	0.129740	0.011182	0.118557	0.045062	0.011896	0.033165	0.023253	0.005605	0.017648
Brag2	0.112107	0.015825	0.096282	0.047284	0.016918	0.030365	0.024772	0.008603	0.016169
Brag3	0.095395	0.039336	0.056059	0.046418	0.018457	0.027961	0.025016	0.009337	0.015679
$\sigma = 0.5$									
	$h = 0.01$*$(0.01)1.00$*			$h = 0.01$*$(0.01)1.00$*			$h = 0.10(0.01)1.00$*		
NW	0.238331	0.039226	0.199105	0.105721	0.020563	0.085158	0.054489	0.009782	0.044707
Boost	0.247049	0.037037	0.210012	0.109199	0.018499	0.090700	0.056834	0.008699	0.048135
Bag1	0.265156	0.010424	0.254732	0.087813	0.018045	0.069768	0.050197	0.016376	0.033821
Bag2	0.194991	0.016550	0.178441	0.087161	0.024954	0.062207	0.053856	0.023111	0.030744
Bag3	0.150827	0.056207	0.094619	0.091095	0.032446	0.058650	0.052591	0.020869	0.031722
Brag1	0.417242	0.021400	0.395842	0.099981	0.019840	0.080142	0.051624	0.015428	0.036197
Brag2	0.305482	0.033010	0.272472	0.097286	0.026574	0.070711	0.054695	0.021294	0.033401
Brag3	0.167704	0.075175	0.092529	0.090373	0.036151	0.054222	0.054123	0.022764	0.031359

Table A.10: Model 3, $X \sim U(-2, 2)$, Triangular kernel

	$n = 50$			$n = 100$			$n = 200$		
	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$
$\sigma = 0.1$									
	$h = 0.01(0.01)0.40$			$h = 0.01(0.01)0.20$			$h = 0.01(0.01)0.20$		
NW	0.035882	0.003409	0.032474	0.024155	0.001698	0.022457	0.018432	0.001078	0.017354
Boost	0.038101	0.002555	0.035546	0.026145	0.001318	0.024827	0.020392	0.000794	0.019598
Bag1	0.034951	0.004218	0.030733	0.023253	0.001795	0.021458	0.018354	0.000980	0.017374
Bag2	0.028911	0.009018	0.019893	0.016296	0.003423	0.012874	0.011271	0.001578	0.009693
Bag3	0.026802	0.009542	0.017260	0.015445	0.003606	0.011838	0.010859	0.001661	0.009198
Brag1	0.034658	0.003137	0.031521	0.023292	0.001694	0.021599	0.018310	0.001019	0.017291
Brag2	0.030605	0.004847	0.025758	0.018567	0.002572	0.015995	0.013847	0.001495	0.012351
Brag3	0.028191	0.006800	0.021391	0.016574	0.003038	0.013536	0.012516	0.001571	0.010945
$\sigma = 0.3$									
	$h = 0.01^*(0.01)1.00^*$			$h = 0.01(0.01)1.00^*$			$h = 0.10^*(0.01)0.30^*$		
NW	0.211157	0.022352	0.188805	0.158395	0.013593	0.144802	0.121663	0.008259	0.113404
Boost	0.234276	0.021010	0.213266	0.174392	0.012752	0.161640	0.133869	0.007725	0.126145
Bag1	0.202399	0.038950	0.163449	0.147066	0.026414	0.120652	0.117789	0.010244	0.107545
Bag2	0.160305	0.054983	0.105322	0.111824	0.035730	0.076094	0.076647	0.014006	0.062641
Bag3	0.110344	0.039520	0.070825	0.089725	0.025558	0.064167	0.073642	0.013423	0.060218
Brag1	0.208122	0.021698	0.186425	0.150697	0.016816	0.133881	0.119066	0.009588	0.109478
Brag2	0.173954	0.028323	0.145631	0.122089	0.022487	0.099603	0.095018	0.012858	0.082160
Brag3	0.116582	0.043051	0.073531	0.093984	0.027705	0.066278	0.086603	0.013721	0.072881
$\sigma = 0.5$									
	$h = 0.01^*(0.01)1.00^*$			$h = 0.01^*(0.01)1.00^*$			$h = 0.10^*(0.01)0.40^*$		
NW	0.435106	0.043666	0.391441	0.337000	0.036623	0.300376	0.267728	0.019840	0.247888
Boost	0.487367	0.041926	0.445441	0.365115	0.035310	0.329805	0.290496	0.019464	0.271032
Bag1	0.389234	0.067633	0.321601	0.257042	0.071455	0.185587	0.239394	0.027681	0.211713
Bag2	0.291029	0.087800	0.203229	0.213686	0.087487	0.126199	0.170250	0.035250	0.135000
Bag3	0.195319	0.064702	0.130617	0.169667	0.060072	0.109596	0.159453	0.031447	0.128006
Brag1	0.438596	0.044997	0.393599	0.290769	0.054922	0.235847	0.245898	0.026940	0.218958
Brag2	0.346846	0.054992	0.291854	0.253767	0.063728	0.190039	0.203887	0.034118	0.169769
Brag3	0.196429	0.079221	0.117208	0.173112	0.071731	0.101382	0.173002	0.034593	0.138409

Table A.11: Model 3, $X \sim N(0,1)$, $N(0,1)$ kernel

	$n = 50$			$n = 100$			$n = 200$		
	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$
$\sigma = 0.1$									
	$h = 0.01(0.01)1.00$			$h = 0.01(0.01)0.70$			$h = 0.01(0.01)0.40$		
NW	0.034620	0.003514	0.031106	0.022628	0.001634	0.020994	0.017061	0.000981	0.016080
Boost	0.035124	0.002552	0.032573	0.024284	0.001182	0.023102	0.019150	0.000688	0.018462
Bag1	0.033238	0.005257	0.027981	0.021354	0.001877	0.019477	0.016983	0.000880	0.016103
Bag2	0.028267	0.009345	0.018922	0.015108	0.003206	0.011902	0.010647	0.001425	0.009222
Bag3	0.026880	0.010759	0.016121	0.014270	0.003558	0.010713	0.009975	0.001525	0.008450
Brag1	0.033034	0.004119	0.028916	0.021275	0.001701	0.019573	0.016971	0.000902	0.016070
Brag2	0.030300	0.005966	0.024334	0.016742	0.002470	0.014272	0.012903	0.001378	0.011525
Brag3	0.028005	0.007932	0.020072	0.014736	0.002898	0.011837	0.010992	0.001458	0.009534
$\sigma = 0.3$									
	$h = 0.01^*(0.01)1.50^*$			$h = 0.01(0.01)1.00^*$			$h = 0.01(0.01)1.00$		
NW	0.202699	0.017256	0.185443	0.152068	0.011389	0.140679	0.122894	0.006854	0.116040
Boost	0.226782	0.016085	0.210697	0.168655	0.009945	0.158709	0.138334	0.006044	0.132291
Bag1	0.192088	0.014701	0.177387	0.143958	0.016918	0.127040	0.117974	0.008007	0.109967
Bag2	0.124554	0.022861	0.101692	0.099347	0.024180	0.075167	0.075295	0.011179	0.064116
Bag3	0.100500	0.033894	0.066607	0.083226	0.020623	0.062603	0.067663	0.010716	0.056947
Brag1	0.206554	0.010391	0.196163	0.146659	0.013089	0.133569	0.119537	0.007413	0.112125
Brag2	0.154488	0.015906	0.138581	0.113503	0.018141	0.095362	0.086932	0.010166	0.076766
Brag3	0.105183	0.034582	0.070602	0.085532	0.022236	0.063296	0.071009	0.011221	0.059788
$\sigma = 0.5$									
	$h = 0.01^*(0.01)1.00^*$			$h = 0.01^*(0.01)1.00^*$			$h = 0.20^*(0.01)2.00^*$		
NW	0.471763	0.031639	0.440124	0.344337	0.026605	0.317732	0.278745	0.017525	0.261220
Boost	0.512044	0.031132	0.480912	0.381931	0.025990	0.355941	0.305747	0.016483	0.289264
Bag1	0.520793	0.009155	0.511637	0.352460	0.021714	0.330746	0.233851	0.035942	0.197909
Bag2	0.282227	0.014386	0.267841	0.225521	0.029964	0.195558	0.177206	0.045003	0.132202
Bag3	0.178530	0.047117	0.131413	0.171684	0.037260	0.134424	0.149809	0.032176	0.117633
Brag1	0.542120	0.012196	0.529924	0.370595	0.022074	0.348521	0.252317	0.025368	0.226949
Brag2	0.384636	0.019769	0.364866	0.294275	0.029112	0.265164	0.202916	0.031931	0.170984
Brag3	0.178007	0.061426	0.116580	0.162653	0.041569	0.121084	0.156890	0.035912	0.120978

Table A.12: Model 3, $X \sim N(0, 1)$, Triangular kernel

$n = 50$, $\sigma = 0.2$, $h = 0.05(0.01)0.4$			
Procedure	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$
NW	0.132777	0.020800	0.111977
Bag1	0.123854	0.039302	0.084551
Bag2	0.163837	0.079290	0.084547
Bag3	0.168479	0.083822	0.084657
Brag1	0.120232	0.032620	0.087613
Brag2	0.144020	0.050111	0.093908
Brag3	0.151318	0.061679	0.089639
Boost_1	0.107578	0.013564	0.094013
Boost_6	0.100103	0.008785	0.091318
Bagboost1	0.088696	0.011500	0.077197
Bagboost2	0.105146	0.035950	0.069196
Bagboost3	0.105118	0.036394	0.068724
Bragboost1	0.090425	0.008776	0.081649
Bragboost2	0.092180	0.020623	0.071557
Bragboost3	0.095531	0.025461	0.070069

Table A.13: Model 1, $X \sim U(-2, 2)$, $N(0, 1)$ kernel

$n = 50$, $\sigma = 0.2$, $h = 0.05(0.01)1.00$			
Procedure	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$
NW	0.180075	0.028676	0.151399
Bag1	0.150128	0.061837	0.088291
Bag2	0.207548	0.113063	0.094485
Bag3	0.207644	0.116629	0.091015
Brag1	0.144149	0.053383	0.090766
Brag2	0.178992	0.079628	0.099365
Brag3	0.179939	0.087388	0.092551
Boost_1	0.138999	0.016251	0.122749
Boost_6	0.122473	0.009550	0.112923
Bagboost1	0.106575	0.019597	0.086978
Bagboost2	0.138751	0.054581	0.084170
Bagboost3	0.129117	0.053201	0.075916
Bragboost1	0.108060	0.014628	0.093432
Bragboost2	0.119745	0.031705	0.088040
Bragboost3	0.109975	0.034964	0.075011

Table A.14: Model 1, $X \sim U(-2, 2)$, Triangular kernel

$n = 200$, $\sigma = 0.2$, $h = 0.05(0.01)0.40$			
Procedure	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$
NW	0.025603	0.003611	0.021992
Bag1	0.024538	0.004476	0.020062
Bag2	0.026760	0.008034	0.018726
Bag3	0.026931	0.008577	0.018353
Brag1	0.024573	0.004261	0.020313
Brag2	0.025914	0.006916	0.018998
Brag3	0.025835	0.007580	0.018255
Boost_1	0.020747	0.002070	0.018677
Boost_6	0.018946	0.001342	0.017604
Bagboost1	0.018153	0.000697	0.017455
Bagboost2	0.017874	0.002333	0.015541
Bagboost3	0.017523	0.002422	0.015101
Bragboost1	0.018319	0.000793	0.017526
Bragboost2	0.018274	0.002519	0.015754
Bragboost3	0.017583	0.002328	0.015255

Table A.15: Model 1, $X \sim U(-2, 2)$, Triangular kernel

$n = 50$, $\sigma = 0.2$,$h = 0.05(0.01)0.40$			
Procedure	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$
NW	5.060624	2.662221	2.398403
Bag1	4.697235	2.471375	2.225860
Bag2	5.462628	4.285872	1.176756
Bag3	5.447989	4.497159	0.950830
Brag1	4.892580	2.594423	2.298158
Brag2	5.960503	3.824263	2.136240
Brag3	6.165079	4.212036	1.953043
Boost_1	4.405221	2.167701	2.237519
Boost_6	3.743244	1.673363	2.069881
Bagboost1	3.367535	1.501856	1.865680
Bagboost2	3.866745	2.824655	1.042089
Bagboost3	3.922203	3.127266	0.794937
Bragboost1	3.461800	1.557399	1.904401
Bragboost2	4.024126	2.292369	1.731758
Bragboost3	4.399643	2.608586	1.791057

Table A.16: Model 1, $X \sim N(0, 1)$, $N(0, 1)$ kernel

$n = 50$, $\sigma = 0.2$, $h = 0.05(0.01)1.00$			
Procedure	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$
NW	6.422288	3.650138	2.772150
Bag1	6.180954	3.535980	2.644973
Bag2	6.919521	5.643666	1.275855
Bag3	6.777449	5.788931	0.988518
Brag1	6.376428	3.659505	2.716923
Brag2	7.610161	5.231266	2.378895
Brag3	8.019158	5.911543	2.107615
Boost_1	5.959722	3.196174	2.763547
Boost_6	5.483972	2.749084	2.734888
Bagboost1	5.297899	2.696847	2.601052
Bagboost2	5.883082	4.593879	1.289203
Bagboost3	5.570216	4.617744	0.952472
Bragboost1	5.474436	2.811805	2.662631
Bragboost2	6.580505	4.177448	2.403057
Bragboost3	6.853388	4.546462	2.306926

Table A.17: Model 1, $X \sim N(0, 1)$, Triangular kernel

$n = 100$, $\sigma = 0.2$,$h = 0.05(0.01)0.80$			
Procedure	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$
NW	4.908290	2.496482	2.411808
Bag1	4.484781	2.231793	2.252989
Bag2	5.044859	3.826342	1.218517
Bag3	4.969224	4.004503	0.964721
Brag1	4.708724	2.377372	2.331353
Brag2	5.613323	3.367845	2.245478
Brag3	5.942209	3.846219	2.095990
Boost_1	4.207734	1.940218	2.267516
Boost_6	3.647236	1.500287	2.146949
Bagboost1	3.542911	1.470446	2.072464
Bagboost2	3.974214	2.873138	1.101076
Bagboost3	3.905867	3.037202	0.868665
Bragboost1	3.636137	1.535817	2.100320
Bragboost2	4.352513	2.352687	1.999826
Bragboost3	4.674462	2.708035	1.966427

Table A.18: Model 1, $X \sim N(0, 1)$, Triangular kernel

$n = 50$, $\sigma = 0.2$, $h = 0.02(0.01)0.30$			
Procedure	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$
NW	0.130711	0.019579	0.111131
Bag1	0.126284	0.024555	0.101729
Bag2	0.153453	0.059467	0.093986
Bag3	0.157421	0.066637	0.090784
Brag1	0.126545	0.021167	0.105378
Brag2	0.143036	0.036645	0.106391
Brag3	0.151462	0.047630	0.103831
Boost_1	0.107177	0.012690	0.094487
Boost_6	0.100626	0.007611	0.093015
Bagboost1	0.093375	0.006690	0.086685
Bagboost2	0.103540	0.027417	0.076122
Bagboost3	0.105699	0.031195	0.074503
Bragboost1	0.094734	0.006659	0.088074
Bragboost2	0.095720	0.016767	0.078953
Bragboost3	0.099895	0.021216	0.078679

Table A.19: Model 2, $X \sim U(-2, 2)$, $N(0, 1)$ kernel

$n = 50$, $\sigma = 0.2$, $h = 0.10(0.01)0.70$			
Procedure	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$
NW	0.147179	0.022984	0.124195
Bag1	0.138131	0.037400	0.100731
Bag2	0.178268	0.077563	0.100704
Bag3	0.180575	0.087128	0.093448
Brag1	0.137742	0.033443	0.104299
Brag2	0.161534	0.055040	0.106495
Brag3	0.167096	0.065599	0.101496
Boost_1	0.120425	0.013271	0.107154
Boost_6	0.111843	0.007244	0.104599
Bagboost1	0.106775	0.009396	0.097378
Bagboost2	0.116453	0.032778	0.083675
Bagboost3	0.112143	0.038154	0.073988
Bragboost1	0.110364	0.008096	0.102269
Bragboost2	0.104526	0.019904	0.084622
Bragboost3	0.103105	0.025947	0.077158

Table A.20: Model 2, $X \sim U(-2, 2)$, Triangular kernel

$n = 100$, $\sigma = 0.2$, $h = 0.10(0.01)0.60$			
Procedure	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$
NW	0.057544	0.006636	0.050908
Bag1	0.055527	0.011017	0.044510
Bag2	0.065867	0.022300	0.043567
Bag3	0.065940	0.024704	0.041237
Brag1	0.054991	0.010175	0.044816
Brag2	0.061872	0.017338	0.044534
Brag3	0.062084	0.019851	0.042233
Boost_1	0.045213	0.004179	0.041034
Boost_6	0.040247	0.001974	0.038273
Bagboost1	0.037865	0.001382	0.036482
Bagboost2	0.038800	0.005558	0.033242
Bagboost3	0.037314	0.006524	0.030790
Bragboost1	0.038725	0.001316	0.037409
Bragboost2	0.038257	0.004643	0.033614
Bragboost3	0.037075	0.005365	0.031710

Table A.21: Model 2, $X \sim U(-2, 2)$, Triangular kernel

$n = 50$, $\sigma = 0.2$, , $h = 0.02(0.01)0.30$			
Procedure	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$
NW	0.862168	0.386647	0.475521
Bag1	0.825814	0.361956	0.463858
Bag2	0.890312	0.656365	0.233947
Bag3	0.890944	0.688100	0.202844
Brag1	0.855549	0.390513	0.465036
Brag2	1.035639	0.635467	0.400172
Brag3	1.065652	0.714175	0.351477
Boost_1	0.690410	0.223350	0.467060
Boost_6	0.593710	0.093357	0.500353
Bagboost1	0.606297	0.132318	0.473979
Bagboost2	0.622998	0.369589	0.253409
Bagboost3	0.601739	0.389896	0.211844
Bragboost1	0.608404	0.119880	0.488524
Bragboost2	0.691895	0.284224	0.407671
Bragboost3	0.769527	0.390555	0.378972

Table A.22: Model 2, $X \sim N(0, 1)$, $N(0, 1)$ kernel

$n = 50$, $\sigma = 0.2$, $h = 0.10(0.01)0.60$			
Procedure	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$
NW	1.023647	0.542390	0.481257
Bag1	0.964050	0.496216	0.467835
Bag2	1.023517	0.793908	0.229608
Bag3	1.012390	0.816974	0.195416
Brag1	0.987574	0.520754	0.466820
Brag2	1.148062	0.761921	0.386141
Brag3	1.186449	0.849471	0.336978
Boost_1	0.881053	0.393971	0.487083
Boost_6	0.801946	0.274474	0.527472
Bagboost1	0.826011	0.314292	0.511718
Bagboost2	0.823354	0.578255	0.245099
Bagboost3	0.788175	0.588981	0.199194
Bragboost1	0.828464	0.307596	0.520868
Bragboost2	0.936133	0.523358	0.412775
Bragboost3	0.985059	0.622872	0.362187

Table A.23: Model 2, $X \sim N(0, 1)$, Triangular kernel

$n = 100$, $\sigma = 0.2$, $h = 0.10(0.01)0.50$			
Procedure	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$
NW	0.707152	0.286807	0.420346
Bag1	0.680231	0.274136	0.406096
Bag2	0.720249	0.507425	0.212824
Bag3	0.704261	0.523343	0.180918
Brag1	0.701646	0.289692	0.411955
Brag2	0.831159	0.454873	0.376286
Brag3	0.854335	0.510982	0.343353
Boost_1	0.550203	0.160444	0.389759
Boost_6	0.469219	0.080117	0.389103
Bagboost1	0.490021	0.096042	0.393980
Bagboost2	0.459687	0.250286	0.209401
Bagboost3	0.435771	0.264982	0.170789
Bragboost1	0.484678	0.089543	0.395135
Bragboost2	0.531410	0.195201	0.336209
Bragboost3	0.578307	0.262748	0.315559

Table A.24: Model 2, $X \sim N(0, 1)$, Triangular kernel

$n = 200$, $\sigma = 0.2$, $h = 0.01(0.01)0.40$			
Procedure	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$
NW	0.421142	0.128283	0.292858674
Bag1	0.410686	0.125301	0.285385256
Bag2	0.454375	0.286687	0.167688654
Bag3	0.442490	0.297276	0.145213801
Brag1	0.422972	0.129509	0.293463541
Brag2	0.539802	0.235613	0.304188817
Brag3	0.547012	0.265537	0.281475067
Boost_1	0.298332	0.053374	0.244958
Boost_6	0.261411	0.024724	0.236687078
Bagboost1	0.275312	0.030070	0.245242049
Bagboost2	0.246539	0.102274	0.144264546
Bagboost3	0.235902	0.110250	0.125651817
Bragboost1	0.273590	0.027589	0.246000509
Bragboost2	0.305871	0.074108	0.231763558
Bragboost3	0.313943	0.093417	0.220525838

Table A.25: Model 2, $X \sim N(0, 1)$, Triangular kernel

$n = 50$, $\sigma = 0.1$, $h = 0.01(0.01)0.60$			
Procedure	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$
NW	0.022010	0.003907	0.018103
Bag1	0.020603	0.004640	0.015962
Bag2	0.022359	0.008726	0.013634
Bag3	0.022697	0.010159	0.012538
Brag1	0.020780	0.003774	0.017006
Brag2	0.021096	0.005533	0.015563
Brag3	0.022270	0.007944	0.014326
Boost_1	0.021131	0.002847	0.018284
Boost_6	0.021122	0.002154	0.018968
Bagboost1	0.018299	0.001756	0.016543
Bagboost2	0.018920	0.004987	0.013933
Bagboost3	0.018693	0.005883	0.012810
Bragboost1	0.019304	0.001597	0.017706
Bragboost2	0.018896	0.003375	0.015521
Bragboost3	0.018171	0.004107	0.014064

Table A.26: Model 3, $X \sim U(-2, 2)$, $N(0, 1)$ kernel

$n = 100$, $\sigma = 0.1$, $h = 0.05(0.01)0.20$			
Procedure	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$
NW	0.009669	0.001717	0.007952
Bag1	0.009680	0.002084	0.007596
Bag2	0.010714	0.003692	0.007021
Bag3	0.010683	0.003786	0.006897
Brag1	0.009733	0.002057	0.007676
Brag2	0.010568	0.003185	0.007383
Brag3	0.010650	0.003451	0.007199
Boost_1	0.008592	0.001224	0.007368
Boost_6	0.008201	0.000823	0.007378
Bagboost1	0.007845	0.000644	0.007201
Bagboost2	0.008086	0.001719	0.006367
Bagboost3	0.008145	0.001727	0.006418
Bragboost1	0.008084	0.000884	0.007200
Bragboost2	0.008563	0.001958	0.006604
Bragboost3	0.008014	0.001540	0.006474

Table A.27: Model 3, $X \sim U(-2, 2)$, $N(0, 1)$ kernel

$n = 50$, $\sigma = 0.1$, $h = 0.01(0.01)1.00$			
Procedure	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$
NW	0.023455	0.003972	0.019483
Bag1	0.021008	0.005076	0.015932
Bag2	0.022832	0.009030	0.013802
Bag3	0.023609	0.011291	0.012318
Brag1	0.020996	0.004352	0.016644
Brag2	0.021654	0.006220	0.015434
Brag3	0.022580	0.008388	0.014192
Boost_1	0.021771	0.002664	0.019107
Boost_6	0.021153	0.002036	0.019118
Bagboost1	0.019742	0.001463	0.018279
Bagboost2	0.019368	0.004514	0.014854
Bagboost3	0.018796	0.006000	0.012796
Bragboost1	0.021000	0.001242	0.019757
Bragboost2	0.018955	0.002906	0.016049
Bragboost3	0.017890	0.003808	0.014082

Table A.28: Model 3, $X \sim U(-2, 2)$, Triangular kernel

$n = 100$, $\sigma = 0.1$,$h = 0.05(0.01)0.50$			
Procedure	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$
NW	0.009956	0.001543	0.008413
Bag1	0.009802	0.001889	0.007913
Bag2	0.010688	0.003338	0.007351
Bag3	0.010648	0.003707	0.006940
Brag1	0.009860	0.001814	0.008046
Brag2	0.010503	0.002836	0.007667
Brag3	0.010442	0.003202	0.007240
Boost_1	0.009122	0.001149	0.007973
Boost_6	0.008652	0.000718	0.007934
Bagboost1	0.008306	0.000476	0.007830
Bagboost2	0.008348	0.001362	0.006986
Bagboost3	0.008220	0.001520	0.006700
Bragboost1	0.008368	0.000595	0.007773
Bragboost2	0.008476	0.001448	0.007028
Bragboost3	0.008093	0.001324	0.006769

Table A.29: Model 3, $X \sim U(-2, 2)$, Triangular kernel

$n = 200$, $\sigma = 0.1$, $h = 0.10(0.01)0.40$			
Procedure	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$
NW	0.004738	0.000650	0.004088
Bag1	0.004572	0.000734	0.003837
Bag2	0.004765	0.001298	0.003467
Bag3	0.004773	0.001363	0.003410
Brag1	0.004593	0.000731	0.003862
Brag2	0.004801	0.001219	0.003582
Brag3	0.004786	0.001313	0.003473
Boost_1	0.004149	0.000378	0.003771
Boost_6	0.004139	0.000215	0.003924
Bagboost1	0.003913	0.000132	0.003781
Bagboost2	0.003750	0.000444	0.003306
Bagboost3	0.003753	0.000468	0.003286
Bragboost1	0.003933	0.000181	0.003752
Bragboost2	0.003902	0.000553	0.003349
Bragboost3	0.003767	0.000464	0.003303

Table A.30: Model 3, $X \sim U(-2, 2)$, Triangular kernel,

Appendix B

Results of Simulation Study II

			SimIIA NW				SimIIB Boosted NW					
σ	$f(x)$	Kernel	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$	h	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$	h	T	$Gain(\%)$
0.2	U(-2,2)	N(0,1)	0.11216	0.02495	0.08720	0.10	0.08325	0.00985	0.07341	0.17	4	25.77
		Triangular	0.13420	0.03670	0.09750	0.28	0.09421	0.01356	0.08064	0.38	3	29.80
	N(0,1)	N(0,1)	1.35239	0.97106	0.38133	0.36	0.69556	0.39541	0.30016	0.43	3	48.57
		Triangular	2.01762	1.54902	0.46860	1.18	0.98115	0.71657	0.26458	1.49	2	51.37
0.6	U(-2,2)	N(0,1)	0.36243	0.09492	0.26751	0.17	0.36243	0.09492	0.26751	0.17	0	0.00
		Triangular	0.37114	0.10487	0.26627	0.42	0.37114	0.10487	0.26627	0.42	0	0.00
	N(0,1)	N(0,1)	1.75525	0.99081	0.76444	0.37	1.40091	0.70998	0.69093	0.65	2	20.19
		Triangular	2.31383	1.58238	0.73145	1.21	1.55342	0.91112	0.64230	1.55	1	32.86
1	U(-2,2)	N(0,1)	0.68116	0.21736	0.46380	0.25	0.68116	0.21736	0.46380	0.25	0	0.00
		Triangular	0.69640	0.21614	0.48026	0.58	0.69640	0.21614	0.48026	0.58	0	0.00
	N(0,1)	N(0,1)	2.50502	1.16810	1.33693	0.42	2.10880	1.14957	0.95923	0.90	2	15.82
		Triangular	2.88656	1.67368	1.21288	1.27	2.26861	1.22661	1.04200	1.85	1	21.41

Table B.1: Model 1, $n = 50$

			SimIIA NW				SimIIB Boosted NW					
σ	$f(x)$	Kernel	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$	h	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$	h	T	$Gain(\%)$
0.2	U(-2,2)	N(0,1)	0.04799	0.00609	0.04190	0.07	0.03205	0.00384	0.02822	0.18	6	33.21
		Triangular	0.05103	0.01034	0.04069	0.20	0.03474	0.00340	0.03133	0.36	4	31.94
	N(0,1)	N(0,1)	0.78158	0.54651	0.23506	0.29	0.39523	0.19708	0.19815	0.34	4	49.43
		Triangular	1.25810	0.93336	0.32473	0.94	0.66397	0.45865	0.20532	1.13	3	47.22
0.6	U(-2,2)	N(0,1)	0.19326	0.04338	0.14988	0.13	0.18438	0.03603	0.14835	0.18	1	4.59
		Triangular	0.19347	0.04250	0.15098	0.31	0.18630	0.03401	0.15229	0.41	1	3.71
	N(0,1)	N(0,1)	1.13185	0.59632	0.53553	0.30	0.95720	0.43822	0.51898	0.41	1	15.43
		Triangular	1.52501	1.03775	0.48726	1.00	1.08704	0.67430	0.41274	1.31	1	28.72
1	U(-2,2)	N(0,1)	0.39100	0.10625	0.28475	0.18	0.39100	0.10625	0.28475	0.18	0	0.00
		Triangular	0.39242	0.09653	0.29589	0.41	0.39242	0.09653	0.29589	0.41	0	0.00
	N(0,1)	N(0,1)	1.73320	0.78182	0.95138	0.35	1.47794	0.81881	0.65914	0.76	2	14.73
		Triangular	1.98587	1.17042	0.81545	1.07	1.57580	0.88300	0.69280	1.58	1	20.65

Table B.2: Model 1, $n = 00$

			SimIIA NW				SimIIB Boosted NW					
σ	$f(x)$	Kernel	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$	h	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$	h	T	$Gain(\%)$
0.2	U(-2,2)	N(0,1)	0.02422	0.00351	0.02072	0.06	0.01574	0.00166	0.01409	0.16	6	35.01
		Triangular	0.02437	0.00380	0.02057	0.15	0.01680	0.00165	0.01515	0.34	5	31.08
	N(0,1)	N(0,1)	0.37953	0.24181	0.13772	0.22	0.18509	0.06196	0.12313	0.26	4	51.23
		Triangular	0.68697	0.51773	0.16923	0.75	0.38288	0.22959	0.15329	0.82	3	44.27
0.6	U(-2,2)	N(0,1)	0.10877	0.02641	0.08236	0.11	0.09936	0.01525	0.08411	0.19	3	8.65
		Triangular	0.10794	0.02501	0.08293	0.26	0.10069	0.01747	0.08322	0.35	1	6.72
	N(0,1)	N(0,1)	0.64589	0.27937	0.36652	0.24	0.60886	0.19902	0.40984	0.30	1	5.73
		Triangular	0.87851	0.52065	0.35786	0.76	0.72766	0.37942	0.34825	0.93	1	17.17
1	U(-2,2)	N(0,1)	0.22657	0.05472	0.17186	0.14	0.22299	0.04504	0.17796	0.19	1	1.58
		Triangular	0.22564	0.05296	0.17268	0.33	0.22540	0.04642	0.17898	0.44	1	0.11
	N(0,1)	N(0,1)	1.09889	0.41015	0.68874	0.29	1.06187	0.50075	0.56113	0.49	1	3.37
		Triangular	1.23097	0.61518	0.61579	0.84	1.21999	0.42709	0.79291	1.00	1	0.89

Table B.3: Model 1, $n = 200$

			SimIIA NW				SimIIB Boosted NW					
σ	$f(x)$	Kernel	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$	h	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$	h	T	$Gain(\%)$
0.2	U(-2,2)	N(0,1)	0.12402	0.02750	0.09653	0.10	0.08677	0.00980	0.07696	0.18	5	30.04
		Triangular	0.13143	0.02921	0.10222	0.25	0.09520	0.00804	0.08717	0.34	3	27.56
	N(0,1)	N(0,1)	0.64757	0.23007	0.41750	0.16	0.46663	0.06675	0.39988	0.22	3	27.94
		Triangular	0.80018	0.36415	0.43604	0.45	0.62299	0.15567	0.46733	0.57	3	22.14
0.6	U(-2,2)	N(0,1)	0.38441	0.09020	0.29421	0.16	0.37773	0.08534	0.29239	0.22	1	1.74
		Triangular	0.39162	0.08898	0.30264	0.38	0.39162	0.08898	0.30264	0.38	0	0.00
	N(0,1)	N(0,1)	1.28009	0.26589	1.01420	0.19	1.28009	0.26589	1.01420	0.19	0	0.00
		Triangular	1.34477	0.38768	0.95709	0.51	1.34477	0.38768	0.95709	0.51	0	0.00
1	U(-2,2)	N(0,1)	0.73133	0.19017	0.54116	0.22	0.73133	0.19017	0.54116	0.22	0	0.00
		Triangular	0.74452	0.20129	0.54324	0.53	0.74452	0.20129	0.54324	0.53	0	0.00
	N(0,1)	N(0,1)	2.45844	0.47581	1.98263	0.26	2.45844	0.47581	1.98263	0.26	0	0.00
		Triangular	2.38849	0.45438	1.93411	0.59	2.38849	0.45438	1.93411	0.59	0	0.00

Table B.4: Model 2, $n = 50$

			SimIIA NW				SimIIB Boosted NW					
σ	$f(x)$	Kernel	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$	h	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$	h	T	$Gain(\%)$
0.2	U(-2,2)	N(0,1)	0.05152	0.00654	0.04499	0.07	0.03266	0.00381	0.02886	0.18	6	36.61
		Triangular	0.05353	0.00767	0.04585	0.18	0.03573	0.00306	0.03268	0.37	5	33.24
	N(0,1)	N(0,1)	0.35205	0.08344	0.26861	0.14	0.28178	0.02106	0.26072	0.17	2	19.96
		Triangular	0.46737	0.16024	0.30713	0.40	0.37050	0.05793	0.31258	0.48	2	20.73
0.6	U(-2,2)	N(0,1)	0.20248	0.04746	0.15503	0.13	0.18627	0.02891	0.15737	0.20	2	8.00
		Triangular	0.20241	0.04161	0.16081	0.30	0.18952	0.03057	0.15896	0.40	1	6.37
	N(0,1)	N(0,1)	0.84433	0.12027	0.72406	0.17	0.84433	0.12027	0.72406	0.17	0	0.00
		Triangular	0.89646	0.16379	0.73267	0.43	0.89646	0.16379	0.73267	0.43	0	0.00
1	U(-2,2)	N(0,1)	0.40986	0.10324	0.30663	0.17	0.40268	0.08758	0.31510	0.23	1	1.75
		Triangular	0.41025	0.09311	0.31714	0.39	0.41010	0.08530	0.32481	0.52	1	0.04
	N(0,1)	N(0,1)	1.64430	0.52593	1.11837	0.30	1.64430	0.52593	1.11837	0.30	0	0.00
		Triangular	1.68025	0.32707	1.35318	0.61	1.68025	0.32707	1.35318	0.61	0	0.00

Table B.5: Model 2, $n = 100$

			SimIIA NW				SimIIB Boosted NW					
σ	$f(x)$	Kernel	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$	h	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$	h	T	$Gain(\%)$
0.2	U(-2,2)	N(0,1)	0.02553	0.00382	0.02171	0.06	0.01593	0.00168	0.01425	0.16	6	37.60
		Triangular	0.02589	0.00416	0.02172	0.15	0.01715	0.00169	0.01546	0.34	5	33.74
	N(0,1)	N(0,1)	0.19762	0.03579	0.16183	0.11	0.16560	0.01518	0.15042	0.14	1	16.20
		Triangular	0.26276	0.07862	0.18414	0.33	0.20468	0.02480	0.17988	0.42	2	22.11
0.6	U(-2,2)	N(0,1)	0.11227	0.02838	0.08389	0.11	0.09995	0.01532	0.08463	0.19	3	10.97
		Triangular	0.11129	0.02364	0.08764	0.25	0.10180	0.01687	0.08493	0.40	2	8.52
	N(0,1)	N(0,1)	0.61171	0.10892	0.50279	0.16	0.61171	0.10892	0.50279	0.16	0	0.00
		Triangular	0.63317	0.12309	0.51007	0.40	0.63317	0.12309	0.51007	0.40	0	0.00
1	U(-2,2)	N(0,1)	0.23221	0.05882	0.17339	0.14	0.22434	0.04569	0.17865	0.19	1	3.39
		Triangular	0.23115	0.05663	0.17451	0.33	0.22703	0.04316	0.18387	0.43	1	1.78
	N(0,1)	N(0,1)	1.19150	0.38664	0.80486	0.26	1.19150	0.38664	0.80486	0.26	0	0.00
		Triangular	1.24868	0.30600	0.94268	0.57	1.24868	0.30600	0.94268	0.57	0	0.00

Table B.6: Model 2, $n = 200$

			SimIIA NW				SimIIB Boosted NW					
σ	$f(x)$	Kernel	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$	h	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$	h	T	$Gain(\%)$
0.1	U(-2,2)	N(0,1)	0.01896	0.00333	0.01563	0.10	0.01636	0.00199	0.01437	0.18	3	13.71
		Triangular	0.01929	0.00391	0.01539	0.26	0.01746	0.00224	0.01523	0.38	2	9.47
	N(0,1)	N(0,1)	0.03253	0.00157	0.03095	0.07	0.03253	0.00157	0.03095	0.07	0	0.00
		Triangular	0.03047	0.00251	0.02796	0.22	0.03047	0.00251	0.02796	0.22	0	0.00
0.3	U(-2,2)	N(0,1)	0.07373	0.02230	0.05143	0.21	0.07373	0.02230	0.05143	0.21	0	0.00
		Triangular	0.07484	0.02194	0.05290	0.49	0.07484	0.02194	0.05290	0.49	0	0.00
	N(0,1)	N(0,1)	0.19109	0.09916	0.09193	0.49	0.19109	0.09916	0.09193	0.49	0	0.00
		Triangular	0.18387	0.01746	0.16641	0.44	0.18387	0.01746	0.16641	0.44	0	0.00
0.5	U(-2,2)	N(0,1)	0.13903	0.05191	0.08712	0.33	0.13903	0.05191	0.08712	0.33	0	0.00
		Triangular	0.14102	0.05262	0.08840	0.77	0.14102	0.05262	0.08840	0.77	0	0.00
	N(0,1)	N(0,1)	0.20341	0.16609	0.03732	5.00*	0.20341	0.16609	0.03732	5.00*	0	0.00
		Triangular	0.20365	0.16327	0.04038	5.00*	0.20365	0.16327	0.04038	5.00*	0	0.00

Table B.7: Model 3, $n = 50$ (* indicates that the maximum of the h grid was used)

			SimIIA NW				SimIIB Boosted NW					
σ	$f(x)$	Kernel	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$	h	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$	h	T	$Gain(\%)$
0.1	U(-2,2)	N(0,1)	0.00924	0.00183	0.00741	0.09	0.00734	0.00086	0.00648	0.19	6	20.59
		Triangular	0.00932	0.00160	0.00773	0.21	0.00770	0.00093	0.00677	0.37	3	17.40
	N(0,1)	N(0,1)	0.02290	0.00173	0.02117	0.08	0.02290	0.00173	0.02117	0.08	0	0.00
		Triangular	0.02106	0.00152	0.01954	0.19	0.02106	0.00152	0.01954	0.19	0	0.00
0.3	U(-2,2)	N(0,1)	0.03969	0.00983	0.02986	0.16	0.03969	0.00983	0.02986	0.16	0	0.00
		Triangular	0.03981	0.01052	0.02929	0.39	0.03981	0.01052	0.02929	0.39	0	0.00
	N(0,1)	N(0,1)	0.13943	0.06297	0.07646	0.35	0.13943	0.06297	0.07646	0.35	0	0.00
		Triangular	0.13933	0.01736	0.12197	0.43	0.13933	0.01736	0.12197	0.43	0	0.00
0.5	U(-2,2)	N(0,1)	0.07846	0.02290	0.05555	0.23	0.07846	0.02290	0.05555	0.23	0	0.00
		Triangular	0.07911	0.02313	0.05598	0.54	0.07911	0.02313	0.05598	0.54	0	0.00
	N(0,1)	N(0,1)	0.18353	0.16744	0.01609	5.00*	0.18353	0.16744	0.01609	5.00*	0	0.00
		Triangular	0.18159	0.16197	0.01962	3.85	0.18159	0.16197	0.01962	3.85	0	0.00

Table B.8: Model 3, $n = 100$ (* indicates that the maximum of the h grid was used)

			SimIIA NW				SimIIB Boosted NW					
σ	$f(x)$	Kernel	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$	h	MISE	$\int \text{Bias}^2 dx$	$\int \text{Var} dx$	h	T	$Gain(\%)$
0.1	U(-2,2)	N(0,1)	0.00453	0.00098	0.00355	0.08	0.00348	0.00045	0.00303	0.18	6	23.19
		Triangular	0.00450	0.00073	0.00377	0.18	0.00362	0.00039	0.00323	0.34	3	19.54
	N(0,1)	N(0,1)	0.01802	0.00137	0.01665	0.08	0.01802	0.00137	0.01665	0.08	0	0.00
		Triangular	0.01673	0.00123	0.01550	0.19	0.01673	0.00123	0.01550	0.19	0	0.00
0.3	U(-2,2)	N(0,1)	0.02143	0.00484	0.01659	0.13	0.02110	0.00413	0.01698	0.18	1	1.54
		Triangular	0.02131	0.00478	0.01653	0.31	0.02131	0.00478	0.01653	0.31	0	0.00
	N(0,1)	N(0,1)	0.10070	0.04062	0.06009	0.28	0.10070	0.04062	0.06009	0.28	0	0.00
		Triangular	0.10767	0.03247	0.07520	0.58	0.10767	0.03247	0.07520	0.58	0	0.00
0.5	U(-2,2)	N(0,1)	0.04459	0.01221	0.03238	0.18	0.04459	0.01221	0.03238	0.18	0	0.00
		Triangular	0.04473	0.01187	0.03287	0.42	0.04473	0.01187	0.03287	0.42	0	0.00
	N(0,1)	N(0,1)	0.16210	0.10052	0.06158	0.51	0.16210	0.10052	0.06158	0.51	0	0.00
		Triangular	0.17580	0.11026	0.06555	1.25	0.17580	0.11026	0.06555	1.25	0	0.00

Table B.9: Model 3, $n = 200$

Appendix C

Graphs and figures

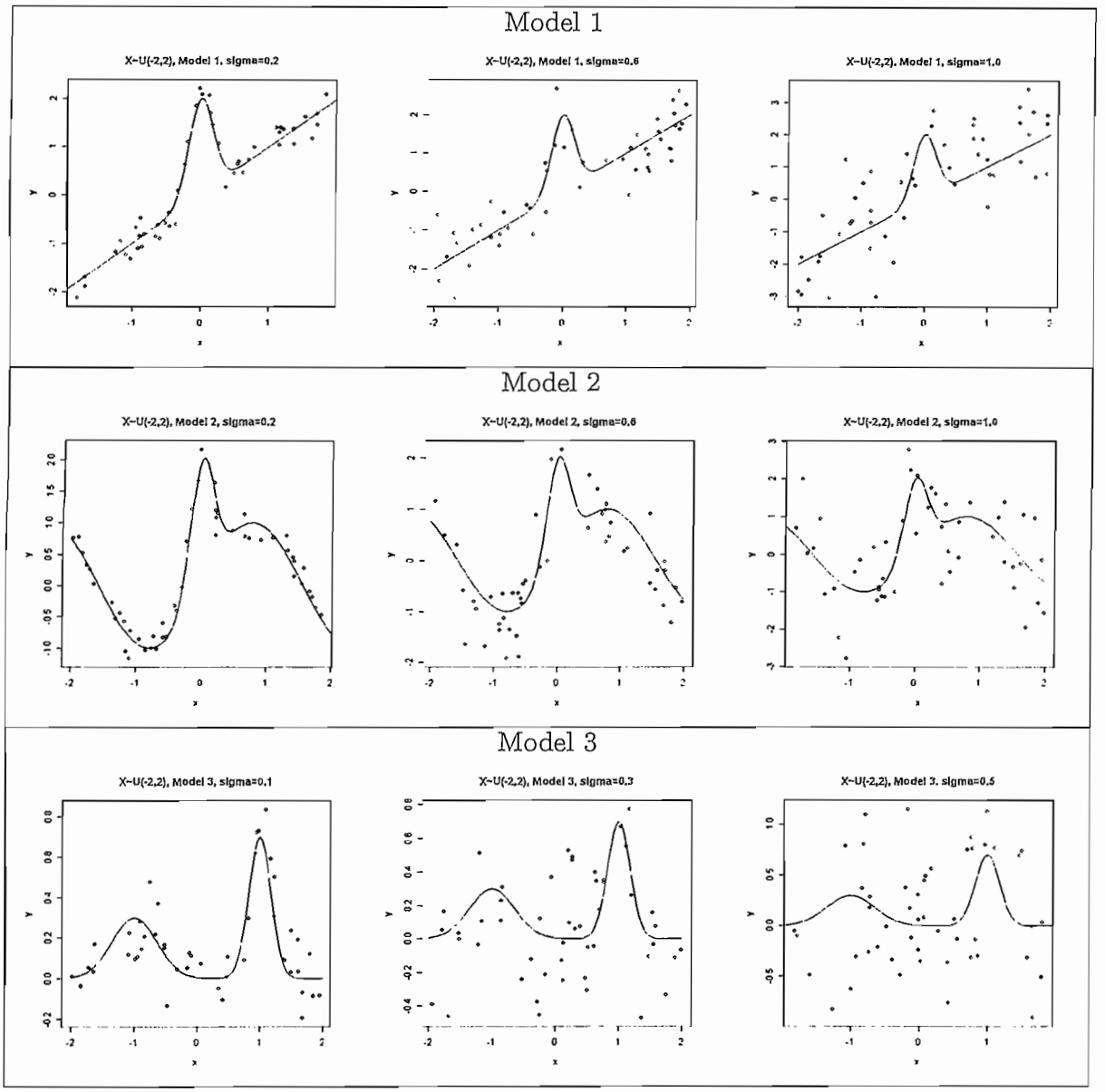


Figure C.1: Examples of a single sample of size $n = 50$ drawn from the different simulation setup scenarios explored in this study, where $X \sim U(-2, 2)$.

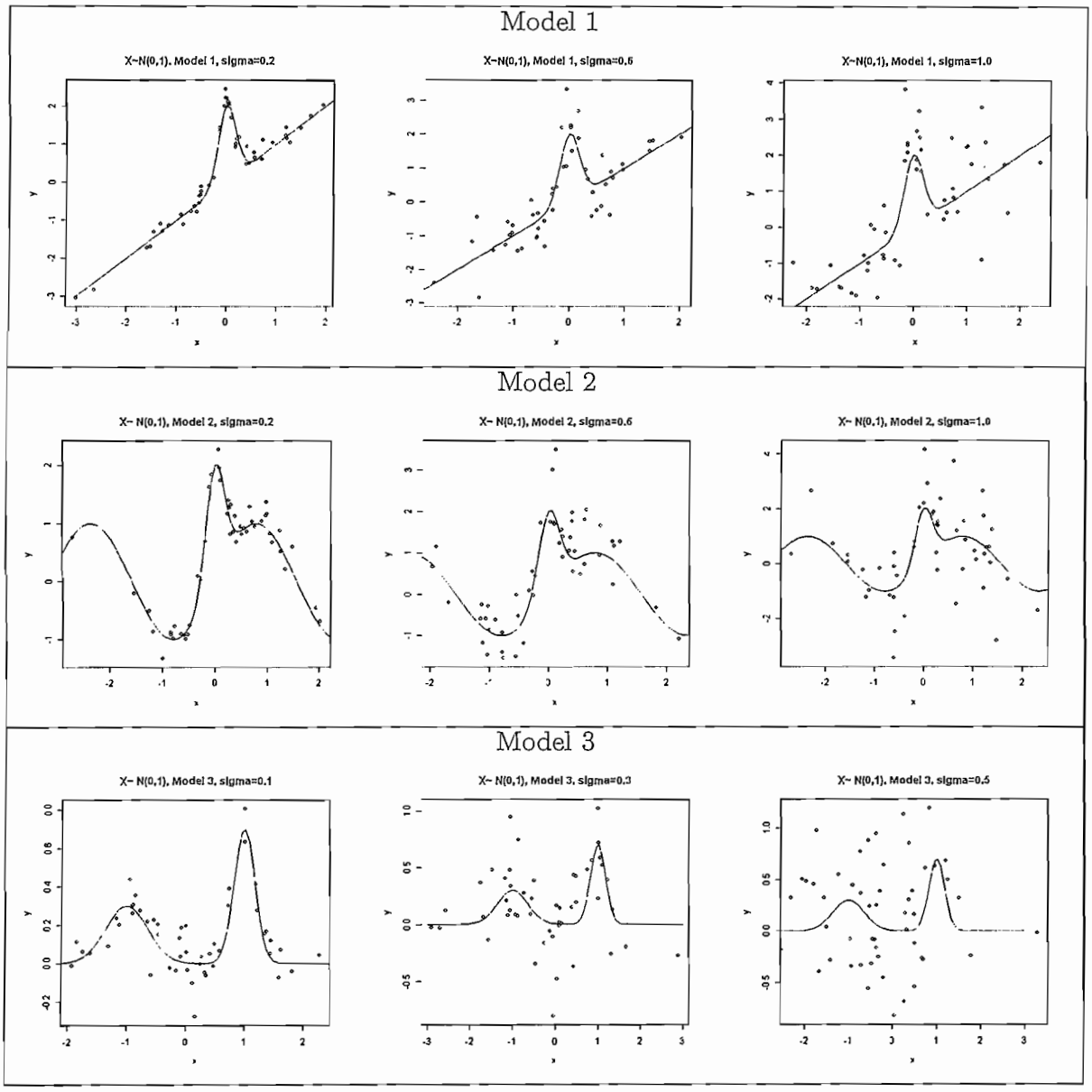


Figure C.2: Examples of a single sample of size $n = 50$ drawn from the different simulation setup scenarios explored in this study, where $X \sim N(0, 1)$.

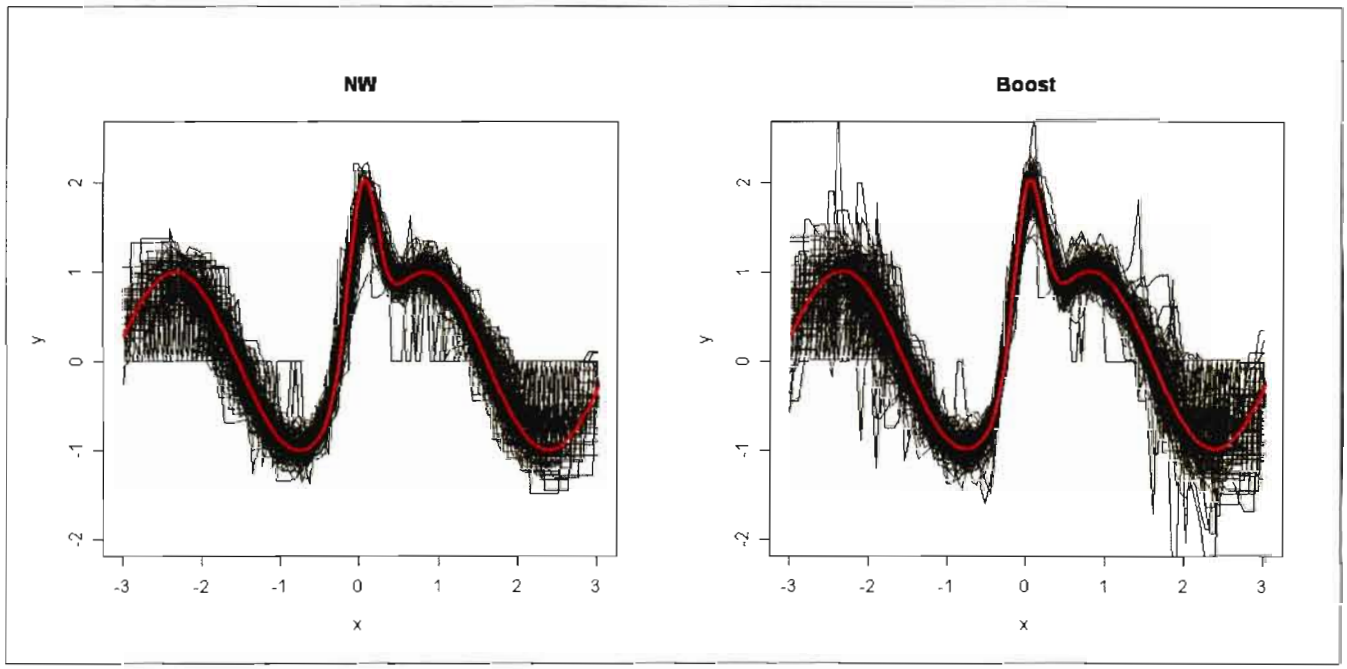


Figure C.3: The effect of NW and Boost for the $n = 50$, $X \sim N(0, 1)$, triangular kernel, $\sigma = 0.2$ scenario. The black lines represent the estimates for each of the 200 Monte Carlo simulations. The red line indicates $m(x)$.

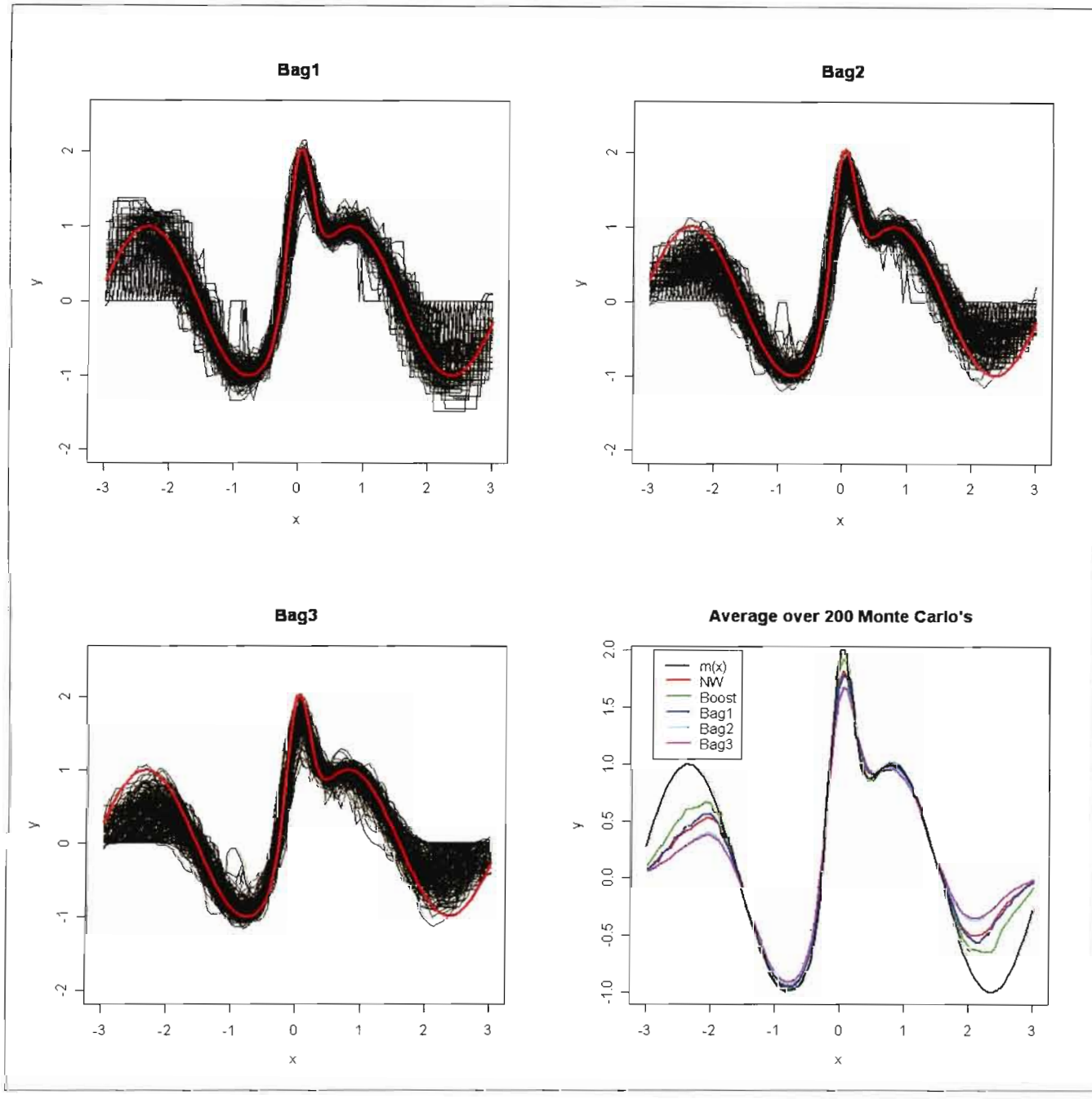


Figure C.4: *Top left, top right and bottom left:* The effect of Bag1, Bag2 and Bag3 for the $n = 50$, $X \sim N(0, 1)$, triangular kernel, $\sigma = 0.2$ scenario. The black lines represent the estimates for each of the 200 Monte Carlo simulations. The red line indicates $m(x)$. *Bottom right:* The average estimates over the 200 Monte Carlo simulations for NW, Boost, Bag1, Bag2 and Bag3, as indicated in the legend.

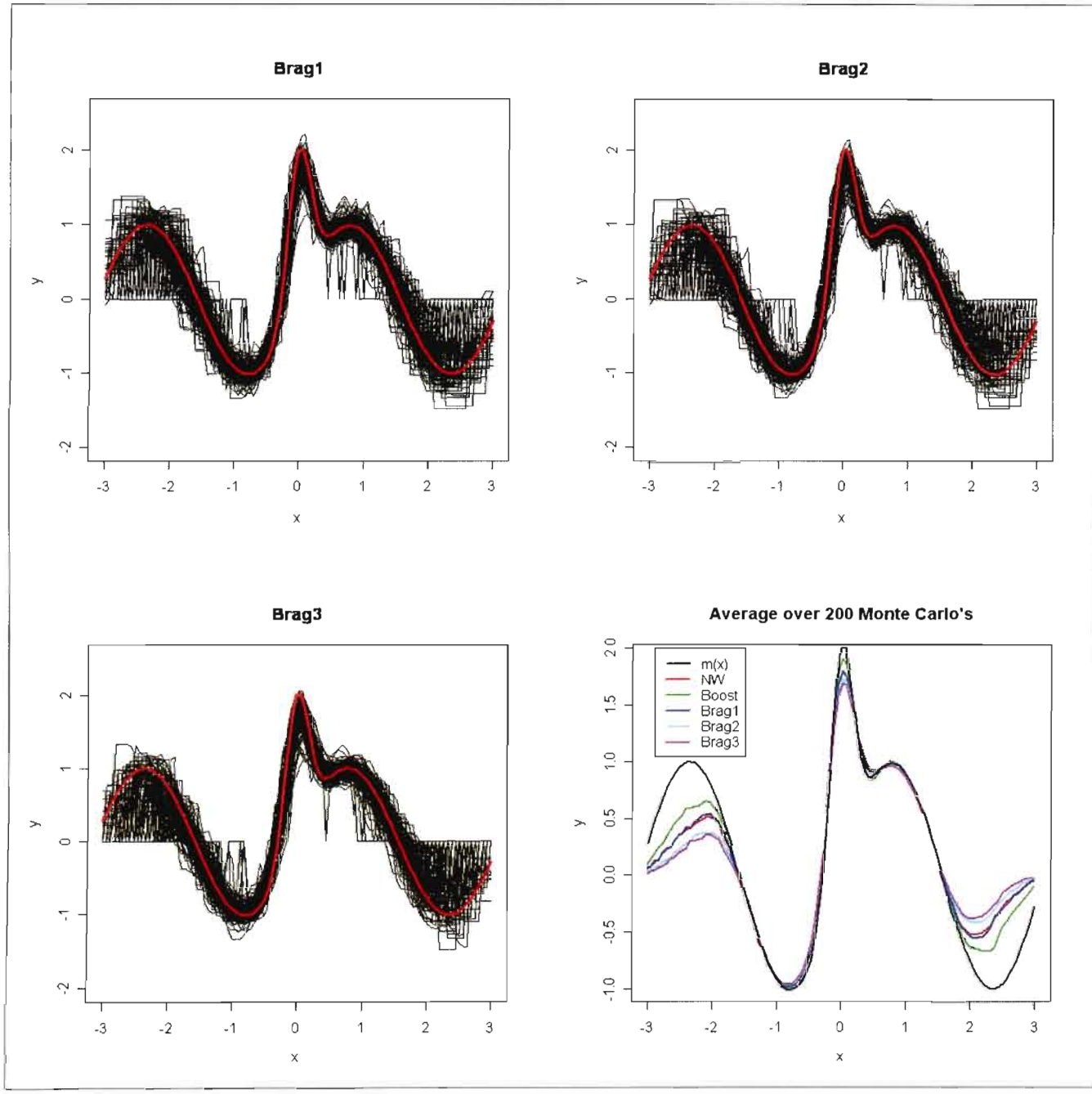


Figure C.5: *Top left, top right and bottom left:* The effect of Brag1, Brag2 and Brag3 for the $n = 50$, $X \sim N(0, 1)$, triangular kernel, $\sigma = 0.2$ scenario. The black lines represent the estimates for each of the 200 Monte Carlo simulations. The red line indicates $m(x)$. *Bottom right:* The average estimates over the 200 Monte Carlo simulations for NW, Boost, Brag1, Brag2 and Brag3, as indicated in the legend.

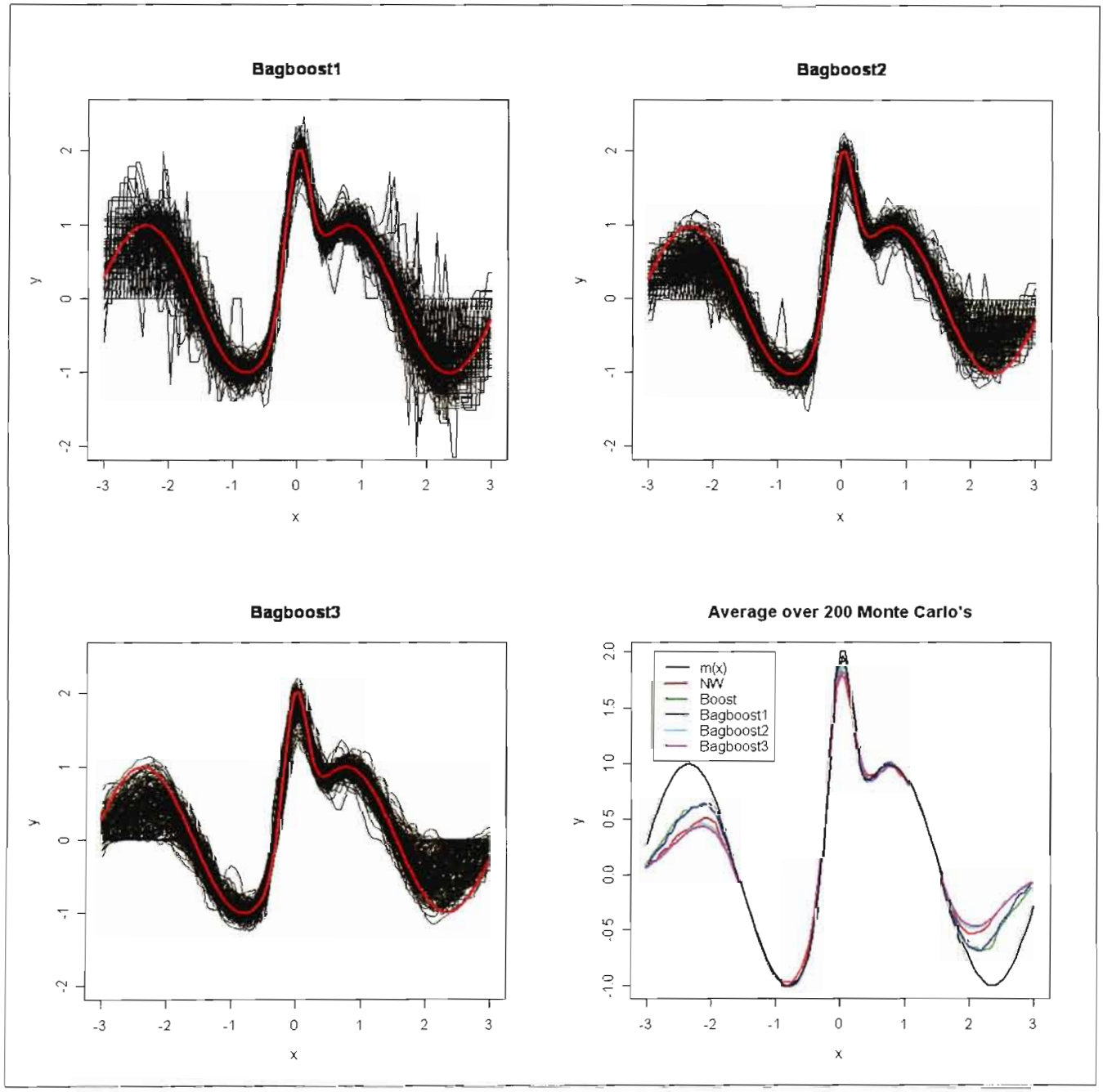


Figure C.6: *Top left, top right and bottom left:* The effect of Bagboost1, Bagboost2 and Bagboost3 for the $n = 50$, $X \sim N(0, 1)$, triangular kernel, $\sigma = 0.2$ scenario. The black lines represent the estimates for each of the 200 Monte Carlo simulations. The red line indicates $m(x)$. *Bottom right:* The average estimates over the 200 Monte Carlo simulations for NW, Boost, Bagboost1, Bagboost2 and Bagboost3, as indicated in the legend.

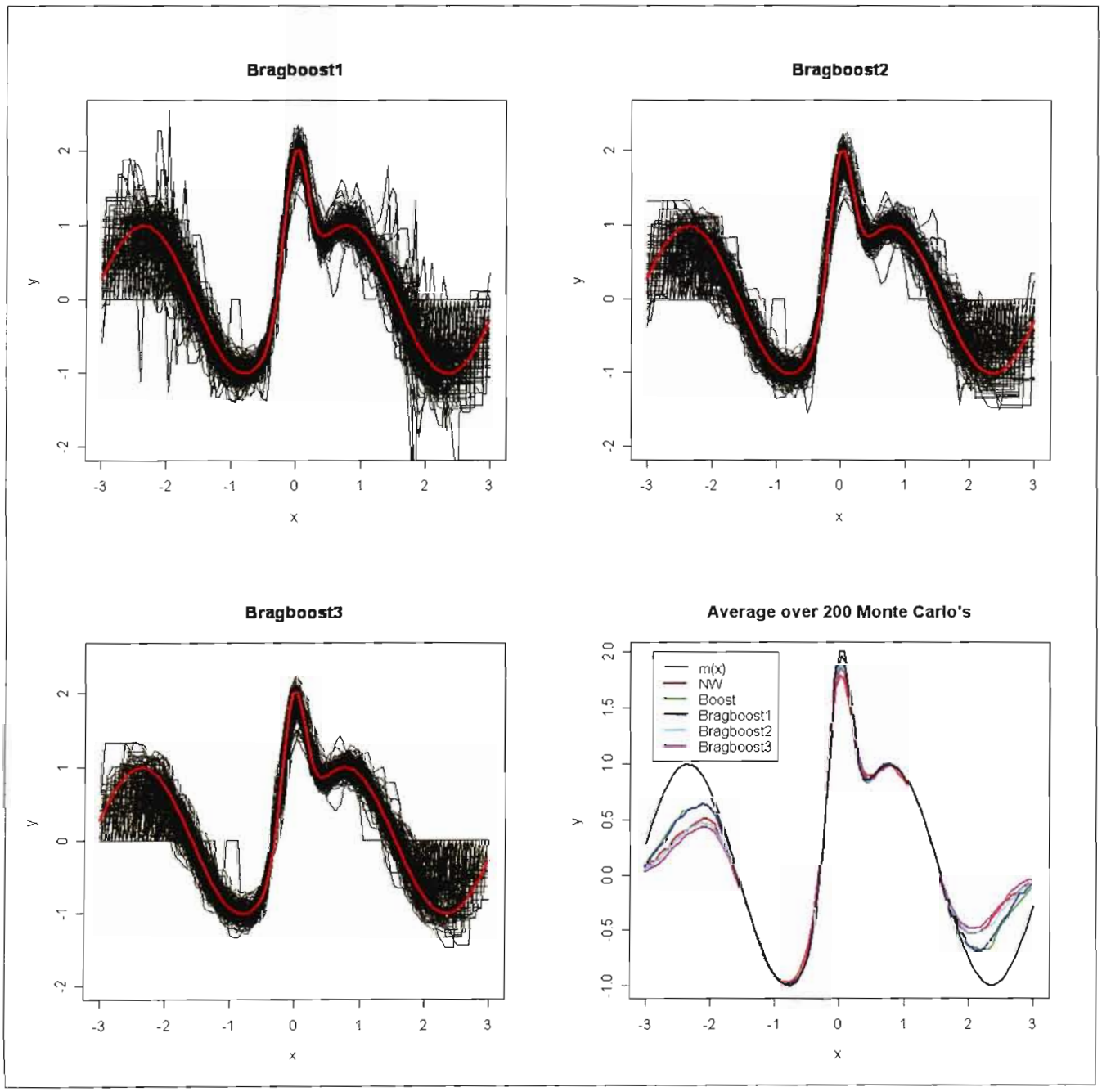


Figure C.7: *Top left, top right and bottom left:* The effect of Bragboost1, Bragboost2 and Bragboost3 for the $n = 50$, $X \sim N(0, 1)$, triangular kernel, $\sigma = 0.2$ scenario. The black lines represent the estimates for each of the 200 Monte Carlo simulations. The red line indicates $m(x)$. *Bottom right:* The average estimates over the 200 Monte Carlo simulations for NW, Boost, Bragboost1, Bragboost2 and Bragboost3, as indicated in the legend.

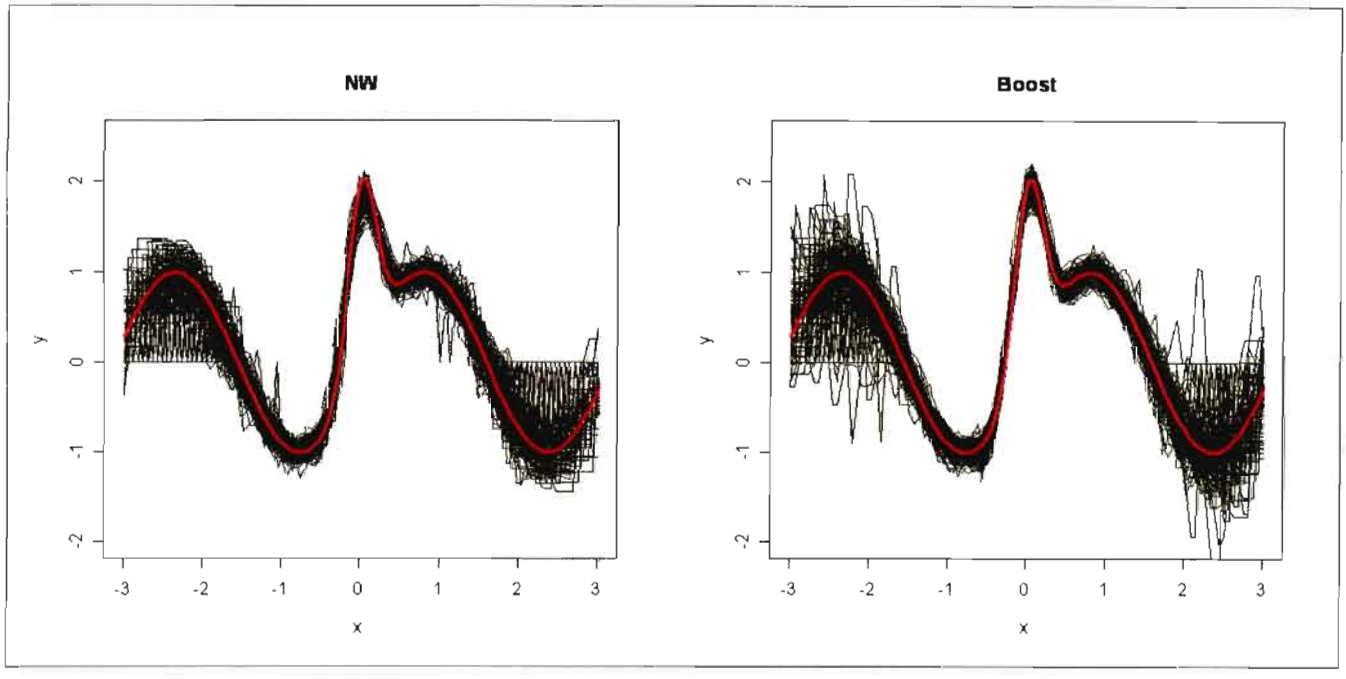


Figure C.8: The effect of NW and Boost for the $n = 100$, $X \sim N(0, 1)$, triangular kernel, $\sigma = 0.2$ scenario. The black lines represent the estimates for each of the 200 Monte Carlo simulations. The red line indicates $m(x)$.

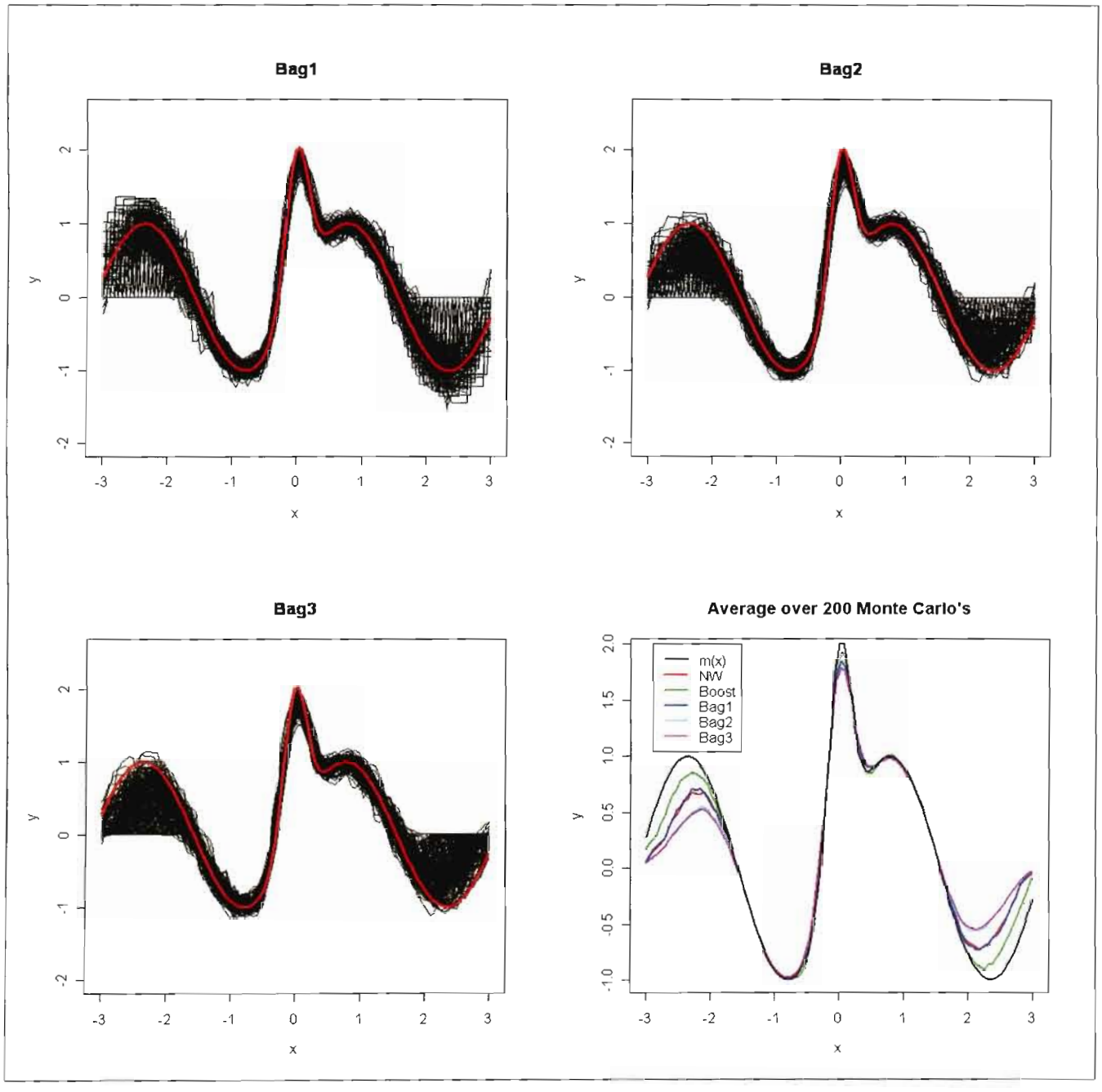


Figure C.9: *Top left, top right and bottom left:* The effect of Bag1, Bag2 and Bag3 for the $n = 100$, $X \sim N(0,1)$, triangular kernel, $\sigma = 0.2$ scenario. The black lines represent the estimates for each of the 200 Monte Carlo simulations. The red line indicates $m(x)$. *Bottom right:* The average estimates over the 200 Monte Carlo simulations for NW, Boost, Bag1, Bag2 and Bag3, as indicated in the legend.

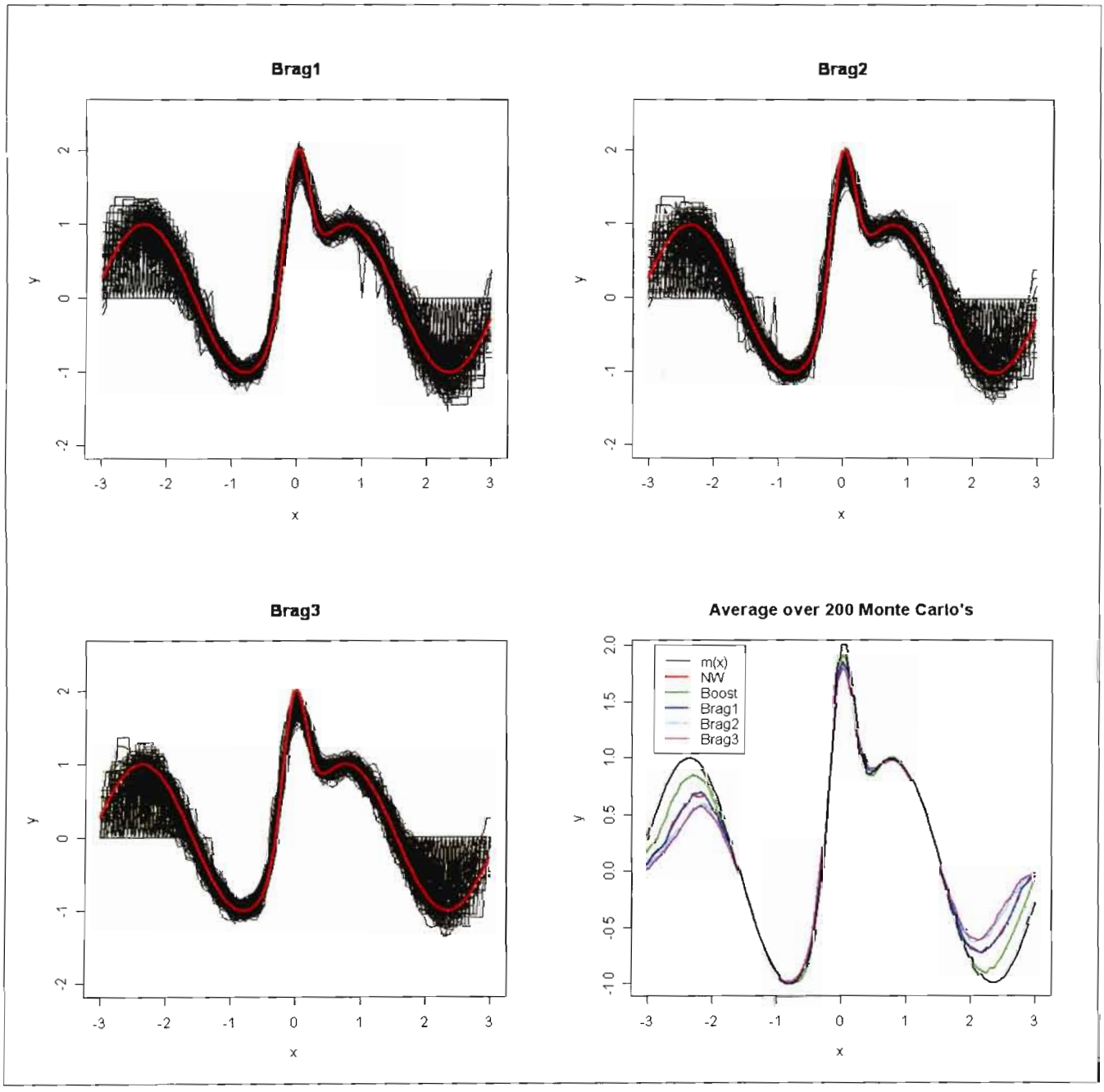


Figure C.10: *Top left, top right and bottom left:* The effect of Brag1, Brag2 and Brag3 for the $n = 100$, $X \sim N(0, 1)$, triangular kernel, $\sigma = 0.2$ scenario. The black lines represent the estimates for each of the 200 Monte Carlo simulations. The red line indicates $m(x)$. *Bottom right:* The average estimates over the 200 Monte Carlo simulations for NW, Boost, Brag1, Brag2 and Brag3, as indicated in the legend.

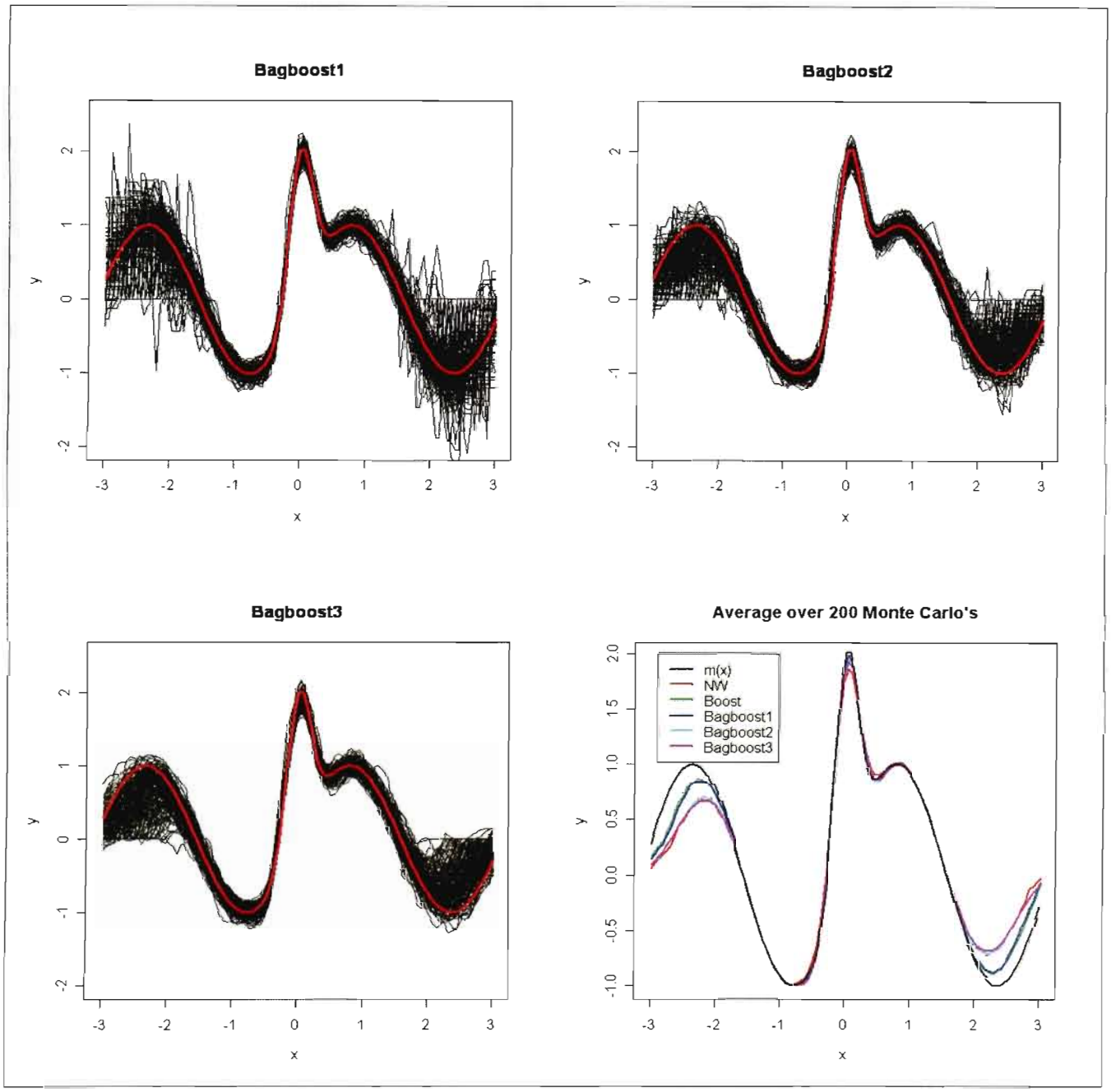


Figure C.11: *Top left, top right and bottom left:* The effect of Bagboost1, Bagboost2 and Bagboost3 for the $n = 100$, $X \sim N(0, 1)$, triangular kernel, $\sigma = 0.2$ scenario. The black lines represent the estimates for each of the 200 Monte Carlo simulations. The red line indicates $m(x)$. *Bottom right:* The average estimates over the 200 Monte Carlo simulations for NW, Boost, Bagboost1, Bagboost2 and Bagboost3, as indicated in the legend.

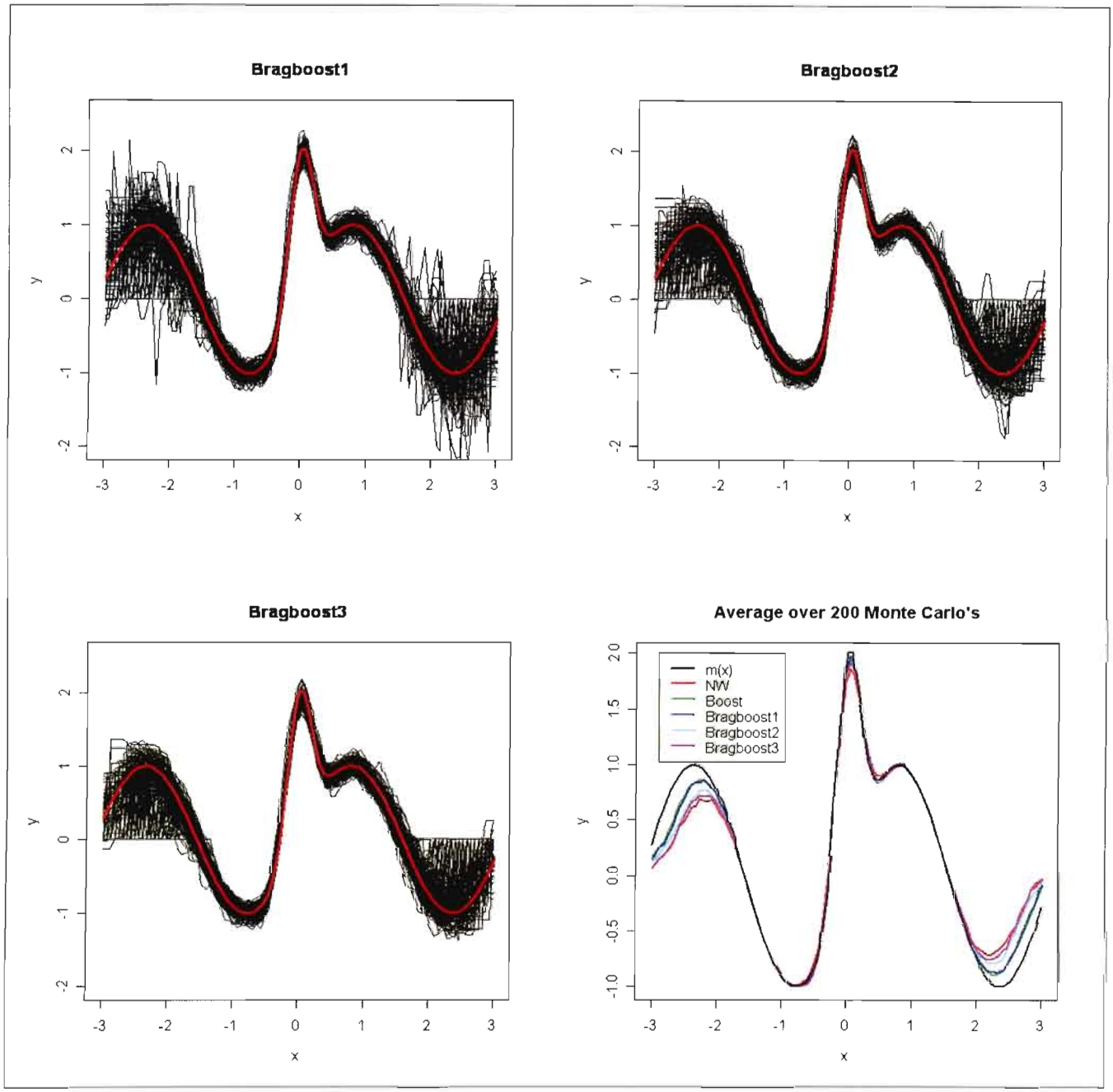


Figure C.12: *Top left, top right and bottom left:* The effect of Bragboost1, Bragboost2 and Bragboost3 for the $n = 100$, $X \sim N(0,1)$, triangular kernel, $\sigma = 0.2$ scenario. The black lines represent the estimates for each of the 200 Monte Carlo simulations. The red line indicates $m(x)$. *Bottom right:* The average estimates over the 200 Monte Carlo simulations for NW, Boost, Bragboost1, Bragboost2 and Bragboost3, as indicated in the legend.

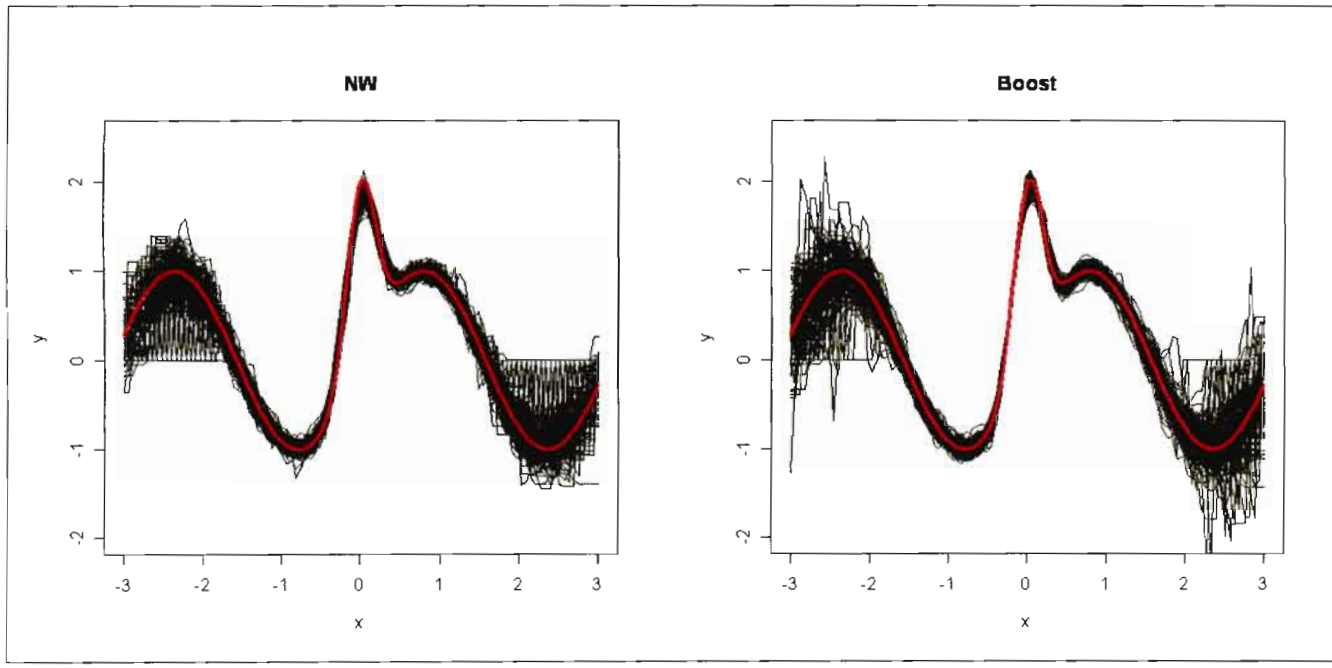


Figure C.13: The effect of NW and Boost for the $n = 200$, $X \sim N(0, 1)$, triangular kernel, $\sigma = 0.2$ scenario. The black lines represent the estimates for each of the 200 Monte Carlo simulations. The red line indicates $m(x)$.

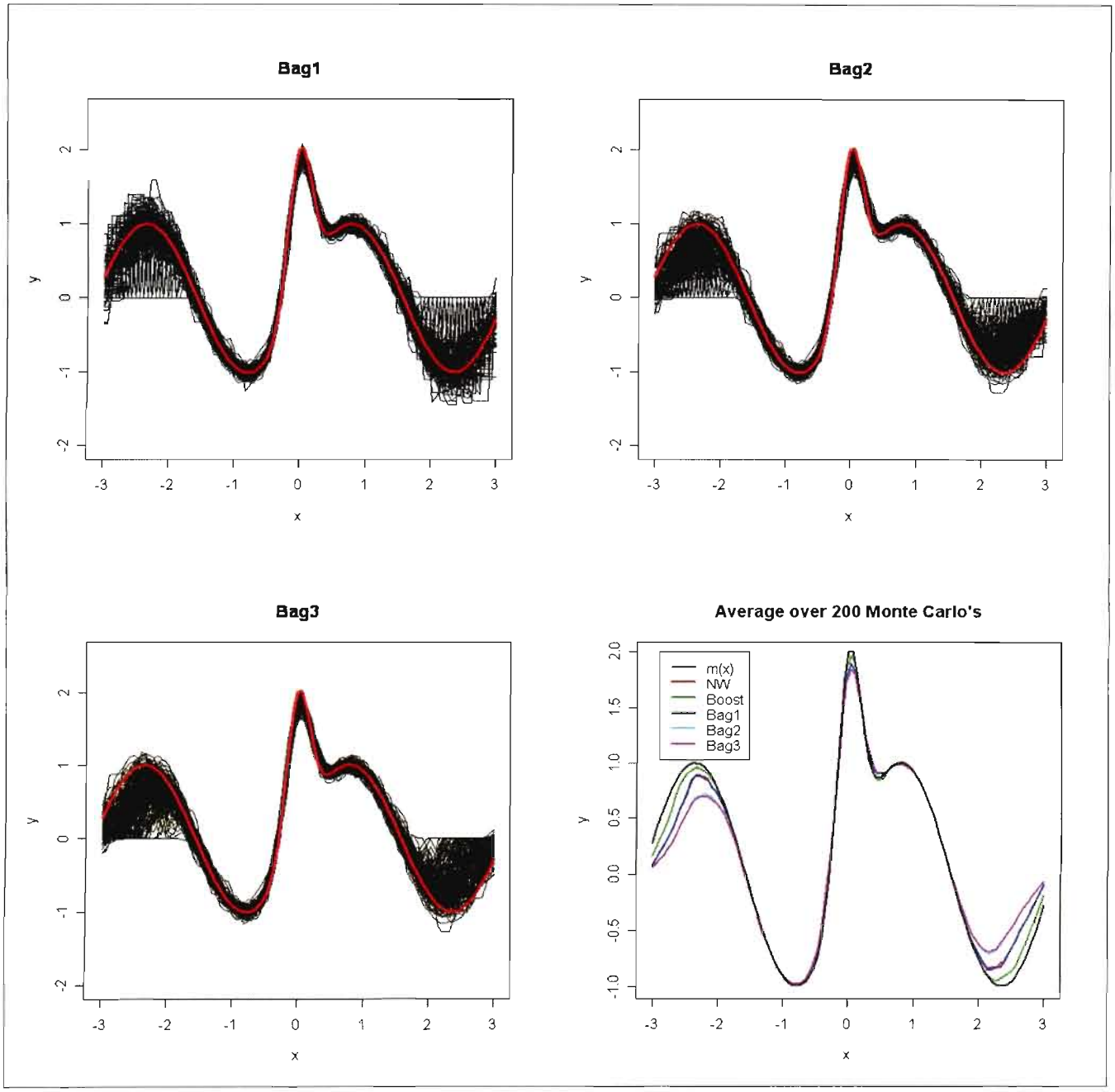


Figure C.14: *Top left, top right and bottom left:* The effect of Bag1, Bag2 and Bag3 for the $n = 200$, $X \sim N(0,1)$, triangular kernel, $\sigma = 0.2$ scenario. The black lines represent the estimates for each of the 200 Monte Carlo simulations. The red line indicates $m(x)$. *Bottom right:* The average estimates over the 200 Monte Carlo simulations for NW, Boost, Bag1, Bag2 and Bag3, as indicated in the legend.

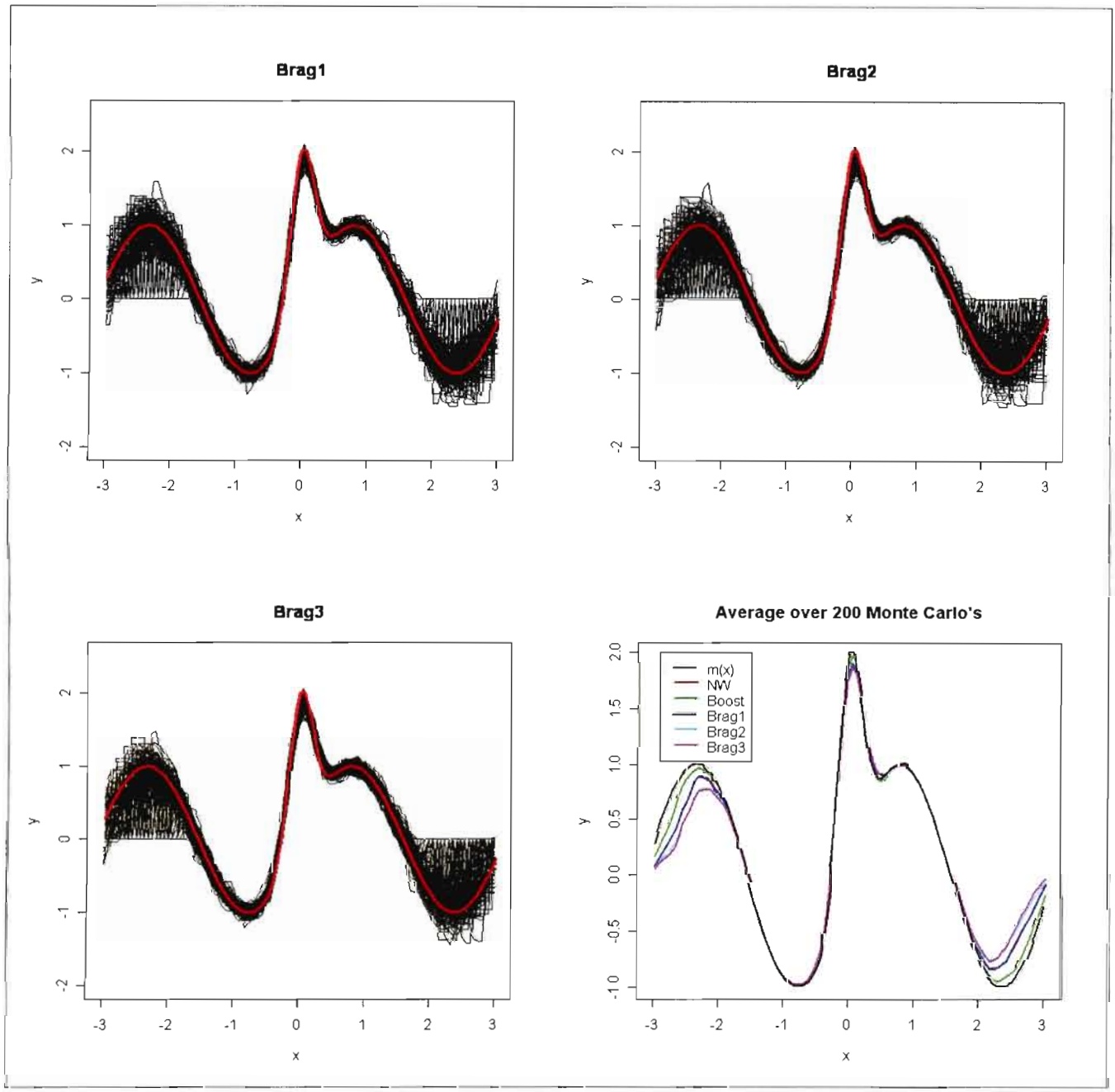


Figure C.15: *Top left, top right and bottom left:* The effect of Brag1, Brag2 and Brag3 for the $n = 200$, $X \sim N(0, 1)$, triangular kernel, $\sigma = 0.2$ scenario. The black lines represent the estimates for each of the 200 Monte Carlo simulations. The red line indicates $m(x)$. *Bottom right:* The average estimates over the 200 Monte Carlo simulations for NW, Boost, Brag1, Brag2 and Brag3, as indicated in the legend.

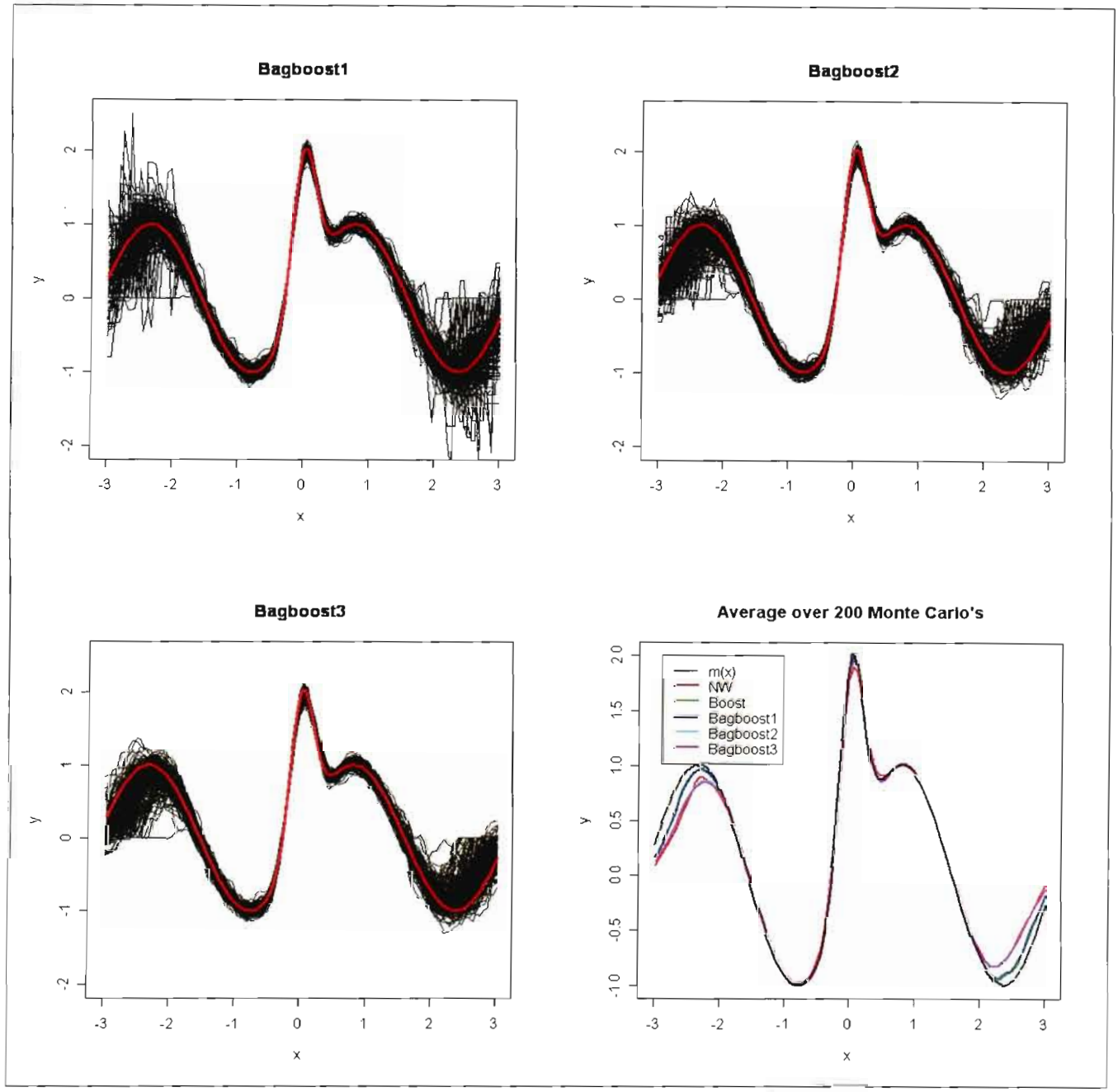


Figure C.16: *Top left, top right and bottom left:* The effect of Bagboost1, Bagboost2 and Bagboost3 for the $n = 200$, $X \sim N(0, 1)$, triangular kernel, $\sigma = 0.2$ scenario. The black lines represent the estimates for each of the 200 Monte Carlo simulations. The red line indicates $m(x)$. *Bottom right:* The average estimates over the 200 Monte Carlo simulations for NW, Boost, Bagboost1, Bagboost2 and Bagboost3, as indicated in the legend.

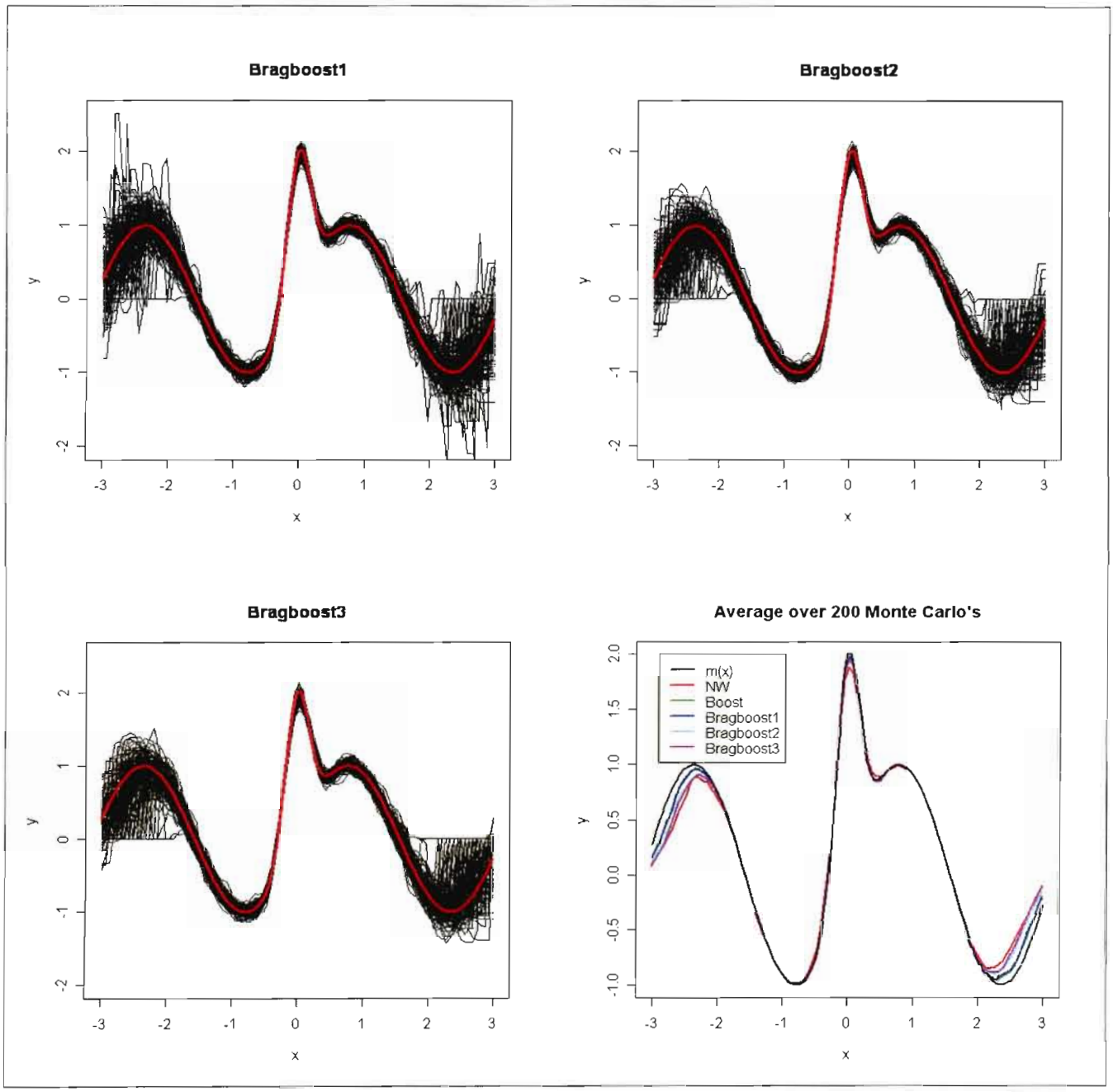


Figure C.17: *Top left, top right and bottom left:* The effect of Bragboost1, Bragboost2 and Bragboost3 for the $n = 200$, $X \sim N(0, 1)$, triangular kernel, $\sigma = 0.2$ scenario. The black lines represent the estimates for each of the 200 Monte Carlo simulations. The red line indicates $m(x)$. *Bottom right:* The average estimates over the 200 Monte Carlo simulations for NW, Boost, Bragboost1, Bragboost2 and Bragboost3, as indicated in the legend.

Bibliography

Akaike, H. (1970). Statistical predictor information, *Annals of the institute of statistical mathematics* **22**: 203–217.

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

Allen, D. M. (1974). The relationship between variable and data augmentation and a method for prediction, *Technometrics* **16**: 125–127.

Altman, N. S. (1992). An introduction to kernel and nearest-neighbor nonparametric regression, *The American Statistician* **46**(3): 175–185.

Barutçuoğlu, Z. and Alpaydin, E. (2003). A comparison of model aggregation methods for regression, *Proceedings of the joint international conference of artificial neural networks and neural information procession*, pp. 76–83.

Benedetti, J. K. (1977). On the nonparametric estimation of regression functions, *Journal of the Royal Statistical Society, Series B* **39**: 248–253.

Borra, S. and Di Ciaccio, A. (2001). Performance evaluation of bagging and boosting in nonparametric regression, *Metron* **59**: 141–156.

Breiman, L. (1996a). Bagging predictors, *Machine Learning* **24**: 123–140.

Breiman, L. (1996b). Heuristics of instability and stabilization in model selection, *The Annals of Statistics* **24**(6): 2350–2383.

Breiman, L. (1996c). Out-of-bag estimation, *Technical report*, Statistics Department, University of California.

Breiman, L. (1999). Prediction games and arcing algorithms, *Neural Computation* **11**(7): 1493–1517.

Bühlmann, P. (2003). Bagging, subagging and bragging for improving some prediction algorithms, *in* M. G. Akritas and D. N. Politis (eds), *Recent advances and trends in nonparametric statistics*, Elsevier, Amsterdam, pp. 9–34.

Bühlmann, P. (2004). Bagging, boosting and ensemble methods, *in* J. E. Gentle, W. Härdle and Y. Mori (eds), *Handbook of Computational Statistics: Concepts and Methods*, Springer, Berlin, pp. 877–907.

Bühlmann, P. (2006). Boosting for high-dimensional linear models, *The Annals of Statistics* **34**(2): 559–583.

Bühlmann, P. and Hothorn, T. (2007). Boosting algorithms: regularization, prediction and model fitting, *Statistical Science* **22**(4): 477–505.

Bühlmann, P. and Yu, B. (2002). Analyzing bagging, *The Annals of Statistics* **30**(4): 927–961.

Bühlmann, P. and Yu, B. (2003). Boosting with the l_2 loss: Regression and classification, *Journal of the American Statistical Association* **98**(462): 324–339.

Bühlmann, P. and Yu, B. (2006). Sparse boosting, *Journal of Machine Learning Research* **7**: 1001–1024.

Buja, A. and Stuetzle, W. (2006). Observations on bagging, *Statistica Sinica* **16**: 323–351.

Buja, A., Mease, D. and Wyner, A. J. (2007). Comment: Boosting algorithms: Regularization, prediction and model fitting, *Statistical Science* **22**(4): 506–512.

Cheng, K. F. and Lin, P. (1981). Nonparametric estimation of a regression function, *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete* **57**: 223–233.

Cheng, P. E. and Cheng, K. F. (1987). Robust nonparametric estimation of a regression function, *Sankhya: The Indian Journal of Statistics, Series B* **49**(1): 9–22.

Cheng, P. E. and Cheng, K. F. (1990). Asymptotic normality for robust r-estimators of regression function, *Journal of Statistical Planning and Inference* **24**(2): 137–149.

Chu, C.-K. and Marron, J. S. (1991). Choosing a kernel regression estimator, *Statistical Science* **6**(4): 404–419.

Cox, D. D. (1983). Asymptotics for m-type smoothing splines, *The Annals of Statistics* **11**: 530–551.

Craven, P. and Wahba, G. (1979). Smoothing noisy data with spline functions, *Numerische Mathematik* **31**: 377–403.

Croux, C., Joossens, K. and Lemmens, A. (2007). Trimmed bagging, *Computational Statistics and Data Analysis* **52**: 362–368.

Davison, A. C. and Hinkley, D. V. (1997). *Bootstrap methods and their applications*, Cambridge University Press, Cambridge.

Drucker, H., Schapire, R. E. and Simard, P. Y. (1993). Boosting performance in nearal networks, *International journal of pattern recognition and artificial intelligence* **7**(4): 705–719.

Duffy, N. and Helmbold, D. (2000). Leveraging for regression, *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*, Morgan Kaufmann, San Francisco, pp. 208–219.

Efron, B. (1979). Bootstrap methods: another look at the jackknife, *The Annals of Statistics* **7**(1): 1–26.

Efron, B. and Tibshirani, R. J. (1993). *An Introduction to the Bootstrap*, Vol. 57 of *Monographs on Statistics and Applied Probability*, Chapman & Hall, New York.

Epanechnikov, V. A. (1969). Nonparametric estimates of a multivariate probability density., *Theory of Probability and Its Applications* **14**: 153–158.

Eubank, R. L. (1988). *Spline Smoothing and Nonparametric Regression*, Vol. 90 of *Statistics: Textbooks and Monographs*, Marcel Dekker, New York.

Fan, J. (1993). Local linear regression smoothers and their minimax efficiencies, *The Annals of Statistics* **21**(1): 196–216.

Fan, J. and Gijbels, I. (1992). Variable bandwidth and local linear regression smoothers, *The Annals of Statistics* **20**(4): 2008–2036.

Fan, J. and Gijbels, I. (1996). *Local Polynomial Modelling and Its Applications*, Vol. 66 of *Monographs on Statistics and Applied Probability*, Chapman and Hall, London.

Freund, Y. (1995). Boosting a weak learning algorithm by majority, *Information and Computation* **121**: 256–285.

Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting, *Journal of Computer and System Sciences* **55**(1): 119–139.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine, *The Annals of Statistics* **29**(5): 1189–1232.

Friedman, J. H. and Hall, P. (2007). On bagging and nonlinear estimation, *Journal of Statistical Planning and Inference* **137**: 669–683.

Friedman, J. H., Hastie, T. J. and Tibshirani, R. J. (2000). Additive logistic regression: A statistical view of boosting, *The Annals of Statistics* **28**(2): 337–374.

Gasser, T. and Müller, H.-G. (1979). Kernel estimation of regression functions, *in* T. Gasser and M. Rosenblatt (eds), *Smoothing Techniques for Curve Estimation*, Vol. 757 of *Lecture Notes in Mathematics*, Springer-Verlag, Heidelberg, pp. 23–68.

Gasser, T. and Müller, H.-G. (1984). Estimating regression functions and their derivatives by the kernel method, *Scandinavian Journal of Statistics* **11**(3): 171–185.

Gasser, T., Müller, H.-G. and Mammitzsch, V. (1985). Kernels for nonparametric curve estimation., *Journal of the Royal Statistical Society, Series B* **47**(2): 238–252.

Hall, P. and Robinson, A. P. (2009). Reducing variability of crossvalidation for smoothing-parameter choice, *Biometrika* **96**(1): 175–186.

Härdle, W. (1987). Algorithm as 222: Resistant smoothing using the fast fourier transform, *Journal of the Royal Statistical Society. Series C (Applied Statistics)* **36**(1): 104–111.

Härdle, W. (1990). *Applied Nonparametric Regression*, Cambridge University Press, Cambridge.

Härdle, W. and Marron, J. S. (1985a). Asymptotic nonequivalence of some bandwidth selectors in nonparametric regression, *Biometrika* **72**(2): 481–484.

Härdle, W. and Marron, J. S. (1985b). Optimal bandwidth selection in nonparametric regression function estimation, *The Annals of Statistics* **13**(4): 1465–1481.

Härdle, W., Hall, P. and Marron, J. S. (1988a). How far are automatically chosen regression smoothing parameters from their optimum?, *Journal of the American Statistical Association* **83**(401): 86–95.

Härdle, W., Janssen, P. and Serfling, R. (1988b). Strong uniform consistency rates for estimators of conditional functionals, *The Annals of Statistics* **16**: 1428–1449.

Huber, P. J. (1979). Robust smoothing, *in* R. L. Launer and G. N. Wilkinson (eds), *Robustness in statistics*, Academic Press, New York.

Huber, P. J. (1981). *Robust Statistics*, Wiley, New York.

Jacod, J. and Protter, P. (2004). *Probability essentials*, 2nd edn, Springer-Verlag, Berlin.

Kahaner, D., Moler, C. and Nash, S. (1989). *Numerical Methods and Software.*, Prentice Hall, Englewood Cliffs, NJ.

Kearns, M. and Valiant, L. G. (1994). Cryptographic limitations on learning boolean formulae and finite automata, *Journal of the association for computing machinery.* **41**(1): 67–95.

Kutner, M. H., Li, W., Nachtsheim, C. J. and Neter, J. (2005). *Applied Linear Statistical Models*, McGraw-Hill, New York.

Loftsgaarden, D. O. and Quesenberry, C. P. (1965). A nonparametric estimate of a multivariate density function, *The Annals of Mathematical Statistics* **36**(3): 1049–1051.

Lugosi, G. and Vayatis, N. (2004). On the bayes-risk consistency of regularized boosting methods, *The Annals of Statistics* **32**(1): 30–55.

Lutz, R. W. and Bühlmann, P. (2006). Conjugate direction boosting, *Journal of computational and graphical statistics* **15**(2): 287–311.

Lutz, R. W., Kalisch, M. and Bühlmann, P. (2008). Robustified l-2 boosting, *Computational Statistics and Data Analysis* **52**: 3331–3341.

Mack, Y. P. and Müller, H.-G. (1989). Convolution type estimators for nonparametric regression, *Statistics and Probability Letters* **7**: 229–239.

Mallows, C. L. (1980). Some theory of nonlinear smoothers, *The Annals of Statistics* **8**(4): 695–715.

Marron, J. S. and Nolan, D. (1988). Canonical kernels for density estimation, *Statistics and Probability Letters* **7**(3): 195–199.

Marzio, M. D. and Taylor, C. C. (2008). On boosting kernel regression, *Journal of statistical planning and inference* **138**: 2483–2498.

Meir, R. and Rätsch, G. (2003). An introduction to boosting and leveraging, *in* S. Mendelson and A. Smola (eds), *Advanced Lectures on Machine Learning*, Vol. 2600 of *Lecture Notes in Artificial Intelligence*, Springer, New York, pp. 119–183.

Müller, H.-G. (1988). *Nonparametric Regression Analysis of Longitudinal Data*, Vol. 46 of *Lecture Notes in Statistics*, Springer-Verlag, Berlin.

Müller, H.-G. (1993). Local regression: automatic kernel carpentry (comment), *Statistical Science* **8**(2): 134–139.

Nadaraya, E. A. (1964). On estimating regression, *Theory of Probability and Its Applications* **9**: 141–142.

Owen, A. B. (1987). Nonparametric conditional estimation, *Technical report*, Stanford University, Stanford, California.

Parzen, E. (1962). On estimation of a probability density function and mode., *The Annals of Mathematical Statistics* **33**(3): 1065–1076.

Peterson, M. L., Molinaro, A. M., Sinisi, S. E. and Van der Laan, M. J. (2007). Cross-validated bagged learning, *Journal of Multivariate Analysis* **98**: 1693–1704.

Pino-Mejías, R., Jiménez-Gamero, M.-D., Cubiles-de-la Vega, M.-D. and Pascual-Acosta, A. (2008). Reduced bootstrap aggregation of learning algorithms, *Pattern Recognition Letters* **29**: 265–271.

Priestley, M. B. and Chao, M. T. (1972). Nonparametric function fitting, *Journal of the Royal Statistical Society, Series B* **34**: 385–392.

Rätsch, G., Demiriz, A. and Bennett, K. (2002). Sparse regression ensembles in infinite and finite hypothesis spaces, *Machine Learning* **48**: 189–218.

Rice, J. A. (1984a). Bandwidth choice for nonparametric regression, *The Annals of Statistics* **12**: 1215–1230.

Rice, J. A. (1984b). Boundary modification for kernel regression, *Communications in Statistics: Theory and Methods* **13**(7): 893–900.

Rice, J. A. (1995). *Mathematical Statistics and Data Analysis*, 2nd edn, Duxbury Press, California.

Rosenblatt, M. (1956). Remarks on some nonparametric estimates of a density function, *The Annals of Mathematical Statistics* **27**(3): 832–837.

Rudin, C., Schapire, R. E. and Daubechies, I. (2007). Analysis of boosting algorithms using the smooth margin function, *The Annals of Statistics* **35**(6): 2723–2768.

Schapire, R. (1990). The strength of weak learnability, *Machine Learning* **5**(2): 197–227.

Schapire, R. E. (1999). A brief introduction to boosting, *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*.

Schapire, R. E., Freund, Y., Bartlett, P. and Lee, W. S. (1998). Boosting the margin: A new explanation for the effectiveness of voting methods, *The Annals of Statistics* **26**: 1651–1686.

Schmid, M. and Hothorn, T. (2008). Boosting additive models using component-wise p-splines, *Computational Statistics and Data Analysis* **53**: 298–311.

Shibata, R. (1981). An optimal selection of regression variables, *Biometrika* **68**: 45–54.

Silverman, B. W. (1984). Spline smoothing: the equivalent variable kernel method, *The Annals of Statistics* **12**(3): 898–916.

Silverman, B. W. (1985). Some aspects of the spline smoothing approach to non-parametric regression curve fitting, *Journal of the royal statistical society. Series B (Methodological)*. **47**(1): 1–21.

Skurichina, M. and Duin, R. P. W. (1998). Bagging for linear classifiers, *Pattern Recognition* **31**(7): 909–930.

Stone, M. (1974). Cross-validatory choice and assessment of statistical predictions, *Journal of the Royal Statistical Society. Series B (Methodological)* **36**(2): 111–147.

Stute, W. (1984). Asymptotic normality of nearest neighbor regression function estimates, *The Annals of Statistics* **12**(3): 917–926.

Swanepoel, J. W. H. (1986). A note on proving that the (modified) bootstrap works, *Communications in Statistics: Theory and Methods* **15**(11): 3193–3203.

Swanepoel, J. W. H. (1988). Point estimation based on approximating functionals and the bootstrap, *Technical report*, Dept. of Statistics, Potchefstroom University.

Swanepoel, J. W. H. (1990). A review of bootstrap methods, *South African Statistical Journal* **24**: 1–34.

Takewaza, K. (2006). *Introduction to Nonparametric Regression*, Wiley, Hoboken, New Jersey.

Tukey, J. W. (1977). *Exploratory Data Analysis*, Addison-Wesley, Philippines.

Valiant, L. G. (1984). A theory of the learnable, *Communications of the ACM* **27**(11): 1134–1142.

Velleman, P. F. (1980). Definition and comparison of robust nonlinear data smoothing algorithms, *Journal of the American Statistical Association* **75**(371): 609–615.

Wahba, G. (1977). A survey of some smoothing problems and the method of generalized cross-validation for solving them, *in* P. R. Krishnaiah (ed.), *Applications of Statistics*, Amsterdam: North-Holland, pp. 507–523.

Wahba, G. (1990). *Spline models for observational data*, Vol. 59 of *Series in Applied Mathematics*, SIAM, Philadelphia.

Wand, M. P. and Jones, M. C. (1995). *Kernel Smoothing*, Vol. 60 of *Monographs on Statistics and Applied Probability*, Chapman and Hall, London.

Watson, G. S. (1964). Smooth regression analysis, *Sankhyā Series A* **26**: 359–372.

Yao, Y., Rosasco, L. and Caponetto, A. (2007). On early stopping in gradient descent learning, *Constructive Approximation* **26**: 289–315.

Zhou, Z. and Yang, Y. (2005). Adapt bagging to nearest neighbor classifiers, *Journal of computer science and technology* **20**(1): 48–54. http://cs.nju.edu.cn/zhouzh/zhouzh.files/publication/jcst05.pdf. Date of use: 7 April 2009.