An Introduction to Coding Theory and the Two–Part Minimum Description Length Principle

Thomas C. M. Lee*

Department of Statistics, Colorado State University

October 15, 2000

Summary

This article provides a tutorial introduction to the so–called two–part minimum description length (MDL) principle proposed by Rissanen. This two–part MDL principle is a powerful methodology for solving many statistical model selection problems. However, it seems that this powerful methodology is only adopted by a small number of statisticians to tackle a small number of problems. One plausible reason for this is that the coding theory results required by the MDL principle are somewhat new to most statisticians, and that there are not many readily accessible articles introducing these results appearing in the statistical literature. The first part of this article is devoted to a discussion of such coding theory results. Then, in the second part of the article, the two–part MDL principle is introduced and explained. In doing so, only those coding theory results that are presented in the first part of the article are used. Finally, the applicability of the two–part MDL principle is demonstrated by applying it to tackle four different statistical problems.

Keywords: coding theory, minimum description length principle, model selection, stochastic complexity.

1 Introduction

Suppose a set of observed data is given for analysis. A data analyst typically begins with examining the data in various ways, such as plotting the data and looking for patterns. After a close study of the data, the analyst proposes a number of plausible models and proceeds with model–fitting.

*Postal: Department of Statistics, Colorado State University, Fort Collins, CO 80523-1877, USA. Email: tlee@stat.colostate.edu.

He/she should not have any difficulties in fitting various models if he/she is well trained statistically. After all the proposed models are fitted, the next task is to compare them and possibly pick the most suitable one. If there is a hierarchical structure for the proposed models, methods such as Akaike's Information Criterion (Akaike 1974) and Mallows' C_p (Mallows 1973) can be adopted for this task, even though none of these methods are widely accepted as perfect. If the models are of different forms, cross–validation type methods can be applied, but usually they are computationally demanding. Under this situation, Rissanen suggests using the code length for encoding the data as a means for comparing model complexity and fidelity to the data: the "best" model is the one that produces the shortest code length. Of course comparing code lengths is neither the only nor the best approach for such purposes, but still is a sensible one. This is because a common feature shared by both a good encoding (or compression) scheme and a good statistical model is their ability to capture the regularities, or patterns, present in the data. Rissanen developed and termed one version of this code length comparison idea the minimum description length (MDL) principle, and the aim of this article is to provide a quick tutorial introduction to this principle.

We shall focus our discussion on the so–called two–part MDL, which was developed by Rissanen in a series of papers published from the late seventies to the mid eighties. Much of the work is summarized (or referenced) in Rissanen (1989), and a brief but informative review of this book is given by Speed (1991). Success in applying this two–part MDL principle to tackle different statistical problems has been frequently reported in the literature; see Section 5 for references.

Different variants and improved versions of the two–part MDL, such as predictive least–squares and stochastic complexity, have also been developed by Rissanen (e.g., Rissanen 1996). Since the present article aims to provide a quick and more focused tutorial on the two–part MDL, such variants will not be discussed. For discussions on these related issues, we refer the reader to the excellent reviews provided by Gruñwald (1998), Hansen & Yu (1999), Gruñwald (2000) and Lanterman (2000).

This article shall proceed as follows. In Section 2 we motivate our discussion by a classical model selection problem: the (global) polynomial regression problem. Section 3 presents some basic coding theory results. These coding theory results will be used in Section 4 to introduce the two–part MDL principle. Then in Section 5 the two–part MDL principle is applied to tackle five different statistical problems. Concluding remarks are offered in Section 6.

2 An Example: Polynomial Regression

The following example will be used to motivate our discussion. Suppose we observe n pairs of measurements $(x_1, y_1), \ldots, (x_n, y_n)$, which satisfy

$$y_i = g(x_i) + \epsilon_i, \qquad \epsilon_i \sim \text{ iid } N(0, \sigma^2),$$

where g is a polynomial with unknown order and coefficients. That is, $g(x) = a_0 + a_1 x + \ldots + a_p x^p$, where p and the a_l's are unknown. The goal is to estimate p, as well as the a_l's. Thus, it is a model selection problem in which the candidate models are of different dimensions. In the sequel we write $\boldsymbol{x} = (x_1, \ldots, x_n)^T$ and $\boldsymbol{y} = (y_1, \ldots, y_n)^T$.

Well known statistical methods like Mallows' C_p and cross–validation can be applied to achieve our goal. However, we shall look at the problem from a different angle: we seek the best scheme to encode (or compress) the measurements so that they can be transmitted in the most economical way. If we denote the code length of an object z as $L(z)$, then, in other words, we seek the encoding scheme that produces the smallest $L(\boldsymbol{x}) + L(\boldsymbol{y})$. Here the code length $L(z)$ can be loosely interpreted as the amount of memory required to store z. Of course, the magnitude of $L(z)$ is not unique and is dependent on the method that is used to encode z.

One naive method for encoding $\boldsymbol{x}$ and $\boldsymbol{y}$ is to encode them one component at a time, and the corresponding code length admits the following expression:

$$L(\text{“naive”}) = L(\boldsymbol{x}) + L(\boldsymbol{y}) = L(x_1) + \ldots + L(x_n) + L(y_1) + \ldots + L(y_n).$$

However, if we suspect g is of order $p = 2$, i.e., $g(x) = a_0 + a_1 x + a_2 x^2$, then we can encode $\boldsymbol{x}$ and $\boldsymbol{y}$ by encoding $\boldsymbol{x}$, $\hat{\boldsymbol{\theta}}_2 = (\hat{a}_0, \hat{a}_1, \hat{a}_2)^T$ and $\boldsymbol{r} = (r_1, \ldots, r_n)^T$. Here $\hat{a}_l$ is an estimate of a_l, and $r_i = y_i - (\hat{a}_0 + \hat{a}_1 x_i + \hat{a}_2 x_i^2)$ is the ith residual, *conditioned on* $\hat{\boldsymbol{\theta}}_2$. The resulting code length is of the form

$$L(\text{“}p = 2\text{”}) = L(\boldsymbol{x}) + L(\hat{\boldsymbol{\theta}}_2) + L(\boldsymbol{r}|\hat{\boldsymbol{\theta}}_2) =$$
$$L(x_1) + \ldots + L(x_n) + L(\hat{a}_0) + L(\hat{a}_1) + L(\hat{a}_2) + L(r_1|\hat{\boldsymbol{\theta}}_2) + \ldots + L(r_n|\hat{\boldsymbol{\theta}}_2).$$

If the true p is in fact 2 and the $\hat{a}_l$'s are good estimates, then, by the model assumption, the r_i's are approximately independent Gaussians. By utilizing this fact, as we shall see, one can encode $\boldsymbol{r}$ in such a way that $L(\boldsymbol{r}|\hat{\boldsymbol{\theta}}_2) < L(\boldsymbol{y})$. Therefore if the saving of the code length gained, $L(\boldsymbol{y}) - L(\boldsymbol{r}|\hat{\boldsymbol{\theta}}_2)$, is larger than the overhead code length $L(\hat{\boldsymbol{\theta}}_2)$, we are then better off in using the “$p = 2$” encoding scheme rather than the naive scheme. Indeed, this is generally true if the true value of p is 2.

There is no reason why we cannot suspect p to have another value. Hence we can compute

$$L(\text{“}p=j\text{”}) = L(\boldsymbol{x}) + L(\hat{\boldsymbol{\theta}}_j) + L(\boldsymbol{r}|\hat{\boldsymbol{\theta}}_j)$$

for various values of j, and obtain a family of code lengths $L(\text{“}p=j\text{”})$. Then we can select the best encoding scheme as the one that produces the smallest value of $L(\text{“}p=j\text{”})$.

Now it is easy to see how this idea of comparing code lengths can be applied to the original problem of estimating p: one can *define* the best estimated p as the one that permits the smallest $L(\text{“}p=j\text{”})$. *This is the key idea behind Rissanen's contributions.*

In order to estimate p using this idea of code length comparison, we need to compute $L(\text{“}p=j\text{”}) = L(\hat{\boldsymbol{\theta}}_j) + L(\boldsymbol{r}|\hat{\boldsymbol{\theta}}_j)$ for various values of j. Note that the term $L(\boldsymbol{x})$ is dropped as it is constant for all j. This means that we have to equip ourselves with adequate coding theory results, so as to be able to calculate $L(\hat{\boldsymbol{\theta}}_j)$ and $L(\boldsymbol{r}|\hat{\boldsymbol{\theta}}_j)$ (see the next section).

This code length comparison idea can also be extended to other model selection problems. The key is to first split and encode the data into two parts (a fitted candidate model plus data conditioned on the fitted model), and then *define* the "best" model as the one that minimizes the overall data code length. For the above polynomial example, the first part is $\hat{\boldsymbol{\theta}}_j$ while the second part is $(\boldsymbol{r}|\hat{\boldsymbol{\theta}}_j)$.

3 Some Results from Coding Theory

This section aims to provide a gentle description of some basic coding theory results. A useful reference is Cover & Thomas (1991).

3.1 Optimal Prefix Codes

Some definitions: Let A be a finite set and suppose that we would like to transmit a finite sequence of its elements through a telecommunication line. The set A is called an *alphabet*, its elements are called *symbols* and a finite sequence of such symbols is called a *message* (repeated appearance of the same symbol is allowed). Suppose we agree to transmit messages by binary codes. To achieve this we need an encoding scheme: a one–to–one mapping which maps each message to a finite binary string. We call such a binary string the *codeword of a message*, or simply a *message codeword.* The one–to–one property of the encoding scheme is essential, as it allows a decoder informed of the original encoding scheme to uniquely recover any message from its codeword.

Prefix property: One common method for constructing an encoding scheme is as follows. First associate each symbol with a unique finite binary string, which is the *codeword of the symbol* (or the *symbol codeword*). Then the codeword of a message can be formed by concatenating the codewords of the message's symbols. However such a concatenated codeword may not be unique, that is, it does not necessary satisfy the one–to–one property. One way to guarantee this message codeword uniqueness is to ensure that the symbol codewords satisfy the *prefix* property: no symbol codeword is allowed to be a prefix of another symbol codeword. All these ideas can be well explained by the following example. Consider the English letters together with their symbol codewords displayed below.

letter	a	b	d	g	o
symbol codeword	010	001	11	00	10

Notice that the codeword for "g" is a prefix of the codeword for "b". Given the message codeword "00101011", a decoder is not able to decode it as this codeword can be interpreted in two ways: "001-010-11" for "bad" and "00-10-10-11" for "good". Those hyphens were merely used to indicate the locations of the breakpoints, which are of course not known to the decoder. However if we change the codeword of "b" to "011", such ambiguity will be removed because now the set of the symbol codewords satisfy the prefix property; Fig. 1 provides a tree–based illustration. This prefix property is highly desirable since it allows unambiguous, straightforward and instantaneous decoding. Throughout this article we only consider prefix codes.

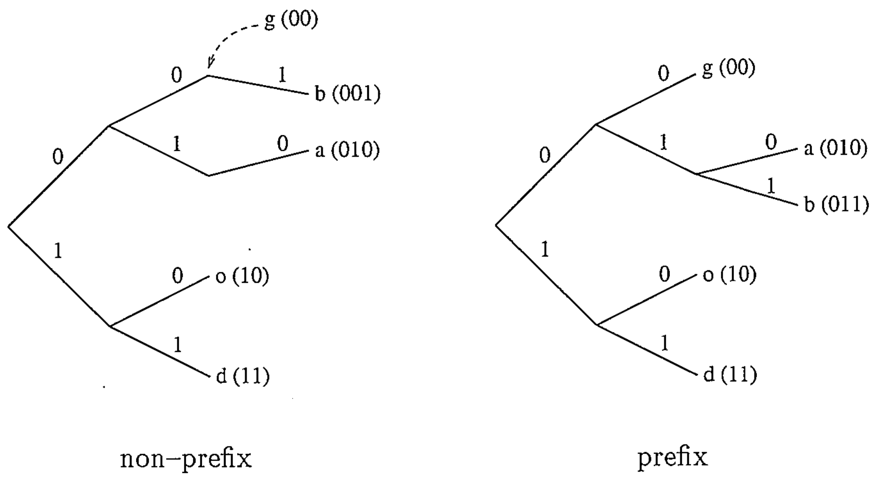

Figure 1: Prefix codes and trees: a symbol codeword of a prefix code cannot be an internal node of a tree.

Kraft inequality: Let $L(\alpha)$ be the length (in number of bits) of the codeword of a symbol $\alpha \in A$. If an encoding scheme (constructed for A) possesses the prefix property, then the following *Kraft inequality* will be satisfied:

$$\sum_{\alpha \in A} 2^{-L(\alpha)} \leq 1.$$

Hence, if the set of lengths $\{L(\alpha), \alpha \in A\}$ of an encoding scheme does not satisfy the Kraft inequality, then it cannot be a prefix one. There is a converse: for any sequence of codeword lengths satisfying the Kraft inequality, there exists a prefix code with the given codeword lengths.

Optimal code length: It should be noted that if the lengths of all symbol codewords are the same, the prefix property will automatically be satisfied. However when constructing an encoding scheme, for the sake of economizing transmission time, we should always aim to minimize the expected mean length of the message codewords. We can achieve this by assigning shorter codewords to more frequent symbols. For example, when transmitting English sentences the most frequent letter "e" should be assigned the shortest codeword.

Let $p(\alpha)$ be the probability of occurrence of the symbol $\alpha \in A$. The expected mean symbol codeword length $E(L)$ (with respect to A and p) is defined as

$$E(L) = \sum_{\alpha \in A} p(\alpha) L(\alpha),$$

and our aim is to design an ideal encoding scheme whose code lengths $\{L(\alpha), \alpha \in A\}$ minimize $E(L)$, subject to the constraints that the symbol codewords are unique and prefix (hence also satisfy the Kraft inequality). Shannon demonstrates that (Shannon & Weaver 1949; see also Rissanen 1989, Section 2.2.2) the ideal code length $L_{\text{OPT}}(\alpha)$ of such an ideal encoding scheme is

$$L_{\text{OPT}}(\alpha) = -\log_2 p(\alpha) \tag{1}$$

for all $\alpha \in A$, and the corresponding optimal value $E(L)_{\text{OPT}}$ for $E(L)$ is:

$$E(L)_{\text{OPT}} = -\sum_{\alpha \in A} p(\alpha) \log_2 p(\alpha) \tag{2}$$

(which is the *entropy* of p). It is true that the ideal code length $L_{\text{OPT}}(\alpha)$ may not be an integer. However we ignore this issue, as methods like Huffman codes are available for obtaining integer $L_{\text{OPT}}(\alpha)$ and yet $E(L)$ is close to being optimal (e.g., see Cover & Thomas 1991, Chapter 5). In general, for a given pair of A and p, it is possible to have more than one encoding scheme which satisfies (1) and gives the optimal code length $E(L)_{\text{OPT}}$.

The above ideas can be abstracted to handle the case when A is infinite and the probability function p is a density function, written as f. In this case the ideal code length of a symbol $\alpha \in A$ is given by

$$L_{\text{OPT}}(\alpha) = -\log_2 f(\alpha). \tag{3}$$

With this abstraction, Rissanen related the problem of finite dimensional parameter estimation to the problem of designing an optimal encoding scheme. This will be discussed in Section 4.

3.2 Encoding Integers and Real Numbers, and Universal Priors

In the last subsection we indicated that, if the density function f of α is known, then the optimal code length $L_{\text{OPT}}(\alpha)$ of α is given by (3). However, in applying the MDL principle (and related techniques), it is often necessary to encode a set of integers and real numbers when no particular density functions are given. Here we describe a prefix encoding scheme for positive integers due to Elias (1975) (see also Rissanen 1989, Section 2.2.4). Using one optimal property of this scheme, we then define various *universal priors.* We also extend this scheme to the encoding of real numbers. It is worth mentioning that it is relatively not that important to us how the coding is exactly done. What is important to us, however, is the minimal code length required to encode our target objects.

Encoding positive integers: Imagine we would like to transmit a large positive integer n by its binary representation (with length about $\log_2 n$ bits). Certainly confusion arises if this binary representation is followed by other binary digits, as the decoder does not know where the binary representation of n ends. We can resolve this issue by supplying the length (using its binary representation) of the binary representation of n as a preamble. This requires an extra length of about $\log_2 \log_2 n$ bits. However the same problem arises as the decoder does not know the length of the preamble's binary representation. We can use the same trick again: supply the length of the preamble by another preamble of length $\log_2 \log_2 \log_2 n$. We can certainly repeat this process until $\log_2 \ldots \log_2 n$ hits a pre–set limit close to zero, and encode n by its binary representation together with a sequence of such binary preambles. (What has been presented is only a brief description of the encoding scheme. The reader should consult the references cited above for details.) It can be shown that this encoding scheme satisfies the prefix property, and that the code length $\log^*$ for this code, as a function of n, is approximately

$$\log^*(n) = \log_2 c + \log_2 n + \log_2 \log_2 n + \ldots, \tag{4}$$

where the sum only includes positive terms and c is a constant approximately equal to 2.865, so that the Kraft inequality holds with equality.

Universal priors: The code length function $\log^*$ has the following optimal property. Let $p(n)$ be any probability function for positive integers satisfying some mild conditions. Rissanen (1983) shows that

$$\lim_{N\to\infty} \frac{-\sum_{n=1}^{N} p(n)\log^*(n)}{-\sum_{n=1}^{N} p(n)\log_2 p(n)} = 1. \tag{5}$$

Recall that the denominator is the optimal expected mean code length (2). Therefore $\log^*(n)$ can be taken as the ideal code length L_{OPT} (see (1)) for large positive integers no matter what $p(n)$ is. There also exist other positive integer encoding schemes which can produce length functions satisfying (5). In fact, Rissanen (1983, Theorem 2) shows that all these optimal length functions have $\log_2 n$ as the dominating term when n is large.

Because of the optimal property (5), Rissanen called the probability function $Q_{\mathrm{I}+}(n) = 2^{-\log^*(n)}$ a *universal prior* for positive integers. This can be extended to be a universal prior $Q_{\mathrm{I}}(n)$ for all integers by dividing $Q_{\mathrm{I}+}(n)$ equally between the positive and negative integers. This can be further extended to be a universal prior for all real numbers by defining $Q_{\mathrm{R}}(x) = Q_{\mathrm{I}}(\lfloor |x| \rfloor)$, where x is a real number and $\lfloor |x| \rfloor$ is the greatest integer less than or equal to $|x|$.

Encoding integers (positive and negative): We can easily modify the above encoding scheme to handle all integers by adding a leading/trailing sign bit to indicate the sign of an integer. In this case the code length is $\log^*(n) + 1$, and again, the dominating term is $\log_2 n$.

Encoding real numbers: First we must realize that in general it takes infinite number of bits to encode a real number x. Therefore in practice we need to truncate x to a given precision δ and transmit the truncated number x_δ: $|x - x_\delta| < \delta$. For example one may truncate the binary real number $x = 1101.101110\ldots$ to $x_\delta = 1101.101$, and so $|x - x_\delta| = 0.000110\ldots < 0.001 = \delta$. Note that $\log_2(1/\delta) = 3$ is the number of fractional binary digits in x_δ.

Now to encode x_δ one can separately encode the integer part $\lfloor x_\delta \rfloor$ and the fractional part of x_δ. It can be shown that the corresponding code length is $L(x_\delta) = \log^*(\lfloor x_\delta \rfloor) + \log_2(1/\delta) \approx \log_2(\lfloor x_\delta \rfloor) + \log_2(1/\delta)$ when x is large. Now since $x \approx x_\delta$, we may just write

$$L(x) = \log_2 x - \log_2 \delta.$$

On the other hand, if the probability density function $\pi(x)$ of x is known ($\pi(x)$ is a standard notation used by Rissanen), then one can show that the corresponding code length for x is approximately (see (3) with $f(x)$ replaced by $\pi(x)$):

$$-\log_2 \pi(x) - \log_2 \delta. \tag{6}$$

When transmitting a real number, one should realize that, as a result of the truncation operation, some accuracy is inevitably lost. Indeed, as we shall see, the issue of "how much should we truncate" plays an important role in the theory of MDL.

3.3 Summary of Code Length Formulae

This section lists some useful code length formulae. The last formula is derived in Section 4.4 below; but for the reason of completeness, it is listed here. We use $L(z)$ to denote the code length of an object z.

CL1 — Let $p(\alpha)$ be the probability of occurrence of a symbol $\alpha \in A$, where A is a *finite* set. Then

$$L(\alpha) = -\log_2 p(\alpha). \tag{7}$$

CL2 — If A is *infinite*, then $p(\alpha)$ is written as $f(\alpha)$, and the *abstracted* code length for α is

$$L(\alpha) = -\log_2 f(\alpha). \tag{8}$$

CL3 — The code length for encoding an integer n with unknown density function is

$$L(n) = \log^*(n) = \log_2 c + \log_2 n + \log_2 \log_2 n + \ldots, \tag{9}$$

where the sum only includes positive terms and $c \approx 2.865$. If n is large,

$$L(n) \approx \log_2 n.$$

CL4 — Let x be a real number and δ be its precision. If the density function of x is unknown, then

$$L(x) = \log_2 x - \log_2 \delta;$$

and if the density function $\pi(x)$ of x is known, then

$$L(x) = -\log_2 \pi(x) - \log_2 \delta. \tag{10}$$

CL5 — If $\hat{\theta}$ is a maximum likelihood estimate computed from n data points, then

$$L(\hat{\theta}) \approx \frac{1}{2} \log_2 n.$$

4 Minimum Description Length Principle

4.1 MDL Generalizes Maximum Likelihood

We mentioned in Section 3.1 that Rissanen related the problem of finite dimensional parameter estimation to the problem of designing an optimal encoding scheme. This can be seen as follows. Suppose a set of data generated from a model f is observed. The model f is completely known except for a vector $\boldsymbol{\theta}$ of parameters, and the goal is to estimate $\boldsymbol{\theta}$. Now imagine that there is an arbitrary alphabet A with $f(\cdot|\boldsymbol{\theta})$ as the density of its symbols, and that the set of the observed data $\boldsymbol{y}$ is a single symbol of A. We want to transmit the data $\boldsymbol{y}$. In order to minimize the expected mean code length, the code length for $\boldsymbol{y}$ should be chosen as (see (3)):

$$L_{\text{OPT}}(\boldsymbol{y}) = \min_{\boldsymbol{\theta}} \{-\log_2 f(\boldsymbol{y}|\boldsymbol{\theta})\}.$$

From this perspective, it is natural to define the estimate $\hat{\boldsymbol{\theta}}$ of $\boldsymbol{\theta}$ as the minimizer of $-\log_2 f(\boldsymbol{y}|\boldsymbol{\theta})$. It is clear that $\hat{\boldsymbol{\theta}}$ is equivalent to the maximum likelihood estimate, but the strength of this idea of minimizing the expected mean code length is that it can be generalized naturally to handle the case when f is only known to be a member of a class of candidate models. The idea is that, in addition to $\boldsymbol{\theta}$, we also choose a "best" model from the model class so that $L(\boldsymbol{y})$ is minimized.

4.2 MDL from Two–Part Codes

Now we consider a more complicated problem: the model, from which the data are generated, is only known to be a member $f_i(\cdot|\boldsymbol{\theta}_i)$ of a class of models

$$\boldsymbol{M} = \{f_i(\cdot|\boldsymbol{\theta}_i); \boldsymbol{\theta}_i \in \Theta_i, \theta_{ij} \sim \pi_{ij}(\theta_{ij}), 1 \leq i \leq m, 1 \leq j \leq k_i\}.$$

In the above m is the number of models in $\boldsymbol{M}$, $\boldsymbol{\theta}_i = (\theta_{i1}, \ldots, \theta_{ik_i})^T$ is a k_i–component vector parameter associated with f_i, Θ_i is the parameter space for $\boldsymbol{\theta}_i$, and $\pi_{ij}(\theta_{ij})$ is a prior density for θ_{ij}. This prior $\pi_{ij}(\theta_{ij})$ is an artificial device introduced merely to simplify the encoding process and does not have a Bayesian interpretation here. More will be said about this in below. It is assumed that every f_i is known except for $\boldsymbol{\theta}_i$, and that different f_i may have different values of k_i; i.e., different number of parameters. Given a set of observed data, our goal is then to choose the "true" f_i from $\boldsymbol{M}$ as well as to estimate the parameter $\boldsymbol{\theta}_i$ associated with it. The polynomial regression problem discussed in Section 2 is an example.

As before, imagine that the set of the observed data $\boldsymbol{y}$ is a single symbol of an alphabet which is to be transmitted. Also imagine that both the encoder and the decoder have agreed on a common model class $\boldsymbol{M}$, and the encoding scheme to be used is based on a model from $\boldsymbol{M}$ with the corresponding $\boldsymbol{\theta}$ estimated from $\boldsymbol{y}$. That is, the encoding scheme is to be based on a fitted model $f_i(\cdot|\hat{\boldsymbol{\theta}}_i)$. Under this situation the encoding of $\boldsymbol{y}$ can be naturally done in two parts. The first part is to encode the fitted model $f_i(\cdot|\hat{\boldsymbol{\theta}}_i)$ that will be adopted to encode $\boldsymbol{y}$, and the second part is to encode $\boldsymbol{y}$ "conditioned on" $f_i(\cdot|\hat{\boldsymbol{\theta}}_i)$. In other words, we split the data into two parts, a "fitted model" part plus a "data given fitted model" part, and encode them separately:

$$L(\text{"data"}) = L(\text{"fitted model"}) + L(\text{"data given fitted model"}).$$

The "best" model is then defined as the one that minimizes $L(\text{"data"})$. For the polynomial regression problem discussed in Section 2, the first part corresponds to the total code length for the coefficients a_l's, while the second part corresponds to the code length for the residuals $\boldsymbol{r}$.

Using this two–part encoding method, the code length $L(\boldsymbol{y})$ for $\boldsymbol{y}$ can be decomposed into

$$L(\boldsymbol{y}) = L(\hat{\boldsymbol{\theta}}_i) + L(\boldsymbol{y}|\hat{\boldsymbol{\theta}}_i),$$

where $L(\hat{\boldsymbol{\theta}}_i)$ and $L(\boldsymbol{y}|\hat{\boldsymbol{\theta}}_i)$ are code lengths for encoding $f_i(\cdot|\hat{\boldsymbol{\theta}}_i)$ and "$\boldsymbol{y}$ conditioned on $f_i(\cdot|\hat{\boldsymbol{\theta}}_i)$" respectively. We remark that a more appropriate notation for the code length of $f_i(\cdot|\hat{\boldsymbol{\theta}}_i)$ is $L(f_i(\cdot|\hat{\boldsymbol{\theta}}_i))$, but for simplicity and to follow the convention of Rissanen, we use $L(\hat{\boldsymbol{\theta}}_i)$; similarly for $L(\boldsymbol{y}|\hat{\boldsymbol{\theta}}_i)$.

Now we derive an expression for $L(\boldsymbol{y})$ and we begin with $L(\hat{\boldsymbol{\theta}}_i)$. To encode $f_i(\cdot|\hat{\boldsymbol{\theta}}_i)$ we can encode f_i and $\hat{\boldsymbol{\theta}}_i$ separately. It takes about $\log^*(m)$ bits to encode f_i, as we can identify each f_i by an index; see **CL3**. Since components of $\hat{\boldsymbol{\theta}}_i = (\hat{\theta}_{i1}, \ldots, \hat{\theta}_{ik_i})^T$ are usually real–valued, it requires about

$$\sum_{j=1}^{k_i} \left\{ -\log_2 \pi_{ij}(\hat{\theta}_{ij}) - \log_2 \delta_j \right\}$$

bits (see **CL4**) to encode them, where δ_j is the precision for $\hat{\theta}_{ij}$. Thus $L(\hat{\boldsymbol{\theta}}_i) = \log^*(m) - \sum_{j=1}^{k_i} \log_2 \pi_{ij}(\hat{\theta}_{ij}) - \sum_{j=1}^{k_i} \log_2 \delta_j$. In most applications, the code length $\log^*(m)$ for identifying f_i will be ignored since it is a constant relative to $\boldsymbol{M}$. Hence we write

$$L(\hat{\boldsymbol{\theta}}_i) = -\sum_{j=1}^{k_i} \log_2 \pi_{ij}(\hat{\theta}_{ij}) - \sum_{j=1}^{k_i} \log_2 \delta_j.$$

Now for $L(\boldsymbol{y}|\hat{\boldsymbol{\theta}}_i)$. Once $f_i(\cdot|\hat{\boldsymbol{\theta}}_i)$ is specified, it takes about $-\log_2 f_i(\boldsymbol{y}|\hat{\boldsymbol{\theta}}_i)$ bits to encode $\boldsymbol{y}$ (see **CL2**), hence

$$L(\boldsymbol{y}|\hat{\boldsymbol{\theta}}_i) = -\log_2 f_i(\boldsymbol{y}|\hat{\boldsymbol{\theta}}_i).$$

Thus the effective code length, or the description length, of $\boldsymbol{y}$ is:

$$\begin{aligned} L(\boldsymbol{y}) &= L(\hat{\boldsymbol{\theta}}_i) + L(\boldsymbol{y}|\hat{\boldsymbol{\theta}}_i) \\ &= -\sum_{j=1}^{k_i} \log_2 \pi_{ij}(\hat{\theta}_{ij}) - \sum_{j=1}^{k_i} \log_2 \delta_j - \log_2 f_i(\boldsymbol{y}|\hat{\boldsymbol{\theta}}_i), \end{aligned} \tag{11}$$

and it should be minimized with respect to f_i, $\hat{\boldsymbol{\theta}}_i$ and the δ_j's. (However notice that if $\hat{\boldsymbol{\theta}}_i$ is not real–valued then $L(\hat{\boldsymbol{\theta}}_i)$ and hence $L(\boldsymbol{y})$ may not be of the general form (11); see Section 5.4 for an example).

4.3 Precision Optimization

With a closer study of the above expression for $L(\boldsymbol{y})$, one can see that there are two types of trade–off involved when one wants to minimize $L(\boldsymbol{y})$. The first trade–off is model complexity, which is associated with the choice of f_i and $\hat{\boldsymbol{\theta}}_i$, or more precisely, the size of k_i — number of parameters in the model. This model complexity trade–off is well known to most statisticians. The second trade–off is about the sizes of the precisions δ_j's. This is a simple, but also a relatively new concept to most statisticians, and is examined in this subsection.

Expression (11) for $L(\boldsymbol{y})$ is not very useful in practice unless a problem concerning the precisions δ_j's is solved: how precise should we encode the estimated parameters $\hat{\theta}_{ij}$? To understand the problem better, consider the following. It is obvious that the length function $L(\boldsymbol{y}|\hat{\boldsymbol{\theta}}_i)$ is minimized when $\hat{\boldsymbol{\theta}}_i$ is the minimizer of $-\log_2 f_i(\boldsymbol{y}|\hat{\boldsymbol{\theta}}_i)$, which is also the maximum likelihood estimate (MLE) of $\boldsymbol{\theta}_i$. In most cases such MLEs are real and require fine precisions for accurate encoding. Such a requirement will result in a large $L(\hat{\boldsymbol{\theta}}_i)$, as fine precisions implies small δ_j's. On the other hand, if we use coarser precisions (larger δ_j's) to encode those MLEs, $L(\hat{\boldsymbol{\theta}}_i)$ will decrease while $L(\boldsymbol{y}|\hat{\boldsymbol{\theta}}_i)$ will generally increase, as the $\hat{\boldsymbol{\theta}}_i$ used for encoding $\boldsymbol{y}$ can deviate more from the MLE, and this causes $-\log_2 f_i(\boldsymbol{y}|\hat{\boldsymbol{\theta}}_i)$ to increase. In other words, we face a trade–off problem: fine precisions produce a small $L(\boldsymbol{y}|\hat{\boldsymbol{\theta}}_i)$ but a large $L(\hat{\boldsymbol{\theta}}_i)$, and vice versa.

One of Rissanen's contributions is that he provided an approximate solution to this trade–off problem (Rissanen 1989, pp 55–56). Briefly, *if $\hat{\theta}_{ij}$ is an MLE computed from n_j data points and if n_j is large, then the precision of $\hat{\theta}_{ij}$ can be effectively encoded with $\frac{1}{2}\log_2 n_j$ bits.* Applying this result, the description length (11) is approximately given by

$$-\log_2 f_i(\boldsymbol{y}|\hat{\boldsymbol{\theta}}_i) - \sum_{j=1}^{k_i} \log_2 \pi_{ij}(\hat{\theta}_{ij}) + \frac{1}{2}\sum_{j=1}^{k_i} \log_2 n_j, \tag{12}$$

where $\hat{\boldsymbol{\theta}}_i$ is the maximum likelihood estimate of $\boldsymbol{\theta}_i$. As a solution to the problem posed at the beginning of this section, f_i should be chosen from M to minimize (12) and $\boldsymbol{\theta}_i$ should be estimated by maximum likelihood. Various ways of handling the term $-\sum \log_2 \pi_{ij}(\hat{\theta}_{ij})$ will be discussed in Section 4.4 below.

Another slightly vague argument which also leads to the use of $\frac{1}{2}\log_2 n_j$ bits to encode the precision of $\hat{\theta}_{ij}$ is given by Hall & Hannan (1988) and Solo (1992). If $\hat{\theta}_{ij}$ is estimated from the data, then there is no need to encode $\hat{\theta}_{ij}$ to an accuracy finer than the standard error of the estimation. For standard parametric problems, if $\hat{\theta}_{ij}$ is estimated from n_j data points, the standard error will be $cn_j^{-\frac{1}{2}}$ where c is (asymptotically) independent of n_j. Thus the code length for the precision is $-\log_2 c + \frac{1}{2}\log_2 n_j$ and $-\log_2 c$ is often ignored if n_j is large.

A more well-known form of (12) is obtained when all the parameters θ_{ij}'s are to be estimated by using all the data points, say n of them. One classical example is subset selection in regression analysis. In this case (12) becomes the MDL criterion defined by Rissanen (1989, pp 56):

$$\mathrm{MDL}(k) = -\log_2 f_i(\boldsymbol{y}|\hat{\boldsymbol{\theta}}_i) - \sum_{j=1}^{k_i} \log_2 \pi_{ij}(\hat{\theta}_{ij}) + \frac{k_i}{2}\log_2 n. \tag{13}$$

4.4 Handling The Prior $\pi_{ij}(\theta_{ij})$

Some remarks have to be made regarding the prior $\pi_{ij}(\theta_{ij})$. From a Bayesian point of view, the role of $\pi_{ij}(\theta_{ij})$ is to reflect one's prior knowledge about θ_{ij}, while Rissanen stresses that his work is non–Bayesian and $\pi_{ij}(\theta_{ij})$ is merely used as an artificial device to minimize the description length. If $\pi_{ij}(\theta_{ij})$ is not known or does not exist at all, he recommends using the universal prior Q_{R} defined in Section 3.2, or any other priors which produce a small description length. Rissanen also states that, if n is large, the choice of $\pi_{ij}(\theta_{ij})$ is relatively unimportant, as the dominating penalty terms in (12) and (13) are $\frac{1}{2}\sum_{j=1}^{k_i}\log_2 n_j$ and $\frac{k_i}{2}\log_2 n$ respectively. In fact, $-\sum_{j=1}^{k_i}\log_2 \pi_{ij}(\hat{\theta}_{ij})$ can be ignored for the following reason. The formulation of the MDL principle allows the user to arbitrarily choose his/her prior $\pi_{ij}(\theta_{ij})$. Therefore he/she can choose a suitable prior $\pi_{ij}(\theta_{ij})$ so that the resulting value of $-\log_2 \pi_{ij}(\hat{\theta}_{ij})$ is small enough to be neglected (e.g., see Cameron, Hannan & Speed 1995). *We shall neglect this term in the rest of this article.*

Therefore, by combining the precision optimization result and the omission of the prior $\pi(\cdot)$, the code length of an MLE $\hat{\theta}$ is $\frac{1}{2}\log_2 n$ if $\hat{\theta}$ is estimated from n data points. This point is summarized in Section 3.3.

4.5 Summary and Remarks

To sum up: the general idea of two–part MDL is to decompose the code length $L(\text{“data”})$ of the full data into two parts

$$L(\text{“data”}) = L(\text{“fitted model”}) + L(\text{“data given fitted model”}),$$

and define the best model as the one that minimizes $L(\text{“data”})$. Typically, calculations of the *model complexity* term $L(\text{“fitted model”})$ involve precision optimization, in which Rissanen's precision result can usually be applied. Calculations of the *data fidelity* term $L(\text{“data given fitted model”})$ is generally straightforward: it is simply the negative of the *conditional likelihood* of the data given the fitted model. Because of this, the negative of $L(\text{“data”})$ can be treated as a penalized likelihood. However, it should be pointed out that in most situations the penalty terms, which are derived from $L(\text{“fitted model”})$, are results from the act of optimizing the precisions, not from any Bayesian assumptions.

5 Examples

In this section the above two–part MDL is applied to tackle four different statistical problems. We shall derive relevant MDL criteria (i.e., code length formulae) for the problems, but we shall *not* discuss the important but separate issue of how to minimize these criteria. Throughout this section we use the generic notation $\hat{\boldsymbol{\theta}}$ to denote a fitted candidate model. That is, if $\boldsymbol{y}$ is the data that we want to encode, the four MDL criteria that are to be constructed admit the common expression $L(\boldsymbol{y}) = L(\hat{\boldsymbol{\theta}}) + L(\boldsymbol{y}|\hat{\boldsymbol{\theta}})$.

5.1 Polynomial Regression Revisited

Here we revisit the polynomial regression problem discussed in Section 2, and we first compute $L(\hat{\boldsymbol{\theta}})$, where $\hat{\boldsymbol{\theta}} = \{\hat{a}_0, \ldots, \hat{a}_p\}$. Since each $\hat{a}_l$ is a real number estimated from n data points, by Rissanen's precision optimization result (see **CL5**), each requires $\frac{1}{2}\log_2 n$ bits to encode. So we have $L(\hat{\boldsymbol{\theta}}) = L(\hat{a}_0, \ldots, \hat{a}_p) = \frac{p+1}{2}\log_2 n$. The remaining part $L(\boldsymbol{y}|\hat{\boldsymbol{\theta}})$ of the overall code length $L(\boldsymbol{y})$ is simply a conditional likelihood. When the errors are Gaussians, it is given by $L(\boldsymbol{y}|\hat{\boldsymbol{\theta}}) = \frac{n}{2}\log(\text{RSS}_p/n)$, where RSS_p is the residual sum of squares: $\text{RSS}_p = \sum_{i=1}^{n}\{y_i - (\hat{a}_0 + \hat{a}_1 x_i + \ldots + \hat{a}_p x_i^p)\}^2$. Upon rearranging terms, we obtain the following classical MDL criterion specific for the polynomial regression problem:

$$\text{MDL}(\text{“polynomial”}) = \frac{(p+1)\log_2 n}{n} + \log_2\left(\frac{\text{RSS}_p}{n}\right). \tag{14}$$

A remark: in order to encode $\hat{\boldsymbol{\theta}}$ in a proper fashion, we first need to encode the order p before the $\hat{a}_l$'s. Also, we need to encode $\boldsymbol{x}$ and an estimate of the noise variance σ^2. However, these code lengths are omitted as they are constants. In the sequel similar omission will be made without being explicitly stated.

5.2 Gaussian Density Mixtures

Suppose we observe n independent realizations $y_1, \ldots, y_n$ generated from the Gaussian mixture density

$$y_i \sim \sum_{j=1}^{p} w_j f(y|\mu_j, \sigma_j^2),$$

where $f(y|\mu_j, \sigma_j^2)$ is the Gaussian probability density function with mean μ_j and variance σ_j^2, and w_j's are weights sum to unity. The number of components p, the weights $\boldsymbol{w} = \{w_1, \ldots, w_p\}$, the means $\boldsymbol{\mu} = \{\mu_1, \ldots, \mu_p\}$ as well as the variances $\boldsymbol{\sigma}^2 = \{\sigma_1^2, \ldots, \sigma_p^2\}$ are all unknown and are the targets of estimation. For identifiability purposes, we assume $\mu_1 < \mu_2 < \ldots < \mu_p$. If p is known *a priori*, all other unknowns can be estimated by the EM algorithm (e.g., see Redner & Walker 1984).

For this mixture problem, a fitted model is $\hat{\boldsymbol{\theta}} = \{\hat{\boldsymbol{w}}, \hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\sigma}}^2\}$. Since that there are $3p$ estimated real parameters in $\hat{\boldsymbol{\theta}}$, and that each of these parameters is to be estimated from n data points, we have $L(\hat{\boldsymbol{\theta}}) = \frac{3p}{2}\log_2 n$ (see **CL5**). The second and last part of code length $L(\boldsymbol{y}|\hat{\boldsymbol{\theta}})$ that we need is given by the negative of the conditional likelihood of $\boldsymbol{y}$. Straightforward algebra shows that

$$\begin{aligned}\text{MDL}(\text{“density mixtures”}) &= L(\hat{\boldsymbol{\theta}}) + L(\boldsymbol{y}|\hat{\boldsymbol{\theta}}) \\ &= \frac{3p}{2}\log_2 n - \sum_{i=1}^{n}\log_2\left\{\sum_{j=1}^{p}\hat{w}_j f(y_i|\hat{\mu}_j, \hat{\sigma}_j^2)\right\},\end{aligned}$$

where $\hat{w}_j$, $\hat{\mu}_j$ and $\hat{\sigma}_j^2$ are elements of $\hat{\boldsymbol{w}}$, $\hat{\boldsymbol{\mu}}$, and $\hat{\boldsymbol{\sigma}}^2$ respectively.

5.3 Fourier Series Regression

This subsection considers the problem of recovering an unknown function f from noisy observations $\boldsymbol{y} = (y_0, \ldots, y_{n-1})^T$. It is assumed that f is compactly supported on $[0,1]$ and that

$$y_i = f\left(\frac{i}{n-1}\right) + \epsilon_i, \qquad \epsilon_i \sim \text{ iid } N(0, \sigma^2).$$

We consider a Fourier series estimator for f:

$$\hat{f}_p(\frac{i}{n-1}) = \hat{a}_0 + \sum_{j=1}^{p} \left\{ \hat{a}_j \cos(\frac{\pi i j}{n-1}) + \hat{b}_j \sin(\frac{\pi i j}{n-1}) \right\},$$

where p is a cutoff frequency and the $\hat{a}_j$'s and $\hat{b}_j$'s are estimated Fourier coefficients given by (e.g., see Fan & Gijbels 1996, Section 2.5.1)

$$\hat{a}_0 = \frac{1}{n}\sum_{i=0}^{n-1} y_i, \qquad \hat{a}_j = \frac{2}{n}\sum_{i=0}^{n-1} \cos(\frac{\pi i j}{n-1}) y_i, \qquad \hat{b}_j = \frac{2}{n}\sum_{i=0}^{n-1} \sin(\frac{\pi i j}{n-1}) y_i, \qquad j = 1, \ldots, p.$$

In order to use this Fourier series estimator, we need to choose p. Now as a fitted model $\hat{\boldsymbol{\theta}} = \{\hat{a}_0, \ldots, \hat{a}_p, \hat{b}_1, \ldots, \hat{b}_p\}$ is composed of $2p+1$ estimated coefficients and they are all estimated from n data points, we have $L(\hat{\boldsymbol{\theta}}) = \frac{2p+1}{2}\log_2 n$ (see **CL5**). It is straightforward to show that $L(\boldsymbol{y}|\hat{\boldsymbol{\theta}}) = \frac{n}{2}\log_2(\text{RSS}_p/n)$ with $\text{RSS}_p = \sum\{y_i - \hat{f}_p(\frac{i}{n-1})\}^2$. Hence we have the following MDL criterion for Fourier series regression:

$$\text{MDL(“Fourier”)} = L(\hat{\boldsymbol{\theta}}) + L(\boldsymbol{y}|\hat{\boldsymbol{\theta}}) = \frac{2p+1}{2}\log_2 n + \frac{n}{2}\log_2\left(\frac{\text{RSS}_p}{n}\right).$$

5.4 Piecewise Constant Regression

We consider the same regression problem as in the last subsection but we estimate f by a piecewise constant function. This is a “non–standard” and interesting MDL application as $\hat{\boldsymbol{\theta}}$ is not only specified by real numbers. The way that we represent and encode $\hat{\boldsymbol{\theta}}$ closely follows the approach of Cameron et al. (1995).

To completely specify a piecewise constant function with say, p continuous segments, one needs to specify (i) the locations of the jump points and (ii) the “height” of each continuous segment. For simplicity, we assume that all jump points are at locations $\frac{i}{n-1}$ for various i. Let n_j be the number of data points in the jth segment. Then $\sum_{j=1}^{p} n_j = n$ and the jth jump point is located at $(\sum_{k=1}^{j} n_k)/(n-1)$. Thus to encode the jump locations, it suffices to encode the p integers $n_1, \ldots, n_p$. As demonstrated by Rissanen (1989, Section 2.2.4), their total code length is $L(n_1, \ldots, n_p) = p\log_2 n$.

Now for the “heights”. Since the height of the jth segment is naturally estimated by averaging those n_j data points that are within the jth segment, the code length for this estimated height is $\frac{1}{2}\log_2 n_j$ (again, see **CL5**). Thus the code length for all p estimated heights is $\frac{1}{2}\sum_{j=1}^{p}\log_2 n_j$. Therefore

$$L(\hat{\boldsymbol{\theta}}) \quad = \quad L(\text{“jump point locations”}) + L(\text{“estimated heights”})$$

$$= p\log_2 n + \frac{1}{2}\sum_{j=1}^{p} \log_2 n_j.$$

As before one can show $L(\boldsymbol{y}|\hat{\boldsymbol{\theta}}) = \frac{n}{2}\log_2(\mathrm{RSS}_p/n)$, thus we have

$$\mathrm{MDL}(\text{"Piecewise Constant"}) = p\log_2 n + \frac{1}{2}\sum_{j=1}^{p}\log_2 n_j + \frac{n}{2}\log_2\left(\frac{\mathrm{RSS}_p}{n}\right).$$

5.5 A Numerical Experiment

Here we report results of a numerical experiment regarding the Fourier series (FS) and the piecewise constant (PC) regression estimators previously discussed.

The experiment was carried out as follows. We first selected two test functions. Test Function 1 was $f(x) = 2\sin(\pi x) + \sin(3\pi x) - \sin(4\pi x) - 2\cos(8\pi x) + \cos(14\pi x)$ and Test Function 2 was the "blocks" function introduced by Donoho & Johnstone (1994): they are displayed in the top row of Fig. 2. Note that FS is particularly suited for Test Function 1 while PC is suited for Test Function 2. We then simulated 100 noisy data sets for each test function by artificially adding independent Gaussian noise. The standard deviation of the noise was $\{\max(f) - \min(f)\}/8$, and the number of observations x was 512 and they were regularly–spaced in $[0, 1]$.

For each simulated noisy data set, we applied the relevant MDL criteria constructed in the last section to select our "best" MDL FS and PC estimates. Finally we compared the code length of the best FS estimate with the best PC estimate and obtained a "grand best" estimate. That is, we were not only applying MDL to select a "best" model from a model class, but also applying it to select a "grand best" model from more than one model class. For Test Function 1, MDL *always* selected FS while for Test Function 2 it *always* selected PC. Fig. 2 also contains some plots for visual evaluation.

5.6 Other Applications

The MDL principle (and its relatives) has also been applied to other estimation problems. For example, Hall & Hannan (1988) and Rissanen, Speed & Yu (1992) applied MDL to the problem of probability density estimation; Saito (1994), Antoniadis, Gijbels & Gregoire (1997) and Lee (2000*b*) consider the use of MDL in the context of nonparametric regression; and Hannan & Rissanen (1988) and Cameron et al. (1995) proposed MDL–based estimators for spectral density estimation.

In addition, different researchers have also applied MDL to other problems in which the encoding of a fitted model $f_i(\cdot|\hat{\boldsymbol{\theta}}_i)$ is more complicated (e.g., tree–based models). For example, Leclerc

(1989), Lee (1998) and Lee (2000*a*) applied the principle to image segmentation; Rissanen (1989, Chapter 7), Quinlan & Rivest (1989), Wallace & Patrick (1993) and Rissanen (1997) have worked on MDL–based classification and machine learning methods; and Lam & Bacchus (1994) and Castillo, Gutiérrez & Hadi (1997, Chapter 11) provide MDL–based solutions to the problem of Bayesian network learning. Of course, the above list is by no means complete, and more comprehensive references can be found in Gruñwald (1998) and Lanterman (2000).

6 Concluding Remarks

What does MDL actually offer? It is *not* the goal of MDL to find the ultimate minimum code length for the data. What MDL can offer is that when a class of candidate models is available, it can be applied to select a best model from this class even when the "true" model may not be a member of it. In fact, MDL can be pushed even further to select a best model from different model classes, just as what was done in Section 5.5. However, one should be very careful in calculating and comparing the code length of models from different classes. Of course, if the "true" model class is known *a priori*, one should always only consider models from this "true" model class.

Relatives of MDL: The above–discussed two–part MDL approach has two close relatives: predictive MDL and stochastic complexity. Due to space limitation, we do not discuss these two variants, and interested readers can consult Rissanen (1989). We conclude this article by pointing out that concepts and theories of MDL are constantly being clarified, refined and added. For a most up–to–date and complete review on some newer versions of MDL, and their relations with other model selection approaches, see Lanterman (2000).

Acknowledgement

The author would like to thank Mark Berman, Victor Solo and Kenneth Wilder for their criticisms on earlier versions of this article, and Murray Cameron for providing a preprint of Cameron et al. (1995). He would also like to thank the referee for many useful comments, and the joint editor, Dr. Elja Arjas, for his editorial arrangement.

References

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

Antoniadis, A., Gijbels, I. & Gregoire, G. (1997), 'Model selection using wavelet decomposition and applications', *Biometrika* **84**, 751–763.

Cameron, M. A., Hannan, E. J. & Speed, T. P. (1995), 'Estimating spectra and prediction variance'. Unpublished manuscript.

Castillo, E., Gutiérrez, J. M. & Hadi, A. S. (1997), *Expert Systems and Probabilistic Network Models*, Springer–Verlag, New York.

Cover, T. M. & Thomas, J. A. (1991), *Elements of Information Theory*, John Wiley & Sons, Inc.

Donoho, D. L. & Johnstone, I. M. (1994), 'Ideal spatial adaptation by wavelet shrinkage', *Biometrika* **81**, 425–455.

Elias, P. (1975), 'Universal codeword sets and representations of the integers', *IEEE Transactions on Information Theory* **IT-21**, 194–203.

Fan, J. & Gijbels, I. (1996), *Local Polynomial Modelling and Its Applications*, Chapman and Hall, London.

Gruñwald, P. (1998), The MDL principle and reasoning under uncertainty, PhD thesis, University of Amsterdam.

Gruñwald, P. (2000), 'Model selection based on minimum description length', *Journal of Mathematical Psychology* **44**, 133–152.

Hall, P. & Hannan, E. J. (1988), 'On stochastic complexity and nonparametric density estimation', *Biometrika* **75**, 705–714.

Hannan, E. J. & Rissanen, J. (1988), 'The width of a spectral window', *Journal of Applied Probability* **25 A**, 119–125.

Hansen, M. H. & Yu, B. (1999), 'Model selection and the principle of minimum description length', Technical report, Bell Labs, Lucent Technologies.

Lam, W. & Bacchus, F. (1994), 'Learning Bayesian belief networks: An approach based on the MDL principle', *Computational Intelligence* **10**, 269–293.

Lanterman, A. D. (2000), 'Schwarz, Wallace, and Rissanen: Intertwining themes in theories of model selection', *International Statistical Review*. **Please add exact reference here.**

Leclerc, Y. G. (1989), 'Constructing simple stable descriptions for image partitioning', *International Journal of Computer Vision* **3**, 73–102.

Lee, T. C. M. (1998), 'Segmenting images corrupted by correlated noise', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **20**, 481–492.

Lee, T. C. M. (2000*a*), 'A minimum description length based image segmentation procedure, and its comparison with a cross–validation based segmentation procedure', *Journal of the American Statistical Association* **95**, 259–270.

Lee, T. C. M. (2000*b*), 'Regression spline smoothing using the minimum description length principle', *Statistics and Probability Letters* **48**, 71–82.

Mallows, C. L. (1973), 'Some comments on C_p', *Technometrics* **15**, 661–675.

Quinlan, J. R. & Rivest, R. L. (1989), 'Inferring decision trees using the minimum description length principle', *Information and Computation* **80**, 227–248.

Redner, R. A. & Walker, H. F. (1984), 'Mixture densities, maximum likelihood and the EM algorithm', *SIAM Review* **26**, 195–239.

Rissanen, J. (1983), 'A universal prior for integers and estimation by minimum description length', *The Annals of Statistics* **11**, 416–431.

Rissanen, J. (1989), *Stochastic Complexity in Statistical Inquiry*, World Scientific, Singapore.

Rissanen, J. (1996), 'Fisher information and stochastic complexity', *IEEE Transactions on Information Theory* **42**, 40–47.

Rissanen, J. (1997), 'Stochastic complexity in learning', *Journal of Computer and System Sciences* **55**, 89–95.

Rissanen, J., Speed, T. & Yu, B. (1992), 'Density estimation by stochastic complexity', *IEEE Transactions on Information Theory* **38**, 315–323.

Saito, N. (1994), Simultaneous noise suppression and signal compression using a library of orthonormal bases and the minimum description length criterion, *in* E. Foufoula-Georgiou & P. Kumar, eds, 'Wavelets in Geophysics', New York: Academic Press, pp. 299–324.

Shannon, C. E. & Weaver, W. (1949), *The Mathematical Theory of Communication*, University of Illinois Press, Urbana and Chicago.

Solo, V. (1992), 'Book review on "The Statistical Theory of Linear Systems", E. J. Hannan and M. Deistler, 1988', *Econometric Theory* **8**, 135–143.

Speed, T. (1991), 'Book review on "Stochastic Complexity in Statistical Inquiry", J. Rissanen, 1989', *IEEE Transactions on Information Theory* **37**, 1739–1740.

Wallace, C. S. & Patrick, J. D. (1993), 'Coding decision trees', *Machine Learning* **11**, 7–22.

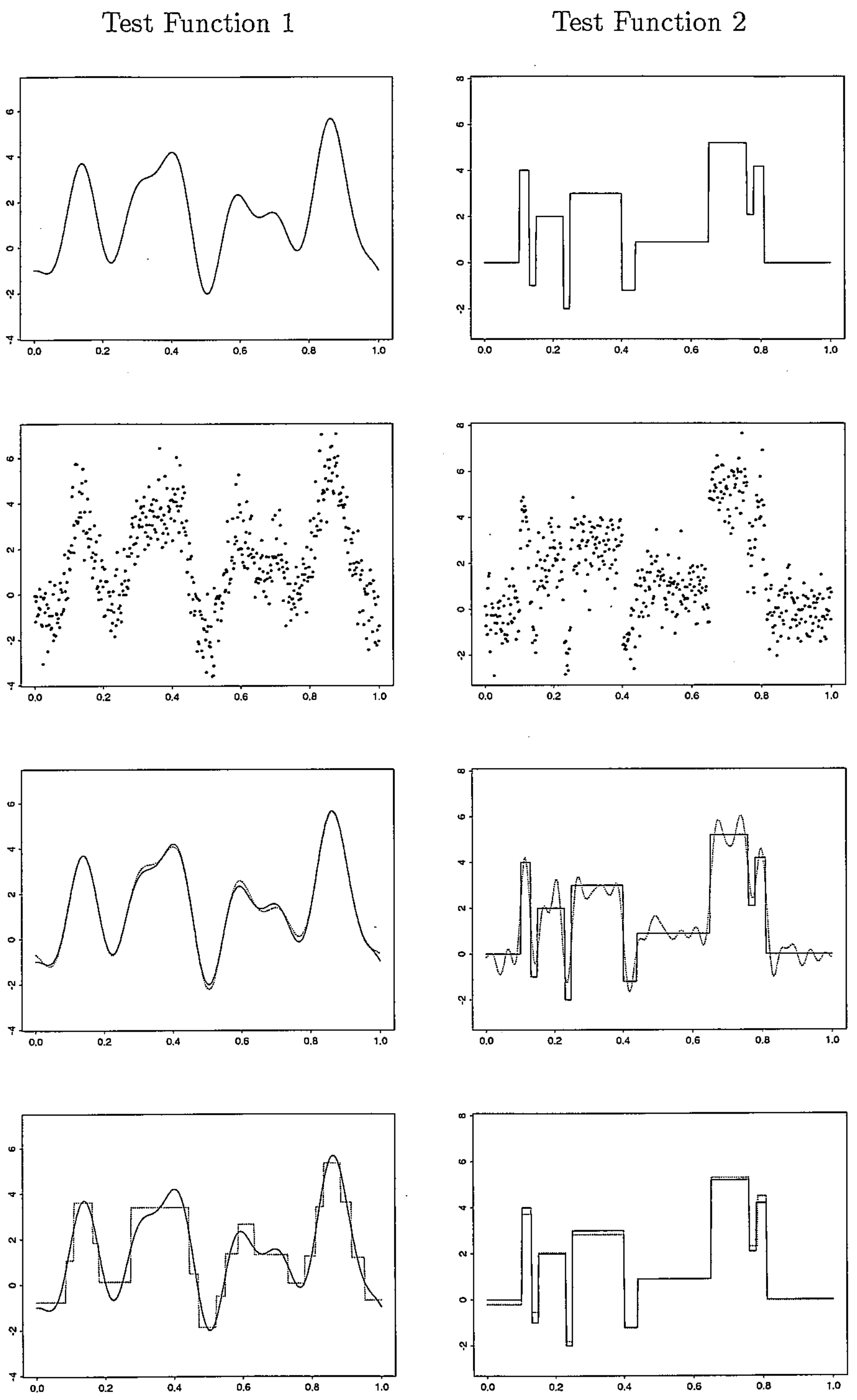

Figure 2: Plots for the numerical experiment. First row: Test Functions; second row: noisy observations; third row: FS estimates (dotted lines); and fourth row: PC estimates (dotted lines).