Added Distributions for use in Clustering (Mixture Modelling), Function Models, Regression Trees, Segmentation, and mixed Bayesian Networks in Inductive Programming 1.2.

Lloyd Allison

Faculty of Information Technology,
Monash University, Clayton, Victoria, Australia 3800.
April 2008.

TR 2008/224

Abstract. *Inductive programming* is a machine learning paradigm combining functional programming (FP) with the information theoretic criterion, *Minimum Message Length* (MML). IP 1.2 now includes the Geometric and Poisson distributions over non-negative integers, and Student's t-Distribution over continuous values, as well as the Multinomial and Normal (Gaussian) distributions from before. All of these can be used with IP's model-transformation operators, and structure-learning algorithms including clustering (mixture-models), classification- (decision-) trees and other regressions, and mixed Bayesian networks, provided only that the types match between each corresponding component Model, transformation, structured model, and variable – discrete, continuous, sequence, multivariate, and so on.

Source code of IP 1.2 is at http://www.allisons.org/ll/FP/IP/ .

Keywords: Bayesian networks, classification, decision trees, function models, Gaussian, Geometric, Haskell, inductive inference, MDL, minimum message length, MML, Multinomial, Poisson, regression, t-Distribution.

1 Introduction

Inductive Programming (IP) [2, 3] is a paradigm that allows the rapid creation of succinct solutions to inductive inference problems in machine learning and data mining. It takes advantage of the compositional properties of (i) Functional Programming (FP), as represented by the programming language Haskell, and (ii) the Minimum Message Length (MML) principle [21, 20], a Bayesian method for inductive inference (data mining, machine learning).

IP has been used to learn various kinds of statistical models from given data including multivariate mixture models (clustering), segmentation models [10], classification- (decision-), regression- and model-trees [3], time-series models, and mixed Bayesian networks [4] over both discrete and continuous variables. (In general a variable can be of an arbitrary type provided that a suitable Model is created for it [2]. For example, sequences can be modelled by a Time-Series model and sequences have been clustered in this way.)

Modules and classes have been somewhat reorganised in IP 1.2, and extra probability distributions (Models) – the Geometric and Poisson distributions over integers and Student's t-Distribution over continuous values – have been added to the previously implemented Multinomial and Normal (Gaussian) Models. All of these distributions can be used immediately with the algorithms for learning structured models – mixture models (clustering), classification- (decision-), regression- and model-trees and other Function Models (regressions), segmentation models of Time-Series data, and Bayesian Networks over mixed discrete and continuous data.

Student's t-Distribution does not have a closed-form estimator. A small library of scientific and numerical functions is new to IP 1.2 to optimise the t's parameters, in the first instance, but it obviously has other uses and will be expanded in future.

2 Background

For completeness, this section gives the obligatory brief introductions to Minimum Message Length (MML) inference and to Inductive Programming (IP).

2.1 MML

Minimum Message Length (MML) [21, 20] is a Bayesian method of inference. It builds on Bayes's theorem [6] and on Shannon's mathematical theory of communication [18], hence 'message':

```
Pr(M&D) = Pr(M).Pr(D|M) = Pr(D).Pr(M|D)
msgLen(E) = -log(Pr(E))
msgLen(M&D) = msgLen(M)+msgLen(D|M) = msgLen(D)+msgLen(M|D)
```

where M is a model (theory, hypothesis, parameter estimate) of prior probability Pr(M) over some data, D, and E is an event of probability Pr(E). MsgLen(E) is the length of a message for E in an optimal code. The units are *nits* for natural logs and *bits* for base 2 logs. MML is related to the later Minimum Description Length (MDL) principle [16, 5] but differs from it, for example, by preferring models that are fully parameterised, to *appropriate precision.* (MML is not equivalent to MAP in general.)

MML imagines a *transmitter* sending a two-part message to a *receiver*. The first part, of length msgLen(M), states a model which answers some inference problem that must be solved. The second part, msgLen(D|M), states the data encoded as if the answer, M, is true. Note that the receiver cannot decode the second part without the first part. There is a trade-off between the complexity of the model, M, and its fit to the data, D|M. The transmitter and receiver can cooperate to design an efficient code but this can only be based on *expected* data. Strict MML (SMML) relies on the design of a full optimal code book. Unfortunately SMML is computationally infeasible for most inference problems [11] but there are efficient and accurate MML approximations [20], notably MML87 [23], for many useful problems and models. The MML87 approximation to the message length for a model M_θ with a k-dimensional parameter θ is given by

$$\text{msgLen}(D,\theta) = \underbrace{-\log h(\theta) + \frac{1}{2}\log|F(\theta)| + \frac{k}{2}\log\kappa_k}_{\text{msgLen}(\theta)} + \underbrace{\frac{k}{2} - \log Pr(D|\theta)}_{msglen(D|\theta)} \qquad (1)$$

where h is the prior on θ, $F(\theta)$ is the Fisher information matrix, $|...|$ takes the determinant, and κ_k is a lattice constant for k dimensions [9], $k_1 = 1/12, k_2 = 3/(36\sqrt{3}), k_3 = 19/(190 * 2^{1/3}), ..., k_\infty = 1/(2\pi e)$.

MML is a natural *compositional* criterion because the complexity of data, models and sub-models are all measured in the same units. Inductive Programming exploits this to build structured models from component models. It also uses transformation operators on models, much as functional programming uses high-order functions to transform functions.

2.2 Inductive Programming

Inductive Programming (IP) provides a library of statistical models, estimators, and operators on these.

Initially, IP was created in response to a quite natural problem in research: Much research in MML, and in machine learning in general, takes the form of creating a new kind of statistical model, plus a learner (estimator), for a fairly specific problem such is unsupervised or supervised classification, or time-series prediction. Often it is carried out by a PhD student who creates a prototype (flaky) implementation of the new model, compares it to earlier competitive models, gets the degree, and departs for fresh fields. Gradually a collection builds up of incompatible, hard to maintain programs. A unified tool-kit, such as IP, can reduce this problem. As an elementary example, there is a *single* implementation of the Normal (Gaussian) distribution in IP and it is used by unsupervised classification, regression-trees and anything else that needs *the* Normal distribution.

In attempting to correct this initial 'problem in research,' IP seeks to create a uniform framework for as much of machine learning as possible. This raises the question of just what are the general tasks, and tools, in machine learning? What are their properties and operators and, crucially, what are their types and type-classes? The polymorphic type system of Haskell [15] has been useful for examining this question and, as part of a first-cut at an answer, there are four type-classes [2, 3]; see figure 1.

`SuperModel` is the super-class of the currently three populated classes of statistical model: `Model`, `FunctionModel` and `TimeSeries`. Every statistical model,

```
class ... SuperModel sMdl where
 prior   :: sMdl -> Probability
 msg1    :: sMdl -> MessageLength
 mixture :: ... mx sMdl -> sMdl
 ...

class Model mdl where
 pr   :: (mdl dataSpace) -> dataSpace -> Probability
 nlPr :: (mdl dataSpace) -> dataSpace -> MessageLength
 msg  :: ... (mdl dataSpace) -> [dataSpace] -> MessageLength
 msg2 :: (mdl dataSpace) -> [dataSpace] -> MessageLength
 ...

class FunctionModel fm where
 condModel :: (fm inSpace opSpace) -> inSpace -> ModelType opSpace
 ...

class TimeSeries tsm where
 predictors :: (tsm dataSpace) -> [dataSpace] -> [ModelType dataSpace]
 ...
```

'`...`' stands for omitted details, '`::`' for 'has type',
'`[t]`' for 'list of a type `t`', and '`->`' for function type.

Fig. 1. Classes of Statistical Model.

`m`, has a prior probability, equivalently, a complexity, that is the first-part of a message, `msg1 m`. A `mixture` can also be formed of two or more statistical models of the same type.

The main duty of a `Model`, `m`, over some data-space (type) is to give the probability, `pr m x`, equivalently the negative log probability, `nlPr m x`, of a datum, `x`, in the data-space. Derived from this is the second-part message length, `msg2 m xs`, of a data-set, `xs`, that is of a list, `[...]`, of data from the data-space. Both message parts together give the total message length, `msg m xs`.

A `FunctionModel` has an *input-space* and an *output-space* and returns a `Model` of the output-space conditional, `condModel`, upon a value from the input-space.

A `TimeSeries` statistical model gives a series of `predictors`, each one being a `Model` for the *next* element in a data-series, depending on the *context* of previous values in the sequence up to that point.

In Haskell, a value has a type and a type can be an instance of one or more type-classes. For example, 3.14 can have the type `Double` and `Double` is an instance of various classes including `Num` (numerical), and `Show` (printable). A Bayesian network [13] learned from multivariate observational data is a directed acyclic graph where each node corresponds to a variable and each edge shows a child's dependency on a parent. In IP, a Bayesian network has been represented as an ordering of the variables plus a `FunctionModel` for each variable conditional upon some or all of its predecessors in the sequence. A conditional probability table (CPT), that is a `FiniteFunctionModel`, is often used for parent-child dependencies on discrete variables but any `FunctionModel` (over the right input and output spaces) can be used [14]. A classification-tree has the type `CTreeType ip op` which actually includes regression-trees when `op` is continuous [3]. `CTreeType` is an instance of class `FunctionModel` and so can be used for parent-child dependencies, as first suggested by Friedman and Goldszmidt [12] (subsequently by Comley and Dowe [8]). The output `Model` of each `FunctionModel` must match the type of its output-space variable; the Multinomial distribution is typically used for a discrete variable and any appropriate continuous distribution for a continuous variable [4].

3 Inductive Programming 1.2 (IP1.2)

IP previously included the Multinomial and Normal (Gaussian) distributions (`Models`). New to IP 1.2 are the Geometric (§3.2) and Poisson (§3.1) distributions, *both* defined for integers ≥ 0 and parameterised on their *means*, and Student's t-Distribution (§3.3) for continuous values. A small library of numerical/ scientific functions (§3.5) is also new. Finally, some file and module names have been changed.

The *Snob* program [21] implements an MML Poisson distribution [20, 22] where two variables are needed, one for a count and one for a duration; in IP (§3.1) a single integer variable is modelled. The Geometric (§3.2) distribution here follows [17] which examined selecting between a Poisson and Geometric, and compared estimators for them.

3.1 Poisson

The Poisson's probability function for an integer datum $x \geq 0$ is

$$Pr(x|\alpha) = \frac{e^{-\alpha}\alpha^x}{x!} \tag{2}$$

The mean and variance are equal to α. The negative log likelihood, L, is

$$L = \alpha - x \log \alpha + \log x!$$

so

$$\frac{dL}{d\alpha} = 1 - \frac{x}{\alpha}$$

Equating the derivative for n data, $x_1, ..., x_n$, to zero, we see that the maximum likelihood estimator is

$$\alpha_{ML} = \frac{1}{n}\sum x_i$$

The second derivative of L is

$$\frac{\partial^2 L}{\partial \alpha^2} = \frac{x}{\alpha^2}$$

Its expectation, the Fisher information, for n data is therefore

$$F_\alpha = \frac{n\alpha}{\alpha^2} = \frac{n}{\alpha} \tag{3}$$

A reasonable prior on α is

$$h_A(\alpha) = \frac{1}{A} e^{-\alpha/A} \tag{4}$$

which has mean A. It is usually easy to come up with a safe value for A by setting it to a "typical expected data value", but if in doubt err towards making A larger rather than smaller. The prior has little effect on the MML estimate of α but it does affect the cost of stating the estimate.

Substituting the prior (4) and Fisher (3) into the message length equation (1), and differentiating

$$\frac{d \text{ msgLen}}{d\ \alpha} = \frac{1}{A} + n - \frac{\sum x_i}{\alpha} - \frac{1}{2\alpha}$$

Setting this to zero gives the MML estimator

$$\alpha_{MML} = \frac{\sum x_i + 1/2}{n + 1/A}$$

with uncertainty region

$$\sqrt{\frac{12}{F_{\alpha_{MML}}}} = \sqrt{\frac{12\alpha_{MML}}{n}}$$

3.2 Geometric

Given a coin where $p = Pr(head)$, the number of tails thrown *before* the first head is thrown is modelled by a Geometric distribution (often the head itself is included, but not here). The distribution's mean, μ, equals $1/p - 1$, so $p = 1/(\mu + 1)$. It is convenient to re-parameterise the distribution on its mean. For an integer datum $x \geq 0$ we have

$$Pr(x|\mu) = \frac{\left(\frac{\mu}{\mu+1}\right)^x}{\mu + 1} \tag{5}$$

$$L = -\log Pr(x|\mu) = x(\log(\mu + 1) - \log \mu) + \log(\mu + 1)$$

The first derivative of L is

$$\frac{d\ L}{d\ \mu} = x\left(\frac{1}{\mu + 1} - \frac{1}{\mu}\right) + \frac{1}{\mu + 1}$$

So the maximum likelihood estimator for n data, $x_1, ..., x_n$, is

$$\mu_{ML} = \frac{1}{n} \sum x_i$$

The second derivative of L is

$$\frac{\partial^2 L}{\partial \mu^2} = x\left(\frac{1}{\mu^2} - \frac{1}{(\mu+1)^2}\right) - \frac{1}{(\mu+1)^2}$$

which for n data has expectation, that is Fisher information,

$$F_\mu = \frac{n}{\mu(\mu+1)}$$

It is convenient to adopt the prior, equation (4), already used for the Poisson distribution, partly because the two distributions are often competitors on a given data-set and doing so puts them on an equal footing [17].

Substituting in the message length equation (1) and differentiating,

$$\frac{d\ \text{msgLen}}{d\ \mu} = \frac{1}{A} + \frac{1}{\mu+1}\left(\sum x_i + n - \frac{1}{2}\right) - \frac{1}{\mu}\left(\sum x_i + \frac{1}{2}\right)$$

Equating this to zero gives a quadratic for μ

$$\frac{\mu^2}{A} + \mu\left(\frac{1}{A} + n - 1\right) - \frac{1}{2} - \sum x_i = 0$$

Taking the admissible (+) solution of the quadratic gives the MML estimator

$$\mu_{MML} = \left(1 - n - \frac{1}{A} + \sqrt{n^2 + \frac{1}{A^2} + 1 + \frac{2n}{A} - 2n + \frac{4}{A}\sum x_i}\right)\frac{A}{2}$$

with uncertainty region

$$\sqrt{\frac{12}{F_{\mu_{MML}}}}$$

It is not immediately obvious that μ_{MML} does indeed tend to the mean for larger and larger data-sets from the same source.

3.3 Student's t-Distribution

Student's t-Distribution [19] for a continuous datum x has the probability density function

$$f(x|\mu,\sigma,\nu) = \frac{\Gamma\frac{\nu+1}{2}}{\sqrt{\mu\nu}\sigma\Gamma\frac{\nu}{2}}\left\{1 + \frac{(x-\mu)^2}{\nu\sigma^2}\right\}^{-(\nu+1)/2}$$

ν is called the *shape* parameter. The mean of the distribution is undefined if $\nu \leq 1$. The variance is $\sigma^2\nu/(\nu-2)$, if $\nu > 2$, otherwise it is undefined. If $\nu = 1$ the t-Distribution is a Cauchy distribution, and if ν is large, say ≥ 30, it is very close to the Normal (Gaussian) distribution, $N(\mu,\sigma)$.

The negative log likelihood, L, for a datum, x is

$$L = \frac{1}{2}\log\pi + \log\Gamma\frac{\nu}{2} - \log\Gamma\frac{\nu+1}{2} - \frac{\nu}{2}\log\nu - \nu\log\sigma + \frac{\nu+1}{2}\log(\nu\sigma^2 + (x-\mu)^2)$$

The first derivatives with respect to the parameters are

$$\frac{\partial\ L}{\partial\ \mu} = -(\nu+1)\frac{x-\mu}{\nu\sigma^2+(x-\mu)^2}$$

$$\frac{\partial\ L}{\partial\ \sigma} = \frac{-\nu}{\sigma} + \nu(\nu+1)\sigma\frac{1}{\nu\sigma^2+(x-\mu)^2}$$

$$\begin{aligned}\frac{\partial\ L}{\partial\ \nu} &= \frac{1}{2}\psi\left(\frac{\nu}{2}\right) - \frac{1}{2}\psi\left(\frac{\nu+1}{2}\right) - \frac{1}{2} - \frac{1}{2}\log\nu - \log\sigma \\ &+ \frac{1}{2}\log(\nu\sigma^2+(x-\mu)^2) + \frac{(\nu+1)\sigma^2}{2(\nu\sigma^2+(x-\mu)^2)}\end{aligned}$$

(Note that ψ and ψ_1 are the first and second derivatives of the *log* Γ function, also known as digamma and trigamma.) The three first derivatives of L, above, give its gradient, ∇L, which is useful when it comes to searching for an estimate.

Turning to the second derivatives of L, first the terms for μ and σ,

$$\frac{\partial^2\ L}{\partial\ \mu^2} = (\nu+1)\left(\frac{1}{\nu\sigma^2+(x-\mu)^2} - \frac{2(x-\mu)^2}{(\nu\sigma^2+(x-\mu)^2)^2}\right)$$

which has expectation

$$F_{11} = \frac{\nu+1}{(\nu+3)\sigma^2}$$

$$\frac{\partial^2\ L}{\partial\ \sigma^2} = \frac{\nu}{\sigma^2} + \nu(\nu+1)\left(\frac{1}{\nu\sigma^2+(x-\mu)^2} - \frac{2\nu\sigma^2}{(\nu\sigma^2+(x-\mu)^2)^2}\right)$$

which has expectation

$$F_{22} = \frac{2\nu}{(\nu+3)\sigma^2}$$

The off-diagonal second derivatives involving μ are all *odd* functions of the data $x-\mu$ so their expectations are zero,

$$F_{12} = F_{21} = F_{13} = F_{31} = 0$$

The Fisher information, $F_{\mu\sigma}$, for n data and holding the shape, ν, constant is therefore

$$F_{\mu\sigma} = F_{11}F_{22}n^2$$

As a sanity check, when ν grows large $F_{\mu\sigma}$ tends to $2n^2/\sigma^4$ which is also the Fisher for the Normal distribution, $N(\mu,\sigma)$, with respect to μ and σ [23].

Next for the second derivatives involving ν,

$$\frac{\partial^2\ L}{\partial\ \nu^2} = \frac{1}{4}\psi_1\left(\frac{\nu}{2}\right) - \frac{1}{4}\psi_1\left(\frac{\nu+1}{2}\right) - \frac{1}{2\nu} + \frac{\sigma^2}{\nu\sigma^2+(x-\mu)^2} - \frac{(\nu+1)\sigma^4}{2(\nu\sigma^2+(x-\mu)^2)^2}$$

which has expectation

$$F_{33} = \frac{1}{4}\psi_1\left(\frac{\nu}{2}\right) - \frac{1}{4}\psi_1\left(\frac{\nu+1}{2}\right) - \frac{\nu+5}{2\nu(\nu+1)(\nu+3)}$$

There is interaction between σ and ν,

$$\frac{\partial^2\ L}{\partial\sigma\partial\nu} = \frac{\partial^2\ L}{\partial\nu\partial\sigma} = -\frac{1}{\sigma} + \frac{(2\nu+1)\sigma}{\nu\sigma^2+(x-\mu)^2} - \frac{\nu(\nu+1)\sigma^3}{(\nu\sigma^2+(x-\mu)^2)^2}$$

which has expectation

$$F_{23} = F_{32} = \frac{-2}{\sigma(\nu+1)(\nu+3)}$$

The Fisher information for n data, for all three parameters, $F_{\mu\sigma\nu}$, is therefore

$$F_{\mu\sigma\nu} = F_{11}(F_{22}F_{33} - F_{23}^2)n^3$$

which, after some rearranging, becomes

$$\frac{n^3}{\sigma^4}\left\{\frac{\nu(\nu+1)}{2(\nu+3)^2}\left(\psi_1\left(\frac{\nu}{2}\right) - \psi_1\left(\frac{\nu+1}{2}\right)\right) - \frac{1}{(\nu+1)(\nu+3)}\right\}$$

agreeing with the equation in [1]. Note that $\log F_{\mu\sigma\nu}$, the present instance of $\log F$ in equation (1), is of the form

$$\log F_{\mu\sigma\nu} = 3\log n - 4\log\sigma + \log\ g(\nu)$$

for an expression g.

There is no closed-form for the MML estimator (nor maximum likelihood (ML)) for the t-Distribution so a numerical search is necessary. This has been implemented, using IP's small numerical library, to minimise the message length, equation (1), subject to the *constraints* $\sigma > 0, 1 \le \nu \le 30$, plus any extra ones that an estimator might impose. The median of the data is a reasonable starting point for μ, the standard deviation for σ, and 6 for ν (according to the grapevine). The default prior is uniform for μ on some range, $1/\sigma$ normalised for σ on some range, and uniform for ν on $[1, 30]$ but, given the numerical search, virtually any prior could be used instead. k_3 is the relevant lattice constant (§2.1). On small data-sets the MML estimator may select slightly larger values of σ and ν than the ML estimator to make $|F|$ smaller.

3.4 Distribution Implementations

To begin to use a new distribution (`Model`) in IP requires at least a function to construct constant, fully parameterised `Models`.

To fit the distribution to data requires an estimator. An estimator is a function from a data-set (list) of data from the appropriate data-space (type) to a `Model` over that data-space. Such an estimator may, for example, be passed as a parameter to the learner of "...-trees" [3] to model data from the tree's output-space, in its leaves, in which case we would learn a regression-tree. (Often there is a function of extra parameters, for example to control the prior, which produces an estimator.)

A *weighted* estimator, for weighted data, is needed to use the distribution in mixture models (clustering). Note that the weights may be fractional. This is because a datum may belong partly to two or more components of the mixture.

Constructors, and weighted and unweighted estimators have been implemented for the Poisson and Geometric distributions, and Student's t-Distribution. `Module T` also includes "extras" to do with the t-Distribution, such as the full Fisher information matrix, which may be useful to implement special cases and other estimators. For example, when holding the shape, ν, constant the reduced Fisher can be got by dropping the last row and column (`dropRowCol`) of the full Fisher. When setting $\mu = 0$ for a linear regression, the first row and column can be dropped.

3.5 Numerical Functions

A small numerical library, `module Sci`, was implemented in Haskell mainly to support the t-Distribution in the first instance. It makes use of Haskell's Array library [15]. There are various efforts by others to interface Haskell to "serious" numerical libraries implemented in C, for example. However the t's needs are quite modest and the present course of action was taken (i) to see how it would work out (interesting), and (ii) to keep IP self-contained, for now. The efficiency question for numerical and array algorithms in functional programming has been well examined [7] but is not a major concern in this instance. As far as writing numerical Haskell code goes, there is obvious potential to exploit features such as infinite series (lazy lists) and polymorphism over the numerical (`Num`) types.

4 Conclusions

Inductive programming (IP) uses the compositional abilities of functional programming (Haskell) and Minimum Message Length (MML) inference. IP 1.2 (§3) now includes the Poisson (§3.1) and Geometric (§3.2) probability distributions, each defined on integers ≥ 0 and parameterised on the distribution mean, and Student's t-Distribution (§3.3) on continuous values.

A small numerical library (§3.5) is also included in IP 1.2 to support tasks such as optimising a function of a *small* number of parameters, for example, fitting a t-Distribution.

The Normal and Multinomial distributions were previously implemented, as were learners for structured-models such as mixtures, classification-trees [3], mixed Bayesian networks [4], and segmentation models [10] which can be used with the new, and earlier, distributions.

References

1. Y. Agusta and D. Dowe. MML clustering of continuous-valued data using Gaussian and t-distributions. *15th Australian Joint AI Conf.*, pages 143–154, 2002.
2. L. Allison. Types and classes of machine learning and data mining. *26th Australasian Computer Science Conference (ACSC)*, pages 207–215, February 2003.
3. L. Allison. Models for machine learning and data mining in functional programming. *J. Functional Programming*, pages 15–32, January 2005. doi:10.1017/S0956796804005301.
4. L. Allison. A programming paradigm for machine learning with a case study of Bayesian networks. *29th Australasian Computer Science Conference (ACSC)*, pages 103–111, February 2006.
5. R. A. Baxter and J. J. Oliver. MDL and MML: Similarities and differences. Technical Report 94/207, Department of Computer Science, Monash University, January 1995.
6. T. Bayes. An essay towards solving a problem in the doctrine of chances. *Phil. Trans. of the Royal Soc. of London*, 53:370–418, 1763. Reprinted in *Biometrika* 45(3/4), pp. 296–315, 1958.
7. M. M. T. Chakravarty and G. Keller. An approach to fast arrays in Haskell. *4th Summer School and Workshop on Advanced Functional Programming*, 2638:27–58, 2002.
8. J. Comley and D. Doew. General Bayesian networks and asymmetric languages. *2nd Hawaii Int. Conf. Statistics and Related Fields (HICS-2)*, June 2003.
9. J. H. Conway and N. J. A. Sloane. *Sphere Packings, Lattices and Groups.* Springer-Verlag, 1988. (See p.61).
10. M. B. Dale, L. Allison, and P. E. R. Dale. Segmentation and clustering as complementary sources of information. *Acta Oecologica*, 31(2):193–202, March 2007.
11. G. E. Farr and C. S. Wallace. The complexity of strict minimum message length inference. *BCS Computer J.*, 45(3):285–292, 2002.
12. N. Friedman and M. Goldszmidt. Learning Bayesian networks with local structure. *UAI'96*, pages 252–262, 1996.
13. K. B. Korb and A. E. Nicholson. *Bayesian Artificial Intelligence.* Chapman and Hall / CRC, 2004.
14. R. T. O'Donnell, L. Allison, and K. B. Korb. Learning hybrid Bayesian networks by MML. *AI 2006: Advances in Artificial Intelligence*, 4304:192–203, 2006. doi:10.1007/11941439_23.
15. S. Peyton-Jones et al. Report on the programming language Haskell-98. 1999. Available at http://www.haskell.org/.
16. J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.
17. D. F. Schmidt and E. Makalic. MMLD inference of the Poisson and geometric models. Technical Report 2008/220, Faculty of Information Technology, Monash University, January 2008.
18. C. E. Shannon. A mathematical theory of communication. *Bell Syst. Technical Jrnl.*, 27:379–423 and 623–656, 1948.
19. Student. The probable error of a mean. *Biometrika*, 6(1):1–25, 1908. (W. S. Gosset writing as Student).
20. C. S. Wallace. *Statistical and Inductive Inference by Minimum Message Length.* Springer-Verlag, 2005. isbn:038723795X.
21. C. S. Wallace and D. M. Boulton. An information measure for classification. *BCS Comput. J.*, 11(2):185–194, 1968.

22. C. S. Wallace and D. Dowe. MML mixture modelling of multi-state, Poisson, von Mises circular and Gaussian distributions. *Proc. 28th Symposium on the Interface*, pages 608–613, 1997.
23. C. S. Wallace and P. R. Freeman. Estimation and inference by compact coding. *J. Royal Statistical Society series B.*, 49(3):240–265, 1987.