Centre for Statistical and Survey Methodology

The University of Wollongong

Working Paper

06-10

Using Infer.NET for Statistical Analyses

S.S.J. WANG & M.P. WAND

Centre for Statistical & Survey Methodology, University of Wollongong, Wollongong NSW 2522. Phone +61 2 4221 5435, Fax +61 2 4221 4845. Email: anica@uow.edu.au

Using Infer.NET for Statistical Analyses

BY S.S.J. WANG & M.P. WAND

Centre for Statistical and Survey Methodology, School of Mathematics and Applied Statistics, University of Wollongong, Wollongong 2522, AUSTRALIA

27th August, 2010

SUMMARY

We demonstrate and critique the new Bayesian inference package Infer.NET in terms of its capacity for statistical analyses. Infer.NET differs from the well-known BUGS Bayesian inference packages in that its main engine is the variational Bayes family of deterministic approximation algorithms rather than Markov chain Monte Carlo. The underlying rationale is that such deterministic algorithms can handle bigger problems due to their increased speed, despite some loss of accuracy. We find that Infer.NET is a well-designed computational framework with intuitive syntax. Nevertheless, the current release is limited in terms of the breadth of models it can handle, and speed of execution on standard hardware platforms.

Keywords: Bayesian inference; Expectation propagation; Mean field approximation; Variational Bayes.

1 Introduction

Infer.NET is a new computational framework for approximate Bayesian inference in hierarchical Bayesian models. The first beta version of Infer.NET was released in December, 2008. Infer.NET can be downloaded from `research.microsoft.com/infernet`. This web-site also contains a two minute mini-clip that explains the essential features of Infer.NET. It points out, for example, that Infer.NET can be used from any of the so-called .NET languages; a family that includes C♯, C++, Visual Basic and IronPython. We work with the first of these languages in the present article. At the time of this writing, the current version of Infer.NET is 2.4 Beta 4 and all advice given in this article is based on that version. Since Infer.NET is in its infancy it is anticipated that new and improved versions will be released quite regularly in the coming years.

Many readers from the statistical community will be familiar with the BUGS (Bayesian inference Using Gibbs Sampling) software products (Spiegelhalter *et al.* 2003), with WinBUGS (Lunn *et al.* 2000) being the most popular of these. Infer.NET is similar to BUGS in that both facilitate the fitting of hierarchical Bayesian models. They differ in their methods for approximate inference. BUGS uses Markov chain Monte Carlo (MCMC) samples from the posterior distributions of parameters of interest. Infer.NET instead uses deterministic approximation methods, known as *variational message passing* and *expectation propagation*, to approximate posterior distributions. Deterministic approximate inference methods have the advantage of being quite fast in comparison with MCMC, and not requiring laborious convergence checks. However, they can be considerably less accurate than MCMC – with the latter having the advantage of improved accuracy through larger samples. Infer.NET has a Gibbs sampling option, which means that it can also perform MCMC-based approximate inference. However, this is for a much narrower class of models compared with WinBUGS.

Variational message passing (VMP) (Winn & Bishop, 2005) is a special case of *variational Bayes* (also known as *mean field approximation*) which, in turn, is a special case of variational approximation. The basic idea of variational Bayes is to approximate joint

posterior densities such as $p(\alpha, \beta, \gamma|\text{observed data})$ by product density forms such as

$$\text{(a)} \quad q_{\alpha,\beta}(\alpha, \beta)\, q_\gamma(\gamma), \quad \text{(b)} \quad q_\alpha(\alpha)\, q_{\beta,\gamma}(\beta, \gamma) \quad \text{or} \quad \text{(c)} \quad q_\alpha(\alpha)\, q_\beta(\beta)\, q_\gamma(\gamma).$$

The choice among (a), (b) and (c) usually involves a trade-off between tractability and minimal imposition of product assumptions. Once the form is settled upon, the optimal q-densities are chosen to minimize the Kullback-Liebler distance from the exact joint posterior. Further details are provided by Bishop (2006, Chapter 10) and Ormerod & Wand (2010). VMP takes advantage of the analytic forms that arise for exponential family distributions with conjugate priors.

Expectation propagation (EP) is a different class of deterministic approximation methods. An early reference is Minka (2001) although similar approaches such as *assumed density filtering* and *moment matching* have a longer history. For certain models, EP has been seen to achieve greater accuracy than VMP (e.g. Bishop, 2006, Section 10.7.1). Other the other hand, models for which EP admits analytic solutions are fewer compared with VMP.

Whilst Infer.NET is geared towards machine learning applications it is, nonetheless, a Bayesian inference engine and therefore has numerous potential statistical applications. In this article we demonstrate and critique the use of Infer.NET in statistical analyses. We start with four elementary examples, and then treat two advanced statistical problems: multivariate classification and additive model analysis. Each example is accompanied by a C♯ script, which is available in a web-supplement to this article.

We find Infer.NET to be a well-structured framework and are attracted by it being script-based environment. However, Infer.NET runs slower on our computers than we expected. In particular, for the examples in Section 2, it is much slower than our homespun R (R Development Core Team, 2010) code for variational Bayes fitting of the same models. Compared with BUGS it is also quite limited in terms of the scope of statistical models it currently can handle.

We introduce the use of Infer.NET for statistical analyses via four simple examples in Section 2. Two advanced examples are described in Section 3. Section 4 explain options for usage of Infer.NET. Some brief comparisons with BUGS are made in Section 5. Our overall evaluation the current version of Infer.NET as a tool for statistical analysis is given in Section 6.

2 Four Simple Examples

We start with four examples involving simple Bayesian models. The first of these is Bayesian simple linear regression. We then describe extensions to (a) binary responses, and (b) random effects. Our last example in this section is concerned with the classical finite normal mixture fitting problem. The simplicity of the examples allows the essential aspects of Infer.NET to be delineated more clearly.

All continuous variables are first transformed to the unit interval and and weakly informative hyperparameter choices are used. The resulting approximate posterior densities are then back-transformed to the original units.

Each example has an accompanying C♯ program containing calls to Infer.NET classes. The full scripts are available as a web-supplement to this article. We highlight the salient code here.

The notation $x_i \overset{\text{ind.}}{\sim} D_i$, $1 \leq i \leq n$, means that the random variables x_i have distribution D_i and are mutually independent. The notation $x \sim \text{Gamma}(A, B)$ means that the random variable x has a Gamma distribution with shape parameter $A > 0$ and rate parameter $B > 0$. The corresponding density function is $p(x) \propto x^{A-1}e^{-Bx}$ for $x > 0$. We use $\boldsymbol{y}$ to denote the $n \times 1$ vector with entries $y_1, \ldots, y_n$. Analogous notation applies to other vectors.

2.1 Simple Linear Regression

Our first example involves the Bayesian simple linear regression model

$$y_i|\beta_0,\beta_1,\tau \overset{\text{ind.}}{\sim} N(\beta_0+\beta_1\,x_i,\tau^{-1}),\quad 1\le i\le n, \tag{1}$$

$$\beta_0,\beta_1 \overset{\text{ind.}}{\sim} N(0,\sigma_\beta^2),\quad \tau\sim\text{Gamma}(A,B), \tag{2}$$

where $\sigma_\beta^2, A, B > 0$ are hyperparameters to be specified by the analyst. Specification of the prior distributions (2) is achieved through the following Infer.NET code:

```
Variable<double> beta0 =
   Variable.GaussianFromMeanAndVariance(0.0,sigsqBeta).Named("beta0");
Variable<double> beta1 =
   Variable.GaussianFromMeanAndVariance(0.0,sigsqBeta).Named("beta1");
Variable<double> tau =
   Variable.GammaFromShapeAndScale(A,1/B).Named("tau");
```

The likelihood (1) is then specified via the loop-type structure:

```
Range index = new Range(n).Named("index");
VariableArray<double> y = Variable.Array<double>(index).Named("y");
VariableArray<double> x = Variable.Array<double>(index).Named("x");
VariableArray<double> mu = Variable.Array<double>(index).Named("mu");
mu[index] = beta0 + beta1*x[index];
y[index] = Variable.GaussianFromMeanAndPrecision(mu[index],tau);
```

The joint posterior density of the model parameters $p(\beta_0,\beta_1,\tau|\boldsymbol{y})$ does not have a closed form expression and Infer.NET will fit the product density approximation

$$p(\beta_0,\beta_1,\tau|\boldsymbol{y})\approx q_{\beta_0}(\beta_0)\,q_{\beta_1}(\beta_1)\,q_\tau(\tau). \tag{3}$$

The milder product density restriction

$$p(\beta_0,\beta_1,\tau|\boldsymbol{y})\approx q_{\beta_0,\beta_1}(\beta_0,\beta_1)\,q_\tau(\tau) \tag{4}$$

can be achieved by treating the regression coefficients as a block; i.e. working with $\boldsymbol{\beta}=[\beta_0\ \beta_1]^T$ rather than the individual coefficients. The required Infer.NET code matches the following alternative expression of (1) and (2):

$$y_i|\boldsymbol{\beta},\tau\sim N(\boldsymbol{\beta}^T\boldsymbol{x}_i,\tau^{-1}),\quad \boldsymbol{\beta}\sim N(\boldsymbol{0},\sigma_\beta^2\boldsymbol{I}),\quad \tau\sim\text{Gamma}(A,B),$$

where $\boldsymbol{x}_i=[1\ x_i]^T$. The prior for $\boldsymbol{\beta}$ is specified via:

```
PositiveDefiniteMatrix SigmaBeta =
            PositiveDefiniteMatrix.IdentityScaledBy(2,sigsqBeta);
Variable<Vector> beta = Variable.VectorGaussianFromMeanAndVariance(
            new Vector(new double[]{0.0,0.0}),SigmaBeta).Named("beta");
```

whilst the likelihood specification is:

```
Range index = new Range(n).Named("index");
VariableArray<double> y = Variable.Array<double>(index).Named("y");
VariableArray<Vector> xvec = Variable.Array<Vector>(index).Named("xvec");
y[index] = Variable.GaussianFromMeanAndPrecision(
           Variable.InnerProduct(beta,xvec[index]),tau);
```

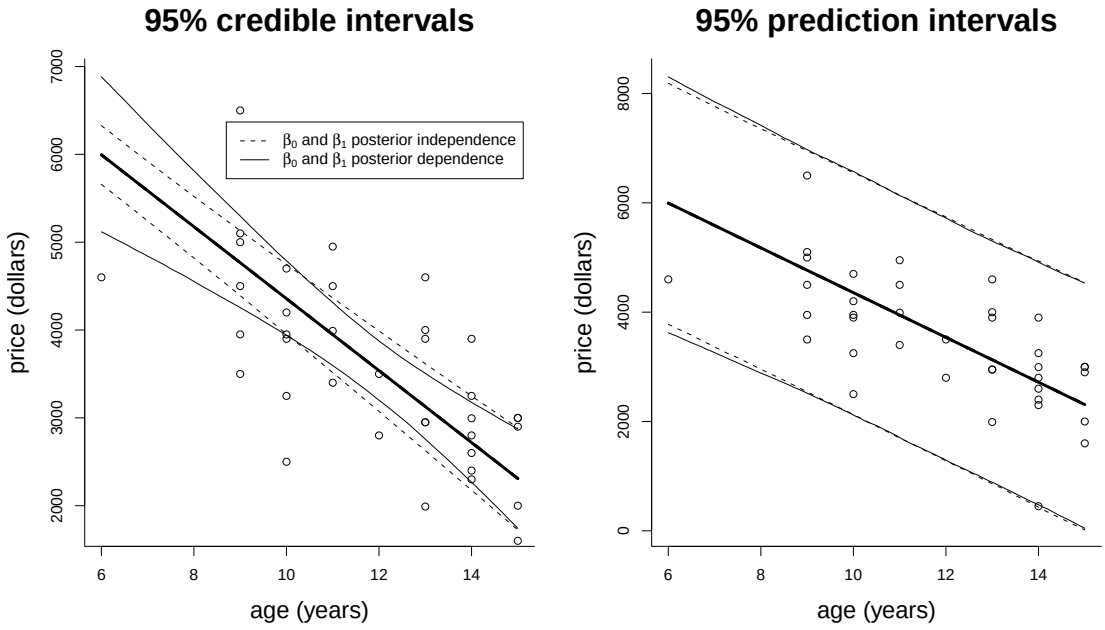

Figure 1: *Fitted regression line, pointwise 95% credible intervals and pointwise 95% Bayesian prediction for data on the age and price of 39 Mitsubishi cars (source: Smith, 1998). The dashed-curve intervals are based on variational Bayes restriction (3). The solid-curve intervals are based on variational Bayes restriction (4).*

Figure 1 displays the fitted regression line, pointwise credible intervals and Bayesian prediction intervals for data on the age and price of $n = 39$ Mitsubishi cars (source: Smith, 1998) with the hyperparameters set at $\sigma^2_{\boldsymbol{\beta}} = 10^8$, $A = B = 0.01$. The approximate posterior densities produced by Infer.NET are shown in Figure 2, but for the error variance $\sigma^2 \equiv 1/\tau$ instead of τ.

Several comments are in order. Firstly, the error variance posterior approximation is unaffected by the type of variational Bayes restriction. But this is far from the case for the regression coefficients. Comparisons with accurate MCMC-based posterior approximations (not shown) shows that variational Bayes approximation (4) is quite accurate in this case. This is to be expected for diffuse independent priors because of orthogonality between $\boldsymbol{\beta}$ and τ in likelihood-based inference. In particular, both variational Bayes approximation (4) and MCMC lead to (for the pre-transformed data)

$$\text{posterior correlation between } \beta_0 \text{ and } \beta_1 \approx -0.92,$$

but variational Bayes approximation (3) forces this value to zero. Hence the posterior covariance matrix of $\boldsymbol{\beta}$ is poorly approximated under (3), leading to strange behaviour in the 95% credible intervals and posterior density functions with incorrect amounts of spread. The prediction intervals are less affected by the differences between (3) and (4) since the error variance posterior contribution dominates.

2.2 Binary Response Regression

Suppose we still observe predictor/response pairs (x_i, y_i) but the $y_i \in \{0, 1\}$. Then appropriate regression models take the form

$$P(y_i = 1|\boldsymbol{\beta}) = F(\boldsymbol{\beta}^T \boldsymbol{x}_i), \qquad \boldsymbol{\beta} \sim N(\mathbf{0}, \sigma^2_\beta \boldsymbol{I}).$$

where $F : \mathbb{R} \to (0, 1)$ is an inverse link function. The most common choices of F correspond to *logistic* regression and *probit* regression. Infer.NET is able to handle both varieties

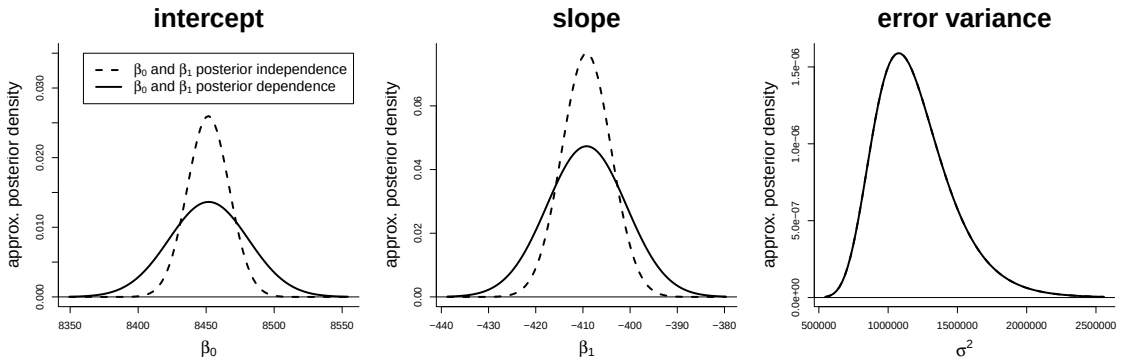

Figure 2: *Variational Bayes approximate posterior density functions produced by Infer.NET for the simple linear regression fit to the Mitsubishi car price/age data. The dashed curves correspond to the variational Bayes restriction of posterior independence between β_0 and β_1, given by (3). The solid curves correspond to a relaxation of this restriction, given by (4). The approximate posterior densities for σ^2 are identical under both (3) and (4).*

of binary response regression, although the most suitable deterministic approximation method differs between the two. Table 1 summarizes this state of affairs.

type	inverse link	approximation method	Infer.NET engine algorithm name
logistic	$e^x/(1+e^x)$	variational Bayes	`VariationalMessagePassing()`
probit	$\Phi(x)$	expectation propagation	`ExpectationPropagation()`

Table 1: *Summary of Infer.NET handling of binary response regression.*

In Infer.NET, both types of binary regression models require that the response variable is converted to be of type Boolean. The likelihood specification for the logistic regression model is:

```
Range index = new Range(n).Named("index");
VariableArray<bool> y = Variable.Array<bool>(index).Named("y");
VariableArray<Vector> xvec = Variable.Array<Vector>(index).Named("xvec");
y[index] = Variable.BernoulliFromLogOdds(
           Variable.InnerProduct(beta,xvec[index]));
```

For probit regression, the last line should be replaced by:

```
y[index] = Variable.IsPositive(Variable.GaussianFromMeanAndVariance(
                                 Variable.InnerProduct(beta,xvec[index]),1));
```

Note that the auxiliary variable version of probit regression (Albert & Chib, 1993) is being used here.
For logistic regression, the inference engine specification is:

```
InferenceEngine engine = new InferenceEngine();
engine.Algorithm = new VariationalMessagePassing();
```

But for probit regression, the `engine.Algorithm` assignment should be:

```
engine.Algorithm = new ExpectationPropagation();
```

Figure 3 shows the fitted probability curves and pointwise 95% credible sets for data on birthweight (grammes) and indicator of bronchopulmonary dysplasia (BPD) (source: Pagano & Gauvreau, 1993) and $\sigma_{\boldsymbol{\beta}}^2 = 10^8$. The two probability curves are very similar, but the credible interval curves differ substantially for higher birthweights.

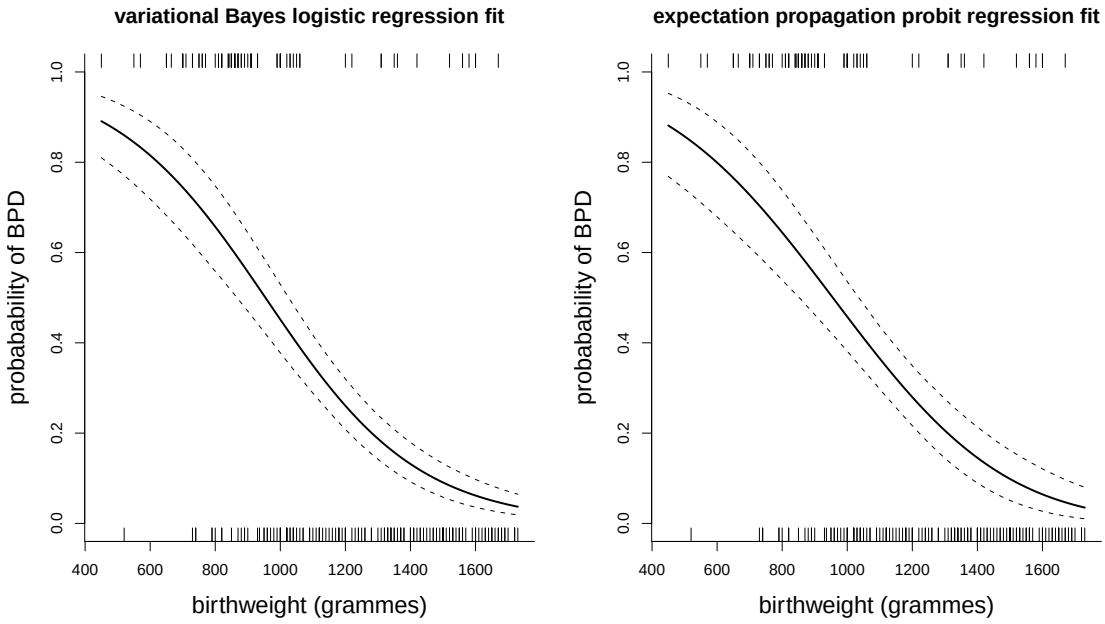

Figure 3: *Binary response regression fits to the bronchopulmonary dysplasia (BPD) data obtained using* Infer.NET*. The solid line is the posterior probability of BPD for a given birthweight. The dashed lines are pointwise 95% credible sets. Left panel: variational Bayes logistic regression. Right panel: expectation propagation probit regression.*

2.3 Random Intercept Model

Our second extension of the linear regression model involves the addition of a random intercept, and represents a simple example of a *mixed* model for grouped data. Specifically, we now consider models of the form

$$\begin{aligned} y_{ij}|\boldsymbol{\beta}, u_i, \tau_\varepsilon &\overset{\text{ind.}}{\sim} N(\boldsymbol{\beta}^T \boldsymbol{x}_{ij} + u_i, \tau_\varepsilon^{-1}), \quad u_1, \ldots, u_m|\tau_u \overset{\text{ind.}}{\sim} N(\mathbf{0}, \tau_u^{-1}) \\ \boldsymbol{\beta} &\sim N(\mathbf{0}, \sigma_\beta^2 \boldsymbol{I}), \quad \tau_u \sim \text{Gamma}(A_u, B_u), \quad \tau_\varepsilon \sim \text{Gamma}(A_\varepsilon, B_\varepsilon) \end{aligned} \tag{5}$$

where y_{ij} is the jth response measurement in the ith group and m is the number of groups. Note that u_i is a random intercept specific to the ith group. Let $\boldsymbol{u}$ denote the vector of u_is.

Figure 4 shows the approximate posterior density obtained from Infer.NET with the variational Bayes product assumption:

$$q_{\boldsymbol{\beta},\boldsymbol{u},\tau_u,\tau_\varepsilon}(\boldsymbol{\beta}, \boldsymbol{u}, \tau_u, \tau_\varepsilon) = q_{\boldsymbol{\beta},\boldsymbol{u}}(\boldsymbol{\beta}, \boldsymbol{u})\, q_{\tau_u,\tau_\varepsilon}(\tau_u, \tau_\varepsilon). \tag{6}$$

(It turns out that the *induced* factorization $q_{\tau_u,\tau_\varepsilon}(\tau_u, \tau_\varepsilon) = q_{\tau_u}(\tau_u)\, q_{\tau_\varepsilon}(\tau_\varepsilon)$ arises in the optimal solution.) The input data correspond to four longitudinal orthodontic measurements on each of $m = 27$ children (source: Pinheiro & Bates, 2000). The data are available in the R computing environment (R Development Core Team, 2010) via the package nlme (Pinheiro *et al.* 2009), in the object `Orthodont`. The y_{ij} correspond to distances from the pituitary to the pterygomaxillary fissure (mm) and

$$\boldsymbol{\beta}^T \boldsymbol{x}_{ij} = \beta_0 + \beta_1 \texttt{age}_{ij} + \beta_2 \texttt{male}_i$$

where $\texttt{age}_{ij}$ is the age of the child when y_{ij} was recorded and $\texttt{male}_i$ is an indicator variable for the child being male. Note that the posteriors densities are for the variances $\sigma_u^2 \equiv 1/\tau_u$ rather than $\sigma_\varepsilon^2 \equiv 1/\tau_\varepsilon$.

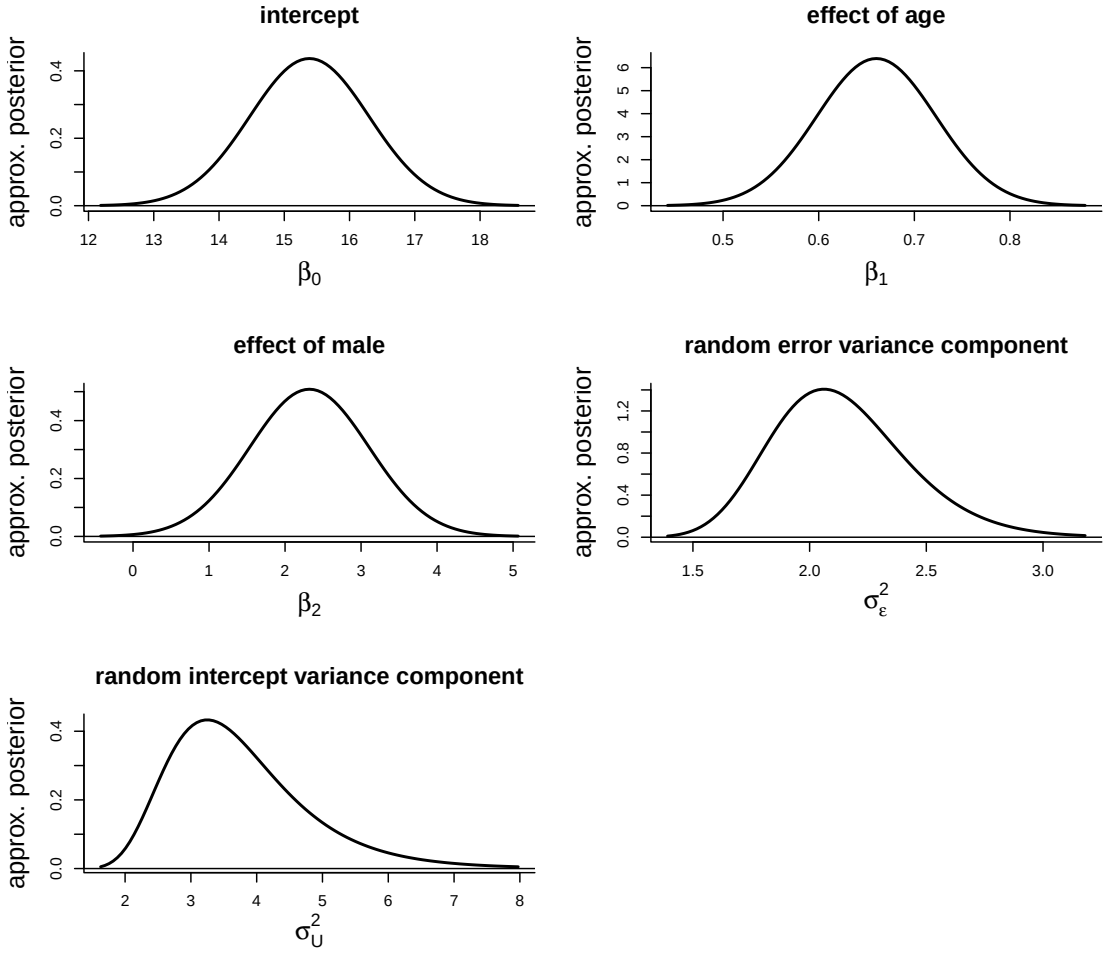

Figure 4: *Variational Bayes approximate posterior density functions produced by Infer.NET for the simple linear mixed model fit to the orthodontic data.*

The results shown in Figure 4 actually correspond to a slight modification of model (5), since Infer.NET does not support its direct fitting under product restriction (6). We will now explain, and justify, the modified model. First note that (5) can be written in matrix form as:

$$\boldsymbol{y}|\boldsymbol{\beta},\boldsymbol{u},\tau_\varepsilon \sim N(\boldsymbol{X}\boldsymbol{\beta}+\boldsymbol{Z}\boldsymbol{u},\tau_\varepsilon^{-1}), \quad \left[\begin{array}{c}\boldsymbol{\beta}\\ \boldsymbol{u}\end{array}\right] \Big| \, \tau_u \sim N\left(\boldsymbol{0},\left[\begin{array}{cc}\sigma_\beta^2\boldsymbol{I} & \boldsymbol{0}\\ \boldsymbol{0} & \tau_u^{-1}\boldsymbol{I}\end{array}\right]\right), \tag{7}$$

$$\tau_u \sim \text{Gamma}(A_u,B_u), \quad \tau_\varepsilon \sim \text{Gamma}(A_\varepsilon,B_\varepsilon)$$

where $\boldsymbol{X}$ contains the $\boldsymbol{x}_{ij}$ and $\boldsymbol{Z} = \boldsymbol{I}_{27} \otimes \boldsymbol{1}_4$ is the indicator matrix for matching the $\boldsymbol{x}_{ij}$s with their corresponding u_i ($1 \le i \le 27, 1 \le j \le 4$). Note that $\boldsymbol{1}_4$ is the 4×1 vector of ones. The posterior density function of $\boldsymbol{\beta}, \boldsymbol{u}$ satisfies

$$p(\boldsymbol{\beta},\boldsymbol{u}|\boldsymbol{y}) \propto \int_0^\infty \int_0^\infty \exp\left\{-\tfrac{1}{2}\,\tau_\varepsilon\|\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta}-\boldsymbol{Z}\boldsymbol{u}\|^2 - \tfrac{1}{2\sigma_\beta^2}\|\boldsymbol{\beta}\|^2 - \tfrac{1}{2}\tau_u\|\boldsymbol{u}\|^2\right\} p(\tau_\varepsilon)p(\tau_u)\,d\tau_\varepsilon\,d\tau_u$$

where $p(\tau_\varepsilon)$ and $p(\tau_u)$ are the posterior density functions of τ_ε and τ_u. An alternative

model, which employs an auxiliary data vector $\boldsymbol{a}$, is:

$$\begin{gathered}\boldsymbol{y}|\boldsymbol{\beta},\boldsymbol{u},\tau_\varepsilon \sim N(\boldsymbol{X}\boldsymbol{\beta}+\boldsymbol{Z}\boldsymbol{u},\tau_\varepsilon^{-1}), \quad \boldsymbol{a}|\boldsymbol{\beta},\boldsymbol{u},\tau_u \sim N\left(\begin{bmatrix}\boldsymbol{\beta}\\ \boldsymbol{u}\end{bmatrix}, \begin{bmatrix}\sigma_\beta^2\boldsymbol{I} & \boldsymbol{0}\\ \boldsymbol{0} & \tau_u^{-1}\boldsymbol{I}\end{bmatrix}\right), \\ \begin{bmatrix}\boldsymbol{\beta}\\ \boldsymbol{u}\end{bmatrix} \sim N\left(\boldsymbol{0},\kappa^{-1}\boldsymbol{I}\right), \qquad \tau_u \sim \text{Gamma}(A_u,B_u), \quad \tau_\varepsilon \sim \text{Gamma}(A_\varepsilon,B_\varepsilon).\end{gathered} \tag{8}$$

where $\kappa > 0$ is a hyperparameter. Let $p_\kappa(\boldsymbol{\beta},\boldsymbol{u}|\boldsymbol{y},\boldsymbol{a})$ be the posterior density function of $(\boldsymbol{\beta},\boldsymbol{u})$ under model (8). Then

$$\begin{aligned}p_\kappa(\boldsymbol{\beta},\boldsymbol{u}|\boldsymbol{y},\boldsymbol{a}) \propto \int_0^\infty \int_0^\infty \exp\Big\{ &-\tfrac{1}{2}\,\tau_\varepsilon\|\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta}-\boldsymbol{Z}\boldsymbol{u}\|^2 - \tfrac{1+\kappa\sigma_\beta^2}{2\sigma_\beta^2}\|\boldsymbol{\beta}\|^2 \\ &-\tfrac{1}{2}\tau_u(1+\kappa/\tau_u)\|\boldsymbol{u}-\boldsymbol{a}\|^2\Big\} p(\tau_\varepsilon)p(\tau_u)\,d\tau_\varepsilon\,d\tau_u.\end{aligned}$$

It is apparent from this that

$$\lim_{\kappa\to 0} p_\kappa(\boldsymbol{\beta},\boldsymbol{u}|\boldsymbol{y},\boldsymbol{a}=\boldsymbol{0}) = p(\boldsymbol{\beta},\boldsymbol{u}|\boldsymbol{y}).$$

Similar results hold for the other posterior density functions. Hence, using (8) with κ set to be a very small number with the auxiliary vector $\boldsymbol{a}$ set to have an observed value $\boldsymbol{0}$ leads to essentially the same results. Figure 5 provides directed acyclic graph representations of the original and modified models.

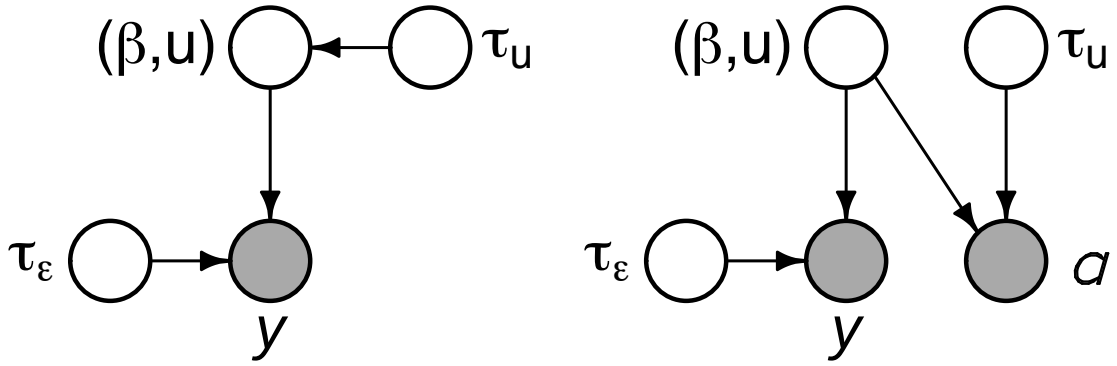

Figure 5: *Directed acyclic graph representations of the original Gaussian linear mixed model (left panel) and a modification for implementation in* Infer.NET. *In the modified model the* $(\boldsymbol{\beta},\boldsymbol{u})$ *node is Gaussian, with mean set to zero and covariance set to a very large multiple of the identity matrix. The shaded nodes correspond to observed data vectors. The* $\boldsymbol{a}$ *node is set to be a vector of zeroes.*

2.4 Normal Mixture Model

The fourth simple example involves fitting a density function of the form

$$f(x) = \sum_{k=1}^K w_k(2\pi)^{-1/2}\tau_k \exp\{-\tfrac{1}{2}\,\tau_k(x-\mu_k)^2\}, \quad w_k,\tau_k > 0 \quad \sum_{k=1}^K w_k = 1 \tag{9}$$

to a univariate sample $\boldsymbol{x} = (x_1,\ldots,x_n)$. This is the classic finite normal mixture problem and has an enormous literature (e.g. McLachlan & Peel, 2000). A variational Bayes algorithm for fitting (9) is described in Section 2.2.5 of Ormerod & Wand (2010) and a description of an appropriate Bayesian model is given there. We describe Infer.NET fitting

of the same model here, but with the addition of choosing the number of components K. Following Section 10.2.4 of Bishop (2006), we choose K to maximize

$$\log \underline{p}(\boldsymbol{x}; q) + \log(K!)$$

where $\log \underline{p}(\boldsymbol{x}; q)$ is the variational Bayes approximation to the marginal log-likelihood. The $\log(K!)$ term accounts for the $K!$ configurations of (w_k, μ_k, σ_k^2) that give rise to the same normal mixture density function.

Infer.NET can compute $\log \underline{p}(\boldsymbol{x}; q)$ by creating a mixture of the current model with an empty model. The learnt mixing weight is then the marginal log-likelihood. Further details on this trick are given in the Infer.NET user guide, where the term *model evidence* is used for $\log \underline{p}(\boldsymbol{x}; q)$. We first need to set up an auxiliary Bernoulli variable as follows:

```
Variable<bool> auxML = Variable.Bernoulli(0.5).Named("auxML");
```

The code for the normal mixture fitting is then enclosed with:

`IfBlock model = Variable.If(auxML);` and `model.CloseBlock();`

The $\log \underline{p}(\boldsymbol{x}; q)$ is then obtained from:

```
double marginalLogLikelihood = engine.Infer<Bernoulli>(auxML).LogOdds;
```

Figure 6 shows the results of this Infer.NET-based approach to fitting a finite normal mixture to the data on eruption durations of a geyser and are available in R via the `MASS` package (Venables and Ripley, 2010), in the data-frame titled `geyser`.

Note that $K = 3$ maximizes $\log \underline{p}(\boldsymbol{x}; q) + \log(K!)$, as shown in the upper panel of Figure 6. The lower panel shows the $K = 3$ Infer.NET fit. Also shown are 95% pointwise credible sets based on Monte Carlo samples of size 10000 from the approximate posterior distributions.

3 Two Advanced Examples

We now describe two advanced examples which illustrate the capabilities of Infer.NET for more challenging data analyses. The first concerns multi-dimensional classification. The second involves fitting an additive model with low-rank smoothing splines.

3.1 Classification via Multivariate Finite Mixtures

This example involves classification of glass samples into one of two types based on their refractive index and chemical content. The training data are from the data frame `Glass` in the R package mlbench (Leisch & Dimitriadou, 2009). The full data set involves 7 types of glass and 9 predictor variables. The first two types (Type 1 and Type 2) comprise 68% of the data with training set sizes of 70 and 76, respectively. We restrict attention to these two classes and use the first seven predictors to build a classifier. These predictors are: refractive index (RI) and content measures of sodium (Na), magnesium (Mg), aluminum (Al), silicon (Si), potassium (K) and calcium (Ca). We fit three-mixture multivariate normal density function to each sample:

$$f(\boldsymbol{x}) = \sum_{k=1}^{3} w_k (2\pi)^{-7/2} |\boldsymbol{T}_k|^{1/2} \exp\{-\tfrac{1}{2} (\boldsymbol{x} - \boldsymbol{\mu}_k)^T \boldsymbol{T}_k (\boldsymbol{x} - \boldsymbol{\mu}_k)\}$$

where $w_1, w_2, w_3 > 0$ and $w_1 + w_2 + w_3 = 1$. For each of $1 \le k \le 3$, the $\boldsymbol{\mu}_k$ and 7×1 vectors and the $\boldsymbol{T}_k$ are symmetric positive definite 7×7 matrices. As priors we used

$$(w_1, w_2, w_3) \sim \text{Dirichlet}\,(\alpha, \alpha, \alpha), \quad \boldsymbol{\mu}_k \overset{\text{ind.}}{\sim} N(\boldsymbol{0}, \sigma_\mu^2 \boldsymbol{I}) \quad \text{and} \quad \boldsymbol{T}_k \overset{\text{ind.}}{\sim} \text{Wishart}(a, \boldsymbol{B}).$$

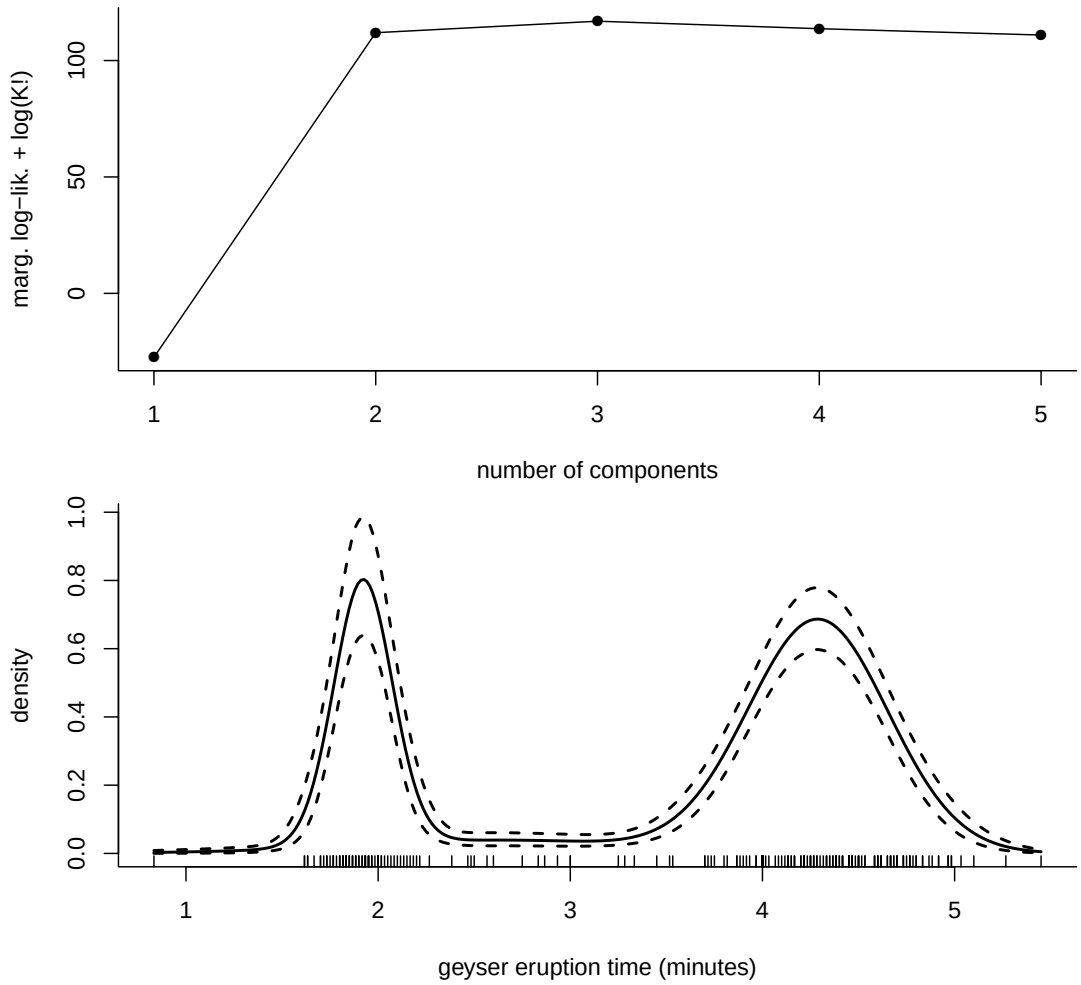

Figure 6: *Upper panel:* $\log \underline{p}(\boldsymbol{x}; q) + \log(K!)$ *versus* K*, where* K *is the number of components in the normal mixture fit to the transformed geyser duration data. Lower panel: fitted normal mixture density for* $K = 3$ *(the* K *that maximizes the criterion of the upper panel plot). The dashed curves correspond to pointwise 95% credible sets.*

The Dirichlet and Inverse-Wishart notation is such that the respective density functions take the form $p(w_1, w_2, w_3) \propto (w_1 w_2 w_3)^{\alpha-1}$ for $w_1, w_2, w_3 > 0$ and $w_1 + w_2 + w_3 = 1$, and

$$p(\boldsymbol{T}_k) \propto |\boldsymbol{T}_k|^{(a-d-1)/2} \exp\{-\tfrac{1}{2}\mathrm{tr}(\boldsymbol{B}\boldsymbol{T}_k)\}, \quad a > 0,\ \boldsymbol{T}_k, \boldsymbol{B} \text{ both positive definite}$$

with $d = 7$. The hyperparameters were set at

$$\alpha = 0.001, \quad a = 100 \quad \text{and} \quad \boldsymbol{B} = 0.01\boldsymbol{I}_7.$$

The Infer.NET code for fitting these to the glass data is similar to that used for fitting the univariate normal mixture models to the geyser data, as described in Section 2.4. One difference is the use of Wishart distributions, rather than Gamma distributions, in the precision matrix prior specification:

```
Range indexK = new Range(K).Named("indexK");
VariableArray<PositiveDefiniteMatrix>
   Tau = Variable.Array<PositiveDefiniteMatrix>(indexK).Named("Tau");
Tau[indexK] = Variable.WishartFromShapeAndScale(aVal,
   PositiveDefiniteMatrix.IdentityScaledBy(d,Bfac)).ForEach(indexK);
```

Figure 7 shows the training data and their classifications according to the Infer.NET normal mixture fits. The training error was 25.3%. An estimate of the test error, obtained using 5-fold cross-validation, is 24.6% with a standard deviation of 9%.

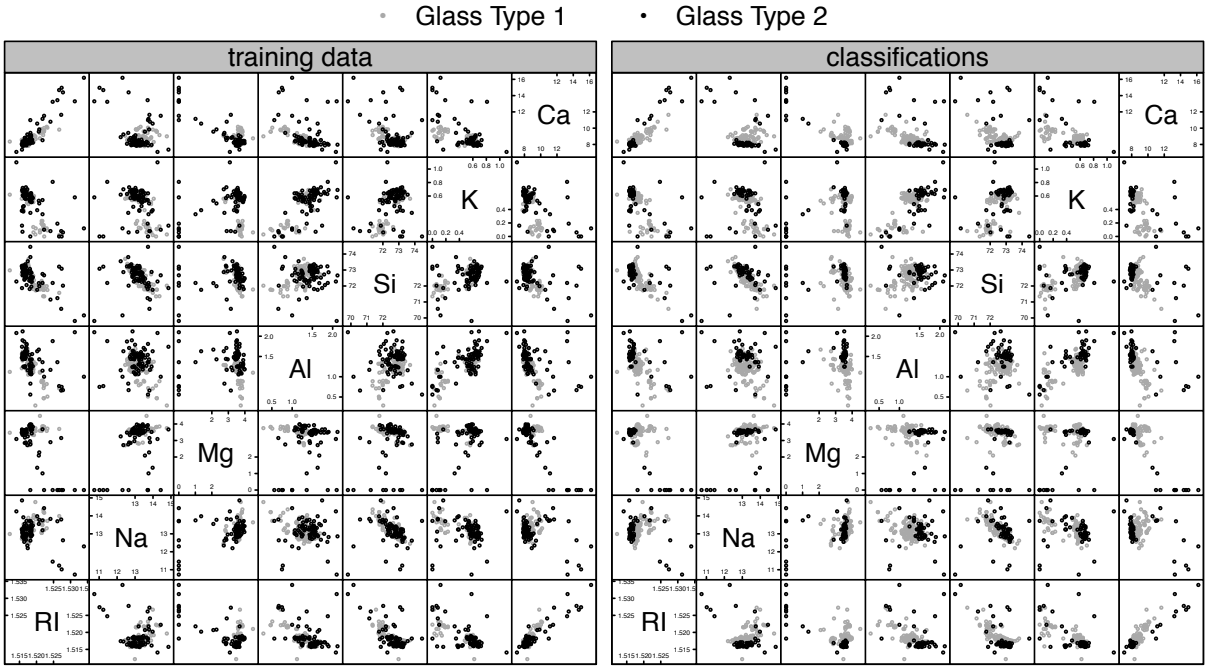

Figure 7: *Left: pairwise scatterplots for the training data in the glass classification example described in the text. Right: the classifications of the training data based on Infer.NET fitting of multivariate normal mixture densities to each of the Glass Type 1 and Glass Type 2 samples.*

3.2 Normal Additive Model

A three-predictor Normal additive model is

$$y_i = \beta_0 + f_1(x_{1i}) + f_2(x_{2i}) + f_3(x_{3i}) + \varepsilon_i, \quad \varepsilon_i \overset{\text{ind.}}{\sim} N(0, \sigma_\varepsilon^2), \tag{10}$$

where, for $1 \leq i \leq n$, the y_i are measurements on a continuous response variable and (x_{1i}, x_{2i}, x_{3i}) are triples containing measurements on three continuous predictor variables. We will model each of the f_j using low-rank smoothing splines with mixed model representation:

$$f_j(x) = \beta_j\, x + \sum_{k=1}^{K_j} u_k z_k^{(j)}(x), \quad u_k \overset{\text{ind.}}{\sim} N(0, \sigma_{uj}^2)$$

where $z_k^{(j)}$ is a set of canonically transformed cubic B-spline basis functions on an interval containing the x_{ji}. Details on computation of the $z_k^{(j)}$ are given in Wand & Ormerod (2008). This model for the f_j corresponds to the type of function estimation performed by the R function `smooth.spline()`. The Bayesian model that we fit in Infer.NET is then:

$$\boldsymbol{y}|\boldsymbol{\beta}, \boldsymbol{u}, \tau_\varepsilon, \tau_{u1}, \tau_{u2}, \tau_{u3} \sim N(\boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{Z}\boldsymbol{u}, \tau_\varepsilon^{-1}\boldsymbol{I}),$$

$$\boldsymbol{u}|\tau_{u1}, \tau_{u2}, \tau_{u3} \sim N\left(\boldsymbol{0}, \begin{bmatrix} \tau_{u1}^{-1}\boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \tau_{u2}^{-1}\boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \tau_{u3}^{-1}\boldsymbol{I} \end{bmatrix}\right),$$

$$\boldsymbol{\beta} \sim N(\boldsymbol{0}, \sigma_\beta^2\boldsymbol{I}), \quad \tau_\varepsilon \sim \text{Gamma}(A_\varepsilon, B_\varepsilon) \quad \tau_{uj} \sim \text{Gamma}(A_{uj}, B_{uj}), \quad j = 1, 2, 3,$$

where

$$\boldsymbol{X} = [1\ x_{1i}\ x_{2i}\ x_{3i}]_{1\leq i\leq n} \qquad \text{and} \qquad \boldsymbol{Z} = [z_k^{(1)}(x_{1i}) \underset{1\leq k\leq K_1}{\mid} z_k^{(2)}(x_{2i}) \underset{1\leq k\leq K_2}{\mid} z_k^{(3)}(x_{3i})]_{1\leq i\leq n}.$$

We fit this model to variables in the Ozone data frame in the R package mlbench (Leisch & Dimitriadou, 2009). The response variable is daily maximum ozone level and the predictor variables are the inversion base height (feet), pressure gradient to the town of Daggett (mm Hg) and inversion base temperature (degrees Fahrenheit) at Los Angeles International Airport. All variables were transformed to the unit interval for Infer.NET fitting and the hyperparameters were fixed at $\sigma_\beta^2 = 10^8$, $A_\varepsilon = B_\varepsilon = A_{uj} = B_{uj} = 0.01$.

The Infer.NET code is similar to that used for the random intercept model analysis of Section 2.3. The only difference is the presence of three random effect dispersion parameters (i.e. τ_{u1}, τ_{u2} and τ_{u3}) rather than one. Note that the trick encompassed in model (8) also needs to employed for current example.

The fitted curves are shown in Figure 8 and show interesting non-linear effects. Similar fits are obtained using standard additive model software such as the `gam()` function in the mgcv package (Wood, 2010).

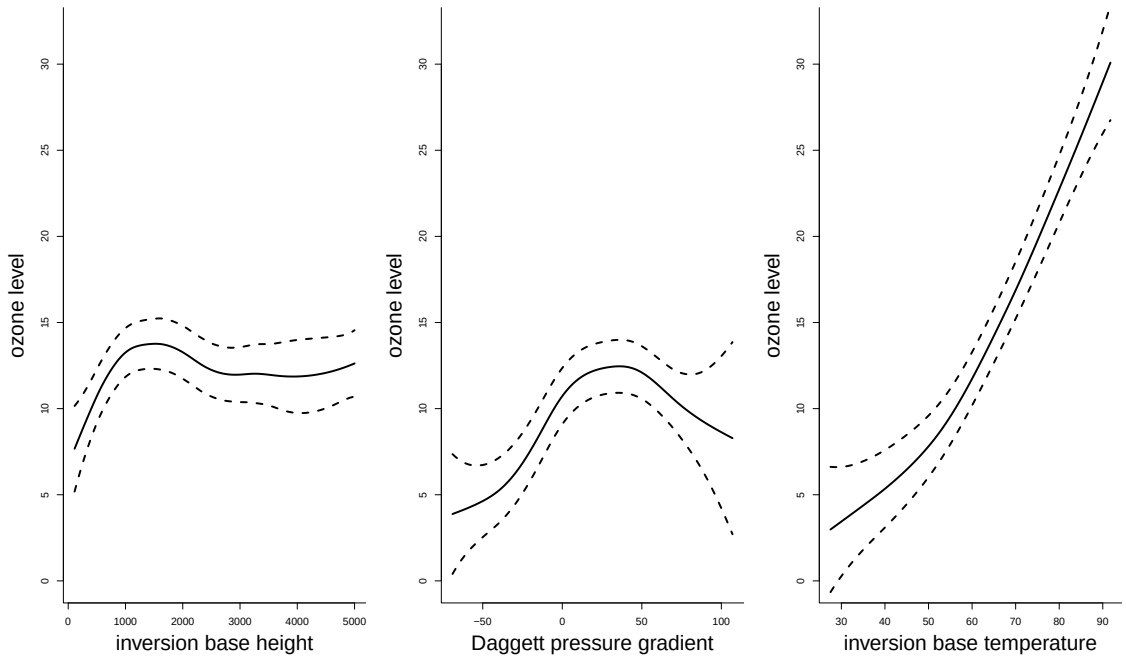

Figure 8: *Variational Bayes additive model fits as produced by Infer.NET for California ozone data. The dashed curves correspond to point-wise 95% credible sets.*

We have explored the generalized response extension of (10), known as *generalized additive models*, in Infer.NET. In principle, binary response models can be handled by combining the concepts of Sections 2.2 and 3.2. However, the examples we have tried to date take several hours to run.

4 How to Use Infer.NET

The core of Infer.NET is a suite of computer programmes compatible with the .NET family of languages, so can be accessed in a variety of ways. In the preparation of this article we have used Infer.NET in two different ways. We now provide brief descriptions.

The first way involves the graphical user interface known as Visual Studio 2008. This facilitates the development of C♯ scripts with calls to various Infer.NET classes. A major advantage of Visual Studio 2008 is its debugging feature, which allows for the efficient development of large and complex scripts. Other features include the ability to generate directed acyclic graphs for models defined by the Infer.NET code and easy access to help pages. The Infer.NET web-site has further details on this approach.

There is, however, the option to simply write C♯ in a preferred text editor and run the code via DOS commands and tailor-made scripts.We found this approach to be very useful for running the examples in Sections 2.2 and 3.2 since we also used R for pre-processing of the data and post-processing of the Infer.NET output. This second approach allowed us to conduct each Infer.NET-based analysis with a single R script and corresponding C♯ script.

5 Comparison with BUGS

Of particular interest to statistical analysts is how Infer.NET compares with BUGS. Such comparison is challenging because of factors such as deciding when a particular algorithm has converged and differences in personal tastes with regard to user interface and syntax. Nevertheless, we briefly make some comparative points concerning accuracy and computation time. These are based solely on the examples of Sections 2 and 3 and, hence, are somewhat limited in their scope.

Infer.NET is inherently inaccurate since it relies on deterministic approximation methods. The first two panels of Figure 2 show that the posterior density functions produced by Infer.NET can be overly narrow for stringent product density restrictions. BUGS, on the other hand, will produce highly accurate posterior density functions if the MCMC sample sizes are sufficiently large. Comparisons with MCMC results show Infer.NET to be quite good for the Normal response regression examples in Sections 2 and 3, provided that the regression coefficients are treated as a block. For the binary response examples of Section 2.2 some inaccuracy is apparent. For example, in the logistic regression example, the population standard deviation of $q^*(\beta_1)$ is 74% of the sample standard deviation of the corresponding MCMC sample (based on a sample size of 100,000).

Table 2 gives some indication of the relative computing times for the examples of Sections 2 and 3. Making these comparisons completely fair is a tall order, since the point at which convergence occurs for underlying approximation algorithms is contentious and problem dependent. Instead, we just report elapsed computing times with the number of variational Bayes or expectation propagation iterations set to 100 and the MCMC sample sizes set at 10000, which was sufficient for convergence in these particular examples. All examples were run on second author's laptop computer (Mac OS X; 2.33 GHz processor, 3 GBytes of random access memory)

Example from Sections 2 and 3	comput. time for Infer.NET (100 VB/EP iterations)	comput. time for BUGS (MCMC samp. size=100000)
Simple linear regression	5 seconds	1 second
Simple logistic regression	5 seconds	5 seconds
Simple probit regression	5 seconds	4 seconds
Random intercept model	38 seconds	2 seconds
Normal mixture model	32 seconds	44 seconds
Multivariate classification	21 seconds	96 seconds
Normal additive model	9.4 minutes	5.2 minutes

Table 2: *Computation times in both Infer.NET and BUGS for each of the examples in Sections 2 and 3 on the second authors' laptop computer (Mac OS X; 2.33 GHz processor, 3 GBytes of random access memory). The times for Infer.NET involve 100 variational Bayes of expectation propagation iterations. The times for BUGS involve MCMC sample sizes of 10000.*

Table 2 reveals that the speed-ups offered by deterministic approximation methods such as variational Bayes are not realized in the current release of Infer.NET. Since speed is the main selling points of deterministic approximation methods, in comparison with

MCMC, Table 2 would indicate that BUGS is the better option at present.

6 Summary

We believe that software engines for fast approximate inference, using deterministic methods such as variational Bayes and expectation propagation, will play a big role in statistical analyses of the future. They have the potential to handle the large and ever-growing data sets and models in the current era of rapid technological change. The emergence of Infer.NET, with its script basis and carefully designed syntax, is an important early development in this direction.

Our examples have demonstrated that Infer.NET can be used, effectively, for both basic and advanced statistical analyses. However, users of the current release should be aware of the following limitations:

- The proportion of statistical models in current popular use that can be handled using Infer.NET is relatively low. For example, Poisson response analogues of the regression examples in Sections 2 and 3 are not supported by Infer.NET. Non-conjugate priors distributions, such as Half-t priors on standard deviation parameters (e.g. Gelman, 2006), are generally not allowed.

- The actual execution times are quite slow and on par with BUGS fitting of the same model. Since MCMC does not suffer from the approximation error inherent in deterministic approximation methods such as variational Bayes there is little to be gained from using the current release of Infer.NET rather than BUGS. As mentioned in the introduction, simple R implementation of variational Bayes algorithms for the examples of Sections 2 and 3 is much faster. We acknowledge that Infer.NET strives for a high level of generality we but we are, nonetheless, surprised at its relative slowness.

We hope that this article will lead to useful discourse on the confluence between Infer.NET and statistical analyses.

Acknowledgments

We are grateful for advice received from Thomas Minka and John Ormerod during the course of this project.

Authors' Note

We have prepared a web-supplement containing each of the C♯ scripts for the examples in this article. Its dissemination will depend on the journal where this article is eventually published. In the meantime, the scripts can be obtained by e-mailing the second author. His current e-mail address is `mwand@uow.edu.au`.

References

Albert, J.H. and Chib, S. (1993). Bayesian analysis of binary and polychotomous response data. *Journal of the American Statistical Association*, **88**, 669–679.

Bishop, C.M. (2006). *Pattern Recognition and Machine Learning.* New York: Springer.

Gelman, A. (2006). Prior distributions for variance parameters in hierarchical models. *Bayesian Analysis*, **1**, 515–533.

Leisch, F. & Dimitriadou, E. (2009). mlbench 1.1. A collection of artificial and real-world machine learning problems. R package. `http://cran.r-project.org`

Lunn, D.J., Thomas, A., Best, N. & Spiegelhalter, D. (2000). WinBUGS – a Bayesian modelling framework: concepts, structure, and extensibility. *Statistics and Computing*, **10**, 325–337.

McLachlan, G.J. & Peel, D. (2000). *Finite Mixture Models*. New York: Wiley.

Minka, T.P. (2001). Expectation propagation for approximate Bayesian inference. In *Proceedings of Conference on Uncertainty in Artificial Intelligence*, 362–369.

Minka, T., Winn, J., Guiver, G. & Kannan, A. (2008). Infer.Net, Microsoft Research Cambridge, Cambridge, UK.

Ormerod, J.T. and Wand, M.P. (2010). Explaining variational approximations. *The American Statistician*, **64**, 140–153.

Pagano, M. and Gauvreau, K. (1993). *Principles of Biostatistics*. Florence, Kentucky: Duxbury.

Pinheiro, J.C. and Bates, D.M. (2000). *Mixed-Effects Models in S and S-PLUS*. New York: Springer.

Pinheiro, J., Bates, D., DebRoy, S., Sarkar, D. & the R Core team. (2009). nlme: linear and nonlinear mixed effects models. R package version 3.1-93.

R Development Core Team (2010). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, `http://www.R-project.org`

Smith, P. (1998). *Into Statistics: A Guide to Understanding Statistical Concepts in Engineering and the Sciences, Second Edition*. Singapore: Springer-Verlag.

Spiegelhalter, D.J., Thomas, A., Best, N.G., Gilks, W.R. & Lunn, D. (2003). BUGS: Bayesian inference using Gibbs sampling. Medical Research Council Biostatistics Unit, Cambridge, UK, `http://www.mrc-bsu.cam.ac.uk/bugs`.

Venables, W.N. & Ripley, B.D. (2009). MASS: functions and datasets to support Venables and Ripley, 'Modern Applied Statistics with S' (4th edition). R package version 7.2-48.

Wand, M.P. and Ormerod, J.T. (2008). On semiparametric regression with O'Sullivan penalized splines. *Australian and New Zealand Journal of Statistics*, **50**, 179–198.

Winn, J. & Bishop, C.M. (2005). Variational message passing. *Journal of Machine Learning Research*, **6**, 661–694.

Wood, S.N. (2010). mgcv 1.5. Routines for generalized additive models and other generalized ridge regression with multiple smoothing parameter selection. R package. `http://cran.r-project.org`