Probabilistic Networks: New Models and New Methods

David J.C. MacKay
Cavendish Laboratory
Madingley Road, Cambridge CB3 0HE
United Kingdom
mackay@mrao.cam.ac.uk

To appear in Proceedings of ICANN'95

Abstract

In this paper I describe the implementation of a probabilistic regression model in BUGS. BUGS is a program that carries out Bayesian inference on statistical problems using a simulation technique known as Gibbs sampling. It is possible to implement surprisingly complex regression models in this environment. I demonstrate the simultaneous inference of an interpolant and an input-dependent noise level.

1 Traditional regression models and their Bayesian interpretation

In traditional regression methods, the interpolant $y(x)$ is represented as a parameterized function $y(x;\mathbf{w})$, and, given data $\{x_n, t_n\}_{n=1}^{N}$, the parameters $\mathbf{w}$ are optimized to minimize the weighted sum of an error function $E_D(\mathbf{w}) = \sum_{n=1}^{N}(t_n - y_n(x_n;\mathbf{w}))^2/2$, and a regularizer $E_W(\mathbf{w})$:

$$M(\mathbf{w}) = \beta \sum_{n=1}^{N}(t_n - y_n(x_n;\mathbf{w}))^2/2 + \alpha E_W(\mathbf{w}) \tag{1}$$

The coefficients α and β are known as hyperparameters, and the regularizer E_W is a function which is smallest for parameter vectors $\mathbf{w}$ that correspond to smooth functions.

In the Bayesian interpretation (reviewed in (MacKay 1992a)), this optimization maps onto a probabilistic model $\mathcal{H}$ where the function $-\beta E_D$ is the log of a likelihood function, $P(\{t_n\}|\mathbf{w},\beta,\mathcal{H}) = \exp(-\beta E_D)/Z_D(\beta)$, and $-\alpha E_W$ is the log of a prior distribution on the parameters, $P(\mathbf{w}|\alpha,\mathcal{H}) = \exp(-\alpha E_W)/Z_W(\alpha)$, so that the minimization of $M(\mathbf{w})$ corresponds to the maximization of the posterior probability:

$$P(\mathbf{w}|\{t_n\},\alpha,\beta,\mathcal{H}) = \frac{P(\{t_n\}|\mathbf{w},\beta,\mathcal{H})P(\mathbf{w}|\alpha,\mathcal{H})}{P(\{t_n\}|\alpha,\beta,\mathcal{H})}. \tag{2}$$

The hyperparameter β defines a noise variance $\sigma_\nu^2 = 1/\beta$, and the model $\mathcal{H}$ assumes that the residuals $(t-y)$ between the data and the unknown function are independent and identically distributed Gaussian variables. If E_W is a quadratic function of $\mathbf{w}$ then the hyperparameter α

defines a variance for the parameters w. An important problem in regression modelling is to control the relative strength of these two hyperparameters. If α/β is too large then the model cannot fit the data well, but if the ratio is too small then overfitting occurs. As reviewed in (MacKay 1992a), we can extend this probabilistic model by including a prior distribution over the hyperparameters $P(\alpha, \beta|\mathcal{H})$ that expresses our lack of knowledge about the noise variance and the variance of the parameters $\mathbf{w}$, and then we can use Bayesian methods to infer the most plausible values for the hyperparameters from the data.

This Bayesian approach to regression modelling has the following advantages. (1) The hyperparameters α, β are controlled using the same data as are used to optimize the parameters $\mathbf{w}$; there is no need for cross-validation. This is true even for models with multiple hyperparameters $\{\alpha\}$ (MacKay 1994; MacKay and Takeuchi 1995). (2) One obtains quantified error bars on model parameters and predictions. (3) One obtains a measure of the effective number of well-determined parameters in a model. (4) Objective model comparison is possible, comparing regression models with different basis functions, for example, or different regularizers (MacKay 1992a). (5) Generalizations to better probabilistic models are easy to formulate.

Modelling an input-dependent noise level

One improvement that we might wish to make is a modification of the noise model, which, in the above model, assumes that all the residuals are independent and Gaussian with identical variance. We might believe that in fact the noise level is a parameterized function $\beta(x; \mathbf{b})$ of the input variable and we might wish to infer it from the data. This inference is not straightforward; a maximum likelihood approach, in which $\mathbf{w}$ and $\mathbf{b}$ are simultaneously varied to maximize the likelihood, leads to a biased estimate of $\beta(x)$—because the optimized function $y(x; \hat{\mathbf{w}})$ unavoidably fits some of the noise so that the noise variance is systematically underestimated. The maximum likelihood solution may even have singularities with the local value of $\beta(x)$ going to infinity in regions where the data points are interpolated perfectly. In (MacKay 1991:Chapter 6) an approximate Bayesian approach to the inference of an input-dependent noise level is described which solves this problem using Gaussian approximations. Here an alternative approach is described using a program that implements Bayesian inference using Gibbs sampling.

2 Markov Chain Monte Carlo

In Bayesian inference, the two key tasks are to write down a probabilistic model for the domain, and to implement inferences given the observed data. Thus in the model with an input-dependent noise level discussed above, the probability of everything is:

$$\begin{aligned} P(\{t_n\}, \mathbf{w}, \mathbf{b}, \alpha_w, \alpha_b|\mathcal{H}) &= \left[\prod_n P\left(t_n | y(x_n; \mathbf{w}), \beta(x_n; \mathbf{b}), \mathcal{H}\right)\right] \times \\ & \quad P(\mathbf{w}|\alpha_w, \mathcal{H}) P(\mathbf{b}|\alpha_b, \mathcal{H}) P(\alpha_w, \alpha_b|\mathcal{H}). \end{aligned}$$

Here an extra hyperparameter α_b has been introduced that defines the expected smoothness of the function $\beta(x; \mathbf{b})$. The input variables $\{x_n\}$ are not being modelled, *i.e.*, they are assumed given under $\mathcal{H}$.

We are interested in the inference, given the data $\{t_n\}_{n=1}^N$, of the functions $y(x)$ and $\beta(x)$, and the prediction of new target variables t_{N+1} at locations x_{N+1}. These inferences are jointly described by a single equation:

$$P(\mathbf{w}, \mathbf{b}, \alpha_w, \alpha_b, t_{N+1}|\{t_n\}_{n=1}^N, \mathcal{H}) = \frac{P(\{t_n\}, \mathbf{w}, \mathbf{b}, \alpha_w, \alpha_b|\mathcal{H})}{P(\{t_n\}_{n=1}^N|\mathcal{H})}. \tag{3}$$

The task is to make a computational implementation of this complicated posterior distribution. There are three principal approaches to implementing Bayesian inference for this sort of problem: approximations based on Gaussians, as for example in (MacKay 1992b); an 'ensemble learning' approach, in which an approximating distribution is optimized by variational free energy minimization, introduced by Hinton and van Camp (1993); and Markov chain Monte Carlo techniques, as reviewed and developed by Neal (1993).

In a Markov chain Monte Carlo (MCMC) approach, the posterior distribution (3) is not represented directly; rather, a procedure is used iteratively to take a state vector $\theta = (\mathbf{w}, \mathbf{b}, \alpha_w, \alpha_b, t_{N+1})$ and generate a new random state vector θ' from a probability distribution $q(\theta'; \theta)$. Then we obtain the next state θ'' by sampling from the distribution $q(\theta''; \theta')$, and so forth. The transition probability q is constructed in such a way that an ergodic Markov process is defined with stationary distribution equal to the desired posterior distribution (3). Thus assymptotically the MCMC sampling procedure produces a sequence of states θ each of which is a sample from the posterior distribution (though consecutive state vectors are not in general *independent* samples from that distribution). Then properties of interest, *e.g.*, moments of predictive distributions, can be estimated from large numbers of samples $\{\theta\}$.

There are two principal ways of constructing transition probabilities q that converge to desired distributions $p(\theta)$.

Metropolis methods. The Metropolis algorithm makes use of a a *proposal density* $g(\theta'|\theta)$, which in the simplest case might be a simple random distribution centred on the current θ. A tentative new state θ' is generated from this distribution, and is accepted with probability:

$$P(\text{accept}) = \min\left(1, \frac{p(\theta|x)}{g(\theta|\theta')} \frac{g(\theta'|\theta)}{p(\theta'|x)}\right)$$

If the step is rejected, then we set $\theta' = \theta$. To compute the acceptance probability we need to be able to compute the probability ratios $p(\theta|x)/p(\theta'|x)$ and $g(\theta|\theta')/g(\theta'|\theta)$. Simple Metropolis algorithms perform poorly in high dimensional problems because they explore the space by a slow random walk. More sophisticated Metropolis algorithms such as hybrid Monte Carlo (see Neal (1993)) make use of proposal densities that give faster movement through the state space.

Gibbs sampling. In Gibbs sampling, each iteration $\theta \rightarrow \theta'$ involves a separate sampling of each variable in turn from its distribution *conditional* on the current values of all the other variables in the model. For many models (though not for general neural networks) these conditional distributions are straightforward to sample from. Conditional distributions that are not of standard form may still be sampled from by 'rejection sampling' if the conditional distribution satisfies certain convexity properties.

Gibbs sampling suffers from the problem of simple Metropolis algorithms that the state space is explored by random walk. However it is a relatively parameter-free method and so is attractive as an implementation strategy.

3 BUGS

A new tool, BUGS, makes simple the implementation of complex Bayesian models by Gibbs sampling. BUGS (Thomas *et al.* 1992) is copyright by the MRC Biostatistics Unit, Robinson Way, Cambridge CB2 2SR, and is available by ftp from `ftp.mrc-bsu.cam.ac.uk`.

In BUGS, a statistical model is expressed using the BUGS language; a compiler processes the model and the available data; and a sampler generates appropriate values of the unknown quantities. BUGS is intended for the analysis of complex models in which there may be many unknown quantities but for which substantial conditional independence assumptions are appropriate.

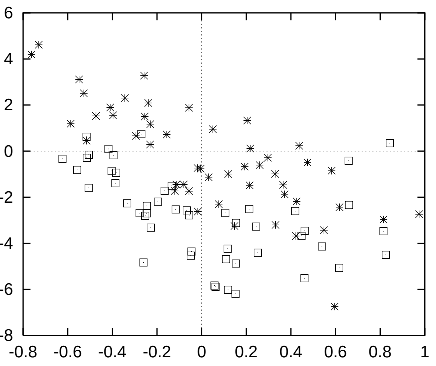

Figure 1: Toy data from two populations differing by an offset `dy`.

4 Inferring an interpolant and an input-dependent noise level with BUGS

A slight generalization of the model given above is made in this work. Data in the form of (x, t) pairs are assumed to be obtained from two populations (figure 1). Both populations depend on the same function $y(x)$ and have the same noise level $\beta(x)$, but there is an offset `dy` added to all the points in one population. We have fifty labelled points from each population. I make the assumption that we are particularly interested in inferring the offset between the populations. This task is motivated by the problem of inferring the difference between the distances to two populations of Cepheid stars (Freedman *et al.* 1994).

The underlying function is described by a linear combination of basis functions $y(x;\mathbf{w}) = \sum_h w_{h=1}^{K_W} \phi_h(x)$. The basis functions ϕ are Legendre polynomials. In neural network terms, this is a two-layer network with a fixed non-linear hidden layer and adaptive linear output connections. Similarly, the varying noise level is written as $\beta(x;\mathbf{b}) = \exp\left(\sum_{h=1}^{K_B} b_h \phi_h(x)\right)$. The parameters of the model are $\mathbf{w} = \{w_h\}_1^{K_W}$ and $\mathbf{b} = \{b_h\}_1^{K_B}$. Gaussian priors on these parameters are specified with hyperparameters α_w and α_b that have broad gamma priors. For convenience I introduce $K = \max(K_W, K_B)$. We obtain the special case of uniform noise by setting $K_B = 1$ and the special case of a straight line relationship by setting $K_W = 2$.

4.1 Defining the model in the BUGS language

BUGS uses a language similar to that of `S-plus`, but access to the latter is not required. Expressing the model in the BUGS language is almost as simple as writing the model on paper. The symbol `<-` defines a deterministic relationship, giving the quantity on the left as a function of the quantity on the right. The symbol `~` specifies a conditional distribution. This model uses normal distributions `dnorm` and gamma distributions `dgamma`. The notation `dnorm(m,beta)` defines a normal distribution of mean m and variance $1/\beta$. The function `inprod(w[],p[])` denotes the inner product of the vectors `w` and `p`.

Here is the file `astro10.bug`, which specifies a model in which the interpolant and the noise level are both described by Legendre polynomials of degree 10. The data from the two populations are stored in vectors `x` and `t` such that population 'a' gave rise to $\{(x_n, t_n)\}_{n=1}^{N_a}$, and population 'b' gave rise to $\{(x_n, t_n)\}_{n=N_a+1}^{N}$.

```
model astro10;

const             # ------- constants -------------------
  N=100,          # Number of data
  Na=50,          # Number of data in first population
  KW=10,          # Number of basis functions for y(x)
  KB=10,          # Number of basis functions for beta(x)
  K=10,           # K=MAX(KW,KB)
  x0=0.0,dx=1.0;  # characteristic range of the x-axis

var                                       # -- variables --
      x[N], dy , y[N], t[N], w[K], b[K],
      phi[K,N],
      alphab, alphaw, beta[N];

data  in "astro.dat" ;
inits in "astro1.in"  ;

{
  dy ~ dnorm( 0.0, 0.0001 ) ;           # noninformative prior for
                                        #    offset between populations

  for (m in 1:N) {                      # recurrence relation to
      phi[1,m] <- 1.0 ;                 #    define Legendre polynomials
      phi[2,m] <- (x[m]-x0)/dx ;
      for (h in 3:K) {
            phi[h,m] <- ( ( 2 * h - 3 ) * (x[m]-x0)/dx * phi[h-1,m]
                          - ( h - 2 ) * phi[h-2,m] ) / ( h - 1 ) ;
      }
  }
  for (h in 1:KW) {
      w[h] ~ dnorm( 0.0 , alphaw ) ;    #  Gaussian prior for w
  }
  for (h in 1:KB) {
      b[h] ~ dnorm( 0.0 , alphab ) ;    #  Gaussian prior for b
  }
  for (h in KW+1:K) {
      w[h] <- 0.0 ;                     #  Excess parameters
  }
  for (h in KB+1:K) {
      b[h] <- 0.0 ;                     #  Excess parameters
  }

  for (m in 1:N) {
      t[m]     ~  dnorm( y[m] , beta[m] ) ;        #  Gaussian residuals
      beta[m]  <- exp( inprod( b[] , phi[,m] ) ) ; #  Noise level
  }
  for (m in 1:Na) {
      y[m]     <- inprod( w[] , phi[,m] ) ;        #  Interpolant
  }
  for (m in Na+1:N) {
      y[m]     <- dy + inprod( w[] , phi[,m] ) ;   #  Other interpolant
  }

  alphab  ~ dgamma( 1.0E-3 , 1.0E-3 ) ; # noninformative priors
  alphaw  ~ dgamma( 1.0E-3 , 1.0E-3 ) ; #    for hyperparameters
}
```

And here is the file `astro1.in`, which specifies the initial conditions for the sampler:

```
list ( alphab=1.0 , alphaw=1.0 , dy=0 )
```

The data file `astro.dat` contains an ASCII list of the values $\{x_n, t_n\}$. The toy data were generated using true values $K_B = K_W = 3$ and `dy` $= 3.0$. The study here examines the inference of the offset `dy` when interpolation models of various orders, with and without the input-dependent noise level, are used.

For each model a burn–in period of 500 iterations was followed by 1000 iterations during which statistics on `dy` were recorded.

4.2 Results

I present, for each of the four models $(K_W, K_B) = (2,1), (2,10), (10,1), (10,10)$, the inferred value of `dy`, in terms of its posterior mean and standard deviation (as estimated by BUGS), and a 95% confidence interval.

`dy` | K_B	1			10		
K_W	mean	sd	2.5%:97.5%	mean	sd	2.5%:97.5%
2	2.44	0.30	1.86 : 3.02	2.76	0.26	2.25 : 3.24
10	2.54	0.31	1.98 : 3.16	2.71	0.27	2.18 : 3.23

The over–simple model $K_B = 1, K_W = 2$, which assumes a linear interpolant and constant noise level, gives a 95% confidence interval for `dy` that only just includes the true value. As is often the case with over-simple models, this model is not only wrong, it gives over-confident predictions too. Changing from the over–simple model to the model with more parameters in the interpolant (moving down from the top left corner) produces a slight increase in the uncertainty of `dy`. The increase is only slight because there are two opposing effects: first, for any particular value of noise level $1/\beta$, the more flexible interpolant is less well determined and the uncertainty in `dy` increases; but second, the greater flexibility of the interpolant allows it to fit the curving shape of the data and makes smaller noise levels $1/\beta$ probable. Small noise levels give more accurate inferences. A similar effect occurs as we increase the number of terms in the representation of $\beta(x)$ (going from left to right). The estimation of `dy` can become more precise, in intuitive terms, because the model is able to discover that some values of x give more reliable measurements than others, so that the inference of `dy` can be based on them, ignoring the more noisy measurements. The net effect is that when we change from the over–simple model to the most flexible model (bottom right), the confidence interval becomes smaller and more accurate.

5 Conclusion

Complex regression models that would take considerable effort to implement by Bayesian methods based on Gaussian approximations can be simulated in BUGS with great ease. BUGS implicitly infers hyperparameters from the data and automatically marginalizes over all the parameters to give the desired predictions. The Gibbs sampling method is not the most efficient of Markov chain Monte Carlo methods, but there may be problems of interest where the convenience of this tool outweighs this drawback. Not all probabilistic models can be implemented in BUGS; the conditional distributions that are sampled from must be log concave. But this paper has demonstrated that interesting new models can be implemented.

Acknowledgements

I thank David Spiegelhalter, Andrew Thomas, Nicky Best and Wally Gilks for creating BUGS and for helpful discussions.

References

FREEDMAN, W. L., MADORE, B. F., MOULD, J. R., HILL, R., and OTHERS. (1994) Distance to the Virgo cluster galaxy M100 from Hubble space telescope observations of cepheids. *Nature* **371**: 757–762.

HINTON, G. E., and VAN CAMP, D., (1993) Keeping neural networks simple by minimizing the description length of the weights. In: *Proceedings of COLT-93*.

MACKAY, D. J. C., (1991) *Bayesian Methods for Adaptive Models*. California Institute of Technology dissertation.

MACKAY, D. J. C. (1992a) Bayesian interpolation. *Neural Computation* **4** (3): 415–447.

MACKAY, D. J. C. (1992b) A practical Bayesian framework for backpropagation networks. *Neural Computation* **4** (3): 448–472.

MACKAY, D. J. C. (1994) Bayesian non-linear modelling for the prediction competition. In *ASHRAE Transactions, V.100, Pt.2*, pp. 1053–1062, Atlanta Georgia. ASHRAE.

MACKAY, D. J. C., and TAKEUCHI, R. (1995) Interpolation models with multiple hyperparameters. In *Maximum Entropy and Bayesian Methods, Cambridge 1994*, ed. by J. Skilling and S. Sibisi, Dordrecht. Kluwer.

NEAL, R. M. (1993) Probabilistic inference using Markov chain Monte Carlo methods. Technical Report CRG–TR–93–1, Dept. of Computer Science, University of Toronto.

THOMAS, A., SPIEGELHALTER, D. J., and GILKS, W. R. (1992) BUGS: A program to perform Bayesian inference using Gibbs sampling. In *Bayesian Statistics 4*, ed. by J. M. Bernardo, J. O. Berger, A. P. Dawid, and A. F. M. Smith, pp. 837–842. Oxford: Clarendon Press.