

THE UNIVERSITY OF CHICAGO

BAYESIAN NEURAL NETWORKS AND VARIABLE SELECTION

A THESIS SUBMITTED TO
THE FACULTY OF THE DIVISION OF THE PHYSICAL SCIENCES
IN CANDIDACY FOR THE DEGREE OF
MASTER OF SCIENCE

DEPARTMENT OF STATISTICS

BY
ROBERT ADAM MOLNAR

CHICAGO, ILLINOIS

JUNE 2010

©2010
Robert Adam Molnar
All Rights Reserved

ABSTRACT

Neural networks have been shown to have good predictive power in complicated multidimensional problems such as credit fraud and cancer detection, and are frequently used for that purpose. However, in a standard network, it is very difficult to determine which variables are important. Additionally, high dimensionality makes it challenging to fit the model without overfitting the coefficients. Various authors have proposed Bayesian methods for neural networks. After describing others' attempts, this paper proposes an alternative. The new model, based on mixture models, adds a switching variable to each predictor inside each node. If the switch is on, the coefficient takes a typical Gaussian distribution, but when off, the coefficient is shrunk to a very small value. The nearly zero coefficients make the model less prone to overfitting errors, while the switches provide evidence of importance.

TABLE OF CONTENTS

ABSTRACT	iii
LIST OF FIGURES	v
Chapter	
1 INTRODUCTION AND BACKGROUND	1
1.1 Origination	1
1.2 Representation and Transformation	2
1.3 Bayesian Networks	5
1.4 Accurate Model Fitting	7
1.4.1 Automatic Relevance Determination	8
1.4.2 Early Stopping	8
1.4.3 Variable Selection	9
1.4.4 Space Search	9
1.4.5 Controlled Size Search	11
2 THE NEW PROPOSAL	13
2.1 Description	13
2.1.1 Mixture Model Regression	13
2.1.2 New Concept	15
2.1.3 Predictor Range and Standard Deviations	17
2.1.4 Posterior Distribution	18
2.2 Implementation	20
REFERENCES	22

LIST OF FIGURES

1.1	Multilayer perceptron with one hidden layer and three nodes.	3
2.1	Comparing small and large normal densities.	14
2.2	Prior distributions for various initial inclusion probabilities.	17
2.3	Prior distributions for various values of σ_{out}	18

CHAPTER 1

INTRODUCTION AND BACKGROUND

This paper looks at a new approaches to selecting variables in Bayesian neural networks. There have been many papers written on neural networks, Bayesian analysis, and variable selection. This chapter reviews prior work on the topics, roughly in chronological order. The new model is described in Chapter 2.

1.1 Origination

Though earlier papers had hypothesized ideas about how information flows through the brain, the 1943 paper of Warren McCulloch and Walter Pitts [31] was the first serious attempt to form a mathematical model of brain activity. (Interestingly, Pitts was at the University of Chicago at the time.) As such, it is generally considered the first paper on neural networks. McCulloch and Pitts noted that the cells of the brain, called neurons, worked in an "all-or-none" fashion. Because "To each reaction of any neuron there is a corresponding assertion of a simple proposition," they attempted to define information transfers between the cells with formulas from propositional logic. Their modeling work formed the basis for the field, including this thesis.

Fifteen years later, a paper in *Psychological Review* by Rosenblatt [40] called the hypothetical nervous system machine a perceptron. Subsequent biological research has shown that the actual human brain works quite differently than the McCulloch-Pitts model, meaning the "neural" network is actually misnamed. (See, for instance, the textbook on modern neural science by Kandel, Schwartz, and Jessell [16]). Despite this, the name neural network (sometimes further clarified as an artificial neural network) has been applied to the computer science and statistics models since at least the 1950s. In the perceptron model, a decision unit takes inputs from neurons, which can be either active or inactive. Rosenblatt called these units A-units; the modern term is node. The node weighs these inputs, and then outputs an active or inactive response based on a fixed threshold. In terms of

regression models, there are binary inputs x_i , a linear weighting equation $T = \sum w_i x_i$, and a threshold t so the output becomes $\hat{y} = I\{T > t\}$.

Initial interest in the perceptron model was high, but subsided quickly. A major problem in the pre-computer era was the difficulty of estimating results. In addition, Minsky and Papert [33] showed in 1969 that perceptrons with one layer of nodes could not completely discriminate two groups that were not linearly separable. This included the XOR gate common in mathematical logic. Since much of the interest in these models came from mathematical logic, including the theory of McCulloch and Pitts, work on neural networks declined for several years.

In the 1980s, computer scientists regained interest in neural networks. Computing power improved, and a learning algorithm published by Rumelhart and others [42] made programming easier. Outputs became more complex than simple step functions, making the nodes what Rosenblatt called continuous transducers, $\hat{y} = f(\sum w_i x_i)$. Most importantly, around 1990, several authors proved that a single layer perceptron model with a continuous sigmoidal function for f can approximate any result arbitrarily closely; see [7], [9], and [14]. An approximation of uniform closeness is more than sufficient for practical work.

1.2 Representation and Transformation

Many researchers, mostly in computer science, look at the solution and application of neural networks, and journals are published such as the IEEE Transactions on Neural Networks. A good introductory summary on neural networks is given by Cheng and Titterton [6]. Several textbooks exist; mathematical statisticians will find Ripley [39] and Bishop [3] informative. This paper deals primarily with MLPs with one hidden layer, because one layer is sufficient for approximation while maintaining computational feasibility.

Figure 1.1 is a conceptual representation of this neural network, known as a single layer perceptron. In the graphic, there are $d = 4$ input variables, x_1, x_2, x_3, x_4 . Subscripts represent variable numbers; computer science literature often refers to the variables as features. Each variable feeds into each function in the single hidden layer, generally called nodes. In this graphic, there are $n = 3$ nodes, labeled as $h_1(X), h_2(X), h_3(X)$. More complicated networks can have multiple hidden layers.

This network has arrows in only one direction. There are no links from one hidden node to another, or from the hidden node back to the inputs. This type of network, called a feed-forward network because it has no sideways or backward input, has the advantage of being directly solvable. A maximum likelihood solution can be found via the slow but effective gradient descent learning algorithm mentioned in Rumelhart [42]. In a network with cycles, it is more difficult to reach a single maximal solution. Additionally, setting prior distributions on parameters that work in the reverse direction would be difficult. The feed-forward models in this paper have analogues to other problems which can be used to help set priors. Other directions do not. While more complex models might reach solutions more quickly, the extra complexity is unneeded for this paper.

In the single layer perceptron, each node is a generalized linear model that uses the variables x_i in an additive combination, then applies a transformation $f()$ to the result. These computed node values are then combined in a potentially different model $g(\sum \alpha_j z_j)$ to provide the predicted value \hat{y} . The transformations $f()$ and $g()$ frequently differ. In

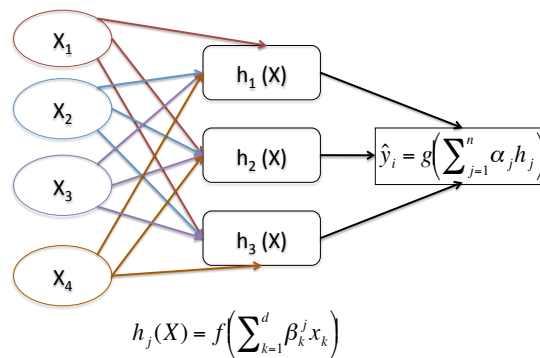


Figure 1.1: Multilayer perceptron with one hidden layer and three nodes.

theory, each h_i node could use a different transformation, but the transformations would have to be selected as part of the process, adding more complexity. One possibility, a genetic

algorithm approach, might work; genetic algorithms and neural networks are occasionally combined, as in Marwala [30]. However, transformation shifts are extremely rare; there is usually one $f()$ and one $g()$. Possibilities include the following:

- $f(u) = u$, the linear transformation. This is sometimes used as the $g()$ function from the nodes to the output, particularly for continuous y . It is not used from variables to nodes. If it was used, the final estimation \hat{y} would be a single generalized linear model: $\hat{y} = g\left(\sum_{i=1}^d(\beta_i^1 + \beta_i^2 + \dots + \beta_i^n)x_i\right)$. It would be easier to use GLM methods. Furthermore, the contributions of individual β_i^j are indistinguishable and not estimable.
- $f(u) = \text{sign}(u)$, the threshold, which returns either $+1$ or -1 . A functionally similar approach is a cutoff that returns 1 if u exceeds a certain value, and 0 otherwise. McCulloch and Pitts used this idea. Because the result of the transformation is not continuous, threshold models will not guarantee a uniform approximation. For this reason, threshold transforms are rarely used.
- $f(u) = \frac{e^u}{1+e^u}$, the logistic transform, returning a value between 0 and 1. This is common, due to the popularity of logistic regression, including the textbook by Hosmer and Lemeshow [15].
- $f(u) = \tanh(u) = \frac{e^u - e^{-u}}{e^u + e^{-u}}$, the hyperbolic tangent transform, returning a value between -1 and $+1$. This is another popular choice, because it is continuous, and a u value of 0 returns $f(u) = 0$.

When going from nodes to the output, the most common choices are the logistic transform when the modeled output is dichotomous, to model percentages, and the identity transform when the modeled output is continuous.

When going from variables to nodes, the two most common options are the logistic and hyperbolic tangent. Both are sigmoidal continuous functions and satisfy the requirement for approximation. There is an equation relating one to the other at a given value h ,

$$\tanh(h) = 2 \text{logit}(2h) - 1$$

The transforms differ in range and shape. The hyperbolic tangent returns values from -1 to $+1$, while the logistic transformation returns values from 0 to 1 . For the hyperbolic tangent, a meaningless node, where every β coefficient equals zero, will return $\tanh(0) = 0$. This makes it relatively easy to ignore meaningless nodes. On the other hand, under the logit, a meaningless node will return $\text{logit}(0) = 0.5$. This is not difficult to adjust, as removing 0.5 converts the logistic range to -0.5 to $+0.5$. After adjustment, the logistic transformation has a slightly gentler slope, though the difference is relatively small. At the steepest point, when the sum of the variables $h = 0$, the logistic has a slope of 14 degrees, while the hyperbolic tangent is at 45 degrees [43]. While the difference in slope might matter in some situations, the primary concern is bookkeeping. Most Bayesian neural network programmers use the hyperbolic tangent, such as Liang [23], and this paper will follow that convention.

1.3 Bayesian Networks

In the late 1980s, authors began to look at applying Bayesian analysis techniques to neural networks, placing prior distributions on the β and α . Buntine and Weigend [5] published an early paper on the subject in 1991, followed by two papers by MacKay [27] and [28]. The Bayesian networks performed very well in a prediction competition sponsored by the American Society of Heating, Refrigeration, and Air Conditioning Engineers, winning the competition to provide accurate measurements of electricity, heating, cooling, and sunlight [29]. Work has continued on Bayesian networks and related models; see a review article by Titterton [46].

Choosing a prior distribution for the parameters is complicated. There are many parameters to estimate. With the β_k^j and α_j over $k = 1, \dots, d$ variables and $j = 1, \dots, n$ nodes. Including an intercept in each node and the output, there are a total of $(d \times n) + (n + 1)$ parameters. In addition, the number of nodes n can be a variable.

Priors should represent what modelers know or believe about the parameter. It is difficult to discern prior belief about coefficients in a large number of hidden nodes. The nodes are exchangeable, because they can be renumbered without affecting the solutions; therefore, distributions did not at first specify different structures in different nodes. For

instance, MacKay fixed the number of hidden nodes, and gave the coefficients independent Gaussian prior distributions. Mathematically, the equations to estimate hidden node values and outputs are shown below. The equations look a little different than those in Figure 1.1, because they now include a separate constant term, called a bias, that the picture lumped into the general X and h vectors. Node bias is represented by β_0 , while prediction bias is represented by α_0 . Notationally, node numbers are superscripts, and variable numbers are subscripts.

$$h_j(X) = f\left(\beta_0 + \sum_{k=1}^d \beta_k^j x_k\right)$$

$$\hat{y} = g\left(\alpha_0 + \sum_{j=1}^n \alpha^j h^j\right)$$

In the Gaussian case, coefficient prior distributions are generally set to have mean zero. The variance parameter is set via a hyperparameter; most common is that the precision, the inverse of the variance, is considered to have a Gamma distribution with specified mean and scale. Although Gamma distributions are not symmetric, and perhaps not of optimal shape, they are flexible, computable, and proper. There are several variance hyperparameters: one for hidden node coefficients, one for hidden node biases, one for output biases, and one for output coefficients. As listed below, these are considered independent.

$$\beta_0 \sim N(0, \sigma_0^2)$$

$$\tau_0 = \sigma_0^{-2} \sim G(\omega_0, \zeta_0)$$

$$\beta_j \sim N(0, \sigma^2)$$

$$\tau = \sigma^{-2} \sim G(\omega, \zeta)$$

$$\alpha_j \sim N(0, \sigma_\alpha^2)$$

$$\tau_\alpha = \sigma_\alpha^{-2} \sim G(\omega_\alpha, \zeta_\alpha)$$

$$\alpha_0 \sim N(0, \sigma_{\alpha_0}^2)$$

$$\tau_{\alpha_0} = \sigma_{\alpha_0}^{-2} \sim G(\omega_{\alpha_0}, \zeta_{\alpha_0})$$

In this initial implementation, all x_k and thus β_k share a single variance hyperparameter. This was quickly improved to help with model fitting.

1.4 Accurate Model Fitting

Neural networks often adapt too heavily to the dataset used for training. At first glance, this doesn't seem like a problem, and sometimes it isn't. When trying to explain the nuances of a dataset, highly specific explanations are acceptable. The problem arises when models are generalized and used for future prediction. As the network focuses on smaller and smaller distinctions in the training set, the emphasis on general patterns decreases. Test sets, and future prediction sets, tend to have the same general patterns but different small distinctions. Thus, while performance on the training set improves by identifying the small distinctions, performance on the test set worsens by disregarding the general patterns. This phenomenon is known as overfitting. In a neural network, overfitting can occur from having too many nodes, and thus too much freedom in the parameters. Overfitting can also be caused by fitting features with no true effect on the response, since each additional predictor adds n parameters.

To avoid overfitting, one option is to fit networks with all potential predictors, then selectively remove predictors that appear less powerful. This approximates the variable selection procedures of linear regression, described in textbooks such as that by Weisberg [47]. This approach works for regression models because measures of individual predictive power exist, such as likelihood ratio tests. The comparison is difficult with neural networks, since over a large number of hidden nodes, it is difficult to quantify the total effect of a predictor. Additionally, unlike linear regression which takes seconds, computing a neural network can take minutes or even hours, making high iteration counts practically unfeasible.

Another option would take subject matter knowledge and build an appropriate prior for the situation. When enough information is known about the predictors to build such a model, it makes perfect Bayesian sense to do just that. Unfortunately, in most cases the necessary expertise is not available. While models should attempt to include known information, an automated procedure is necessary. Several options are available.

1.4.1 Automatic Relevance Determination

MacKay, along with Radford Neal, developed a procedure to deal with such scenarios, called Automatic Relevance Determination (ARD). ARD changes the priors on the β_k^j . Instead of one σ for all of the $k = 1, 2, \dots, d$ input variables, the prior is divided. There is one overall scaling parameter, ν , and then a separate variance parameter for each feature. Mathematically,

$$\begin{aligned}\beta_k &\sim N(0, \nu \times (\sigma_k)^2) \\ \tau_k = (\sigma_k)^{-2} &\sim G(\nu, \zeta_k) \\ \nu &\sim G(\omega_\nu, \zeta_\nu)\end{aligned}$$

The values for ν are set to be very vague, to account for unknown overall variation, while the ζ^k are more precise. Theoretically, this doesn't help, because it makes it impossible to fully identify and distinguish ν and the σ_k . Doubling every σ_k and dividing the ν by 2 would result in the same predictions. However, MacKay and Neal found the separation easier computationally. In the electricity prediction competition [29], ARD models were more accurate than those without the adjustment. Neal included information on ARD in his Ph.D. paper, which was expanded into a manuscript [34].

Most published works using Bayesian neural networks also use ARD. Applications include inferring distillation quality in oil refineries from sensor reports [2], detecting adverse drug reactions [36], and classifying gene expression data [21]. Lampinen and Vehtari [17] provide a review article, including background information and applications predicting concrete quality, approximating a topographic mapping, and tree trunk recognition in forests.

1.4.2 Early Stopping

Another method, early stopping, is frequently used in non-Bayesian neural networks. Coefficients in neural networks are often estimated by iterative adjustment, where the values are updated until they reach a stable plateau. The early stopping approach divides the training data into two parts, which can be labeled the building and validation sets. The network is formed using the building set, as if there were no stopping; the structure is

saved every so often, perhaps every ten iterations. At each save point, performance on the validation set is computed. Theory suggests that when the model begins to become overfit, focusing on specific distinctions instead of general patterns, validation performance will begin to deteriorate. The technique halts model development once the validation results get worse, before the maximum number of iterations. It assumes that the correct depth of fit has been reached. More complicated algorithms exist to determine the stopping point, including one described by Prechelt [37].

Neal [35] tested Bayesian methods against early stopping, and found that on the larger tested sets, ARD and related methods returned more accurate results than the stopping technique. While are other ideas, such as a method devised by Sum et al [45] for node pruning in time series environments, these are less appropriate for a Bayesian environment. Prior information is not involved. There are better options for estimation.

1.4.3 Variable Selection

Another way to prevent overfitting is through variable selection. Not all variables contribute to predictive power. Removing the useless ones will reduce the problem of overfitting. At least conceptually, as data indicates that some predictors are useful and others not, the posterior distribution of any model will change from the prior.

Many authors have approached the problem of variable selection from a Bayesian view, at least as far back as Dempster in 1973 [8]. One approach, model averaging, has been discussed by Raftery, Madigan, and Hoeting [38], among others. A neural network is, in effect, a combination of many small models, so averaging networks is not appropriate. This review considers two other approaches: neural net space search and controlled size search. While both show promise, they are not fully Bayesian with room for prior information.

1.4.4 Space Search

Model selection via a space search was proposed by Herbert Lee in his 1998 Ph.D. thesis [18] and expanded in 2001 [19]. Like Neal, Lee's paper formed a portion of a book [20]. In this approach, the model space contains parameters for the number of hidden nodes n , as well as indicators for all input variables. For example, say there are three input variables,

x_1, x_2, x_3 , and the number of hidden nodes is determined to fall between 1 and 5. There are $2^3 = 8$ potential sets of predictors: { none; $x_1; x_2; x_3; x_1x_2; x_1x_3; x_2x_3; x_1x_2x_3$. } The same set of predictors appears in each of the nodes. Since there are 5 possibilities for number of nodes, the total number of states to search is roughly $5 \times 2^3 = 40$. (Technically, it's $1 + 5 \times (2^3 - 1)$ because if the model selects no predictors, any number of nodes is equivalent.) The program needs to select from the possibilities, comparing them using a performance metric. Lee prefers the Bayesian Information Criterion (BIC) defined by Schwarz [44]. As defined by Schwarz, the BIC equals the logarithm of the model likelihood, minus the number of parameters fit times 0.5 times the log of the number of observations. The best model has the highest penalized likelihood. (BIC is often transformed by multiplying by -2 , $BIC = -2 \log(L) + (\text{params}) \times \log(\text{obs})$. This puts BIC on the same scale as the Akaike Information Criterion [1]. After transformation, the lowest AIC/BIC is best.)

Under BIC state search, each additional parameter adds a penalty of half the $\log(\text{obs})$ to the log likelihood. If adding a coefficient improves the model by more than that amount, estimating the parameter is helpful. However, in neural networks computing the number of parameters can be tricky, because adding a variable or a node does not add just one coefficient. Adding a new x_i adds coefficients in all hidden nodes, a total of n new parameters. Inserting a node requires fitting new β for each variable plus one for the intercept bias.

For the small example above, with fewer than 40 possibilities, a statistician could calculate BIC for each model and choose the best one. For larger problems, an automated state space mechanism would be necessary. Lee proposes a Markov Chain search over the state space called Bayesian Random Searching (BARS). BARS begins with a candidate model and BIC. On each step, with probability $\frac{1}{2}$, the candidate model grows, either by adding a variable or hidden node. The other half of the time, the candidate model shrinks by a variable or node. After the shift, BIC is computed for the new model. Then the Metropolis-Hastings algorithm [32] is applied. If the new candidate has better BIC, it is accepted; if it has worse BIC, it is accepted with probability $\exp(BIC_{cand} - BIC_{old})$. The process repeats for a specified number of iterations, or until the model space has been adequately explored.

According to Lee's book, in a comparison test on loan application modeling, BARS and ARD had similar misclassification rates. Actually, the rate for ARD was slightly less, 29%

compared to 31%. However, Lee’s model found some predictors to be unnecessary, which would allow the bank to reduce the length of the application form. As he notes, the choice of modeling technique can include both predictive power and interpretability. Additionally, a Bayesian approach can include prior information when available. Both Neal’s and Lee’s techniques as implemented do not attempt to include data outside the model. Priors and Bayesian techniques exist to control size and reduce overfitting.

1.4.5 Controlled Size Search

Lee’s approach, while relying on a Bayesian information criterion, is closer to penalized likelihood than a fully Bayesian model. In particular, there is no explicit prior distribution on predictors or the number of predictors. Another researcher, Faming Liang, took a different approach. In a series of four papers published between late 2003 and 2005, [23], [24], [25], and [26], he explores the idea of placing a prior on the number of nonzero coefficients. Of the papers, the 2004 paper published in *Survey Methodology*, the only one with a coauthor (Anthony Kuk), is the best summary.

For a model with d input variables and n nodes, an intercept should be added to each node and from the nodes to the output. Each node in this paper’s model has $(d+1)$ possible β coefficients, and the equation for the output has $(n + 1)$ possible α coefficients. (The Liang and Kuk article uses γ where this paper uses β , and β there for α here.) The prior distributions on the α and β are the same as in the base case, independent of each other:

$$\beta_j \sim N(0, \sigma_\beta^2)$$

$$\alpha_j \sim N(0, \sigma_\alpha^2)$$

For standardized x predictors, Liang and Kuk set the σ^2 variances to 5, a moderate value.

The search model adds to the ordinary network by introducing an indicator variable for each β and α . If the indicator is one, the parameter works as normal, but if the indicator is zero, the parameter is zeroed out. Using I_k^j to represent the indicator for variable k in

node j , the equations for nodes and the output become the following:

$$h_j(X) = f(I_0^j \beta_0 + \sum_{k=1}^d I_k^j \beta_k^j x_k)$$

$$\hat{y} = g(I_0 \alpha_0 + \sum_{j=1}^n I_j \alpha_j h_j)$$

The total number of indicator variables is $U = n(d + 1) + (n + 1)$, but many of them are zero. The number of nonzero connections is subject to a prior probability based on a truncated Poisson distribution, with a minimum of 3 and a maximum of U . Let Λ be the set of nonzero indicator variables. The prior probability is based on the size of Λ :

$$P(\text{size of } \Lambda = m) \propto \frac{\lambda^m}{m!}$$

Liang and Kuk choose the truncated Poisson parameter λ to be about one tenth the size of the dataset. Each model of a given size is treated similarly, so each variable has equal initial probability.

The introduction of indicator variables adds a great deal of flexibility to the model. Given the ability to fit the posterior distribution, this flexibility can improve predictive power. Relying on earlier results by Liang and others, such as evolutionary Monte Carlo [22] and reversible jump computing [13], the authors engage in complicated Monte Carlo programming to produce results. Tracking a full set of separate indicators and changing the size of the model space are difficult tasks, even for excellent programmers.

The models described to this point all use some Bayesian techniques in neural networks. However, none are fully sufficient. Without adjustment, the original Bayesian neural network tends to overfit the model. Early stopping as implemented in non-Bayesian networks is ad hoc and not very efficient. The ARD model applies a single shrinkage factor to all instances of a variable, which reduces overfitting in a sharp uniform way. Lee's BARS model advances by allowing nodes to vary in shape, but relies on information criteria and stepwise selection instead of a fully Bayesian setup. Liang's size search model is Bayesian, but has to control many indicator variables and considers predictors interchangeably, neglecting potential information about the prior. There is still room for improvement.

CHAPTER 2

THE NEW PROPOSAL

2.1 Description

The idea for the main model of this paper is somewhat similar to Liang’s work, in that it can be expressed in a form with indicator variables. However, the basic idea, developed in 2003 before Liang’s work was published, comes from Bayesian space search for regression. It represents the prior distribution for each coefficient as a mixture of normally distributed random variables.

2.1.1 Mixture Model Regression

In 1993, George and McCulloch [11] published a paper presenting a method called Stochastic Search Variable Selection (SSVS). The authors expanded upon the idea in 1997 [12], which was extended to multivariate responses by Brown, Vannucci, and Fearn [4]. The approach described here focuses on part of the 1997 paper, the hierarchical mixture model for linear regression.

The normal linear regression model with response variable Y and potential predictors X_1, \dots, X_d fits β coefficients and an error measure σ through a model with priors assigned to β and σ .

$$Y|\beta, \sigma \sim N(X\beta, \sigma^2 I)$$

Some of the predictors do not have a large effect on the response, and thus have small β_i in the true model. Fitting small coefficients is inefficient and increases the risk of overfitting. Removing these variables from the model would help. Unfortunately, the subset of small coefficients is unknown.

To attempt to determine this subset, define a new vector $R = (r_1, \dots, r_d)'$. The r_i are indicator variables, equaling 0 if X_i has little effect and a small β_i , and 1 if the predictor has a large effect. The prior distribution now must include R . George and McCulloch

suggest a Bernoulli prior for each variable, where $P(r_i = 1) = w_i$ and $P(r_i = 0) = 1 - w_i$. The inclusion prior for each variable is independent of that of other variables. Additionally, the probability can differ for each predictor, representing knowledge about that variable, though examples given generally set all w_i to the same value.

The prior distribution for β, σ , and R can be conjugate, in the same family as the posterior, or non-conjugate. The non-conjugate approach is more interesting for this thesis, where each component of β looks like a mixture of two normal distributions, one with very low variance, and one with greater dispersion.

$$\pi(\beta_i|r_i) = (1 - r_i)N(0, \nu_0 r_i) + r_i N(0, \nu_1 r_i)$$

The two ν are variance hyperparameters. ν_0 values should be small, because they occur when $r_i = 0$ and the variable is considered to have little effect. On the other hand, ν_1 values should be set based on the size and form of variables in the model. For example, Figure 2.1 shows the difference between the densities of two normal distributions when $\nu_0 = 1, \nu_1 = 100$, corresponding to standard deviations of 1 and 10 respectively.

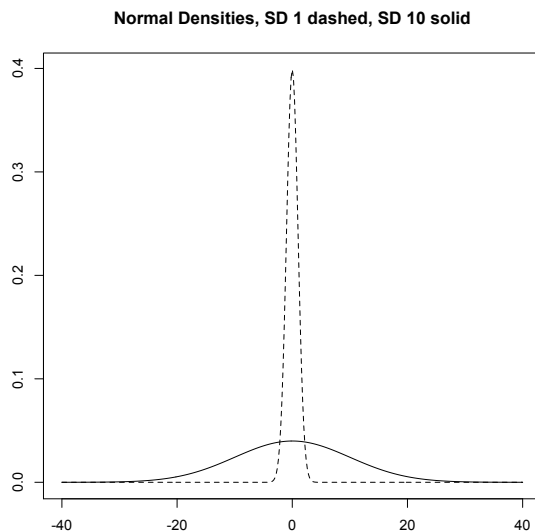


Figure 2.1: Comparing small and large normal densities.

When a $\nu_0 = 0$, this zeroes out the β contribution as in Liang's model. George and

McCulloch briefly explored this approach. They mentioned that ordinary MCMC is not possible with zeroes, and complicated computing like that of Green [13] is necessary. The work in this paper utilizes non-zero values, for which computing is more direct.

2.1.2 New Concept

This paper takes the idea for prior distribution shape for a given β_i from George and McCulloch, and applies it to neural networks. Writing things out formally, the model attempts to predict Y by creating \hat{Y} . For a continuous observation Y , \hat{Y} is a linear combination of the node values z_i , as in linear regression. In the formulation from Chapter 1, the transformation $g()$ from nodes to output is the identity transform. Given n nodes and an intercept term as the $(n + 1)$ st term,

$$\hat{Y} = \alpha_1 h_1 + \alpha_2 h_2 + \dots + \alpha_n h_n + \alpha_{n+1} (1)$$

Each node value is the result of a model with a logistic transform $f()$ on the d variables. Each node also has an intercept, represented by a τ . For an observation, the value from node j is a logistic transformation of a generalized additive model:

$$h_j = \frac{e^{\tau^j + \beta_1^j x_1 + \dots + \beta_d^j x_d}}{1 + e^{\tau^j + \beta_1^j x_1 + \dots + \beta_d^j x_d}}$$

To fully specify the model, prior distributions are necessary. For the step from nodes n to the output Y , the parameters are specified as in linear regression.

$$Y \sim X\alpha + e \quad \text{Var}(e_i) = \sigma_y^2$$

Prior distributions are required for the α and σ_y^2 . Following the specification for linear regression in textbooks such as Rossi, Allenby, and McCulloch [41], the prior distribution for σ_y^2 , the variance of the linear regression from the nodes to the output is a conjugate prior - the inverse scaled chi-square distribution with a fixed number of degrees of freedom ν and scaling value SSQ . Given this value, the prior for α coefficients has zero mean, and

variance based on σ_y^2 and a fixed precision matrix A .

$$\alpha | \sigma_y^2 \sim N(0, \sigma_y^2 * A^{-1})$$

$$\sigma_y^2 \sim SSQ * \nu / \chi_\nu^2$$

Given node values, reasonable starting choices for the fixed parameters are $\nu = 5$, $SSQ = 0.25$, and A as a diagonal matrix with 0.1 on the central diagonal and 0 elsewhere.

Inside a node, the node intercept β is modeled as shown above. The on-off indicator variable r_i has a binomial prior with initial probability of inclusion p_i . The position on the curve γ is a standard normal distribution. The standard deviations σ_{in} and σ_{out} are fixed values, based on the range of the predictors. Examining predictor range and determining appropriate deviations is more fully considered in the next section.

$$\beta_i^j = \gamma_i^j * (\sigma_{out} + r_i^j(\sigma_{in} - \sigma_{out}))$$

$$r_i^j \sim Bern(p_i)$$

$$\gamma_i^j \sim N(0, 1)$$

Node intercepts τ^j are on the same scale as the β parameters for included predictors, with a normal distribution, zero mean, and standard deviation σ_{in} .

$$\tau^j \sim N(0, \sigma_{in}^2)$$

This implementation assumes that each node must have an intercept. Therefore, a node cannot be empty, since at least the intercept is considered non-zero. This choice is similar to ARD, but differs from Liang's approach and BARS. It is more appropriate for interpretation and unbiasedness. However, the tradeoff is that if multiple nodes have only intercepts, estimation is not identifiable.

2.1.3 Predictor Range and Standard Deviations

Scale poses a problem when finding reasonable parameter estimates. Potential predictors can have any numeric value. Different ranges make it difficult to quantify the size of an “on” indication. To eliminate these scale problems, the response Y and predictors X are standardized into a range from -1 to $+1$. This places all variables on the same scale, and allows the model to use a single σ_{in} and σ_{out} for all predictors. (However, it does mean that the model might predict \hat{Y} values beyond the limits of $+1$ and -1 . This is a drawback balanced by the simplified scale.) On this scale, for a logistic or hyperbolic tangent function, a move of 3 units would be very substantial. With this guidance, σ_{in} was set to 1.5. Setting a 100-to-1 distinction between the standard deviations, a large and clear dichotomy, yields $\sigma_{out} = 0.015$. The distinction might be smaller.

The shape of the prior distribution for each X_i variable is affected by the ratio of the two standard deviations; it is also influenced by p_i , the chosen initial probability of inclusion. Figure 2.2 shows the shape of the prior distribution holding constant $\sigma_{in} = 1.5$ and $\sigma_{out} = 0.100$, but changing the initial inclusion probability to $p = 0.1, 0.5$, and 0.9 .

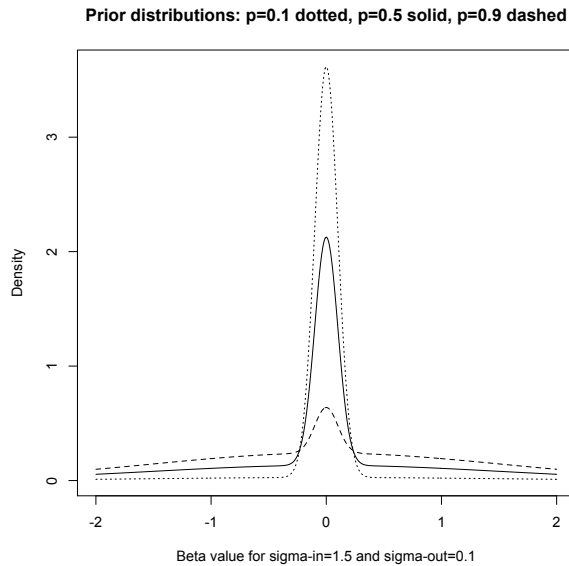


Figure 2.2: Prior distributions for various initial inclusion probabilities.

Figure 2.3 examines the effect of various σ_{out} choices, retaining $\sigma_{in} = 1.5$ and $p_i = 0.5$.

For the strict case used in the code, $\sigma_{out} = 0.015$ and a 1:100 ratio of standard deviations, the graph appears close to that of an indicator function. Moving to a 1:15 ratio, a standard deviation of 0.1, is shown as the solid line, and a 1:3 ratio appears as the dashed line. There are many possibilities to consider for standard deviation ratio and inclusion probability.

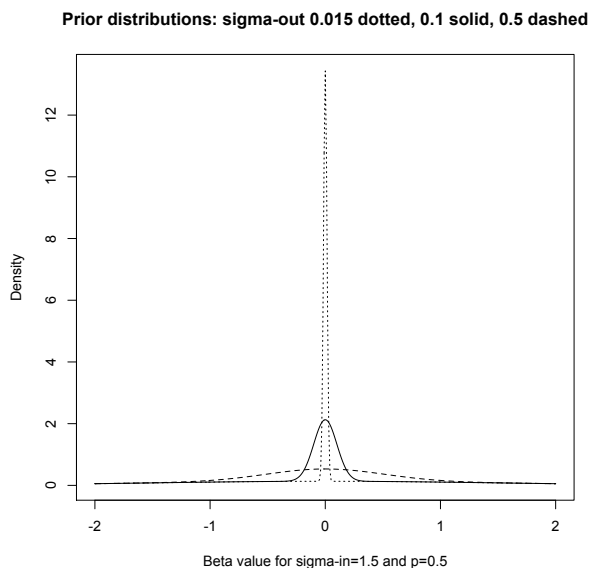


Figure 2.3: Prior distributions for various values of σ_{out} .

2.1.4 Posterior Distribution

To accurately estimate the model, the posterior distribution - the likelihood times the prior - must be found. The likelihood and prior consist of many multiplications. Computationally, it is easier to work with the natural logarithm of the posterior, which involves mostly addition, and then exponentiate as needed. Additionally, certain processes, such as Langevin proposals, directly utilize the log posterior. Therefore, this section develops the log posterior for the proposed model.

Assuming linear regression, with standard deviation σ_y and Gaussian errors, gives the likelihood for one observation:

$$(2\pi)^{-\frac{1}{2}} \sigma_y^{-1} e^{-\frac{1}{2\sigma_y^2}(y-\hat{y})^2}$$

The log likelihood becomes $-\frac{1}{2} \log(2\pi) - \log(\sigma_y) - \frac{1}{2\sigma_y^2} (y - \hat{y})^2$ and when summed over all k observations the log likelihood is

$$-\frac{k}{2} \log(2\pi) - k \log(\sigma_y) - \sum_{\zeta=1}^k \frac{1}{2\sigma_y^2} (y_\zeta - \hat{y}_\zeta)^2$$

Several things contribute to the prior: the overall prediction, the α node coefficients, the τ node intercepts, and the β node coefficients. Furthermore, because of the structure, the β can be broken into switching components r_i^j and normal variates γ_i^j . To begin, the node coefficient prior distribution involves a joint prior on α and σ_α . Given the σ_α , the α are treated as independent Gaussian with mean zero and standard deviation σ_α times a constant. One parameter has a prior of $(2\pi)^{-\frac{1}{2}} \sigma_\alpha^{-1} e^{-\frac{1}{2\sigma_\alpha^2} (\alpha_i)^2}$. Over all n nodes, plus the intercept, the log prior for α becomes

$$-\frac{n+1}{2} \log(2\pi) - (n+1) \log(\sigma_\alpha) - \sum_{i=1}^{n+1} \frac{1}{2\sigma_\alpha^2} \alpha_i^2$$

The standard deviation σ_α has a scaled inverse chi-square distribution, where $\frac{1}{\sigma_\alpha}$ is a chi-square. With a fixed number of degrees of freedom ν , and a fixed scaling parameter SSQ the log prior becomes

$$(\nu/2) \log(SSQ \nu/2) - \log \Gamma(\nu/2) - \frac{\nu SSQ}{2\sigma_\alpha} - (1 + \nu/2) \log(\sigma_\alpha)$$

The node intercepts τ_i are specified as Gaussian, with mean zero and standard deviation σ_{in} , the standard deviation of variables in a node. Over all n nodes, the log prior becomes

$$-\frac{n}{2} \log(2\pi) - n \log(\sigma_{in}) - \sum_{i=1}^n \frac{1}{2\sigma_{in}^2} \tau_i^2$$

Next, the β_i^j for node i and variable j are a function of the switches and deviates, $\beta_i^j = \gamma_i^j * (\sigma_{out} + r_i^j (\sigma_{in} - \sigma_{out}))$. The r_i^j have a binomial prior, while the γ_i^j have a Gaussian distribution with mean zero and variance 1. The prior (not the log prior) for a γ_i^j is a Gaussian density, $(2\pi)^{-\frac{1}{2}} e^{-\frac{1}{2}(\gamma_i^j)^2}$, similar to the α and τ . The binomial prior for an r_i^j

involves the initial inclusion probability p^j , which can be different for each variable. The prior is $(p^j)^{r_i^j} (1 - p^j)^{(1-r_i^j)}$. Taking the logarithm, and performing a double summation over the n nodes and v vars, yields

$$-\frac{nv}{2} \log(2\pi) + \sum_{i=1}^n \sum_{j=1}^v r_i^j \log(p^j) + (1 - r_i^j) \log(1 - p^j) - \frac{1}{2}(\gamma_i^j)^2$$

Putting it all together, the log posterior is the sum of the parts above, the log likelihood and the log prior.

2.2 Implementation

As seen in the previous section, the posterior distribution for a Bayesian neural network is quite complex. Like many Bayesian models, sampling from this distribution is extremely difficult. As part of their work, Neal [34] and Liang [23] wrote substantially new computational methods.

Unlike those works, no new methods are proposed in this paper. Instead, the implementation relies on many iterations of the Metropolis-Hastings algorithm [32]. The process rotates between updating α node coefficients and β predictor coefficients, while holding the other group constant. Since the values from the nodes to the output, α (and σ_α), look like linear regression coefficients based on node values h , these can be sampled efficiently using a Gibbs step [10] for Bayesian linear regression. The implementation uses a modified version of the code provided in the book by Rossi, Allenby, and McCulloch [41] that simplifies the procedure to take only one draw at a time.

On the other hand, as Lee notes in his book [20], the β values inside nodes have no simple prior distribution, and a Metropolis step is necessary. The initial implementation tracks the values in terms of r_i and the value on the normal curve γ . A move is proposed and the new posterior distribution is computed. If the posterior improves, the move is accepted. If the posterior does not improve, there is still a nonzero probability that the move will be made. This probabilistic move allows the traversal to escape local maxima and other sticking points. There are several different potential moves: an independence sampler, redraw all of one type of variable (τ, γ, r_i) , redraw a node, flip an r_i , and a symmetric

random walk on γ . The probability of each proposal can be adjusted by options in the program. Finding the “best” move proposal distribution is impossible in general, given the wide variety of data suited for this model. The programmed options allow for flexibility. There are, of course, many other ways to propose moves, but these suffice as a complete workable set.

REFERENCES

- [1] Akaike, H. (1974). A New Look at the Statistical Model Identification, *IEEE Transactions on Automatic Control* 19: 716-723.
- [2] Barbosa, C. H., et al. (2002). Bayesian Neural Networks on the Inference of Distillation Product Quality. *Proceedings of the VII Brazilian Symposium on Neural Networks*.
- [3] Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. Oxford: Oxford University Press.
- [4] Brown, P. J., M. Vannucci, and T. Fearn. (1998). Multivariate Bayesian Variable Selection and Prediction. *Journal of the Royal Statistical Society, Series B (Statistical Methodology)* 60: 627-641.
- [5] Buntine, W. L., and A. S. Weigend (1991). Bayesian back-propagation. *Complex Systems* 5: 603-643.
- [6] Cheng, B., and D. M. Titterton (1994). Neural Networks: A Review from a Statistical Perspective. *Statistical Science* 9: 2-30.
- [7] Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems* 2: 303-314.
- [8] Dempster, A. P. (1973). Alternatives to least squares in multiple regression. In *Multivariate Statistical Inference*, editors D. G. Kabe and R. P. Gupta, pages 25-40. New York: Elsevier.
- [9] Funahashi, K. (1989). On the approximate realization of continuous mappings by neural networks. *Neural Networks* 2: 183-192.
- [10] Geman, S., and D. Geman. (1984). Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 6: 721-741.
- [11] George, E. I., and R. E. McCulloch. (1993). Variable selection via Gibbs sampling. *Journal of the American Statistical Society* 88: 881-889.
- [12] George, E. I., and R. E. McCulloch. (1997). Approaches for Bayesian variable selection. *Statistica Sinica* 7: 339-373.
- [13] Green, P. J. (1995). Reversible jump Monte Carlo computation and Bayesian model determination. *Biometrika* 82: 711-732.

- [14] Hornik, K., M. Stinchcombe and H. White (1990). Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural Networks* 3: 551-560.
- [15] Hosmer, D. W., and S. Lemeshow. (2001). *Applied Logistic Regression, second edition*. New York: Wiley.
- [16] Kandel, Eric R., James H. Schwartz, and Thomas M. Jessell. (2001). *Principles of Neural Science, fourth edition*. New York: McGraw-Hill.
- [17] Lampinen, J., and A. Vektari. (2001). Bayesian approach for neural networks - review and case studies. *Neural Networks* 14: 257-274.
- [18] Lee, H. K. H. (1998). Model selection and model averaging for neural networks. Ph. D. thesis. Pittsburgh: Department of Statistics, Carnegie Mellon University.
- [19] Lee, H. K. H. (2001). Model selection and neural network classification. *Journal of Classification* 18: 227-243.
- [20] Lee, H. K. H. (2004). *Bayesian Nonparametrics via Neural Networks*. Philadelphia: Society for Industrial and Applied Mathematics.
- [21] Li, Yi, C. Campbell, and M. Tipping. (2002). Bayesian automatic relevance determination algorithms for classifying gene expression data. *Bioinformatics* 18: 1332-1339.
- [22] Liang, F., and W. H. Wong. (2001). Real parameter evolutionary Monte Carlo with applications in Bayesian mixture models. *Journal of the American Statistical Association* 96: 653-666.
- [23] Liang, F. (2003). An Effective Bayesian Neural Network Classifier with a Comparison Study to Support Vector Machine. *Neural Computation* 15: 1959-1989.
- [24] Liang, F., and A. Y. C. Kuk. (2004). A Finite Population Estimation Study with Bayesian Neural Networks. *Survey Methodology* 30: 219-234.
- [25] Liang, F. (2005a). Bayesian neural networks for nonlinear time series forecasting. *Statistics and Computing* 15: 13-29.
- [26] Liang, F. (2005b). Evidence Evaluation for Bayesian Neural Networks Using Contour Monte Carlo. *Neural Computation* 17: 1385-1410.
- [27] Mackay, D. J. C. (1992a). Bayesian interpolation. *Neural Computation* 4: 415-447.
- [28] Mackay, D. J. C. (1992b). A practical Bayesian framework for backpropagation networks. *Neural Computation* 4: 448-472.

- [29] Mackay, D. J. C. (1994). Bayesian non-linear modelling for the prediction competition. *ASHRAE Transactions* vol 100, pt 2, pages 1053-1062.
- [30] Marwala, T. (2007). Bayesian training of neural networks using genetic programming. *Pattern Recognition Letters* 28: 1452-1458.
- [31] McCulloch, W. S., and W. Pitts (1943). A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics* 5: 115-133.
- [32] Metropolis, N.; Rosenbluth, A. W.; Rosenbluth, M.N.; Teller, A.H.; Teller, E. (1953). Equations of State Calculations by Fast Computing Machines. *Journal of Chemical Physics* 21: 1087-1092.
- [33] Minsky, M., and S. Papert. (1969). *Perceptrons*. Cambridge, MA: MIT Press.
- [34] Neal, R. (1996). *Bayesian Learning for Neural Networks*. Lecture Notes in Statistics, no. 118. New York: Springer.
- [35] Neal, R. (1998). Assessing Relevance Determination Methods using DELVE. In *Neural Networks and Machine Learning*, C. M. Bishop (editor), pages 97-129. New York: Springer.
- [36] Orre, R., A. Lansner, A. Bate, and M. Lindquist. (2000). Bayesian neural networks with confidence estimations applied to data mining. *Computational Statistics and Data Analysis* 34: 473-493.
- [37] Prechelt, L. (1997). Early Stopping - but when? In *Neural Networks: Tricks of the Trade*, pages 55-69. Lecture Notes in Computer Science, no. 1524. New York: Springer.
- [38] Raftery, A. E., D. Madigan, and J. A. Hoeting. (1997). Bayesian Model Averaging for Linear Regression Models. *Journal of the American Statistical Association* 92: 179-191.
- [39] Ripley, B. D. (1996). *Pattern Recognition and Neural Networks*. Cambridge: Cambridge University Press.
- [40] Rosenblatt, M. (1958). The perceptron: a probabilistic model for information storage and recognition in the brain. *Psychological Review* 65: 386-408.
- [41] Rossi, P., G. Allenby, and R. E. McCulloch. (2005). *Bayesian Statistics and Marketing*. Hoboken, New Jersey: John Wiley and Sons.
- [42] Rumelhart, D. E., J. L. McClelland, and the PDP Research Group, eds. (1986). *Parallel Distributed Processing, Vols. 1 and 2*, Cambridge, MA: MIT Press.

- [43] Samarasinghe, S. (2007). *Neural Networks for Applied Sciences and Engineering: From Fundamentals to Complex Pattern Recognition*. Boca Raton, FL: Auerbach Publications.
- [44] Schwarz, G. (1978). Estimating the dimension of a model. *Annals of Statistics* 6: 461-464.
- [45] Sum, J., et al. (1999). An Adaptive Bayesian Pruning for Neural Networks in a Non-Stationary Environment. *Neural Computation* 11: 965-976.
- [46] Titterington, D. M. (2004). Bayesian Methods for Neural Networks and Related Models. *Statistical Science* 19: 128-139.
- [47] Weisberg, S. (1985). *Applied Linear Regression*, second edition. New York: Wiley.