MML Inference of Single-layer Neural Networks

Enes Makalic, Lloyd Allison and David L. Dowe

Abstract

Inference of the optimal neural network architecture for a specific dataset is a long standing and difficult problem. Although a number of researchers have proposed various model selection procedures, the problem still remains largely unsolved. The architecture of the neural network, (the number of hidden layers, hidden neurons, inputs, etc.) directly affects its performance. A network that is too simple will not learn the problem sufficiently well, resulting in poor performance. Conversely, a complex network can overfit and exhibit poor generalisation capabilities. This paper introduces a novel selection criterion, based on Minimum Message Length (MML), for inference of single hidden layer, fully-connected, feedforward neural networks. The criterion performance is demonstrated on several artificial and real datasets. Furthermore, the MML criterion is compared against an MDL-based criterion and variations of the Akaike's Information Criterion (AIC) and Bayesian Information Criterion (BIC). In all tests considered, the MML criterion never overfitted and performed as well as, and often better than other model selection criteria.

1 Introduction

Artificial neural networks are an efficient tool for classification and regression problems. At the present time the most popular neural network type in use is the Multilayer Perceptron (MLP) [11, 10]. MLPs are characterised by the number of hidden layers, hidden neurons and connections between the layers. The architecture of a network must be determined separately for each problem - there is no single, universal architecture suitable for all tasks. For the purposes of this paper, we are only concerned with single hidden layer feedforward neural networks. These networks are frequently used and can model any continuous function [13, 18, 19].

We note that the architecture of the network directly influences the success of the training process. If we choose a network that is too small, the network will not be able to learn the problem sufficiently well. In contrast, a network that is too large will over-fit and will commonly exhibit low generalisation performance. Consequently, the task of selecting the initial architecture is of significant importance to researchers in this field. To date, much of the research in the architecture selection area has concentrated on two approaches: constructive [25] and pruning [24]. Constructive algorithms start with a trivial network and progressively add more hidden neurons until a satisfactory solution is found. When to add extra neurons and when to stop the growing process are two important problems a constructive algorithm must address. Conversely, pruning algorithms start with a complex network and prune it by removing unnecessary components. The primary difficulty with pruning algorithms is the strategy used to decide whether a network component is removed.

This paper introduces a novel architecture selection criterion for neural networks based on Minimum Message Length (MML) [6, 7, 5]. The new criterion is neither a constructive nor a pruning strategy. Rather, it is an objective function that estimates the goodness of an inferred model. As such, it can be included with any architecture selection algorithm. We compare our criterion to several variations of the Akaike's Information Criterion (AIC), Bayesian Information Criterion (BIC) and a version of the Minimum Description Length (MDL) criterion.

Section 2 briefly introduces some related research in neural network architecture selection. Descriptions of MML and MML87 are given in Section 3 and Section 3.1. MML87 inference of neural networks is examined in detail in Section 4. All test results and discussion are given in Section 5. Limitations of the current approach and a brief summary of the main findings are given in Section 6 and Section 7 respectively.

2 Related Research

Several information theory based selection criteria that are applicable to neural network architecture selection exist. A number of researchers have proposed Bayesian inference methods that do not limit the complexity of the network model (see e.g. [12, 23]). Instead, neural networks that are as large as computationally feasible are chosen. It is argued that provided the priors for such models are properly specified, the networks will not overfit. However, due to high computational times associated with large networks, one is still interested in finding the smallest optimal network sufficient for a problem.

A criterion which is commonly used in model selection is the Akaike Information Criterion (AIC). AIC is based on the simple idea of penalising the likelihood function to prevent overfitting. A number of different variations of AIC exist. A list of common AIC variations [20] is shown in Table 1. The model with the minimum AIC is said to be 'optimal'.

	Definition
N	Size of dataset
M	Number of parameters (e.g. network weights and biases)
$(y-t)^2$	Squared error
AIC_1	$\log\left(\frac{\sum(y-t)^2}{N}\right) + \frac{2\log M}{N}$
AIC_2	$\log\left(\frac{\sum(y-t)^2}{N}\right) + \frac{2\sqrt{M}}{N}$
AIC_3	$\log\left(\frac{\sum(y-t)^2}{N}\right) + \frac{2M}{N}$
AIC_4	$\log\left(\frac{\sum(y-t)^2}{N}\right) + \frac{2M^2}{N}$
AIC_C	$\log\left(\frac{\sum(y-t)^2}{N}\right) + \frac{2M}{N-M-1}$

Table 1: Variations of the AIC

An alternative information criterion that also penalises the likelihood is Bayesian Information Criterion (BIC). BIC can be used to approximate the posterior probabilities of models and is derived from log of the Bayes factor for comparing a model to the null model. A list of BIC variations [20] is shown in Table 2. A model with minimum BIC is said to be optimal.

	Definition
N	Size of dataset
M	Number of parameters (e.g. network weights and biases)
$(y-t)^2$	Squared error
BIC_1	$\log\left(\frac{\sum(y-t)^2}{N}\right) + \frac{\log M \log N}{N}$
BIC_2	$\log\left(\frac{\sum(y-t)^2}{N}\right) + \frac{\sqrt{M}\log N}{N}$
BIC_3	$\log\left(\frac{\sum(y-t)^2}{N}\right) + \frac{M\log N}{N}$
BIC_4	$\log\left(\frac{\sum(y-t)^2}{N}\right) + \frac{M^2\log N}{N}$

Table 2: Variations of the BIC

Additionally, a number of Minimum Description Length (MDL) criteria for architecture selection have been developed. Brake et al [15, 22] define a MDL cost of a neural network, MDL_V, as:

$$\mathrm{MDL}_V = \log(k) + 2 \log \log(k) + k \times l + k(k-1) + m \times l \text{ bits} \tag{1}$$

where, k is the total number of neurons in the network, m is the number of directed arcs and l is the precision (in bits) with which weights are coded. Interestingly, no method of calculating l is derived. Instead, the authors somewhat arbitrarily choose l as 16 bits for integers and 32 bits for real numbers. The encoding of the data given the network structure is dependent on the underlying data form. If the data is binary, or bipolar, this is straightforward. Otherwise, for discrete data, each datum is encoded using 16 bits. In comparison, real valued data is encoded using 32 bits for each datum. If the network incorrectly classifies a pattern, the pattern 'is transmitted using the input together with the correct output'. The model with the minimum MDL_V is said to be optimal.

3 Overview of MML

Minimum Message Length (MML) [6, 7, 5] inductive inference is an objective function that can estimate the goodness of an inferred model. In the context of this paper, we are given a dataset and wish to infer the optimal neural network architecture for that dataset. Obviously, a large number of possible neural networks exist which solve the problem. Using MML, one can perform a direct comparison between alternative hypotheses (that is, neural networks) and select one which is optimal; that is, the smallest network architecture that can do the problem sufficiently well.

The main idea behind MML is often explained using the following simple scenario. Assume there exists a sender wishing to transmit a message to a receiver via a noiseless transmission channel (see Figure 1). MML states that the message is transmitted in two parts:

1. an encoding of the model θ, and
2. an encoding of the data given the model, $x|\theta$.

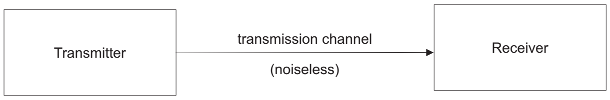

Figure 1: Sender-Receiver model

A model that minimises this two part message is deemed optimal. The first part of the message is arranged to be the answer to an inference problem that we are interested in. For example, the optimal neural network architecture for a particular dataset. The second part of the message reflects how well the model fits the data. Obviously, a simple model will have a shorter first part of the message than a complex model. However, the complex model may fit the data better and have a shorter second part of the message. When using MML, we optimise this trade-off between the model complexity and how well the model fits the data. This is illustrated in Figure 2.

3.1 MML87 Inference

The most commonly used MML approximation in practice is MML87 [7, 5]. MML87 is an efficient approximation to strict MML [9, 5, 14] and states that the total message length for a model Θ with

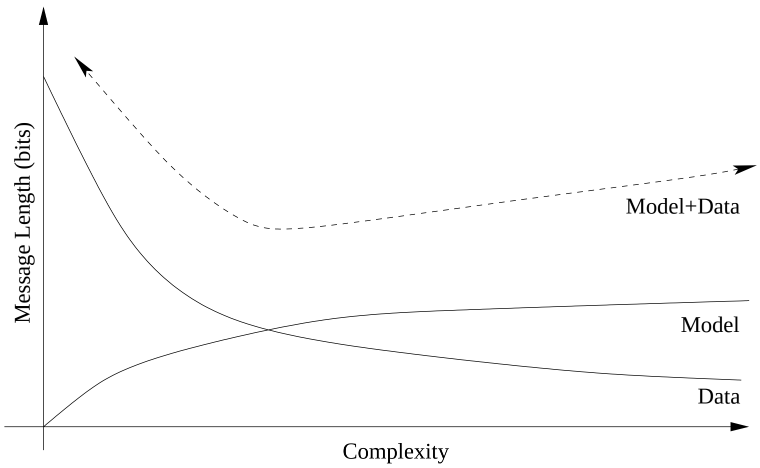

Figure 2: Trade-off between model and data complexity

parameters $\vec{\theta}$ is:

$$\begin{aligned}\text{msgLen}\,(\Theta) &= -\log\left(\frac{h(\vec{\theta})}{\kappa_n^{n/2}\sqrt{F(\vec{\theta})}}\right) - \log f(x|\vec{\theta}) + \frac{n}{2} \\ &= -\log h(\vec{\theta}) + \frac{1}{2}\log F(\vec{\theta}) - \log f(x|\vec{\theta}) + \frac{n}{2}\left(\log k_n + 1\right) \end{aligned} \tag{2}$$

where $h\left(\vec{\theta}\right)$ is the prior probability, $f(x|\vec{\theta})$ is the likelihood function, n is the number of parameters, κ is a dimension constant[1] and $F(\vec{\theta})$ is the determinant of the expected Fisher information matrix, whose entries (i, j) are:

$$\sum_{x\in\mathcal{X}} f(x|\vec{\theta})\frac{\partial^2}{\partial\theta_i\partial\theta_j}\left(-\log f(x|\vec{\theta})\right) \tag{3}$$

The Fisher information determines how sensitive the likelihood function is to the parameters $\vec{\theta}$. In the context of this paper, the Fisher information states how sensitive the output of a neural network is to the weights and biases. For example, if the output of a neural network is not very sensitive to a particular weight (or bias), we can state that weight (or bias) less precisely. Conversely, a weight that affects the network output significantly must be stated more accurately. A model that minimises the total message length (2) is said to be 'optimal'.

4 MML Inference of Neural Networks

The MML87 based neural network selection criterion is introduced in this section. Note that only the most significant formulae are provided here - the appendix contains complete derivations for each equation.

[1] Also called lattice constants. The first two are: $\kappa_1 = \frac{1}{12}$, $\kappa_2 = \frac{5}{36\sqrt{3}}$.

4.1 Notation

Consider a single-layer feedforward network comprising I inputs, H hidden neurons and O outputs. We can write a single output of the aforementioned network, $y_o^{(2)}$, as:

$$y_o^{(2)} = \sum_{h=1}^{H} \left(w_{ho}^{(2)} y_h^{(1)} \right) + b_o^{(2)} \tag{4}$$

where $\vec{w}^{(2)}$ and $b^{(2)}$ are the output weights and biases respectively. That is, $w_{ho}^{(2)}$ denotes the weight from hidden neuron h to output neuron o and $b_o^{(2)}$ denotes the bias for output neuron o.

The output of hidden neuron h with inputs $\vec{x} \in \mathbb{R}^I$ is:

$$y_h^{(1)} = f^{(1)} \left(\sum_{i=1}^{I} \left(w_{ih}^{(1)} x_i \right) + b_h^{(1)} \right) \tag{5}$$

where $f^{(1)}$ is a non-linear transfer function; $w_{ih}^{(1)}$ denotes the weight from input i to hidden neuron h and $b_h^{(1)}$ denotes the bias for hidden neuron h.

We are given a data set $D = \{(\vec{x_1}, \vec{t_1}), \ldots, (\vec{x}_p, \vec{t_p}), \ldots (\vec{x}_N, t_N)\}$ and wish to minimise the following error (performance) function:

$$E\left(\vec{t}, \vec{y}^{(2)}\right) = \sum_{p=1}^{N} \sum_{o=1}^{O} \left(y_{po}^{(2)} - t_{po} \right)^2 \tag{6}$$

This is commonly approximated by minimising the error for each pattern p:

$$E_p\left(\vec{t}_p, \vec{y}_p^{(2)}\right) = \sum_{o=1}^{O} \left(y_{po}^{(2)} - t_{po} \right)^2 \tag{7}$$

4.2 Likelihood Function

Assuming the target data from the function is generated with additive Gaussian noise and the data samples are independent, the likelihood function is:

$$f(\vec{t}|\vec{x}, \vec{w}, \vec{\beta}) = \prod_{p=1}^{N} \prod_{o=1}^{O} \frac{\beta_o}{\sqrt{2\pi}} e^{-\frac{\beta_o^2}{2}\left(y_{po}^{(2)} - t_{po}\right)^2} \tag{8}$$

where $\vec{w} = \{\vec{w}^{(1)}, \vec{w}^{(2)}, \vec{b}^{(1)}, \vec{b}^{(2)}\}$ (that is, all network weights). The standard deviation of network output o, β_o, is $\beta_o = \frac{1}{\sigma_o}$. Therefore, the negative log-likelihood function is:

$$L(\vec{t}|\vec{x}, \vec{w}, \vec{\beta}) = \sum_{p=1}^{N} \sum_{o=1}^{O} \left(\frac{\beta_o^2}{2} \left(y_{po}^{(2)} - t_{po} \right)^2 - \log \beta_o \right) + \frac{N \times O}{2} \log 2\pi \tag{9}$$

4.3 Fisher Information

The Fisher information matrix determines how sensitive the likelihood function is to the parameters (see Section 3.1) - that is, network weights and biases. Since most neural networks will have a large number of parameters, the Fisher information for the entire network is difficult to compute and prone to numerical instabilities [21]. Subsequently, we opted for the following block-diagonal approximation. The matrix is divided into blocks where each block corresponds to a single neuron. The approximation assumes individual neurons are independent. A diagram of the block-diagonal approximation is given in Figure 3.

The determinant of the full Fisher information matrix is equal to the product of the individual block determinants. This property reduces the time taken to evaluate the full determinant as well as the probability of numerical errors.

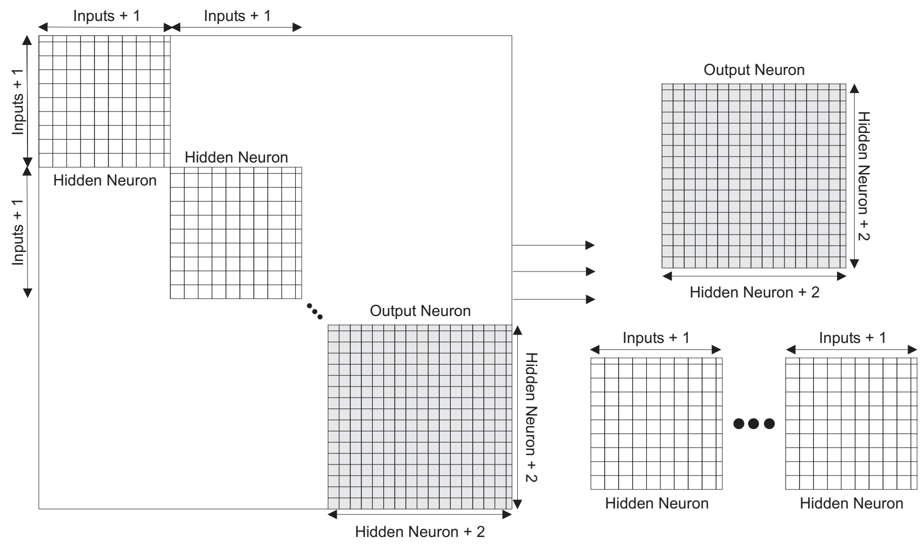

Figure 3: Fisher information block-diagonal approximation

4.3.1 Fisher Information for an Output Neuron

Given a feedforward network with H hidden neurons, the Fisher information matrix for output neuron o can be written as:

$$F = \beta_o^2 F_o \tag{10}$$

where:

$$F_o = \begin{pmatrix} \left(y_{p1}^{(1)}\right)^2 & \left(y_{p1}^{(1)} y_{p2}^{(1)}\right) & \dots & \left(y_{p1}^{(1)} y_{pH}^{(1)}\right) & \left(y_{p1}^{(1)}\right) & 0 \\ \left(y_{p2}^{(1)} y_{p1}^{(1)}\right) & \left(y_{p2}^{(1)}\right)^2 & \dots & \left(y_{p2}^{(1)} y_{pH}^{(1)}\right) & \left(y_{p2}^{(1)}\right) & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \left(y_{pH}^{(1)} y_{p1}^{(1)}\right) & \left(y_{pH}^{(1)} y_{p2}^{(1)}\right) & \dots & \left(y_{pH}^{(1)}\right)^2 & \left(y_{pH}^{(1)}\right) & 0 \\ \left(y_{p1}^{(1)}\right) & \left(y_{p2}^{(1)}\right) & \dots & \left(y_{pH}^{(1)}\right) & N & 0 \\ 0 & 0 & \dots & 0 & 0 & \frac{2N}{\beta_o^4} \end{pmatrix} \tag{11}$$

Note that the summation over all patterns has been omitted for reasons of clarity.

4.3.2 Fisher Information for a Hidden Neuron

The Fisher Information for a hidden neuron h is:

$$F = \sum_{o=1}^{O} \left(\beta_o w_{ho}^{(2)}\right)^2 F_h \tag{12}$$

where:

$$F_h = \begin{pmatrix} (x_{p1} f')^2 & \left(x_{p1} x_{p2} (f')^2\right) & \dots & \left(x_{p1} x_{pH} (f')^2\right) & \left(x_{p1} (f')^2\right) \\ \left(x_{p2} x_{p1} (f')^2\right) & (x_{p2} f')^2 & \dots & \left(x_{p2} x_{pH} (f')^2\right) & \left(x_{p2} (f')^2\right) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \left(x_{pH} x_{p1} (f')^2\right) & \left(x_{pH} x_{p2} (f')^2\right) & \dots & (x_{pH} f')^2 & \left(x_{pH} (f')^2\right) \\ \left(x_{p1} (f')^2\right) & \left(x_{p2} (f')^2\right) & \dots & \left(x_{pH} (f')^2\right) & (f')^2 \end{pmatrix} \tag{13}$$

Note that $f' = f_{ph}^{(1)\prime}$ and the sum over all patterns is omitted from each matrix entry for reasons of clarity.

4.3.3 Fisher Information for the Neural Network

From (10) and (12), the Fisher information for the entire neural network is:

$$F = \left(\prod_{o=1}^{O} \left| \beta_o^2 F_o \right| \right) \left(\prod_{h=1}^{H} \left| \sum_{o=1}^{O} \left(\beta_o w_{ho}^{(2)} \right)^2 F_h \right| \right) \tag{14}$$

Hence:

$$\frac{1}{2} \log F = \frac{1}{2} \sum_{o=1}^{O} \left(\log \left(\left| \beta_o^2 F_o \right| \right) \right) + \frac{1}{2} \sum_{h=1}^{H} \left(\log \left(\left| \sum_{o=1}^{O} \left(\beta_o w_{ho}^{(2)} \right)^2 F_h \right| \right) \right) \tag{15}$$

where:

$$\log \left(\left| \beta_o^2 F_o \right| \right) = 2(H+2) \log \beta_o + \log |F_o| \tag{16}$$

$$\log \left(\left| \sum_{o=1}^{O} \left(\beta_o w_{ho}^{(2)} \right)^2 F_h \right| \right) = (I+1) \log \left(\sum_{o=1}^{O} \left(\beta_o w_{ho}^{(2)} \right)^2 \right) + \log |F_h| \tag{17}$$

Due to numerical issues, $|F_o|$ and $|F_h|$ may be evaluated as zero or very small positive (or negative) numbers. This can often cause (16) and (17) to become negative, which is not acceptable. In such cases, (16) and/or (17) are set to zero.

4.4 Priors

The prior on a network weight or bias is:

$$h_w(w) = \frac{1}{40} \operatorname{sech}^2 \left(\frac{w}{20} \right) \tag{18}$$

The prior on β_o is:

$$h_\beta(\beta_o) = \frac{1}{\beta_o} \tag{19}$$

4.5 Message Length

Substituting (9), (15), (18) and (19) into (2) gives the total message length for a single hidden layer neural network:

$$\begin{aligned} \text{msgLen} &= \log^* H - \log \left(h_w(\vec{w}) h_\beta(\vec{\beta}) \right) + \frac{1}{2} \log F + L + \frac{n}{2} (\log k_n + 1) \\ &= \log^* H - \log (h_w(\vec{w})) - \log \left(h_\beta(\vec{\beta}) \right) + \frac{1}{2} \log F + L + \frac{n}{2} (\log k_n + 1) \end{aligned}$$

$$
\begin{aligned}
= \quad & \log^* H - \log\left(\prod_{m=1}^{M} \frac{1}{40}\operatorname{sech}^2\left(\frac{w_m}{20}\right)\right) + \log\left(\prod_{o=1}^{O} \beta_o\right) \\
& + \frac{1}{2}\sum_{o=1}^{O}\left(\log\left(\left|\beta_o^2 F_o\right|\right)\right) + \frac{1}{2}\sum_{h=1}^{H}\left(\log\left(\left|\sum_{o=1}^{O}\left(\beta_o w_{ho}^{(2)}\right)^2 F_h\right|\right)\right) \\
& + \sum_{p=1}^{N}\sum_{o=1}^{O}\left(\frac{\beta_o^2}{2}\left(y_{po}^{(2)} - t_{po}\right)^2 - \log\beta_o\right) + \frac{N \times O}{2}\log 2\pi \\
& + \frac{n}{2}\left(\log k_n + 1\right) \qquad (20)
\end{aligned}
$$

where M is the total number of weights and biases in the network. A network that minimises (20) is deemed optimal. The $\log^* H$ [16] in (20) denotes the cost of transmitting the network architecture - that is, the number of hidden neurons. Individual connections need not be transmitted under the assumption of fully connected networks. We define $\log^*$ as:

$$
\log^* H \approx \log(H) + \log(\log(H)) + \log(2.865) \qquad (21)
$$

4.5.1 MML Estimator for β_o

Assuming a single output network, the MML estimator for β_o is:

$$
\hat{\beta}_{MML} = \sqrt{\frac{N - M - 2}{\sum_{p=1}^{N}\left(y_{po}^{(2)} - t_{po}\right)^2}} \qquad (22)
$$

For multiple output networks, the MML estimator for β_o is not analytically solvable. Instead, we use (22) as an approximation to the MML estimator for β in all networks. Furthermore, for (22) to be meaningful, N must be greater than $(M + 2)$. Intuitively, inference of a model where the number of parameters is greater than the number of data does not make sense (although it can be done). Hence, the above constraint is minor.

5 Results and Discussion

The outlined MML87 based neural network criterion was tested on several artificial problems, where the optimal neural network architecture is known. Furthermore, a number of real datasets, for which the optimal neural network architecture is unknown, were also used. We compared our criterion against AIC and BIC variations and a version of the MDL selection criterion. Table 3 shows all functions used during testing. The neural network software was implemented in a data mining environment CDMS [17].

Function name	Function definition	Optimal network size
f_1	$f_1(x) = x^2 + N\left(0, \sigma^2\right)$	1 - 2 - 1
f_2	$f_2(x) = e^{-x}\sin(2\pi x) + N\left(0, \sigma^2\right)$	1 - 5 - 1
f_3	$f_3(x) = x(x-1)(x+1) + N\left(0, \sigma^2\right)$	1 - 3 - 1
Iris	UCI Repository of machine learning databases[8]	Unknown

Table 3: Functions used for criterion evaluation

For each artificial function, the testing procedure consisted of the following steps:

1. Train a neural network of optimal[2] size using the conjugate gradient with Polak-Ribier updates training algorithm. Since we know what the answer is for artificial problems, this is a reasonable thing to do.

2. Add noise to the output of this trained neural network. The neural network from (1) is then the *true* function we will try to infer.

3. Attempt to infer the original network (plus noise) from (1) using each of the following criteria: MML87, AIC, BIC and MDL_V.

The size of each data set was varied between $N = 25$ and $N = 400$. The noise standard deviation was also varied from $\sigma = 0.01$ to $\sigma = 0.35$. Furthermore, all function inputs were uniformly sampled in range $(-1, 1)$. During the testing process, the effect of dataset size and noise standard deviation was investigated for all criteria.

5.1 Function f_1

This is the simplest function and was originally modelled with a two hidden neuron network. MML inferred the optimal architecture for $\sigma \leq 0.15$ even with 25 data points. When $\sigma > 0.15$, MML underfitted and inferred one hidden neuron as the optimal architecture (see Fig. 4). Improvements in inference results were seen as the dataset was increased to 50 samples. Here, MML inferred the optimal architecture for all noise levels where $\sigma \leq 25$. When the amount of noise was too high ($\sigma > 0.25$), MML again underfitted. This is of course the proper thing to do. For $N = 100$, MML inferred the optimal architecture for all noise levels apart from $\sigma = 0.35$. When $\sigma = 0.35$, the MML criterion underfitted (see Fig. 4). Given datasets of 200 and 400 samples, our criterion inferred the optimal architecture for all noise levels considered.

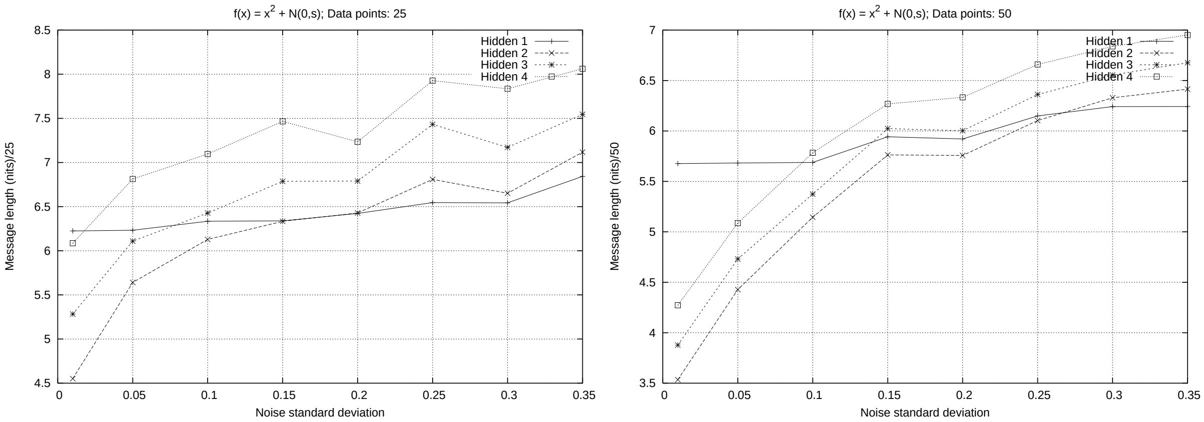

Figure 4: MML Inference of Function $f1$, $N = 25$ and $N = 50$

The MDL_V criterion inferred the optimal architecture for $\sigma \leq 0.05$ when using a dataset of 25 points. As the noise level was increased, $\sigma > 0.05$, the MDL_V criterion overfitted and inferred three and four hidden neurons as optimal (see Fig. 5). For $N = 50$, slightly improved inference results were observed. The MDL_V criterion inferred the optimal architecture for $\sigma = 0.01, 0.05, 0.15, 0.20$ and 0.30. Overfitting was again observed for $\sigma = 0.10, 0.25$, where MDL_V inferred three hidden neurons rather than two. For the largest amount of noise, MDL_V underfit - a network of one hidden neuron was inferred. This is

[2] An optimal network in this case is one that is much better than a smaller network; this *optimal* network therefore has very good prediction capabilities

illustrated in Fig. 5. Moreover, overfitting was also observed even for 'large' amounts of data - 100, 200 and 400 data points respectively.

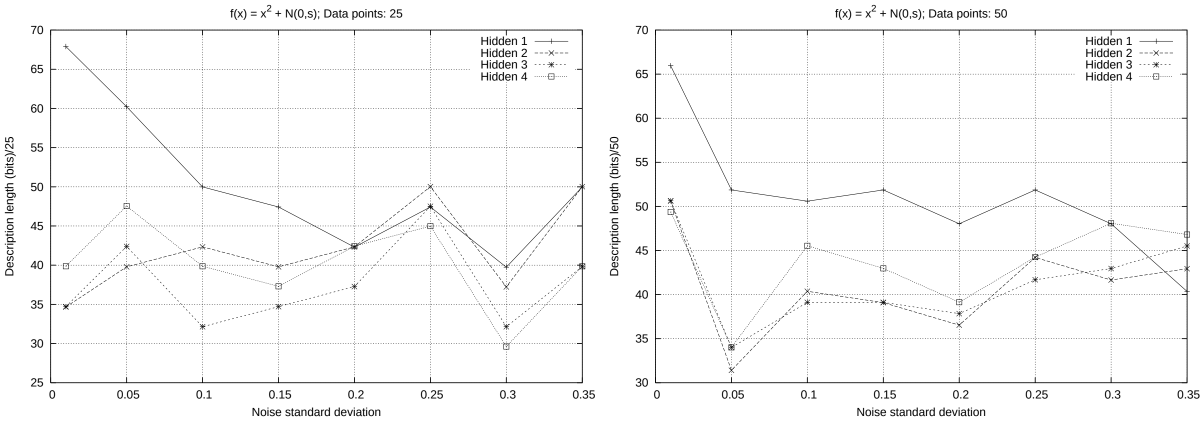

Figure 5: MDL_V Inference of Function $f1$, $N = 25$ and $N = 50$

In comparison, the AIC_1 and AIC_2 criteria inferred the optimal architecture for $\sigma \leq 0.05$ using a dataset of 25 points. For larger noise levels, both criteria overfitted and inferred much larger networks (from three to seven hidden neurons) as optimal (see Fig. 6). This is not desirable, especially for such a small dataset. As the amount of data was increased, similar overfitting was observed. Conversely, AIC_3 performed somewhat better. The criterion overfitted for all dataset sizes, albeit less often than AIC_1 and AIC_2. Interestingly, AIC_4 performed better still. For $N = 25$, a neural network of one hidden neuron was inferred as optimal for all noise levels considered. As the dataset size was increased, the performance of AIC_4 improved (see Fig. 6). For example, when $N \geq 200$, AIC_4 inferred the optimal architecture for all noise levels. The AICC criterion did not perform as well as AIC_4 - the optimal architecture was inferred less often. Additionally, overfitting was observed for some datasets (for example, $N = 25$).

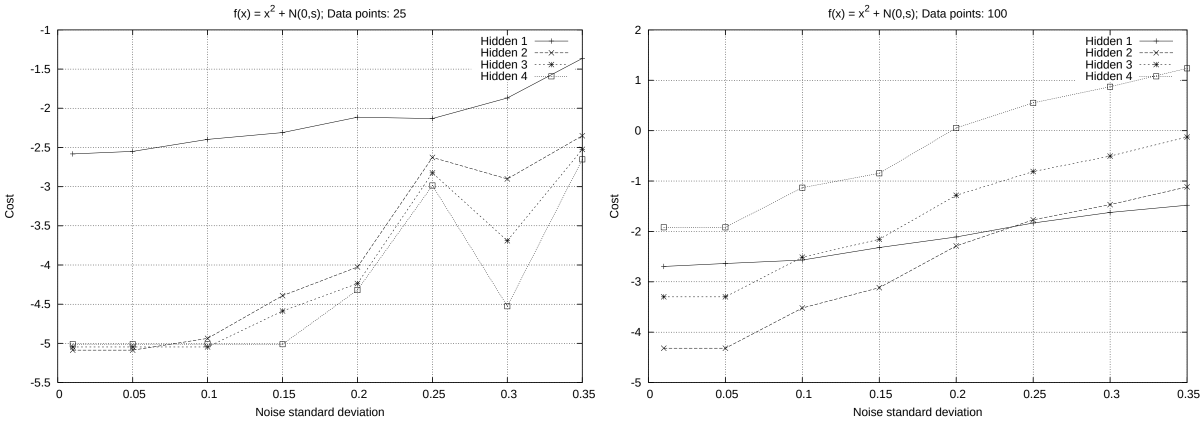

Figure 6: Inference of Function $f1$: AIC_2 (left) and AIC_4 (right)

The different variations of the BIC performed similarly to the aforementioned AIC-based methods. Both BIC_1 and BIC_2 did rather poorly, commonly overfitting regardless of the dataset size. BIC_3

performed much better although overfitting was still observed for $N = 25$ (see Fig. 7). For $N \geq 50$, BIC_3 inferred the optimal architecture for all noise levels considered. Comparatively, BIC_4 did reasonably well as no overfitting was observed for all dataset sizes and noise levels. However, BIC_4 often penalised the likelihood too much and inferred the simplest model as optimal. For example, when $N \leq 50$, BIC_4 inferred one hidden neuron as optimal for all noise levels considered (see Fig. 7).

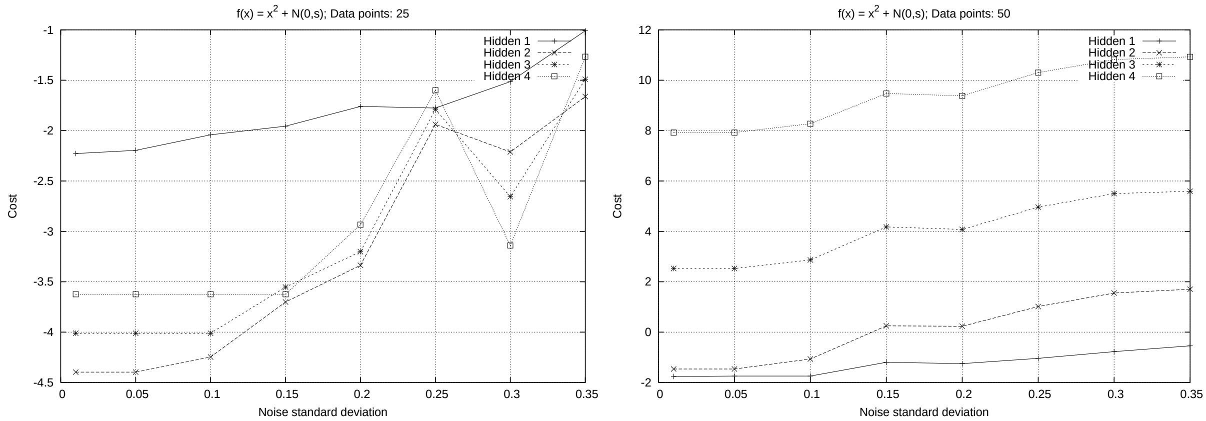

Figure 7: Inference of Function $f1$: BIC_3 (left) and BIC_4 (right)

5.2 Function f_2

Function f_2 is more difficult to model than f_1. Given a dataset of $N = 25$ samples, the MML criterion inferred a network with one hidden neuron as optimal for all noise levels considered. This is a reasonable thing to do for the small dataset. For $N = 50$, MML inferred the optimal architecture when $\sigma = 0.01$ (see Fig. 8). As the noise was increased, MML inferred three and four hidden neurons as optimal. For the largest amount of noise, our criterion inferred the simplest model (one hidden neuron). Similar results were observed when the dataset size was increased to $N = 100$ (see Fig. 8).

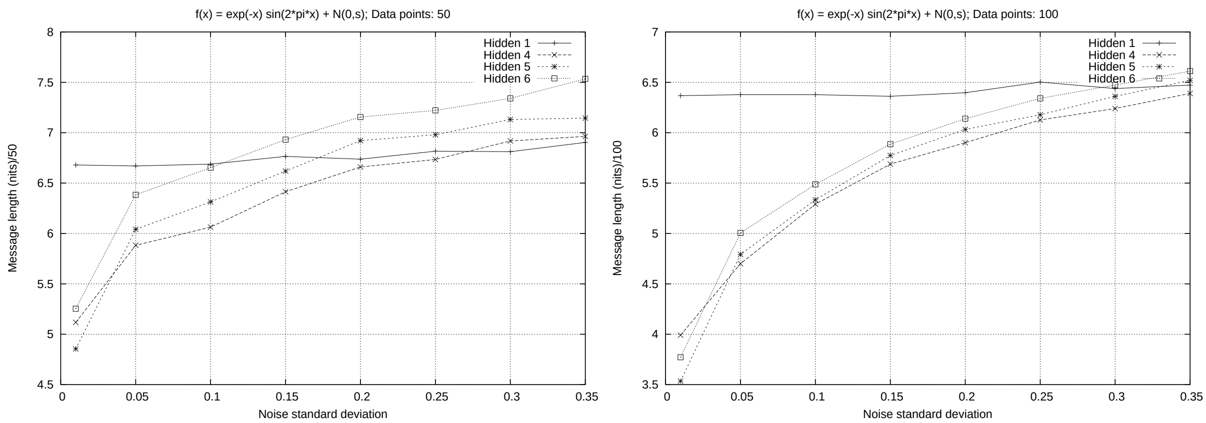

Figure 8: MML Inference of Function $f2$, $N = 50$ and $N = 100$

The MDL_V criterion inferred the optimal network size when $\sigma \leq 0.05$ and $N = 25$. For larger noise levels, MDL_V criterion underfitted - commonly inferring a network of three or four hidden neurons as optimal. For $N \geq 50$, similar results were observed (see Fig. 9). The optimal architecture was inferred for low noise levels, while increased noise levels caused underfitting. No overfitting was observed for all datasets and noise levels tested.

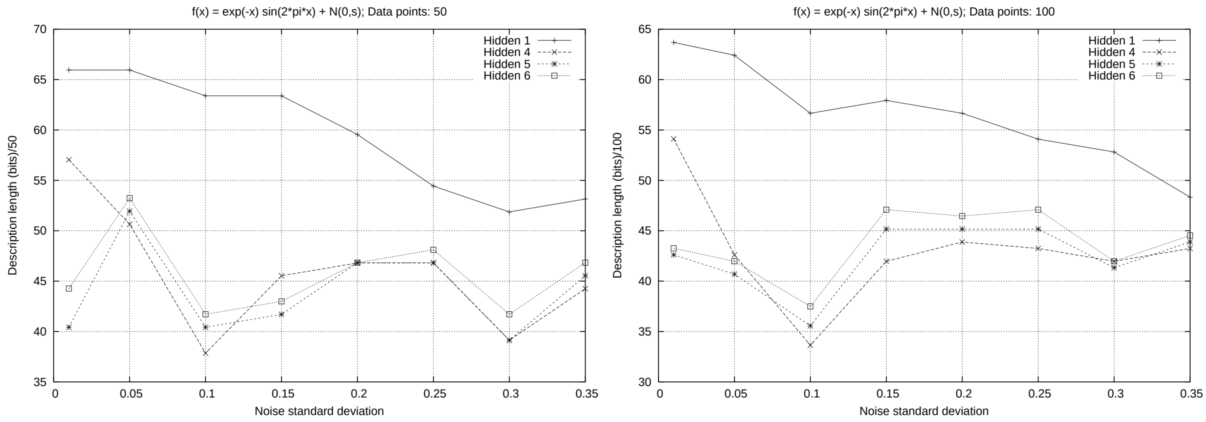

Figure 9: MDL Inference of Function $f2$, $N = 50$ and $N = 100$

Conversely, the AIC_1 and AIC_2 criteria performed very poorly for all datasets. Even when run on a small dataset, $N = 25$, both AIC_1 and AIC_2 overfitted. For example, AIC_1 inferred seven hidden neurons as optimal for $\sigma = 0.10, 0.15$. As the dataset size was increased, significant overfitting was still observed for both criteria. When $N \geq 200$, AIC_2 commonly inferred six or more hidden neurons as optimal rather than five. The AIC_3 criterion performed significantly better than both AIC_1 and AIC_2. Here, no overfitting was observed for all datasets tested. For $N = 25$, AIC_3 inferred three hidden neurons as optimal for most noise levels. When $\sigma = 0.10$, AIC_3 inferred the optimal architecture. As the dataset size was increased, $N \geq 50$, three or four hidden neurons were almost always inferred as optimal.

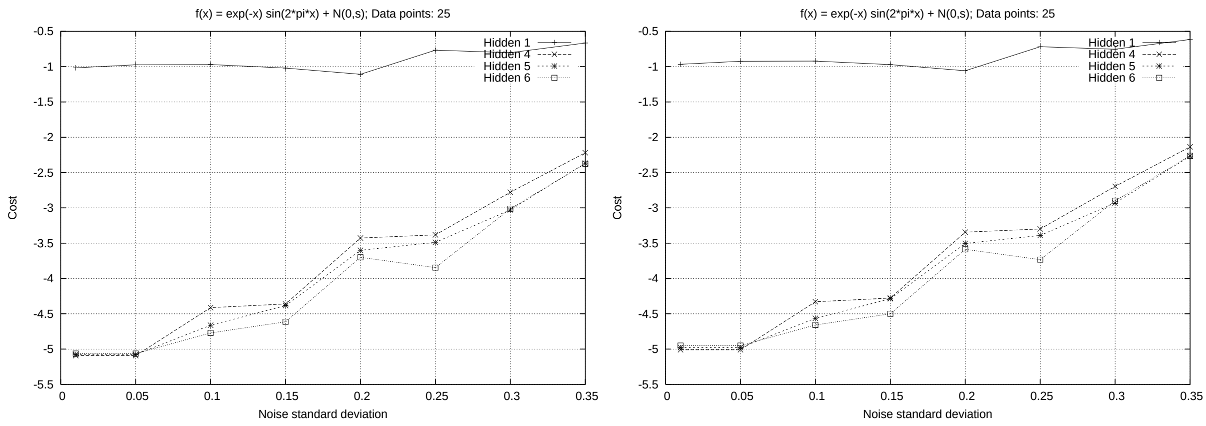

Figure 10: Inference of Function $f2$: AIC_1 (left) and AIC_2 (right)

Additionally, the likelihood penalty of AIC_4 caused the criterion to often underfit. For example, when $N = 25$, AIC_4 inferred one hidden neuron as optimal for all noise levels considered. Similarly, when

$N = 50$ one hidden neuron was inferred as optimal for $\sigma \geq 0.15$. When $\sigma < 0.15$, AIC_4 inferred three hidden neurons. Although the inference improved for larger datasets, AIC_4 never inferred the optimal architecture. In comparison, the AICC criterion inferred three hidden neurons for $N = 25$ and $\sigma \leq 0.30$. When $\sigma = 0.35$, the simplest network of one hidden neuron was inferred as optimal. For $N = 50$, AICC inferred three hidden neurons as optimal for all noise levels tested. As the amount of data was increased, similar results were obtained. No overfitting was observed in all tests.

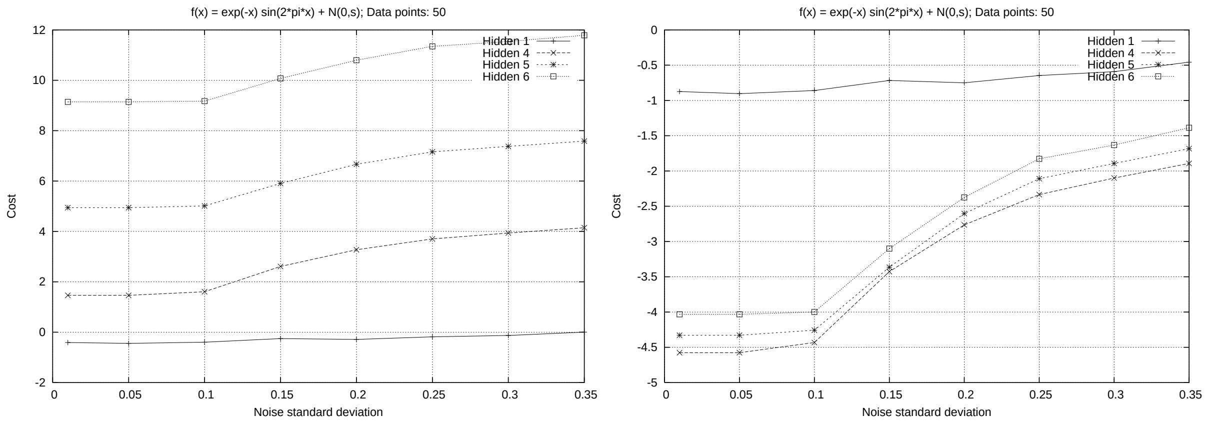

Figure 11: Inference of Function $f2$: AIC_4 (left) and AICC (right)

Moreover, the BIC_1 and BIC_2 criteria again performed poorly. Both criteria commonly overfitted even for small datasets (see Fig. 12). Adding more data did not stop overfitting. For example, when $N = 200$ both BIC_1 and BIC_2 again overfitted and inferred six or more hidden neurons as optimal. Conversely, BIC_3 performed better on all datasets considered. For $N \leq 200$, BIC_3 inferred three or four hidden neurons as optimal for almost all noise levels considered. Furthermore, BIC_4 inferred one hidden neuron as optimal for all $N \leq 50$. When $N = 100$ and $\sigma \leq 0.05$, a network of three hidden neurons was inferred as optimal. No overfitting was observed for BIC_4 in all tests.

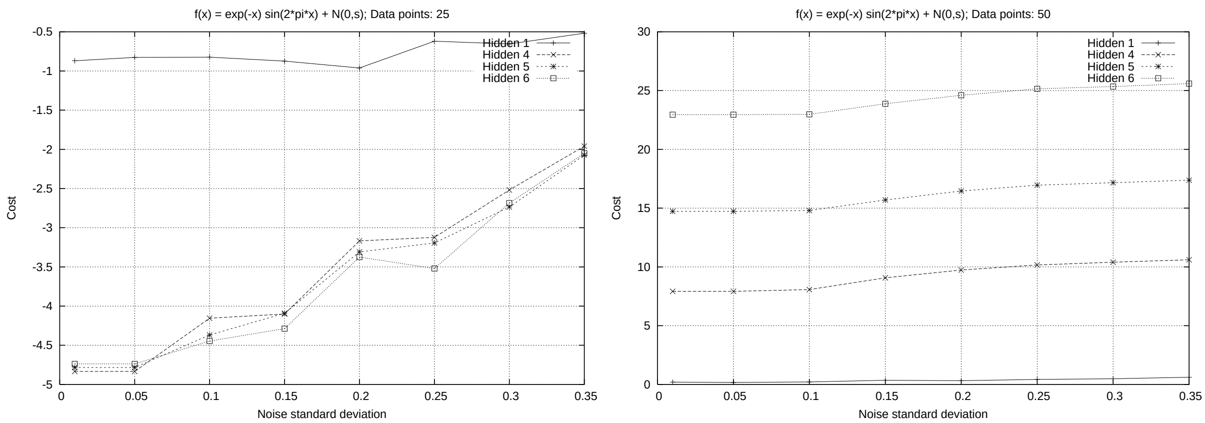

Figure 12: Inference of Function $f2$: BIC_2 (left) and BIC_4 (right)

5.3 Function f_3

In comparison to previous functions, the *optimal* neural network for function f_3 had three hidden neurons. Given only 25 data points, the MML criterion inferred a network of one hidden neuron as optimal for all noise levels. This is illustrated in Fig. 13. For $N = 50$, MML inferred the optimal architecture when $\sigma = 0.01$. MML underfitted for $\sigma = 0.05$ and inferred the optimal network to have two hidden neurons. As the noise was increased, MML inferred one hidden neuron as optimal (see Fig. 13). This is of course a reasonable thing to do.

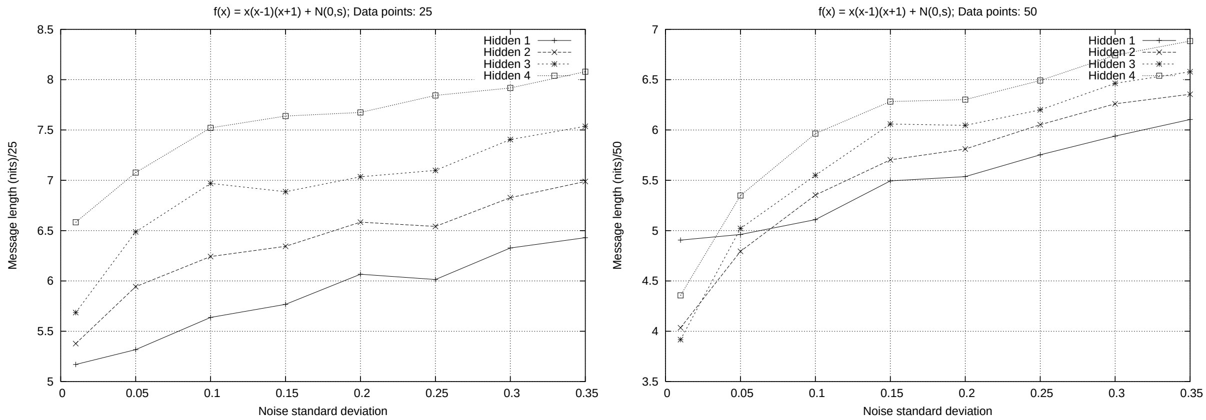

Figure 13: MML Inference of Function $f3$, $N = 25$ and $N = 50$

The MDL_V criterion inferred the optimal architecture when $N = 25$ and $\sigma = 0.01, 0.05, 0.20, 0.25, 0.35$. For all other noise levels (that is, $\sigma = 0.10, 0.15, 0.30$), MDL_V inferred two hidden neurons as optimal. This is shown in Fig. 14. Similar results were observed for $N = 50$. Here, MDL_V inferred the optimal architecture for $\sigma \leq 0.05$. However, the MDL_V criterion overfitted for larger data sets. When $N = 200$ and $\sigma = 0.2$, MDL_V inferred four hidden neurons as optimal rather than three (see Fig. 14).

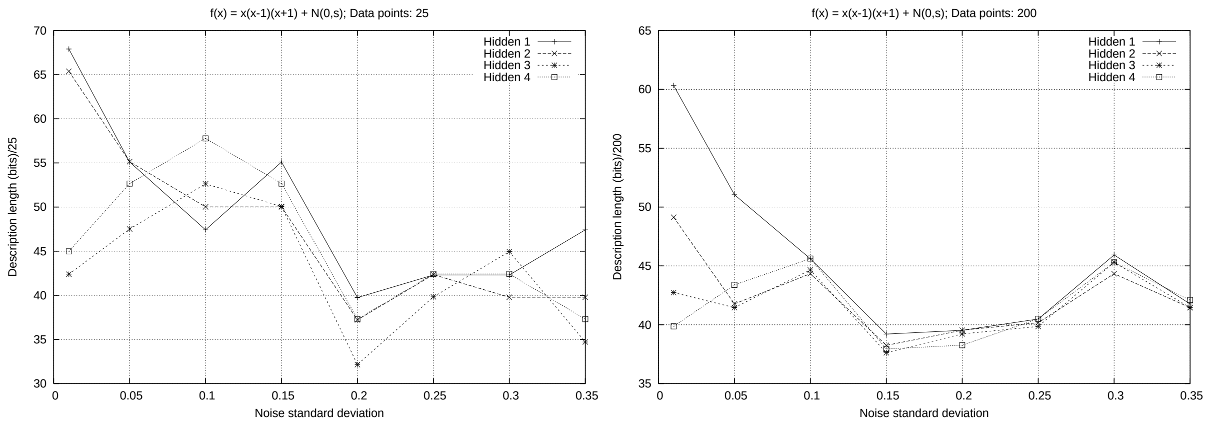

Figure 14: MDL Inference of Function $f3$, $N = 25$ and $N = 200$

Conversely, the AIC_1 and AIC_2 criteria again performed rather poorly. Significant overfitting was observed even for small datasets, such as $N = 25$. Increasing the amount of data did not improve the performance of either method as overfitting was still common (see Fig. 15). In comparison, given a dataset of 25 samples, AIC_3 inferred two hidden neurons as optimal for $\sigma \leq 0.10$. When $\sigma = 0.20$, AIC_3 overfitted and inferred four hidden neurons as optimal, rather than three. This is shown in Fig. 15. No overfitting was observed for larger datasets.

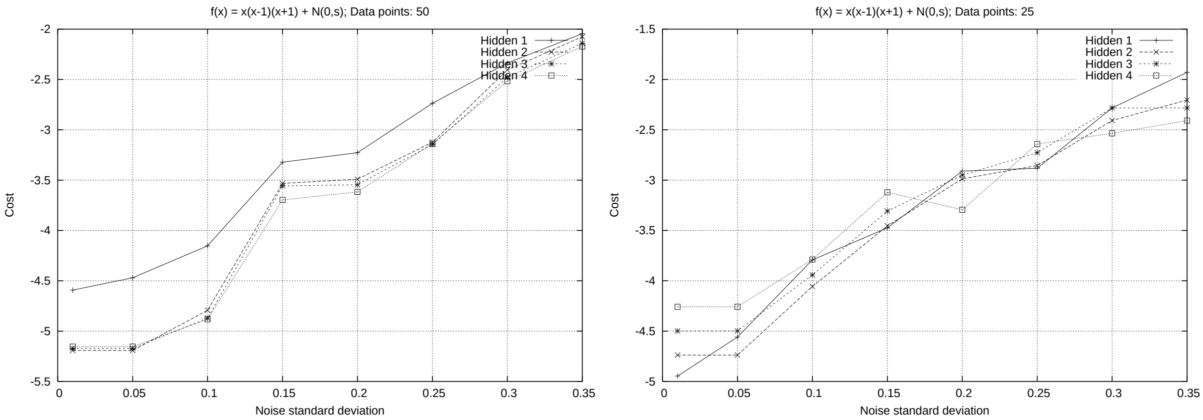

Figure 15: Inference of Function $f3$: AIC_2 (left) and AIC3 (right)

The AIC_4 criterion inferred the simplest network as optimal for $N \leq 50$. As the amount of data was increased, AIC_4 inferred one or two hidden neurons as optimal. Interestingly, AIC_4 never inferred the optimal architecture. Moreover, the AICC criterion performed rather similarly and inferred one or two hidden neurons as the optimal architecture for all datasets. Both AIC_4 and AICC did not overfit for all datasets and noise levels tested.

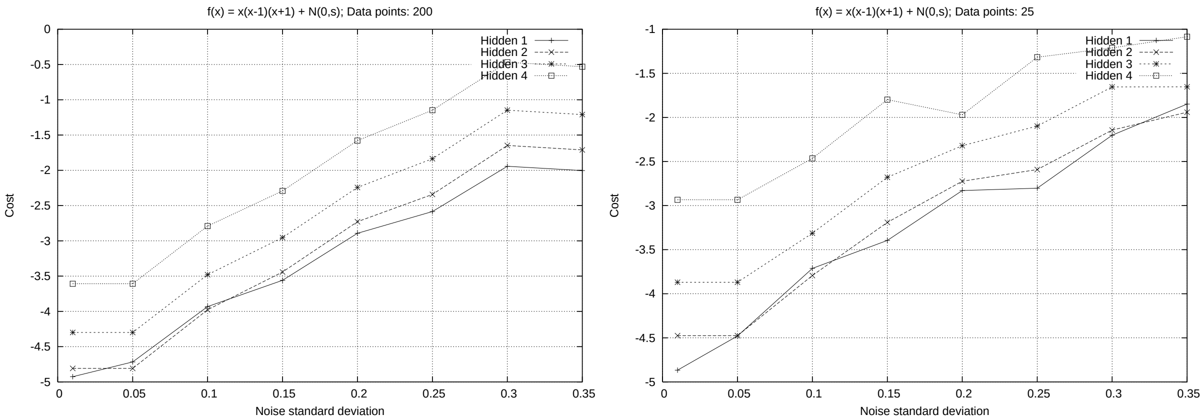

Figure 16: Inference of Function $f3$: AIC_4 (left) and AICC (right)

As in f_1 and f_2, the BIC_1 and BIC_2 criteria performed poorly. Overfitting was observed for some noise levels in all datasets tested. In comparison, BIC_3 inferred one or two hidden neurons as optimal for all datasets and noise levels. The last BIC variant examined, namely BIC_4, did not do well for function f_3. BIC_4 inferred the simplest network as optimal for all $N \leq 200$.

5.4 Iris Dataset

The iris dataset is a classification task and requires a network to classify between three different types of iris plants. The dataset comprises three classes of 50 instances each. The first two input attributes, namely sepal length and sepal width, were removed due to low class correlation. The optimal neural networks inferred by each criterion are shown in Table 4. The consensus is that a two hidden neuron network is optimal for this dataset [3, 2]. The following criteria inferred the optimal network size: MML, MDL_V, AIC_3 and AICC. The remaining criteria either overfitted (AIC_1, AIC_2, BIC_1 and BIC_2) or underfitted (AIC_4, BIC_3 and BIC_4). These conclusions are in agreement with the results observed for the three artificial datasets.

Criterion name	Optimal network (I-H-O)
MML	$1-2-3$
MDL_V	$1-2-3$
AIC_1	$1-3-3$
AIC_2	$1-3-3$
AIC_3	$1-2-3$
AIC_4	$1-1-3$
AICC	$1-2-3$
BIC_1	$1-3-3$
BIC_2	$1-3-3$
BIC_3	$1-1-3$
BIC_4	$1-1-3$

Table 4: Inferred optimal networks for the Iris dataset

6 Limitations and Future Work

Although the MML criterion performed well on the artificial and real datasets, several limitations of the current method exist. Firstly, our criterion is only applicable to single hidden layer neural networks. This is not a major limitation as such networks can approximate any real, continuous function to arbitrary accuracy. In all conducted tests, maximum likelihood estimators for the network weights (and biases) were used. It would be of interest to obtain MML estimators and publish a comparison between the two. Additionally, the current MML approximation is mainly suitable for regression problems and may not perform well when applied to n-ary classification.

Future work will include an extension of the current criterion to networks with multiple hidden layers. Evaluation of more priors for network parameters and the robustness of MML to the choice of prior is necessary. Stochastic methods using Markov Chain Monte Carlo (MCMC) techniques [4] are also of interest. Moreover, MML inference of neural networks for classification is being looked at.

7 Conclusion

The architecture selection problem in neural networks is a difficult research topic. This paper has introduced a novel selection criterion for single hidden layer, feedforward neural networks. In particular, we have addressed the issue of architecture selection in neural networks for regression. The new criterion is based on MML87 and uses a block-diagonal Fisher Information approximation that assumes neuron independence. Tests on real and artificial datasets were conducted to analyse the criterion performance. Our criterion performed very well in all test cases as compared to the MDL_V criterion and several variations of the AIC and BIC criteria. For all test functions, MML did not once overfit. When the amount of data was too small (as compared to the parameter space), MML underfitted as is only reasonable. Further application of the MML criterion to music genre classification [1] has shown promising results. Conversely, the MDL_V criterion performed inconsistently and has overfitted functions f_1 and f_3. The AIC_1, AIC_2, BIC_1 and BIC_2 criteria performed rather poorly in almost all test cases. These criteria overfitted even for small dataset sizes. BIC_4 often underfitted given datasets comprising 100 or more samples due to the severe likelihood function penalty. The remaining criteria, namely AIC_3, AIC_4, AICC and BIC_3, performed reasonably well on most functions tested. However, overfitting was still evident especially for AIC_3 and BIC_3.

8 Appendix

8.1 Equation Derivations

8.1.1 Likelihood Function

$$\begin{aligned}
L(\vec{t}|\vec{x},\vec{w},\vec{\beta}) &= -\log f(t|\vec{x},\vec{w},\vec{\beta}) \\
&= -\log\left(\prod_{p=1}^{N}\prod_{o=1}^{O}\frac{\beta_o}{\sqrt{2\pi}}e^{-\frac{\beta_o^2}{2}\left(y_{po}^{(2)}-t_{po}\right)^2}\right) \\
&= -\sum_{p=1}^{N}\sum_{o=1}^{O}\log\left(\frac{\beta_o}{\sqrt{2\pi}}e^{-\frac{\beta_o^2}{2}\left(y_{po}^{(2)}-t_{po}\right)^2}\right) \\
&= -\sum_{p=1}^{N}\sum_{o=1}^{O}\left(\log\frac{\beta_o}{\sqrt{2\pi}}-\frac{\beta_o^2}{2}\left(y_{po}^{(2)}-t_{po}\right)^2\right) \\
&= \sum_{p=1}^{N}\sum_{o=1}^{O}\left(\frac{\beta_o^2}{2}\left(y_{po}^{(2)}-t_{po}\right)^2-\log\frac{\beta_o}{\sqrt{2\pi}}\right) \\
&= \sum_{p=1}^{N}\sum_{o=1}^{O}\left(\frac{\beta_o^2}{2}\left(y_{po}^{(2)}-t_{po}\right)^2-\log\beta_o+\frac{1}{2}\log 2\pi\right) \\
&= \sum_{p=1}^{N}\sum_{o=1}^{O}\left(\frac{\beta_o^2}{2}\left(y_{po}^{(2)}-t_{po}\right)^2-\log\beta_o\right)+\frac{N\times O}{2}\log 2\pi
\end{aligned}$$

8.1.2 Fisher Information

First Partial Derivatives for the Output Neuron

First partial derivatives of (23):

$$\begin{aligned}
\frac{\partial L}{\partial \beta_o} &= \sum_{p=1}^{N}\left(\beta_o\left(y_p^{(2)}-t_p\right)^2-\frac{1}{\beta_o}\right) \\
&= \beta_o\sum_{p=1}^{N}\left(y_p^{(2)}-t_p\right)^2-\frac{N}{\beta_o} \\
\frac{\partial L}{\partial w_{ho}^{(2)}} &= \sum_{p=1}^{N}\frac{\beta_o^2}{2}2\left[y_{ph}^{(1)}\left(y_{po}^{(2)}-t_{po}\right)\right] \\
&= \beta_o^2\sum_{p=1}^{N}\left[y_{ph}^{(1)}\left(y_{po}^{(2)}-t_{po}\right)\right]
\end{aligned}$$

$$
\begin{aligned}
\frac{\partial L}{\partial b_o^{(2)}} &= \sum_{p=1}^{N} \frac{\beta_o^2}{2} 2 \left(y_{po}^{(2)} - t_{po} \right) \\
&= \beta_o^2 \sum_{p=1}^{N} \left(y_{po}^{(2)} - t_{po} \right)
\end{aligned}
$$

Second Partial Derivatives for the Output Neuron

Second partial derivatives of (23) are:

$$
\begin{aligned}
\frac{\partial^2 L}{\partial \beta^2} &= \sum_{p=1}^{N} \left(y_{po}^{(2)} - t_{po} \right)^2 + \frac{N}{\beta_o^2} \\
\frac{\partial^2 L}{\partial \left(w_{ho}^{(2)} \right)^2} &= \beta_o^2 \sum_{p=1}^{N} \left(y_{ph}^{(1)} \frac{\partial y_{po}^{(2)}}{\partial w_{ho}^{(2)}} \right) \\
&= \beta_o^2 \sum_{p=1}^{N} \left(y_{ph}^{(1)} \right)^2 \\
\frac{\partial^2 L}{\partial \left(b_o^{(2)} \right)^2} &= \beta_o^2 \sum_{p=1}^{N} \left(\frac{\partial y_{po}^{(2)}}{\partial b_o^{(2)}} \right) \\
&= \beta_o^2 \sum_{p=1}^{N} (1) \\
&= N \beta_o^2 \\
\frac{\partial^2 L}{\partial b_o^{(2)} \, \partial w_{ho}^{(2)}} &= \beta_o^2 \sum_{p=1}^{N} \left(\frac{\partial y_{po}^{(2)}}{\partial w_{ho}^{(2)}} \right) \\
&= \beta_o^2 \sum_{p=1}^{N} \left(y_{ph}^{(1)} \right) \\
\frac{\partial^2 L}{\partial w_{go}^{(2)} \, \partial w_{ho}^{(2)}} &= \beta_o^2 \sum_{p=1}^{N} \left(y_{pg}^{(1)} \frac{\partial y_{po}^{(2)}}{\partial w_{ho}^{(2)}} \right) \\
&= \beta_o^2 \sum_{p=1}^{N} \left(y_{pg}^{(1)} y_{ph}^{(1)} \right) \\
\frac{\partial^2 L}{\partial \beta \, \partial b_o^{(2)}} &= 2\beta_o \sum_{p=1}^{N} \left(\left(y_{po}^{(2)} - t_{po} \right) \frac{\partial y_{po}^{(2)}}{\partial b_o^{(2)}} \right) \\
&= 2\beta_o \sum_{p=1}^{N} \left(y_{po}^{(2)} - t_{po} \right) \\
\frac{\partial^2 L}{\partial \beta_o \, \partial w_{ho}^{(2)}} &= 2\beta_o \sum_{p=1}^{N} \left(\left(y_{po}^{(2)} - t_{po} \right) \frac{\partial y_{po}^{(2)}}{\partial w_{ho}^{(2)}} \right) \\
&= 2\beta_o \sum_{p=1}^{N} \left[y_{ph}^{(1)} \left(y_{po}^{(2)} - t_{po} \right) \right]
\end{aligned}
$$

Expectations:

$$\begin{aligned}
E_t\left\{\frac{\partial^2 L}{\partial\beta_o\,\partial b_o^{(2)}}\right\} &= 0 \\
E_t\left\{\frac{\partial^2 L}{\partial\beta_o\,\partial w_{ho}^{(2)}}\right\} &= 0 \\
E_t\left\{\frac{\partial^2 L}{\partial\beta_o^2}\right\} &= \frac{2N}{\beta_o^2}
\end{aligned}$$

First Partial Derivatives for a Hidden Neuron

First partial derivatives of (23) for hidden neuron h:

$$\begin{aligned}
\frac{\partial L}{\partial w_{ih}^{(1)}} &= \sum_{p=1}^{N}\sum_{o=1}^{O}\left(\frac{\partial}{\partial w_{ih}^{(1)}}\left\{\frac{\beta_o^2}{2}\left(y_{po}^{(2)}-t_{po}\right)^2-\log\beta_o\right\}\right) \\
&= \sum_{p=1}^{N}\sum_{o=1}^{O}\left(\frac{\beta_o^2}{2}2\left(y_{po}^{(2)}-t_{po}\right)\frac{\partial y_{po}^{(2)}}{\partial w_{ih}^{(1)}}\right) \\
&= \sum_{p=1}^{N}\sum_{o=1}^{O}\left(\beta_o^2\left(y_{po}^{(2)}-t_{po}\right)w_{ho}^{(2)}x_{pi}f_{ph}^{(1)\prime}\right) \\
&= \sum_{p=1}^{N}\left(x_{pi}f_{ph}^{(1)\prime}\sum_{o=1}^{O}\left(\beta_o^2 w_{ho}^{(2)}\left(y_{po}^{(2)}-t_{po}\right)\right)\right) \\
\frac{\partial L}{\partial b_h^{(1)}} &= \sum_{p=1}^{N}\sum_{o=1}^{O}\left(\frac{\partial}{\partial b_h^{(1)}}\left\{\frac{\beta_o^2}{2}\left(y_{po}^{(2)}-t_{po}\right)^2-\log\beta_o\right\}\right) \\
&= \sum_{p=1}^{N}\sum_{o=1}^{O}\left(\frac{\beta_o^2}{2}2\left(y_{po}^{(2)}-t_{po}\right)\frac{\partial y_{po}^{(2)}}{\partial b_h^{(1)}}\right) \\
&= \sum_{p=1}^{N}\sum_{o=1}^{O}\left(\beta_o^2\left(y_{po}^{(2)}-t_{po}\right)w_{ho}^{(2)}f_{ph}^{(1)\prime}\right) \\
&= \sum_{p=1}^{N}\left(f_{ph}^{(1)\prime}\sum_{o=1}^{O}\left(\beta_o^2 w_{ho}^{(2)}\left(y_{po}^{(2)}-t_{po}\right)\right)\right)
\end{aligned}$$

Second Partial Derivatives for a Hidden Neuron

Second partial derivatives of (23) for hidden neuron h:

$$\begin{aligned}
\frac{\partial^2 L}{\partial\left(w_{ih}^{(1)}\right)^2} &= \sum_{p=1}^{N}\left(x_{pi}\frac{\partial}{\partial w_{ih}^{(1)}}\left\{f_{ph}^{(1)\prime}\sum_{o=1}^{O}\left(\beta_o^2 w_{ho}^{(2)}\left(y_{po}^{(2)}-t_{po}\right)\right)\right\}\right) \\
&= \sum_{p=1}^{N}\left(x_{pi}\left(\frac{\partial A}{\partial w_{ih}^{(1)}}B+A\frac{\partial B}{\partial w_{ih}^{(1)}}\right)\right) \\
\frac{\partial^2 L}{\partial\left(b_h^{(1)}\right)^2} &= \sum_{p=1}^{N}\left(\frac{\partial}{\partial b_h^{(1)}}\left\{f_{ph}^{(1)\prime}\sum_{o=1}^{O}\left(\beta_o^2 w_{ho}^{(2)}\left(y_{po}^{(2)}-t_{po}\right)\right)\right\}\right)
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{p=1}^{N} \left(\frac{\partial A}{\partial b_h^{(1)}} B + A \frac{\partial B}{\partial b_h^{(1)}} \right) \\
\frac{\partial^2 L}{\partial w_{ih}^{(1)}\ \partial b_h^{(1)}} &= \sum_{p=1}^{N} \left(x_{pi} \frac{\partial}{\partial b_h^{(1)}} \left\{ f_{ph}^{(1)\prime} \sum_{o=1}^{O} \left(\beta_o^2 w_{ho}^{(2)} \left(y_{po}^{(2)} - t_{po} \right) \right) \right\} \right) \\
&= \sum_{p=1}^{N} \left(x_{pi} \left(\frac{\partial A}{\partial b_h^{(1)}} B + A \frac{\partial B}{\partial b_h^{(1)}} \right) \right) \\
\frac{\partial^2 L}{\partial w_{ih}^{(1)}\ \partial w_{jh}^{(1)}} &= \sum_{p=1}^{N} \left(x_{pi} \frac{\partial}{\partial w_{jh}^{(1)}} \left\{ f_{ph}^{(1)\prime} \sum_{o=1}^{O} \left(\beta_o^2 w_{ho}^{(2)} \left(y_{po}^{(2)} - t_{po} \right) \right) \right\} \right) \\
&= \sum_{p=1}^{N} \left(x_{pi} \left(\frac{\partial A}{\partial w_{jh}^{(1)}} B + A \frac{\partial B}{\partial w_{jh}^{(1)}} \right) \right)
\end{aligned}
$$

where:

$$
\begin{aligned}
A &= f_{ph}^{(1)\prime} \\
B &= \sum_{o=1}^{O} \beta_o^2 w_{ho}^{(2)} \left(y_{po}^{(2)} - t_{po} \right)
\end{aligned}
$$

Expectations:

$$
\begin{aligned}
E_t \left\{ \frac{\partial^2 L}{\partial \left(w_{ih}^{(1)} \right)^2} \right\} &= \sum_{p=1}^{N} \left(x_{pi} A \frac{\partial B}{\partial w_{ih}^{(1)}} \right) \\
&= \sum_{p=1}^{N} \left(x_{pi} f_{ph}^{(1)\prime} \left(\sum_{o=1}^{O} \beta_o^2 w_{ho}^{(2)} \frac{\partial y_{po}^{(2)}}{\partial w_{ih}^{(1)}} \right) \right) \\
&= \sum_{p=1}^{N} \left(x_{pi} f_{ph}^{(1)\prime} \left(\sum_{o=1}^{O} \beta_o^2 w_{ho}^{(2)} w_{ho}^{(2)} x_{pi} f_{ph}^{(1)\prime} \right) \right) \\
&= \sum_{p=1}^{N} \left(\left(x_{pi} f_{ph}^{(1)\prime} \right)^2 \sum_{o=1}^{O} \left(\beta_o w_{ho}^{(2)} \right)^2 \right) \\
&= \sum_{o=1}^{O} \left(\beta_o w_{ho}^{(2)} \right)^2 \sum_{p=1}^{N} \left(x_{pi} f_{ph}^{(1)\prime} \right)^2 \\
E_t \left\{ \frac{\partial^2 L}{\partial \left(b_h^{(1)} \right)^2} \right\} &= \sum_{p=1}^{N} \left(A \frac{\partial B}{\partial b_h^{(1)}} \right) \\
&= \sum_{p=1}^{N} \left(f_{ph}^{(1)\prime} \left(\sum_{o=1}^{O} \beta_o^2 w_{ho}^{(2)} \frac{\partial y_{po}^{(2)}}{\partial b_h^{(1)}} \right) \right) \\
&= \sum_{p=1}^{N} \left(f_{ph}^{(1)\prime} \left(\sum_{o=1}^{O} \beta_o^2 w_{ho}^{(2)} w_{ho}^{(2)} f_{ph}^{(1)\prime} \right) \right) \\
&= \sum_{p=1}^{N} \left(\left(f_{ph}^{(1)\prime} \right)^2 \sum_{o=1}^{O} \left(\beta_o w_{ho}^{(2)} \right)^2 \right)
\end{aligned}
$$

$$
\begin{aligned}
&= \sum_{o=1}^{O}\left(\beta_o w_{ho}^{(2)}\right)^2 \sum_{p=1}^{N}\left(f_{ph}^{(1)\prime}\right)^2 \\
E_t\left\{\frac{\partial^2 L}{\partial w_{ih}^{(1)}\,\partial b_h^{(1)}}\right\} &= \sum_{p=1}^{N}\left(x_{pi}A\frac{\partial B}{\partial b_h^{(1)}}\right) \\
&= \sum_{p=1}^{N}\left(x_{pi}f_{ph}^{(1)\prime}\left(\sum_{o=1}^{O}\beta_o^2 w_{ho}^{(2)}\frac{\partial y_{po}^{(2)}}{\partial b_h^{(1)}}\right)\right) \\
&= \sum_{p=1}^{N}\left(x_{pi}f_{ph}^{(1)\prime}\left(\sum_{o=1}^{O}\beta_o^2 w_{ho}^{(2)} w_{ho}^{(2)} f_{ph}^{(1)\prime}\right)\right) \\
&= \sum_{p=1}^{N}\left(x_{pi}\left(f_{ph}^{(1)\prime}\right)^2 \sum_{o=1}^{O}\left(\beta_o w_{ho}^{(2)}\right)^2\right) \\
&= \sum_{o=1}^{O}\left(\beta_o w_{ho}^{(2)}\right)^2 \sum_{p=1}^{N}\left(x_{pi}\left(f_{ph}^{(1)\prime}\right)^2\right) \\
E_t\left\{\frac{\partial^2 L}{\partial w_{ih}^{(1)}\,\partial w_{jh}^{(1)}}\right\} &= \sum_{p=1}^{N}\left(x_{pi}A\frac{\partial B}{\partial w_{jh}^{(1)}}\right) \\
&= \sum_{p=1}^{N}\left(x_{pi}f_{ph}^{(1)\prime}\left(\sum_{o=1}^{O}\beta_o^2 w_{ho}^{(2)}\frac{\partial y_{po}^{(2)}}{\partial w_{jh}^{(1)}}\right)\right) \\
&= \sum_{p=1}^{N}\left(x_{pi}f_{ph}^{(1)\prime}\left(\sum_{o=1}^{O}\beta_o^2 w_{ho}^{(2)} w_{ho}^{(2)} x_{pj} f_{ph}^{(1)\prime}\right)\right) \\
&= \sum_{p=1}^{N}\left(x_{pi}x_{pj}\left(f_{ph}^{(1)\prime}\right)^2 \sum_{o=1}^{O}\left(\beta_o w_{ho}^{(2)}\right)^2\right) \\
&= \sum_{o=1}^{O}\left(\beta_o w_{ho}^{(2)}\right)^2 \sum_{p=1}^{N}\left(x_{pi}x_{pj}\left(f_{ph}^{(1)\prime}\right)^2\right)
\end{aligned}
$$

References

[1] A. C. Bickerstaffe and E. Makalic. MML Classification of Music Genres. In *Proc. 16th Australian Joint Conf. on Artificial Intelligence (to appear)*, Perth, Australia, 2003.

[2] A. P. Engelbrecht. A New Pruning Heuristic Based on Variance Analysis of Sensitivity Information. *IEEE Transactions on Neural Networks*, 12(6):1386–1399, 2001.

[3] A. P. Engelbrecht, L. Fletcher and I. Cloete. Variance Analysis of Sensitivity Information for Pruning Feedforward Neural Networks. In *IEEE Intl. Joint Conf. on Neural Networks*, Washington DC, USA, 1999.

[4] C. Andrieu, J. F. G. de Freitas, A. Doucet, and M. Jordan. An Introduction to MCMC for Machine Learning. *Machine Learning*, 50:5–43, 2003.

[5] C. S. Wallace and D. L. Dowe. Minimum Message Length and Kolmogorov complexity. *Computer Journal*, 42:270–283, 1999.

[6] C. S. Wallace and D. M. Boulton. An Information Measure for Classification. *Computer Journal*, 11(2):195–209, 1968.

[7] C. S. Wallace and P. R. Freeman. Estimation and inference by compact encoding (with discussion). *Journal of the Royal Statistical Society, series B*, 49:240–265, 1987.

[8] C.L. Blake and C.J. Merz. UCI Repository of machine learning databases, `http://www.ics.uci.edu/$\sim$mlearn/MLRepository.html`, 1998.

[9] C.S. Wallace and D.M. Boulton. An invariant Bayes method for point estimation. *Classification Society Bulletin*, 3:11–34, 1975.

[10] D. E. Rumelhart, G. E. Hinton and R. J. Williams. Learning Internal Representations by Error Propagation. *Parallel Distributed Processing*, 1:318–362, 1986.

[11] D. E. Rumelhart, G. E. Hinton and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323(9):533–536, 1986.

[12] D. MacKay. A practical Bayesian framework for backpropagation networks. *Neural Computation*, 4:448–472, 1992.

[13] G. Cybenko. Approximations by superpositions of sigmoidal functions. *Math. Control, Signals, Systems*, 2:303–314, 1989.

[14] G. E. Farr and C. S. Wallace. The Complexity of Strict Minimum Message Length Inference. *Computer journal*, 45:285–292, 2002.

[15] G. te Brake and J.N. Kok and P.M.B. Vitányi. Model Selection for neural networks: comparing MDL and NIC. In M. Verleysen, editor, *Proceedings European Symposium on Artificial Neural Networks*, pages 31–36. D facto, 1994.

[16] J. Rissanen. Modeling by Shortest Data Description. *Automatica*, 14:465–471, 1978.

[17] J. W. Comley, L. Allison and L. J. Fitzgibbon. Flexible decision trees in a general data-mining environment. In *Fourth Intl. Conf. on Intelligent Data Engineering and Automated Learning (IDEAL-2003)*, Hong Kong, 2003.

[18] K. Funahashi. On the approximate realization of continuous mappings by neural networks. *Neural Networks*, 2:183–192, 1989.

[19] K. Hornik, M. Stinchcombe and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

[20] M. Qi and P. Zhang. An investigation of model selection criteria for neural network time series forcasting. *European Journal of Operational Research*, 132:666–680, 2001.

[21] P. van der Smagt and G. Hirzinger. Why feed-forward networks are in a bad shape. In *Proceedings International Conference on Artificial Neural Networks*, pages 159–164, Skovde, Sweden, 1998.

[22] Q. Gao, M. Li and P.M.B. Vitanyi. Applying MDL to learning best model granularity. *Artificial Intelligence*, 121(1–2):1–29, 2000.

[23] R. M. Neal. *Bayesian Learning for Neural Networks*, volume 118. Springer-Verlag, Lecture Notes in Statistics, 1996.

[24] R. Reed. Pruning algorithms - a survey. *IEEE Transactions on Neural Networks*, 4(5):740–747, 1993.

[25] T. Y. Kwok and D. Y. Yeung. Constructive algorithms for structure learning in feedforward neural networks for regression problems. *IEEE Transactions on Neural Networks*, 8(3):630–645, 1997.