Model Selection for Neural Networks: Comparing MDL and NIC

Guido te Brake[1] & Joost N. Kok
Department of Computer Science, Utrecht University.
Paul M.B. Vitányi
Centre for Mathematics and Computer Science, Amsterdam.
E-mail `g.m.tebrake@twi.tudelft.nl`.

Abstract

We compare the MDL and NIC methods for determining the correct size of a feedforward neural network. The NIC method has to be adapted for this kind of networks. We include an experiment based on a small standard problem.

1. Introduction

The Minimum Description Length Principle is an elegant method to determine the appropriate complexity (size) of a model. It has been applied to different kinds of models, and recently also to neural networks. The Network Information Criterion for probabilistic neural networks is an instance of Akaike's AIC and can also be used for selection of a model. We discuss both principles, change the NIC, and apply them to a small standard problem: the mapping of a two-joint robot arm.

2. Minimum Description Length

The Minimum Description Length (MDL) principle is introduced by Rissanen in 1978 and states that the best hypothesis is the hypothesis with the shortest description length [6]. The description length of a hypothesis is the sum of the number of bits needed for coding the hypothesis, and the number of bits needed for coding the exceptions. It is possible to derive the MDL principle from Bayes' formula [1].

The MDL principle can be explained in terms of a communication problem [5]. Assume given a set of set of n objects with several attributes, and a predicate on the objects. A friend has the same set of objects, but does not know the predicate. We can sent him n bits, i.e. the truth-value for every object. When the classification does not depend on the attributes then we can send no shorter message but if there is a relation between the values of the predicate and the attributes of the objects then we can send an encoded hypothesis together with a list of exceptions. If the hypothesis too simple, then it has a small encoding but the list of exceptions might be big. If the hypothesis is complex but without exceptions, then it can be the case that the coding length is longer than n. According to MDL, the best hypothesis is the hypothesis that gives the shortest total length. It is assumed, and validated by experience, that this is the one that grasps the important relations in the data, and ignores the noise. It

[1] Current adress: TWI/SSOR, Mekelweg 4, 2628 CD Delft, the Netherlands

follows from this philosophy that the hypothesis selected by the MDL principle can be expected to be a good predictor for the classification of objects that are not elements of the data set. Neural networks can be coded in the following way. Both the topology and the weigths on the links are coded. Assume that the network contains k nodes numbered $\{1, \ldots, k\}$. The code starts with the number k. Next a list of k bias values is encoded using l bits for each bias value. We need $k \times (k-1)$ bits to describe which nodes are connected by directed arcs (possibly in two ways). The weight for each link is given using a precision of l bits. This description takes at most $\log(k) + 2\log\log(k) + k \times l + k(k-1) + m \times l$ bits, where m is the number of directed arcs (links). The encoding of the data for the neural network depends on its form. It is straightforward if the input and output elements are in $\{0, 1\}$. If an example is not correctly classified then it is transmitted using the input together with the correct output. If the input or output elements contain integers or reals then they have to be encoded. For integers, one takes as many bits as MaxInt needs, and for real numbers usually twice as many bits. For the real numbers such an encoding introduces a new problem: when is output correct? We consider it correct if the distance between the output vector and the target vector is under a small fixed value.

3. Network Information Criterion

We base our discussion on [4]. In that paper the neural networks are required to be probabilistic networks. We show that the derivation of the Network Information Criterion (NIC) is also valid for deterministic networks with differentiable transfer functions. This implies for example that the Heaviside function can not be used as a transfer function, but that the standard sigmoidial transfer functions are allowed.

Assume a neural network that outputs $f(x, \theta)$, where x are the inputs and θ is the weight vector $\{\theta_1, \ldots, \theta_k\}$. Define a function d that compares the desired output y with the output of the network with weights θ and input x by taking the square of norm of the difference of the two vectors: $d(x, y, \theta) = \|y - f(x, \theta)\|^2$. By ∇ we denote the vector of partial derivatives to the weights $\theta_1, \ldots, \theta_k$: $\nabla = (\frac{\partial}{\partial \theta_i})_{i=1}^{k}$, and $\nabla\nabla = (\frac{\partial^2}{\partial \theta_i \theta_j})_{i=1, j=1}^{k, \ k}$ By the assumption that the transfer functions are differentiable, these derivatives of the function d exist. Inputs x are taken from their domain with distribution $q(x)$. In the sequel we assume that all examples are taken from the same distribution. We define the expected loss $D(g, f, \theta)$ between the function g and its neural implementation f using weights θ by

$$D(g, f, \theta) = \int d(x, g(x), \theta) q(x) dx$$

(for the probabilistic networks of [4] the expected loss term is more complicated due to the probalistic nature of the network). We do not know the distribution $q(x)$, but we can use the training set to find an estimate $q^*(x)$ of $q(x)$:

$$q^*(x) = \frac{1}{t} \sum_{i=1}^{t} \delta(x - x_i) = \frac{1}{t} \#\{i : x = x_i\}.$$

If the number of elements t in the training set is large enough, then $q^*(x)$ approximates $q(x)$. If we take $q^*(x)$ as distribution we obtain the expected loss D^*:

$$D^*(g,f,\theta) = \int d(x,g(x),\theta)q^*(x)dx = \frac{1}{t}\sum_{i=1}^{t} d(x_i,g(x_i),\theta)$$

Hence $D^*(g,f,\theta)$ is the average error of the examples in the training set. Consider the learning rule

$$\theta := \theta - \epsilon(\nabla d)(x,g(x),\theta) \tag{1}$$

where ϵ is a constant called the learning rate. This is the standard learning rule for feedforward networks performing gradient descent on d. Assume that we choose x according to the distribution $q^*(x)$. We have the following variant of a theorem in [4] in which *trace* denotes the trace of a matrix (i.e. the sum of the diagonal elements):

Theorem 1 *Suppose we want to model function g with a neural network $\lambda x.f(x,\theta)$ using weights θ using learning rule 1. Assume that repeated application of the learning rule using a training set gives a fixed point θ. Let θ_0 be the weight vector that minimizes $\lambda\theta.D(g,f,\theta)$. Then the expected loss D on all inputs relates to the expected loss on the training set D^* as follows:*

$$\langle D(g,f,\theta)\rangle = \langle D^*(g,f,\theta)\rangle + \frac{1}{t}\mathit{trace}(GQ^{-1}) + O(t^{-\frac{3}{2}}).$$

$$G = \mathit{Variance}[(\nabla d)(x,g(x),\theta_0)],\ Q = \mathit{ExpectedValue}[(\nabla\nabla d)(x,g(x),\theta_0)]$$

(x is selected in accordance with probability distribution $q(x)$).

This theorem is the basis of the Network Information Criterion. Let F be a set of models in which we allow a varying number of weights, i.e. F contains functions f with the same codomain, and only the number of weights varies. Fix a training set with t examples. For each $f \in F$, let θ_f denote the weights after training with learning rule $(*)$. We now define the Network Information Criterion as follows: Choose f such that $NIC(f)$ is minimized:

$$NIC(f) = D^*(g,f,\theta_f) + \frac{1}{t}\mathit{trace}(G^*_f Q^{*-1}_f)$$

$$G^*_f = \mathit{Variance}[\nabla d(x,g(x),\theta_f)],\ Q^*_f = \mathit{ExpectedValue}[\nabla\nabla d(x,g(x),\theta_f)]$$

(x is selected in accordance with probability distribution of the *training* set $q^*(x)$). The model that minimizes the NIC is optimal in the average loss sense. We can use this criterion to select a particular model from a set of possible models. Note that $NIC(f)$ only refers to the training set.

4. Comparing MDL and NIC

The major difference between MDL and NIC is that the MDL principle is about code lengths where NIC is about the expected average error for new examples. MDL can be used for model selection, where the assumption is that it selects a model which also optimally classifies new examples. NIC also predicts the performance of the model[2]. Another major difference is that the NIC selects the model with the smallest error measure, depending on the chosen error type. (In our experiment this is the total squared sum.) The MDL principle uses a list of the misclassified examples and hence for real numbers a threshold is necessary. Both principles have a training error part and an added complexity term. The importance of the complexity term decreases when the number of training examples increases. This is the decision that both criteria take: how complex may a model be to be justified for the training set of this size.

A difference between the two criteria is the effective number of parameters in the complexity term of NIC vs. the actual number of parameters in the complexity term of MDL. The effective number of parameters for NIC equals the effective number of parameters of Moody [3, 4]. If the effective number of parameters of NIC is much lower than then the actual number of parameters of MDL then the two methods might give preference to different models.

Murata and Usui have done some small-scale experiments with the NIC criterion [3]. There are some problems with NIC if the weights are small. The MDL method seems to work well for neural networks. In practice, application of the NIC criterion seems to be more difficult and computationally more expensive (allthough at first sight it is mathematically more attractive).

5. Robot arm experiment

In this experiment we model a robot arm with two degrees of freedom (see Figure 1). This standard problem is also discussed in [2]. The problem is to construct a feedforward network that associates the (y_1, y_2) coordinates to the (θ_1, θ_2) coordinates. Let r_1 and r_2 be the lengths of the two arms. Then the relationship is given by

$$y_1 = r_1 \cos(\theta_1) + r_2 \cos(\theta_1 + \theta_2), \quad y_2 = r_1 \sin(\theta_1) + r_2 \sin(\theta_1 + \theta_2).$$

As in [2] we take $r_1 = 2$ and $r_2 = 1.3$. One hundred random examples from a restricted range are constructed to which a little gaussian noise is added to the outputs. The training set consists of random examples taken from two separate areas of the domain. The first angle θ_1 was choosen between 90 and 150 degrees or between 180 and 240 degrees and the second angle between 30 and 150 degrees. We want the network not just to interpolate between the given examples, but also to extrapolate. Therefore we use two test sets: one using the same domain as the training set and a second test set in which θ_1 ranges between 0 and 270 and θ_2 between 0 and 180 degrees.

We want to choose a three layer feedforward network, with in the second layer nodes with sigmoidial transfer functions and in the third layer output nodes with

[2] According to Rissanen, coding is the same as prediction, and in [6] he gives error bounds for prediction using the MDL principle (Predictive MDL).

[3] Murata, personal communication.

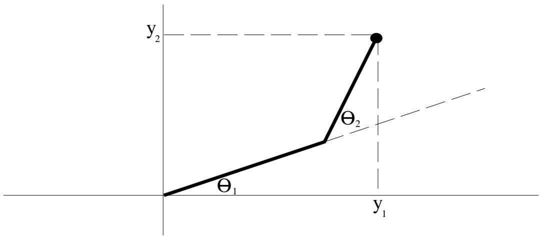

Figure 1: The robot arm

linear transfer functions. The number of nodes in the hidden layer is varied. We want choose the network using only the training set. After that, the models are tested with the two test sets.

During the experiments we noticed that the second test was more interesting in the sense that for the first test set the increase in error for the optimal network and bigger networks was small. For the second test set the slope of the error after the optimal network size was steep.

In figure 2 the error on the training set and prediction error are shown for different networks. Thirty different random training sets are used and the network was trained using the backpropagation algorithm in 10^6 training cycles. We see that the optimal network is a network with 8 hidden nodes.

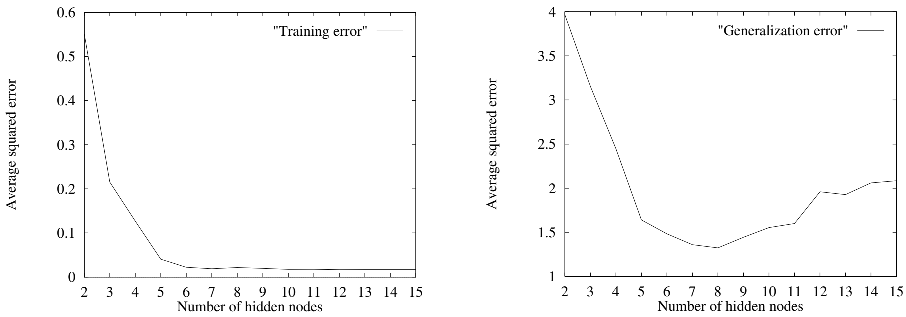

Figure 2: Error on the training set and second test set

In figure 3 we have the results of MDL and NIC. MDL observes that the total code length has a minimum at seven hidden nodes, being only one node away from the optimal network size.

The NIC is only valid if the distribution of the examples in the test set is equal with the distribution of the samples in the training set, which is the case with the first training set. NIC gives a good estimation of the error for this test set. In the right picture of figure 3 we can see this. The lowest two lines are the complexity term and the error on the training set. The NIC measure (the prediction of the generalization error) and the generalization error are plotted on top of eachother, and are seen as one line.

It is interesting to apply the NIC to a problem where the generalization error get worse for larger networks when the examples in the test set are taken from the same distibution as the examples in the training set.

There are some problems in the computation of the complexity part due to numerically unstable computations.

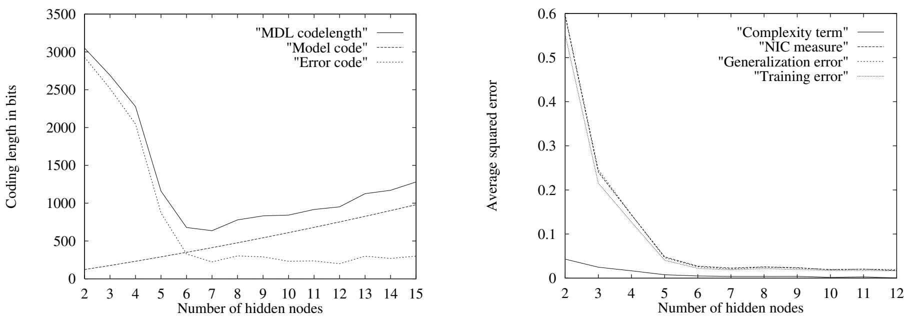

Figure 3: MDL (left) and NIC (right).

6. Conclusions

Application of statistical principles for model selection like MDL and NIC to neural networks works well for the small experiment. For a better comparison it is necessary to apply both principles to more difficult problems.

The MDL method is suited for classification problems in which error coding is straightforward. Applying the MDL method is simple, and it is computationally not expensive compared to the NIC.

The NIC method is suited for problems with a continuous error measure (like the robot arm problem). The NIC only predicts the error when the test examples are taken from the same distribution. When the test domain differs from the training domain, NIC looses its theoretical basis. However, if the distributions are not too different NIC can be succesfully applied. NIC requires a continuous transfer function in the neural network. Computationally NIC is more difficult than MDL principle because inverses of matrices have to be computed.

References

[1] M. Li and P.M.B. Vitányi. *An introduction to Kolmogorov Complexity and its applications*. Springer Verlag, 1993.

[2] David J.C. MacKay. A practical bayesian framework for backpropagation networks. *Neural Computation*, 4(3):448–472, 1992.

[3] J.E. Moody. The effective number of parameters; an analysis of generalization and regularization in nonlinear learning systems. In J.E. Moody, S.J. Hanson, and R.P. Lippmann, editors, *Advances in Neural Information Processing Systems 4*. Morgan Kaufmann Publishers, 1992.

[4] N. Murata, S. Yoshizawa, and S. Amari. A criterion for determining the number of parameters in an artificial neural network model. In T. Kohonen,

K. Mäkisara, O. Simula, and J. Kangas, editors, *Artificial Neural Networks*, volume 1, pages 9–14. ICANN-91, 1991. Espoo, Finland.

[5] J. Quinlan and R. Rivest. Inferring decision trees using the minimum description length principle. *Information and Computing*, 80:227–248, 1989.

[6] J. Rissanen. Stochastic complexity and modeling. *Annals of Statistics*, 14:1080–1100, 1986.