IJCNN'98, Anchorage, Alaska
May 4-5, pp. 2384-2389

USING AN MDL-BASED COST FUNCTION WITH NEURAL NETWORKS

Harri Lappalainen

Helsinki University of Technology
Neural Networks Research Centre
P.O.Box 2200
FIN-02015 HUT, FINLAND
E-mail: Harri.Lappalainen@hut.fi

ABSTRACT

The minimum description length (MDL) principle is an information theoretically based method to learn models from data. This paper presents how to efficiently use an MDL-based cost function with neural networks. As usual, the cost function can be used to adapt the parameters in the network, but it can also include terms to measure the complexity of the structure of the network and can thus be applied to determine the optimal structure. The basic idea is to convert a conventional neural network such that each parameter and each output of the neurons is assigned a mean and a variance. This greatly simplifies the computation of the description length and its gradient with respect to the parameters, which can then be adapted using standard gradient descent.

1. INTRODUCTION

Intuitively, the idea behind MDL is that in order to be able to compress any given data one has to capture its regularities and structure. There are information-theoretical reasons why minimising the description length should yield models with good generalisation properties [5], and it has been shown in practise that description length has strong correlation with the expected classification error of an independent test set [3].

To understand the principle behind MDL, suppose there is a sender who wants to transmit a given data to a receiver. We shall assume that the sender will use a two-part coding scheme: he will first send a model of the data and then the data which is compressed using the model. The more bandwidth is used for the model the more the data is compressed. At some point, however, the cost of using a longer model becomes higher than the gain in compression of the data. Clearly, the length of the model should not exceed the length of the description of the data without using any model. Figure 1 shows schematically the relation between model complexity and the description length of the model and the data. Of course it is not necessary to actually code and transmit the model and the data in order to estimate the optimal model. It is enough to calculate what would be the description length if the data were compressed.

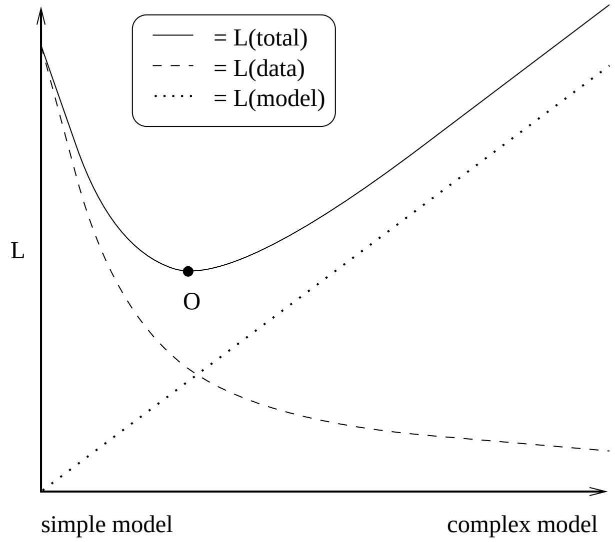

Figure 1: The relation between model complexity and description length is shown schematically. Increasing model complexity allows better coding of data, but with the cost of increasing the length of model description. The best model has the minimum total description length (point O).

The terms under- and over-fitting can be seen from MDL point of view. The optimal model is by our definition the one that has the minimum description length. Under-fitting means that the complexity of a model is below, and over-fitting that the complexity is above that of the optimal model.

It is important to recognise that the optimal model is not unique. Before transmitting the model and the data, the sender and the receiver must have agreed on how the message is coded, or otherwise the receiver would not know how to interpret it. The description length of the information used in the interpretation cannot be measured, because we would need to know how to interpret the coded interpretation, etc. Therefore there must always be some aspect of the model whose description length is neglected, namely the coding scheme. It determines the description length of the data, which means that models can only be compared with respect to a given coding scheme. Using Bayesian terms, the coding scheme implicitly defines a prior over the models. It is, however, often much easier to incorporate prior knowledge in the design of the code than assign a Bayesian prior over the models.

1.1. MDL-based cost functions for supervised and unsupervised learning

A common problem in pattern recognition is that one wants to recognise objects in a given raw data. One might have a large amount of raw data available, but only a small amount of labelled data. The typical approach is to extract features from the raw data in an unsupervised manner and then to find a mapping from the features to the corresponding labels (desired outputs) in a supervised manner. The whole process can be described by $I_U \rightarrow (F_U = I_S) \rightarrow D_S$, where the arrows stands for mappings, I for input, F for feature, D for desired output, U for unsupervised and S for supervised.

Let us denote the model by M. In supervised learning one wants to find a mapping from the input data I_S to the desired output data D_S. The mapping is described by the model M_S, whose structure and parameters are to be estimated. In a conventional neural network, the cost function C measures the error in the mapping: $C(D_S - M_S(I_S))$. The MDL-based cost function for supervised learning is $L(D_S|M_S, I_S) + L(M_S)$: the description length of the output data given the model and the input data plus the description length of the model. The first term corresponds to $C(D_S - M_S(I_S))$ and the second term penalises for the complexity of the model. The input is assumed to be known by both the sender and the receiver and is not included in the description length.

In unsupervised learning one wants to find features F_U which compactly describe the input data I_U. The quality of the features is evaluated by estimating how much information about inputs they preserve, that is, how well the inputs can be reconstructed from the features. We shall consider a generative model M_U which defines the *reconstruction* mapping: $\hat{I}_U = M_U(F_U)$. The reconstruction error which measures the quality of the features is then $C(I_U - M_U(F_U))$. We shall use vector quantisation to illuminate the use of a generative model.

In vector quantisation, the model consists of model vectors $\boldsymbol{m}_i$, which directly define the reconstruction mapping M_U: given an index i, the reconstruction is $\boldsymbol{m}_i$. The index i thus plays the role of the feature. The quality of the feature is measured with the reconstruction error C, or quantisation error as it is often called in connection with vector quantisation. The feature is defined as the index that minimises the reconstruction error, or to put it in other words, the index of the closest model vector: $F_U = M_U^{-1}(I_U) = \arg\min_F C(I_U - M_U(F))$. The model can be adapted by minimising the reconstruction error, that is, by moving the closest model vector even closer to the input. The features and the model are thus both defined as the minimum of the reconstruction error.

In addition to the term measuring the reconstruction error, the MDL-based cost function for unsupervised learning has terms penalising the complexity of the model and the features: $L(I_U|M_U, F_U) + L(M_U) + L(F_U)$. The cost function is used in exactly the same way as with vector quantisation: the features and the model are found by minimising it.

The features are often considered a subset of the parameters of the model, because they too have to be estimated from the data and are included in the cost function. The only difference between the features and the parameters is that the features change for each input element, whereas the parameters are constant for different data elements.

2. MEASURING THE DESCRIPTION LENGTH

Typically neural networks have relatively simple structure but many parameters, and it is usually easy to design a satisfactory coding for the structure. We shall therefore concentrate on measuring the description length of the parameters of the model.

If the transmitted data consists of discrete elements with known probabilities, then the length needed in transmission can be computed. It is well-known that if the probability of a data element D is $P(D)$, then the transmission of the element D using the optimal code takes $-\log_2 P(D)$ bits [9, 8]. We shall hereafter measure information in base e instead of base 2: $-\log_2 P(D)$ bits equals $-\ln P(D)$ nats, where 1 nat equals $\log_2 e$ bits.

The output data and the parameters in the network have continuous values instead of discrete. Communicating a random continuous value at an infinite precision would require an infinite number of bits, and therefore the values have to be discretised. If the discretisation is done with precision ϵ_x, then the range of values is divided into small bins with size ϵ_x. Suppose we know the probability distribution $p(x)$. If ϵ_x is small enough, $p(x)$ is approximately linear throughout each bin and the probability of each bin is approximately $p(x)\epsilon_x$, which means that the description length L of the value x is

$$L(x) = -\ln p(x)\epsilon_x = -\ln p(x) - \ln \epsilon_x. \qquad (1)$$

Equation 1 shows that in order to compute the description length, *each continuos value must be assigned a probability distribution and an accuracy.*

The model is described by its structure, its parameters $\boldsymbol{\theta} = [\theta_1, \theta_2, \ldots]$ and their accuracies $\epsilon_{\boldsymbol{\theta}}$. We shall consider a coding scheme where the model is used to assign a probability distribution $p(D)$ for each data element $D \in \mathcal{D}$. Some of the parameters may also be hyper parameters which are used to assign probability distribution p_{θ_i} for other parameters. The data is discretised with accuracy $\epsilon_{\mathcal{D}}$, but that can be considered an integral part of the coding scheme. This is because the effect of $\epsilon_{\mathcal{D}}$ does not depend on the model: it always adds a constant $-N \ln \epsilon_{\mathcal{D}}$ term in the description length, and can therefore be omitted (N is the number of data elements). The accuracy of the parameters ($\epsilon_{\boldsymbol{\theta}}$) does depend on the model, however, and must be taken into account.

In the literature, several different ways to measure the description length have been proposed. Our approach dates back to Wallace and Boulton [10] and similar approaches have been used in [4, 11]. Alternative approaches have been presented e.g. in [6, 7].

Suppose the target value for the parameter vector is $\boldsymbol{\theta}$. Let us denote the actual discretised value by $\hat{\boldsymbol{\theta}}$ and the corresponding round-off error by $\tilde{\boldsymbol{\theta}} = \hat{\boldsymbol{\theta}} - \boldsymbol{\theta}$. The errors depend on the locations of the borders between the bins, but there is no reason to favour one discretisation over the others, and therefore we should compute the *expectation* of the description length over the distribution of $\tilde{\boldsymbol{\theta}}$, which is approximately even in the range $[-\epsilon_{\boldsymbol{\theta}}/2, \epsilon_{\boldsymbol{\theta}}/2]$. We are also going to assume that the errors are uncorrelated, that is, $E\{\tilde{\theta}_i \tilde{\theta}_j\} = 0$.

The expected description length can be approximated by utilising the second order Taylor's series expansion of $L(\hat{\boldsymbol{\theta}})$ about $\boldsymbol{\theta}$.

$$L(\hat{\boldsymbol{\theta}}) \approx L(\boldsymbol{\theta}) + \sum_i \frac{\partial L}{\partial \hat{\theta}_i} \tilde{\theta}_i + \frac{1}{2} \sum_i \sum_j \frac{\partial^2 L}{\partial \hat{\theta}_i \partial \hat{\theta}_j} \tilde{\theta}_i \tilde{\theta}_j \qquad (2)$$

Here the derivatives of L are evaluated at $\boldsymbol{\theta}$. Taking expectation over $\tilde{\boldsymbol{\theta}}$ yields the cost function:

$$L_E(\boldsymbol{\theta}, \epsilon_{\boldsymbol{\theta}}) \stackrel{def}{=} E_{\tilde{\boldsymbol{\theta}}}\{L(\hat{\boldsymbol{\theta}})\} \approx L(\boldsymbol{\theta}) + \frac{1}{2} \sum_i \frac{\partial^2 L}{\partial \hat{\theta}_i^2} E\{\tilde{\theta}_i^2\}. \qquad (3)$$

Here we have used $E\{\tilde{\theta}_i\} = 0$ for all i and $E\{\tilde{\theta}_i \tilde{\theta}_j\} = 0$ for all $i \neq j$. The optimal parameters $\boldsymbol{\theta}$ and their accuracies $\epsilon_{\boldsymbol{\theta}}$ can be solved by minimising the expected description length L_E.

Since the cost function includes second order derivatives of L with respect to the parameters, it might be troublesome to use it with a large neural network. First the outputs have to be computed in a feedforward phase as usual. Approximations of the second order derivatives needed in the cost function can be computed in a process analogous to backpropagation. It involves two feedback phases: the first one is used to compute the first order derivatives and the second one to compute the second order derivatives. It is then laborious to use backpropagation to compute the gradient, because it has to be applied to all three phases, resulting in a total of six phases.

Fortunately we can overcome these difficulties by recognising that it is not the second order derivatives we wish to evaluate but the expectation over $\tilde{\boldsymbol{\theta}}$.

3. COMPUTING THE EXPECTATION OF THE DESCRIPTION LENGTH FOR NEURAL NETWORKS

In this section we shall describe a method for computing the expectation of the description length for neural networks in an efficient feedforward process. We shall first write down the feedforward equations for the outputs of the neurons and the description length, and then take expectations of the equations. This results in converted feedforward equations, which automatically yield the expected description length. It is then easy to use backpropagation to adapt the parameters. For the sake of simplicity, we shall only consider supervised learning with multilayer perceptrons (MLP), although the method is general and can be applied to practically any type of neural network, both supervised and unsupervised learning.

3.1. Feedforward equations

We shall consider a standard MLP-network with input, hidden and output layers. In order to be able to write the feedforward equations in a compact form, we shall assign all parameters and outputs of the neurons a unique index. In our notation, ξ_i can mean either the value of a parameter or output of a neuron, that is,

ξ_i are used to denote any value than can be an input for neurons. The set of indices for the parameters is denoted by $\mathcal{P}$ and the transfer functions of the neurons are denoted by f_i. The values ξ_i are defined by equation 4.

$$\xi_i(\hat{\boldsymbol{\theta}}, t) \stackrel{def}{=} \begin{cases} \hat{\theta}_i = \theta_i + \tilde{\theta}_i & \text{parameters } (i \in \mathcal{P}) \\ D_i(t) & \text{desired outputs} \\ I_i(t) & \text{input neurons} \\ f_i(\xi_j | j \in \mathcal{J}_i) & \text{other neurons} \end{cases} \tag{4}$$

Input and output data is parametrised by time t. We have used a shorthand notation for the parameters of a function. For example $f(\xi_j | j \in \{2, 4, 5\}) = f(\xi_2, \xi_4, \xi_5)$. The set $\mathcal{J}_i$ thus contains the indices of the inputs to the neuron i or, equivalently, indices of the parameters of the function f_i. The network is assumed to be strictly feedforward, which means that $j \in \mathcal{J}_i$ implies $j < i$.

For hidden and output neurons the transfer functions f_i are like in any conventional neural network. They can be sums of inputs multiplied by weights, sigmoids, radial basis functions, etc.

The cost function for supervised learning is $L(M_S) + L(D_S|M_S)$ as explained in section 1.1. The description lengths are computed according to equation 1, with the exception that the terms $-\ln \epsilon$ are omitted from $L(D_S|M_S)$. For each D_i we shall assign a function f_j, which is used to compute the terms $-\ln p(D_i)$. The set of indices for these functions is denoted by $\mathcal{L}_\mathcal{D}$. Similarly, the set $\mathcal{L}_\mathcal{P}$ comprises of the indices of functions f_j, which evaluate the terms $-\ln p(\theta_i)$. We can now write down the cost function in terms of ξ_i and ϵ_{θ_i}.

$$L(\hat{\boldsymbol{\theta}}, \boldsymbol{\epsilon_\theta}) = \sum_{i \in \mathcal{L}_\mathcal{P}} \xi_i - \sum_{i \in \mathcal{P}} \ln \epsilon_{\theta_i} + \sum_{t=1}^{N} \sum_{i \in \mathcal{L}_\mathcal{D}} \xi_i(t) \tag{5}$$

The first two terms correspond to $L(M_S)$ and the third term to $L(D_S|M_S)$.

If a parameter θ_i does not have an associated neuron in the set $\mathcal{L}_\mathcal{P}$, it means that we tacitly assume the probability distribution $p(\theta_i)$ to be constant throughout the range of values of θ_i, that is, we assume the value of the parameter to be evenly distributed. It has to be reminded that although the constant term $-\ln p(\theta_i)$ can be omitted when adapting the parameters and their accuracies, it should still be taken into account when models with different parametrisations are compared.

The structure of the network is shown in figure 2. Desired outputs are marked by D, input neurons by I, and other neurons by f. The parameters of the network are not shown. The functions f above the dotted line are the ones used to compute the description length of the parameters and the data.

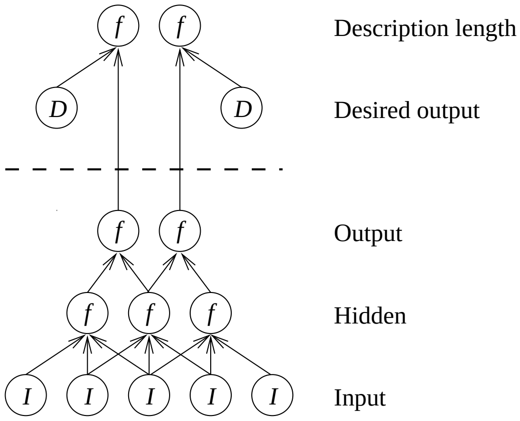

Figure 2: Structure of a MLP network with an MDL-based cost function is shown schematically. The layers below the dotted line are the same as in a conventional MLP: input, hidden and output layers. The functions above the dotted line are used to compute the cost function L.

3.2. Expectation of the description length

Let us define $\mu_i \stackrel{def}{=} E_{\tilde{\boldsymbol{\theta}}}\{\xi_i\}$. Taking expectation over $\tilde{\boldsymbol{\theta}}$ from equation 4 yields

$$\mu_i(\boldsymbol{\theta}, \boldsymbol{\epsilon_\theta}, t) = \begin{cases} \theta_i \\ D_i(t) \\ I_i(t) \\ E\{f_i(\xi_j | j \in \mathcal{J}_i)\} \end{cases} \tag{6}$$

and expectation from equation 5 yields

$$L_E(\boldsymbol{\theta}, \boldsymbol{\epsilon_\theta}) \stackrel{def}{=} E\{L(\hat{\boldsymbol{\theta}}, \boldsymbol{\epsilon_\theta})\} = \sum_{i \in \mathcal{L}_\mathcal{P}} \mu_i - \sum_{i \in \mathcal{P}} \ln \epsilon_{\theta_i} + \sum_{t=1}^{N} \sum_{i \in \mathcal{L}_\mathcal{D}} \mu_i(t). \tag{7}$$

We have dropped the subscripts from the expectation operators, but they are always taken over $\tilde{\boldsymbol{\theta}}$.

3.3. Expectation of a function

Equations 6 and 7 show that we could compute the expected description length if we knew how to evaluate the expectations of the functions f_i.

Let $f_i(\xi_j | j \in \mathcal{J}_i)$ be a function whose expectation value we would like to compute. As before, we shall approximate f_i with its second order Taylor's series expansion. However, we are not going to expand f_i with respect to the discretised parameters $\hat{\boldsymbol{\theta}}$ of the network but the direct parameters ξ_j of the function f_i. The expansion is thus computed about the expectation values

$\mu_j = E\{\xi_j\}$.

$$\xi_i = f_i(\xi_j | j \in \mathcal{J}_i) \approx f_i(\mu_j | j \in \mathcal{J}_i) + \sum_{j \in \mathcal{J}_i} \frac{\partial f_i}{\partial \xi_j}(\xi_j - \mu_j) + \frac{1}{2} \sum_{j \in \mathcal{J}_i} \sum_{k \in \mathcal{J}_i} \frac{\partial^2 f_i}{\partial \xi_j \partial \xi_k}(\xi_j - \mu_j)(\xi_k - \mu_k) \quad (8)$$

Taking expectation from both sides of equation 8 yields

$$\mu_i = E\{f_i(\xi_j | j \in \mathcal{J}_i)\} \approx f_i(\mu_j | j \in \mathcal{J}_i) + \sum_{j \in \mathcal{J}_i} \frac{v_j}{2} \frac{\partial^2 f_i}{\partial \xi_j^2}. \quad (9)$$

Here we have denoted the variance of ξ_j by v_j: $v_j \stackrel{def}{=} E\{(\xi_j - \mu_j)^2\}$. The first order terms disappear because $E\{\xi_j - \mu_j\} = 0$. We have also assumed that for all $j \neq k$, either ξ_j and ξ_k are uncorrelated or $\frac{\partial^2 f_i}{\partial \xi_j \partial \xi_k} = 0$, which removes the second order cross terms.

3.4. Variance of a function

We still need the variances v_j. Since each error $\tilde{\theta}_i$ is evenly distributed in the range $[-\epsilon_{\theta_i}/2, \epsilon_{\theta_i}/2]$, the variances of the parameters are given by $v_i = \epsilon_{\theta_i}^2/12$. The variance of the inputs and desired outputs can be assumed to be zero, or they can be assigned some values if there is prior knowledge about, for example, the equipment used to measure the values.

In order to compute the variance of the function f_i, we shall approximate it with first order Taylor's series expansion.

$$\xi_i = f_i(\xi_j | j \in \mathcal{J}_i) \approx f_i(\mu_j | j \in \mathcal{J}_i) + \sum_{j \in \mathcal{J}_i} \frac{\partial f_i}{\partial \xi_j}(\xi_j - \mu_j) \quad (10)$$

According to this approximation, $\mu_i = E\{\xi_i\} \approx f_i(\mu_j | j \in \mathcal{J}_i)$, which yields

$$v_i = E\{(\xi_i - \mu_i)^2\} \approx \sum_{j \in \mathcal{J}_i} \sum_{k \in \mathcal{J}_i} \frac{\partial f_i}{\partial \xi_j} \frac{\partial f_i}{\partial \xi_k} E\{(\xi_j - \mu_j)(\xi_k - \mu_k)\} \quad (11)$$

Again we can drop out the cross terms if all ξ_j are mutually uncorrelated. This yields our final approximation for the variance of a function:

$$v_i \approx \sum_{j \in \mathcal{J}_i} \left(\frac{\partial f_i}{\partial \xi_j} \right)^2 v_j \quad (12)$$

Notice that when computing the variance, one cannot mix the first and second order approximation for μ_i, since that might result in negative values for v_i.

3.5. Converted feedforward equations

We shall now collect together the results derived in this section.

$$\mu_i(\boldsymbol{\theta}, \boldsymbol{\sigma^2}, t) = \begin{cases} \theta_i \\ D_i(t) \\ I_i(t) \\ f_i(\mu_j | j \in \mathcal{J}_i) + \sum_{j \in \mathcal{J}_i} \frac{v_j}{2} \frac{\partial^2 f_i}{\partial \xi_j^2} \end{cases} \quad (13)$$

$$v_i(\boldsymbol{\theta}, \boldsymbol{\sigma^2}, t) = \begin{cases} \sigma_i^2 = \epsilon_{\theta_i}^2/12 \\ 0 \text{ (or the variance of } D_i(t)) \\ 0 \text{ (or the variance of } I_i(t)) \\ \sum_{j \in \mathcal{J}_i} \left(\frac{\partial f_i}{\partial \xi_j} \right)^2 v_j \end{cases} \quad (14)$$

Substituting $\epsilon_{\theta_i} = (12 v_i)^{1/2}$ into equation 7 yields equation 15 for the expected description length L_E.

$$L_E(\boldsymbol{\theta}, \boldsymbol{\sigma^2}) = \sum_{i \in \mathcal{L}_\mathcal{P}} \mu_i - \sum_{i \in \mathcal{P}} \frac{1}{2} \ln 12 v_i + \sum_{t=1}^{N} \sum_{i \in \mathcal{L}_\mathcal{D}} \mu_i(t) \quad (15)$$

Equations 13 and 14 describe how to convert a standard feedforward network into a network where the output of each neuron is assigned a mean and a variance. Equation 15 then defines the MDL-based cost function for such a network. The gradient of the cost function can be computed using standard backpropagation algorithm.

4. DISCUSSION

4.1. Effects of approximations

In the approximation of the means and variances, the inputs of each function are assumed to be uncorrelated. If there exist i and j such that the value of θ_i is propagated in more than one separate route to the parameters of the function f_j, then the parameters are correlated and the computation of mean and variance may be inaccurate. This should not be a very severe restriction; in an MLP with one hidden layer, for example, no parameter (weight) affects any output neuron in more than one separate route. Should the approximation turn out to be too inaccurate, some of the cross terms may be taken into account.

We have assumed that the errors $\tilde{\boldsymbol{\theta}}$ made in the discretisation of the parameters are mutually uncorrelated. It can be argued that the estimate of the description length would be more accurate if we took into account the dependence between parameters: a change in the value of one parameter might be partially compensated by a suitable change in the values of the others. Assuming uncorrelated errors effectively penalises parametrisations with strong dependencies between parameters, since the description length could be made shorter using a parametrisation which removes the dependencies. Usually it is desirable to favour parametrisations with small dependencies, and thus the assumption of uncorrelated discretisation errors is reasonable.

In order to have a decodable message, the accuracies of the parameter values should be coded and sent before sending the truncated parameters. Wallace and Freeman [11] argue that the values and the accuracies of the parameters are not independent, and one can construct a decodable message with almost the same code length we have used.

4.2. Relation to previous work

If we neglect the effect of finite accuracy to the description length and assume an infinite precision, then the parameter values that minimise L are the ones that maximise the posterior probability of the parameters, that is, the maximum a posteriori estimate. They are not necessarily the same that minimise L when finite accuracy is taken into account, however. Taking the expectation over $\tilde{\boldsymbol{\theta}}$ effectively measures and penalises the sensitivity of L to the values of the parameters, thus finding a flatter minimum of L, which corresponds to more probability mass of the posterior density.

Using such MDL-based arguments, Hochreiter and Schmidhuber have arrived in a very similar algorithm, which they call the flat minimum search (FMS) [2]. Due to the similarity of the penalty for the complexity of the model, the promising results they have achieved should be reproductable with our method. Indeed, in preliminary simulations, we have been able to reproduce the experiment 1 in [2]: noisy classification.

Although the cost function in FMS is very similar to ours, it does not define a description length, and it is thus difficult to include measures of the complexity of the structure of the network. The computation of our cost function appears to be more simple, but, on the other hand, FMS does not assume independence of the parameters of the functions in the network. In FMS the user has to give an extra parameter, a tolerable error, which regulates the trade-off between the description length of the parameters and the data. In our approach, the optimal accuracies for the parameters are estimated directly from the data.

MDL has been used with neural networks to find representations in an unsupervised manner in [1, 12, 13]. The main focus therein has been on discrete valued features, such as the indices of vectors in vector quantisation. This paper concentrates on the coding of real valued parameters and features, and thus complements the previous work.

5. REFERENCES

[1] Geoffrey E. Hinton and Richard S. Zemel. Autoencoders, minimum description length and Helmholz free energy. In Jack D. Cowan, Gerald Tesauro, and Joshua Alspector, editors, *NIPS 6*, pages 3–10, San Francisco, 1994. Morgan Kaufmann.

[2] Sepp Hochreiter and Jürgen Schmidhuber. Flat minima. *Neural Computation*, 9(1):1–42, January 1997.

[3] Bernhard Pfahringer. Compression-based feature subset selection. In P. Turney, editor, *IJCAI-95 Workshop on Data Engineering for Inductive Learning*. IJCAI-95 Workshop Program Working Notes, Montreal, Canada, 1995.

[4] Jorma Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.

[5] Jorma Rissanen. A universal prior for integers and estimation by minimum description length. *Annals of Statistics*, 11(2):416–431, 1983.

[6] Jorma Rissanen. Stochastic complexity. *Journal of the Royal Statistical Society (Series B)*, 49(3):223–239 and 252–265, 1987.

[7] Jorma Rissanen. Fisher information and stochastic complexity. *IEEE Transactions on Information Theory*, 42(1):40–47, January 1996.

[8] Jorma Rissanen and G. G. Langdon, Jr. Universal modeling and coding. *IEEE Transactions on Information Theory*, 27:12–23, 1981.

[9] Claude E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, July 1948.

[10] C. S. Wallace and D. M. Boulton. An information measure for classification. *Computer Journal*, 11(2):185–194, 1968.

[11] C. S. Wallace and P. R. Freeman. Estimation and inference by compact coding. *Journal of the Royal Statistical Society (Series B)*, 49(3):240–265, 1987.

[12] Richard S. Zemel. *A minimum description length framework for unsupervised learning*. PhD thesis, University of Toronto, Canada, 1993.

[13] Richard S. Zemel and Geoffrey E. Hinton. Developing population codes by minimizing description length. In Jack D. Cowan, Gerald Tesauro, and Joshua Alspector, editors, *NIPS 6*, pages 11–18, San Francisco, 1994. Morgan Kaufmann.