Overtraining, Regularization, and Searching for Minimum in Neural Networks

J. Sjöberg, L. Ljung
Department of Electrical Engineering
Linköping University
S-581 83 Linköping, Sweden
Phone: +46 13 281890
E-mail: sjoberg@isy.liu.se

December 2, 1991

Abstract

Neural network models for dynamical systems have been subject of considerable interest lately. They are often characterized by the fact that they use a fairly large amount of parameters. Here we address the problem why this can be done without the usual penalty in terms of a large variance error. We show that *regularization* is a key explanation, and that terminating a gradient search ("backpropagation") before the true criterion minimum is found is a way of achieving regularization. This, among other things, also explains the concept of "overtraining" in neural nets.

1 Introduction

Adaptive Neural Networks (NN) have recently been the subject to substantial interest, see, e.g., [7] or [1]. Of particular interest for modelling and control is their capability to describe the behaviour of, possibly nonlinear, dynamical systems. From a modelling point of view an NN is just one particular parametrization of a nonlinear regression model for the system, like

$$y(t) = g(\varphi(t), \theta) \tag{1.1}$$

see, e.g., [6]. Here $y(t)$ would be the output of the system and $\varphi(t)$ would contain past inputs and outputs. The parameter vector θ contains the weighting coefficients of the network and the mapping $g(\cdot,\cdot)$ is defined by the actual net. In the next section we will give some details of this.

A characteristic feature of NN models is that the dimension of θ is quite high, often several hundred. From an estimation point of view this should raise some worries, since it is bound to give a large "variance error" (i.e., that modelling error that originates from the noise disturbances in the estimation ("training") data set and causes misfits when the model is applied to a validation ("generalization") data set). Nevertheless NNs have shown good abilities for modelling dynamical systems.

In this paper we explain why this is so. The key is the concept of *regularization*. In Section 3 we show how regularization reduces the variance error and in Section 4 we show how typically used estimation ("training") procedures, such as backpropagation, implicitly employs regularization. We also explain the phenomenon of "overtraining" in this way. Section 5 contains an example that illustrates the analysis.

2 Neural Networks as models of dynamical systems

Let us consider a two-layered feedforward neural network with one hidden layer between the input layer and the output layer, see Figure 1. For convenience, let us consider the case when we only have one output unit. The case of several output units is a straight-forward generalization.

This network is a nonlinear function taking an argument for each input unit, and give the value of the output unit as result. The exact behaviour of the network is determined by the values of the *weights* between the input units units and the hidden units, and between the hidden units and the output unit. If we put this mathematically; with I input units the network g is a function $g : \Re^I \rightarrow \Re$. If we represent the values of the input units as a vector, and similarly, if we represent all the weights between the input units and one of the hidden units as a vector, then the input signal to this hidden unit is simply the inner product between the input vector and the weight vector. The output signal from a hidden unit is just the input signal passed through a sigmoidal nonlinearity. Finally the value of the output unit is determined as the inner product of the vector containing the values of the hidden units and the vector of the weights between the hidden layer and the output unit. Put into an equation this network can be written

$$g(X) = \sum_{j=1}^{H} W_{2j}\sigma(W_{1j}^T X + \alpha_j) \tag{2.1}$$

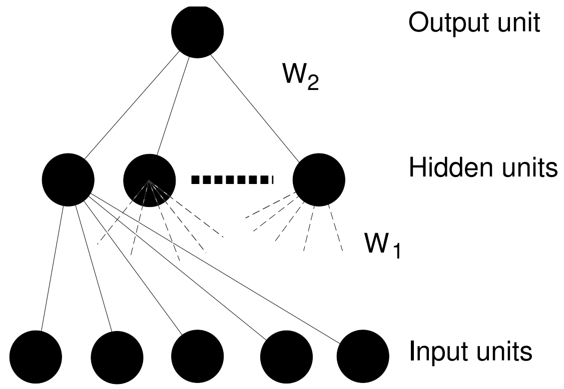

Figure 1: The feedforward neural network considered in this paper.

where H is the number of hidden units, W_{1j} is a column vector containing the weights between the input units and hidden unit j. Similarly W_{2j} is the weight between hidden unit j and the output unit, α_j is a threshold to each hidden unit, and σ is the sigmoidal function chosen as

$$\sigma(x) = \frac{1}{1 + \mathrm{e}^{-x}}. \tag{2.2}$$

The weights, the components of W_1 and W_2, take any real values, however, in most problems they take values not too far from the origin. It is known that this model can approximate any continuous function on a compact set arbitrary well, if there are no limit of the number of hidden units (see [2]).

When the size of the network is decided upon, *i.e.*, the number of input- and hidden units are settled, it remains to find the values of the weights which result in the best approximation of the nonlinear function. This estimation problem is in the NN literature called the *training*, or the *learning*, of the network and to be undertaken one needs data, samples from the function to be approximated. Those data is called the *training set*.

Let us organize the data in the following way; if we have N samples of training data, *i.e.*, N vectors of input values, X_k, sometimes called *input patterns*, and N corresponding output values, Y_k, sometimes called *output patterns* (in the case of regression $X_t = \varphi(t)$, $Y_t = y(t)$, and $t = k$, according to previous section). That is, if f is the nonlinear function which we want to approximate then $Y_k = f(X_k)$ for $k = 1 \ldots N$. Collect all weights and thresholds, which also can be regarded as a weights connected to a unit with the fixed value of 1, into a parameter vector θ. The output of the network will then depend on θ as well as on the input values, current value of X_k. The resulting error obtained by the network, ε can be written

$$\varepsilon(\theta, Z_k) = Y_k - g(\theta, X_k) \tag{2.3}$$

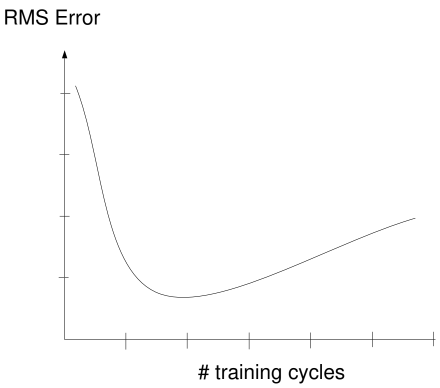

Figure 2: Overtraining: Typical evolution of performance for a NN

where

$$\begin{aligned} Z_k &= (Y_k, X_k), && (2.4) \\ Z^N &= [Z_1, Z_2, ...Z_N]^T. && (2.5) \end{aligned}$$

The most common goal of the training is to obtain the *Least Squares* solution of θ, *i.e.*, θ should be chosen so that the *loss function*,

$$V_N(\theta, Z^N) = \frac{1}{2N} \sum_{k=1}^{N} \varepsilon^2(\theta, Z_k), \tag{2.6}$$

is minimized. In the next section, however, we will see how the loss function can be changed to obtain a better fit, when measured on new data ("a validation data set").

In our research, and to obtained the results presented in the example later in this paper, we used a *Gauss-Newton*-based method (GN), a gradient method of second order, to minimize the loss function. Most commonly used approach is, however, the *backpropagation error algorithm* (BP) which is a gradient method of first order and, hence, converge considerably slower than the GN.

In the BP the weights are changed in the gradient direction of the loss function. A disadvantage is, however, that there are no clues how the step size should be chosen. In the GN method the gradient direction is modified with help of the second derivative of the loss function, the Hessian. In that way one obtain a good value of the step size, and the direction of the the update is also changed in a favorable way. More information about BP can be found in any textbook on neural networks, *e.g.*, [4], and GN is well described in, *e.g.*, [3].

One amazing fact about the neural networks is how good the results can be in spite of a huge number of parameters. When a common model with the same number of parameters one expects a big overfit. In the case with a neural network trained with a gradient method overfitting may happened, however, it does not come instantaneous. Instead the overfitting comes gradually during the training, *i.e.*, if the loss function is computed on independent data after every training cycle it will first decrease and the network learns to approximate the unknown function, and then it will start to grow again. This is the phenomenon which often reefered to as *overtraining*, or *overlearning*, and it typical look like in Figure 2. In the next section we will look closer into overtraining and explain why it comes about, and what can be done to prevent it.

3 Regularization and variance reduction

Consider the following general nonlinear regression problem: We observe $y(t)$ and $\varphi(t)$ for $t = 1, \ldots, N$ and want to estimate the parameter θ in a relationship

$$y(t) = g(\theta, \varphi(t)) \tag{3.1}$$

Assume that the observed data actually can be described by

$$y(t) = g(\theta_0, \varphi(t)) + e(t) \tag{3.2}$$

for a realization of a white noise sequence $\{e(t)\}$. Let

$$\lambda_0 = Ee^2(t) \tag{3.3}$$

We estimate θ by a usual prediction error method:

$$\hat{\theta}_N = \arg\min_{\theta} V_N(\theta) \tag{3.4}$$

where

$$V_N(\theta) = \frac{1}{2N} \sum_{t=1}^{N} (y(t) - g(\theta, \varphi(t))^2 \tag{3.5}$$

This parameter value gives the following model for the relationship between $y(t)$ and $\varphi(t)$:

$$\hat{y}(t) = g(\hat{\theta}_N, \varphi(t)) \tag{3.6}$$

The quality of this model can be evaluated by

$$\bar{V}_N = E\bar{V}(\hat{\theta}_N) \tag{3.7}$$

where

$$\bar{V}(\theta) = E(y(t) - g(\theta, \varphi(t))^2 \tag{3.8}$$

Note that in (3.8) expectation is over $\varphi(t)$ and $e(t)$, while in (3.7) expectation is over the random variable $\hat{\theta}_N$ (which depends on the random variables $\varphi(t)$, $e(t)$ $t \leq N$)

It is a well-known, general result that

$$\bar{V}_N \approx \lambda_0(1 + \frac{1}{2}\frac{d}{N}) \tag{3.9}$$

where

$$d = \dim\theta \tag{3.10}$$

See, e.g., [5], p 418.

This says that when the parametrization is flexible enough to contain a true description of the system (3.2), each estimated parameter gives a contribution to the error (3.9) that does not depend on how important the parameter actually is for the fit. In other words, regardless of how sensitively $V_N(\theta)$ depends on a certain parameter θ_i, its contribution to the variance model error (3.9) still is $\lambda_0/2N$. We can thus have parameters that do no good in improving the fit in (3.5) but just makes the model (3.6) worse when applied to a new data set. This is known as *overfit.*

There could also be parameters that do contribute marginally to the fit, but whose contribution to the error (3.9) is larger. That is to say that if a parameter improves the fit in $\bar{V}_N$, defined in 3.7, by less than $\lambda_0/2N$, it is better to leave it out from the parametrization, because its overall effect on the model will be negative. Let us call such parameters *superfluous.*

In practice, we may have a situation where we suspect that there are too many parameters in a given parametrization, but we cannot *a priori* point to those that are superfluous. This situation apparently is at hand for neural network models. The traditional way to deal with this problem in statistics is to use *regularization.*

In our context this means that we seek to minimize

$$W_N(\theta) = V_N(\theta) + \frac{1}{2}\delta \mid \theta - \theta_0 \mid^2 \tag{3.11}$$

instead of (3.5). (Of course, in practice we do not know θ_0, so that "attraction term" in (3.11) will be around some nominal guess $\theta^{\#}$. We shall discuss that modification later). Let $\hat{\theta}_N^{\delta}$ be the value of θ that minimizes $W_N(\theta)$ so that

$$W'_N(\hat{\theta}_N^{\delta}) = 0 \tag{3.12}$$

This gives, by Taylor's expansion

$$0 = W'_N(\hat{\theta}_N^{\delta}) \approx W'_N(\theta_0) + (\hat{\theta}_N^{\delta} - \theta_0)W''_N(\theta_0) \tag{3.13}$$

so that

$$(\hat{\theta}_N^\delta - \theta_0) \approx -(W_N''(\theta_0))^{-1} W_N'(\theta_0) \tag{3.14}$$

Note that, for large N, due to the law of large numbers

$$W_N''(\theta_0) = V_N''(\theta_0) + \delta I \approx Q + \delta I$$

where

$$Q = E\psi(t)\psi^T(t) \tag{3.15}$$

$$\psi(t) = \frac{d}{d\theta} g(\theta, \varphi(t)) \mid_{\theta=\theta_0} \tag{3.16}$$

Moreover

$$W_N'(\theta_0) = \frac{1}{N} \sum_{t=1}^{N} \psi(t) e(t) \tag{3.17}$$

so

$$E \ N W_N'(\theta_0)(W_N'(\theta_0))^T = \lambda_0 Q \tag{3.18}$$

which shows that

$$P \doteq E(\hat{\theta}_N^\delta - \theta_0)(\hat{\theta}_N^\delta - \theta_0)^T \approx \frac{\lambda_0}{N}(Q + \delta I)^{-1} Q (Q + \delta I)^{-1} \tag{3.19}$$

(These calculations follow exactly the corresponding traditional calculations, e.g., in [5], Chapter 9. All assumptions and formal verifications are entirely analogous).

Now, what is the model quality measure (3.7) in this case?

$$\bar{V}_N^\delta = E\bar{V}(\hat{\theta}_N^\delta)$$

We have, by Taylor's expansion,

$$E\bar{V}(\hat{\theta}_N^\delta) \approx E\left\{\bar{V}(\theta_0) + (\hat{\theta}_N^\delta - \theta_0)\bar{V}'(\theta_0) + \frac{1}{2}(\hat{\theta}_N^\delta - \theta_0)^T \bar{V}''(\theta_0)(\hat{\theta}_N^\delta - \theta_0)\right\} =$$

$$= \lambda_0 + 0 + \frac{1}{2} E \ tr \left\{Q(\hat{\theta}_N^\delta - \theta_0)(\hat{\theta}_N^\delta - \theta_0)\right\} = \lambda_0 + \frac{1}{2} trQP \tag{3.20}$$

Here we used that $\bar{V}'(\theta_0) = 0$ and $\bar{V}''(\theta_0) = Q$. Now plugging in (3.19) gives

$$\bar{V}_N^\delta = \lambda_0 (1 + \frac{1}{2N} trQ(Q + \delta I)^{-1} Q (Q + \delta I)^{-1})$$

Since all the matrices written in *trace* can be diagonalized simultaneously we find that

$$\bar{V}_N^\delta = \lambda_0 (1 + \frac{\tilde{d}}{2N}) \tag{3.21}$$

$$\tilde{d} = \sum_{i=1}^{d} \frac{\sigma_i^2}{(\sigma_i + \delta)^2} \qquad \sigma_i : \text{ eigenvalues of Q} \tag{3.22}$$

First, we notice that for $\delta = 0$ we reobtain (3.9). Second, applications typically show that the spread of the eigenvalues σ_i is quite substantial, so that σ_i is often either significantly larger or significantly smaller than δ. In that case, we can interpret $\tilde{d}$ as follows:

$$\tilde{d} \approx \begin{array}{c}\text{number of eigenvalues of Q} \\ \text{that are larger than } \delta\end{array} \tag{3.23}$$

Here we see the important benefit of *regularization: The superfluous parameters no longer have a bad influence on the model performance.*

We must however also address the problem that in practice we cannot use θ_0 in (3.11) but must use

$$W_N(\theta) = V_N(\theta) + \delta \mid \theta - \theta^{\#} \mid^2$$

for some nominal guess $\theta^{\#}$. Then the estimate $\hat{\theta}_N^{\delta}$ will not converge to the true value θ_0 but to some other value θ^* as $N \to \infty$. This gives an extra contribution to (3.21) which is

$$\bar{V}(\theta^*) - \bar{V}(\theta_0) \approx \frac{1}{2}(\theta^* - \theta_0)^T Q(\theta^* - \theta_0) \tag{3.24}$$

We know that, by definition of θ^* and θ_0

$$\bar{V}'(\theta^*) + \delta(\theta^* - \theta^{\#}) = 0 \tag{3.25}$$

$$\bar{V}'(\theta^*) = \bar{V}'(\theta_0) + Q(\theta^* - \theta_0) = 0 + Q(\theta^* - \theta_0) \tag{3.26}$$

This gives

$$Q(\theta^* - \theta_0) = -\delta(\theta^* - \theta^{\#}) = -\delta(\theta^* - \theta_0 + \theta_0 - \theta^{\#})$$

or

$$(Q + \delta I)(\theta^* - \theta_0) = -\delta(\theta_0 - \theta^{\#}) \tag{3.27}$$

which, inserted into (3.24) gives

$$\bar{V}(\theta^*) - \bar{V}(\theta_0) = \frac{1}{2}\delta^2(\theta_0 - \theta^{\#})^T (Q + \delta I)^{-1} Q (Q + \delta I)^{-1}(\theta_0 - \theta^{\#})$$

Now,

$$\| (Q + \delta I)^{-1} Q (Q + \delta I)^{-1} \| \le \max_i \frac{\sigma_i}{(\sigma_i + \delta)^2} \le \frac{1}{4\delta}$$

so

$$| \bar{V}(\theta*) - \bar{V}(\theta_0) |\leq \frac{\delta}{8}| \theta_0 - \theta^{\#} |^2 \tag{3.28}$$

What all this means is the following: Suppose that we do have some knowledge about the order of magnitude of θ_0 (so that $| \theta_0 - \theta^{\#} |$ is not exceedingly large). Suppose also that there are $d^{\#}(\delta)$ parameters that contribute with eigenvalues of Q that are less than δ. Then introducing regularization - with parameter δ - decreases the model error (3.21) by $(\lambda_0/2)d^{\#}/N$, at the same time as the increase (3.24) due to bias is less than $\delta/8 \cdot | \theta_0 - \theta^{\#} |^2$. Regularization thus leads to a better mean square error of the model if

$$\frac{\lambda_0}{2}\frac{d^{\#}(\delta)}{N} > \frac{\delta}{8} | \theta_0 - \theta^{\#} |^2 \tag{3.29}$$

This expression gives a precise indication of when regularization is good and what levels of δ are reasonable. Note that

$$Q = E\psi(t)\psi^T(t)$$

is a matrix of which we have a good estimate from data. This means that $d^{\#}(\delta)$ is known to us with good approximation. Likewise, we will have a good estimate of λ_0. N is a fixed, known number, and really the only unknown quantity in (3.29) is θ_0. With some measure of the size of $| \theta_0 - \theta^{\#} |$ we should thus choose the regularization parameter δ such that

$$\delta^* = \max\{\delta \mid (3.29) \text{ holds}\} \tag{3.30}$$

Let us conclude this section by deriving how much the estimate $\hat{\theta}_N^\delta$ differs from $\hat{\theta}_N$. We will find use of it in the next section.

We have

$$0 = W_N'(\hat{\theta}_N^\delta) = V_N'(\hat{\theta}_N^\delta) + \delta(\hat{\theta}_N^\delta - \theta^{\#})$$

Also

$$V_N'(\hat{\theta}_N^\delta) = V_N'(\hat{\theta}_N) + V_N''(\hat{\theta}_N)(\hat{\theta}_N^\delta - \hat{\theta}_N) \approx Q(\hat{\theta}_N^\delta - \hat{\theta}_N)$$

We thus have

$$Q(\hat{\theta}_N^\delta - \hat{\theta}_N) = -\delta(\hat{\theta}_N - \hat{\theta}_N + \hat{\theta}_N - \theta^{\#})$$

or

$$(\hat{\theta}_N^\delta - \hat{\theta}_N) = \delta(Q + \delta I)^{-1}(\theta^{\#} - \hat{\theta}_N) \tag{3.31}$$

This can also be written as

$$\hat{\theta}_N^\delta = (I - M_\delta)\hat{\theta}_N + M_\delta\theta^{\#} \tag{3.32}$$

$$M_\delta = \delta(\delta I + Q)^{-1} \tag{3.33}$$

so the regularized estimate is a weighted mean between the unregularized one and the nominal value $\theta^{\#}$.

4 Terminating iterative search for the minimum is regularization

The unregularized estimate $\hat{\boldsymbol{\theta}}_N$, defined by (3.4) is typically computed by iterative search of the following kind:

$$\hat{\boldsymbol{\theta}}_N^{(i+1)} = \hat{\boldsymbol{\theta}}_N^{(i)} - \mu H V_N'(\hat{\boldsymbol{\theta}}_N^{(i)}) \tag{4.1}$$

$$\hat{\boldsymbol{\theta}}_N^{(0)} = \boldsymbol{\theta}^\#$$

Here superscript (i) indicates the i:th iterate. The step size μ is determined by some search along the indicated line. H, is a matrix that may modify the search direction from the negative gradient one, V_N', to some other one. The initial value $\boldsymbol{\theta}^\#$ is natural to use, viewing $\boldsymbol{\theta}^\#$ as some prior guess of $\boldsymbol{\theta}_0$.

Now, note that for any reasonable choices of μ and H (neglecting local minima) we have

$$\lim_{i\to\infty} \hat{\boldsymbol{\theta}}_N^{(i)} = \hat{\boldsymbol{\theta}}_N \tag{4.2}$$

The question we are going to address here is what estimate we have after a finite number of iterations.

Now, by Taylor's expansion

$$V_N'(\hat{\boldsymbol{\theta}}_N^{(i)}) \approx Q(\hat{\boldsymbol{\theta}}_N^{(i)} - \hat{\boldsymbol{\theta}}_N) \tag{4.3}$$

where Q is defined by (3.15). Introduce

$$\tilde{\boldsymbol{\theta}}_N^{(i)} = \hat{\boldsymbol{\theta}}_N^{(i)} - \hat{\boldsymbol{\theta}}_N \tag{4.4}$$

Then (4.1) gives

$$\tilde{\boldsymbol{\theta}}_N^{(k)} \approx (I - \mu H Q)^k (\boldsymbol{\theta}^\# - \hat{\boldsymbol{\theta}}_N) \tag{4.5}$$

This can be written as

$$\hat{\boldsymbol{\theta}}_N^{(k)} = (I - M_k)\hat{\boldsymbol{\theta}}_N + M_k \boldsymbol{\theta}^\# \tag{4.6}$$

$$M_k = (I - \mu H Q)^k \tag{4.7}$$

Compare with (3.31), (3.32)! The point now is that - except in the true Newton case $H = Q^{-1}$ - the matrices M_k and M_δ behave similarly. For simplicity, take $H = I$ (making (4.1) a gradient search scheme), and assume that Q is diagonal with decreasing values along the diagonal (this can always be achieved by a change of variables). Then the j:th diagonal element of M_δ will be

$$M_\delta^{(jj)} = \frac{\delta}{\sigma_j + \delta} \tag{4.8}$$

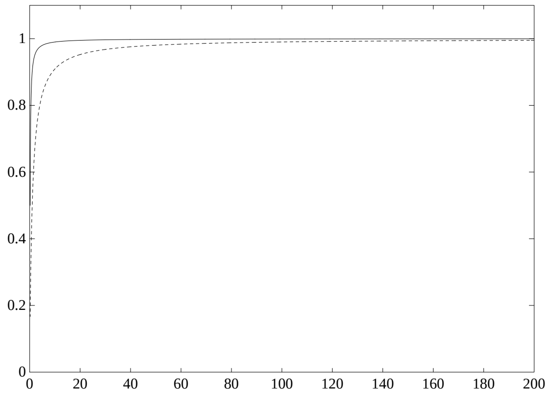

Figure 3: Solid line: M_k^{jj}; dashed line: M_δ^{jj}. $\delta = 0.1$, and $\sigma = 1$. X-axis: $1/\sigma_j$

while the corresponding element of M_k is

$$M_k^{(jj)} = (1 - \mu\sigma_j) \tag{4.9}$$

By necessity, we must choose $\mu < 2/\sigma_1$ to keep the scheme (4.1) convergent. In Fig. 3 we plot these two expressions as functions of $1/\sigma_j$. We see that the effect is pretty much the same. The amount of regularization that is inflicted depends as the size of δ (large δ corresponds to small k).

In summary we have established here that "unfinished" search for the minimization of a criteria function has the same effect as regularization towards the initial value $\boldsymbol{\theta}^{\#}$. Only schemes that allow exact convergence in a finite number of steps do not show this feature.

5 Modeling a hydraulic robot arm

Let us now illustrate what has been said with an example.

The goal is to model the dynamics of a hydraulically controlled robot arm. The position of the robot arm is controlled by the oil pressure in the cylinder which can be controlled by the size of the valve through which the oil streams. The valve size, $u(t)$, and the oil pressure, $y(t)$ are input and output signals, respectively. They are shown in Figures 4 and 5.

As seen in Figure 5 the oil pressure have a very oscillating settling period after a step change of the valve size. These oscillations are caused be the robot arm which oscillate itself after a change in position.

If the oil pressure had been linear dependent on the valve position then, of course, there would be no reason to use a neural network to model this behaviour, a linear black-box model would have been sufficient. The oscillations

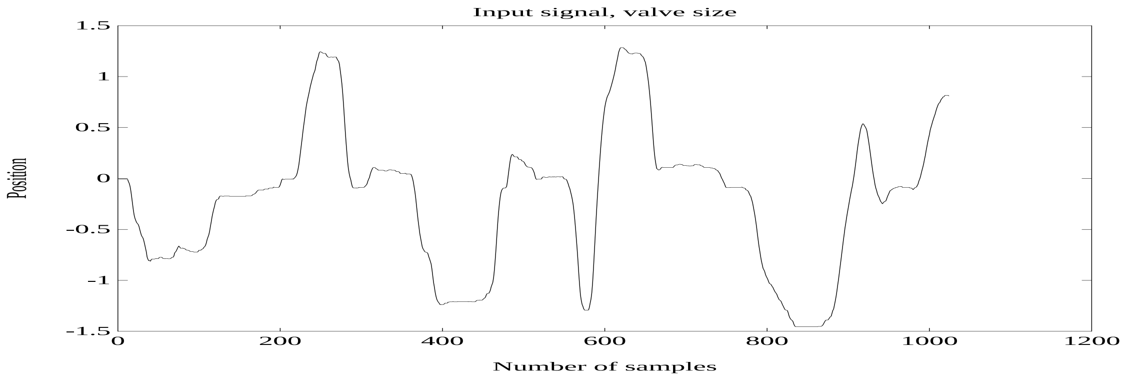

Figure 4: Measured data of valve position.

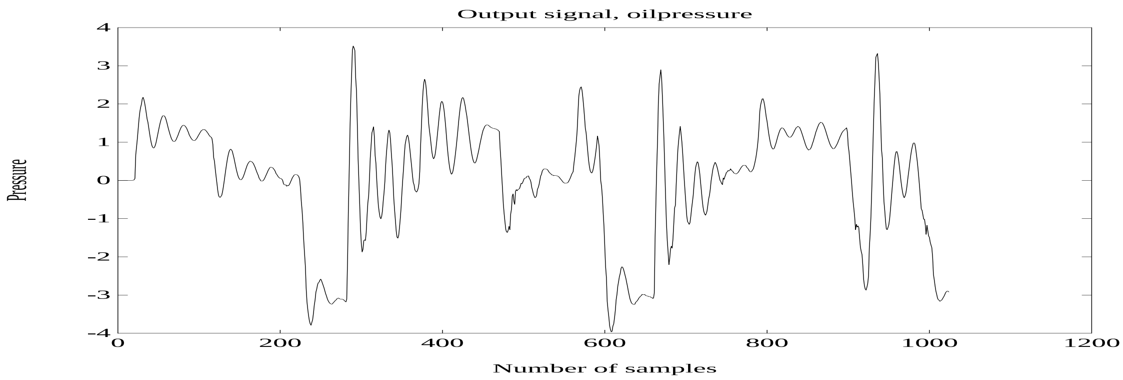

Figure 5: Measured data of oil pressure.

of the oil pressure, however, decay in a nonlinear way, and therefor we tried a network model.

We had 1024 samples available which were divided into two sets, the first 512 samples of the input- and output signal were used for the identification, and the last part of the samples was saved for validation of the model.

The input vector to the network, $\varphi(t)$ consist of values of $y(s)$ and $u(s)$ with $s < t$, and the output is $\hat{y}(t) = g(\varphi(t), \theta)$, the predicted value of $y(t)$. The dimension of $\varphi(t)$ determines the number of input units in the network. After that we haved tried several different sizes of networks, we chose one with five input units and

$$\varphi(t) = [u(t-1)\ u(t-2)\ u(t-3)\ y(t-1)\ y(t-2)]. \tag{5.1}$$

To finally settle the structure of the neural network model the number of hidden units had to be chosen. This number will decide the networks approximation abilities, however, the final model will never contain any information

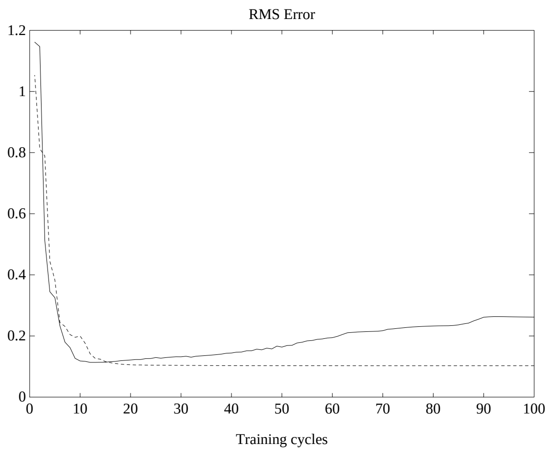

Figure 6: Solid line: no regularization; dashed line: regularized with 10^{-2}.

of the system which is not represented in the data. Therefore there is an upper limit of how many hidden units that can be useful. We chose ten hidden units in our final model, which gave enough flexibilities for a good fit. The model did not improve if the number of hidden units was increased further.

With ten hidden units, five input units, and one output unit we have $10 * (5+1) + (10+1) = 71$ parameters, including all weights and thresholds.

We then trained this network with our 524 training patterns, *i.e.*, θ was estimated using the validation set. In Figure 6 the RMS error computed on the validation set after each iteration of the minimization algorithm is shown for two different regularizations. In the first case the regularization was set to zero, and we can clearly see that overtraining occurs. If, however, the training would have been stopped at ten training cycles, there resulting model would have been good. From ten iterations and onwards the regularizational effect of the algorithm is gradually switched off, as described in the previous section. In the second case (still figure 6) the regularization was chosen to 10^{-2}, and we see that no overlearning comes about. In this case the regularization is not switched off and, hence, the overfit is prevented.

How are the singular values of the Hessian affected by the regularization? In figure 7 the singular values for these two cases are shown. Without regularization the singular values can get arbitrary small, and in this example the smallest is of the order 10^{-7}. The singular values cannot be smaller than the regularization currently used, and we see this when a regularization of 10^{-2} is being used. The former small singular values are set equal the regularization. These singular values corresponds to superfluous parameters virtually switched off by introduction of the regularization.

In respect to these arguments we can regard the singular values larger than

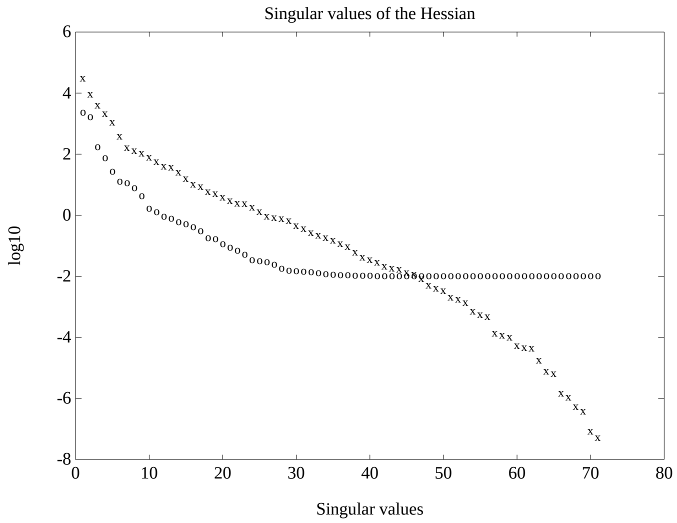

Figure 7: x - no regularization; o - regularized with 10^{-2}.

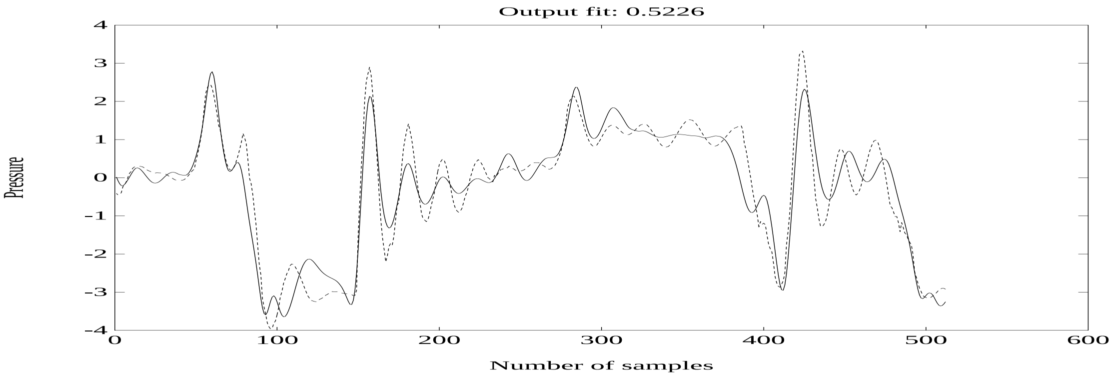

Figure 8: Simulated (solid line) and true output (dashed line).

the regularization as corresponding to they useful parameters. From figure 7 we conclude that one can make use of roughly 30 parameters when identifying this plant with the given data, and with this particular model.

An interesting feature is that if the number of hidden units was increased and the regularization was kept at 10^{-2}, then all additional parameters became superfluous, *i.e.*, the virtual number of parameters stayed at 30. This means that the data does not contain any more information which can be incorporated by an increase of the number of hidden units.

In Figure 8 the NN model is used for simulation on the validation set and compared to the true output of the plant.

An alternative to regularization would be to reduce the number of hidden units and this way get ride of some of the parameters so that no overtraining occurs, *i.e.*, the number of parameters are reduced until no superfluous parameters remains. We tried this in the framework of our NN model, *i.e.*, we applied the same NN model and reduced the number of hidden units until no regularization was necessary to prevent overtraining. We landed on five hidden units, *i.e.*, $5 * (5+1) + (5+1) = 36$ parameters, however, the remaining model did not posses enough flexibility of adaptation, and the overall result was worse then with the larger model in connection with regularization. The conclusion is that by removing superfluous parameters we had also removed some important ones to.

6 Conclusions

We have stressed two aspects of modeling by neural networks in this contribution:

- Regularization is a necessary feature to keep the variance error reasonably small in models employing many parameters.
- performing a limited amount of (gradient) iterations when searching for the criterion minimum is a way to achieve regularization.

One consequence of this is that performing more (gradient) iterations of a non-regularized criterion will make the model worse. This phenomenon is well-known in neural network modeling and is called "overtraining", or "overlearning".

For the minimization we prefer a Gauss-Newton method applied to the regularized criterion. This will assure good numerical properties when computing the search direction (The Hessian of the criterion is well-conditioned) as well as the benefits of regularization.

References

[1] Special issue on neural network in control systems. *IEEE Control Systems Magazine*, 10:303–314, 1989.

[2] G. Cybenko. "Approximation by Superpositions of a Sigmoidal Function". *Mathematics of Control, Signals, and Systems*, 2:303–314, 1989.

[3] J.E. Dennis and R.B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, Englewood Cliffs, New Jersey, 1983.

[4] R. Hecht-Nielsen. *Neurocomputing.* Addison-Wesley, 1990.

[5] L. Ljung. *System Identification: Theory for the User.* Prentice-Hall, Englewood Cliffs, NJ, 1987.

[6] L. Ljung. "Issues in System Identification". *IEEE Control Systems Magazine.*, 11(1):25–29, January 1991.

[7] K.S. Narendra and K. Parthasarathy. "Identification and control of Dynamical systems using neural networks". *IEEE Trans. Neural Networks*, 1:4–27, 1990.