ADAPTIVE REGULARIZATION

L. K. Hansen, C. E. Rasmussen*, C. Svarer, and J. Larsen
CONNECT, Electronics Institute B349
Technical University of Denmark,
DK-2800 Lyngby, Denmark
email: lkhansen,ed,csvarer,jlarsen@ei.dtu.dk

Abstract. Regularization, e.g., in the form of weight decay, is important for training and optimization of neural network architectures. In this work we provide a tool based on asymptotic sampling theory, for iterative estimation of weight decay parameters. The basic idea is to do a gradient descent in the estimated generalization error with respect to the regularization parameters. The scheme is implemented in our ***Designer Net*** **framework for network training and pruning, i.e., is based on the diagonal Hessian approximation. The scheme does not require essential computational overhead in addition to what is needed for training and pruning. The viability of the approach is demonstrated in an experiment concerning prediction of the chaotic Mackey-Glass series. We find that the optimized weight decays are relatively large for densely connected networks in the initial pruning phase, while they decrease as pruning proceeds.**

INTRODUCTION

Learning based on the conventional feed-forward net may be analyzed with statistical methods and the result of such analysis can be applied to model optimization [5, 6, 10, 11, 12]. We have shown how pruning and regularization can be combined to design compact networks for time series prediction [11, 12]. Our "Designer Net" framework is based on the *Optimal Brain Damage* (OBD) method of Le Cun *et al.* [7] and we use simple weight decay for regularization. The benefits from compact architectures are three-fold: Their generalization ability is better, they carry less computational burden, and they are faster to adapt if the environment changes. Further, we have shown how the generalization error of the network may be estimated – without extensive cross-validation – using a modification of Akaike's *Final Prediction Error* (FPE) estimate [1]. The minimal FPE constitutes a useful stopping

*Present address: Dept. of Computer Science, University of Toronto, Canada.

criterion for pruning. However, our previous work has been conditioned on the correct setting of several parameters, most prominently the weight decay parameters. *In this contribution we provide the possibility of adapting regularization parameters within the Designer Net framework.*

The results obtained can be viewed as a *sampling theory* alternative to the Bayesian or *Evidence* based techniques for adaptive regularization developed by MacKay [8, 9]. An analytical comparison of these two techniques has recently been given in [2].

LEARNING

The use of *system identification* tools for neural net learning has been pioneered by Moody (see e.g., [10]) who derived estimators for the average generalization error. The main source of uncertainty in the learning process is the shortage of training data. Other important contributions to uncertainty are: Lack of fit, noise in the training process, and non-stationarity of the data-generating environment. Lack of fit[1] was discussed in, e.g., [5], while noise in the training process has been discussed in [3]. In this presentation we will neglect these three effects. Lack of fit can be minimized by starting the pruning process from large enough networks, while noise in the training process can be relieved by careful search in weight space. Non-stationarity is a hard problem that will be pursued in future work, here we will assume stationarity.

NETWORK ARCHITECTURE AND TRAINING

The basic network is a *tapped delay line architecture* with L input units, n_H hidden sigmoid units and a single linear output unit. The initial network is fully connected between layers and implements a non-linear mapping from lag space $\mathbf{x}(k) = [x(k), ..., x(k-L+1)]$, ($L$ is the length of the tapped delay line), to the real axis:

$$\widehat{y}(k) = F_{\mathbf{u}}(\mathbf{x}(k)) \qquad \widehat{y} \in \mathcal{R}, \tag{1}$$

where $\mathbf{u} = [\mathbf{w}, \mathbf{W}]$ is the N-dimensional weight vector and $\widehat{y}(k)$ is the prediction of the target signal $y(k)$. The particular family of non-linear mappings considered here can be written as:

$$F_{\mathbf{u}}(\mathbf{x}(k)) = \sum_{j=1}^{n_H} W_j \tanh\left(\sum_{i=1}^{L} w_{ij}\, x(k-i-1) + w_{i0}\right) + W_0, \tag{2}$$

where n_H is the number of hidden units, W_j are the hidden-to-output weights, while w_{ij} connect the input and hidden units.

[1] Lack of fit is also sometimes described as "the teacher does not belong to student space" or "incomplete modeling".

A simulator based on *batch mode*, second order local optimization has been developed, as described in [11, 12]. The scheme is based on the *diagonal approximation* of the cost-function Hessian (the second derivative matrix). We use the sum of squared errors to measure the performance of the current network:

$$E_{\text{train}} = \frac{1}{p} \sum_{k=1}^{p} \left[y(k) - F_{\mathbf{u}}(\mathbf{x}(k)) \right]^2 , \tag{3}$$

where p is the number of training examples. To ensure numerical stability and for assisting the pruning procedure we augment the cost-function with a weight decay term:

$$E = E_{\text{train}} + \frac{\alpha_w}{p} \sum_{ij}^{N_w} w_{ij}^2 + \frac{\alpha_W}{p} \sum_{j}^{N_W} W_j^2 , \tag{4}$$

where N_w, N_W are the numbers of weights and thresholds in hidden and output units, respectively. Further, α_w, α_W are the weight decay parameters of the hidden and output layers, respectively. The objective of the training procedure is to optimize the networks ability to predict near future values of a given time series. Hence, the network weights, $\mathbf{u}$, are trained to recognize the short time structure of the training set time series.

PRUNING

The OBD method proposed by Le Cun *et al.* [7] was successfully applied to reduce large networks for recognition of handwritten digits. The basic idea is to estimate the increase in the *training error* when deleting weights. The estimate is formulated in terms of weight *saliencies* s_l:

$$\delta E_{\text{train}} = \sum_{l \in D} s_l \equiv \sum_{l \in D} \left(\frac{2\alpha}{p} + \frac{1}{2} \frac{\partial^2 E_{\text{train}}}{\partial u_l^2} \right) u_l^2 , \tag{5}$$

where u_l is a component of $\mathbf{u}$ and the sum runs over the set D of weights to be deleted. The saliency definition used here takes into account that the weight decay terms force the weights to depart from the minimum of the training set error. As in [7] we approximate the second derivative by the positive semi-definite expression:

$$\frac{\partial^2 E_{\text{train}}}{\partial u_j^2} \approx \frac{2}{p} \sum_{k=1}^{p} \left(\frac{\partial F_{\mathbf{u}}(\mathbf{x}(k))}{\partial u_j} \right)^2 . \tag{6}$$

The major assumptions entering the derivation of OBD are: 1) Terms of third and higher orders in the deleted weights can be neglected. 2) The off-diagonal terms in the Hessian, $\partial^2 E_{\text{train}} / \partial u_l \partial u_{l'}$, can be neglected. Computationally, the second order (diagonal) terms, eq. (6), are reused from the training scheme. We refrain from operations involving the full Hessian, which scales poorly for large networks. The recipe allows for *ranking* the weights according to saliency.

GENERALIZATION

The generalization error is defined as the average squared error on an example from the example distribution function $P(\mathbf{x}, y)$. The examples are assumed to be generated by a *teacher* function of the same form as the model and with a set of *unknown* weights $\mathbf{u}^*$ and degraded by additive noise:

$$y(k) = F_{\mathbf{u}^*}(\mathbf{x}(k)) + \nu(k) \tag{7}$$

where the noise samples $\nu(k)$ are independent identically distributed variables of unknown variance σ^2. Further, we assume that the noise terms are independent of the corresponding inputs. The generalization error of a given network is by definition the average error on a random example. A more interesting quantity is the training set average of the generalization error, viz., the average over an ensemble of networks in which each network is provided with its individual training set. Using the diagonal approximation for the Hessian this error (also referred to as the test error) can be estimated as [6]:

$$\begin{aligned} \widehat{E}_{\text{test}} &= \left(1 + \frac{N_{\text{eff}}}{p}\right)\sigma^2 \\ &\quad +2\sum_{ij}^{N_w} \lambda_{ij} \left(\frac{\alpha_w}{p} \cdot \frac{w^*_{ij}}{\lambda_{ij} + 2\alpha_w/p}\right)^2 \\ &\quad +2\sum_{j}^{N_W} \Lambda_j \left(\frac{\alpha_W}{p} \cdot \frac{W^*_j}{\Lambda_j + 2\alpha_W/p}\right)^2 + R, \end{aligned} \tag{8}$$

with

$$N_{\text{eff}} = \sum_{ij}^{N_w} \left(\frac{\lambda_{ij}}{\lambda_{ij} + 2\alpha_w/p}\right)^2 + \sum_{j}^{N_W} \left(\frac{\Lambda_j}{\Lambda_j + 2\alpha_W/p}\right)^2, \tag{9}$$

where the λ's are the second derivatives already computed in eq. (6): $\lambda_{ij} \equiv \partial^2 E_{\text{train}}/\partial w_{ij}^2$, $\Lambda_j \equiv \partial^2 E_{\text{train}}/\partial W_j^2$. The rest term R contains higher order quantities and terms that do not affect the estimate of the regularization parameters, see [5, 6] for further discussion. The estimate is based on linearization of the networks as regards the fluctuations in the weights resulting from different training sets.

The generalization error estimates were also used for answering the question of how many weights it may be possible to delete in a pruning session in [11, 12]. We applied Akaike's FPE estimate [1] of the test error in terms of the training error which reads:

$$\widehat{E}_{\text{test}} = \frac{p+N}{p-N} E_{\text{train}}, \tag{10}$$

where p is the number of training samples, and N is the number of parameters in the model. The left hand side of eq. (10) is the average generalization error, averaged over all possible training sets of size p.

The relation expresses the fact that the training and test errors are biased estimates of the noise level because each parameter during training has "absorbed" noise from the training samples.

Since we have regularized the training procedure by weight-decay terms α_w, α_W, hence, suppressed the ability of the (otherwise) ill-determined parameters to model noise, we need to modify the standard FPE estimate by replacing the total number of parameters with the *effective* number of parameters, see [10, 11, 12][2]:

$$\widehat{E}_{\text{test}} = \frac{p + N_{\text{eff}}}{p - N_{\text{eff}}} E_{\text{train}}, \tag{11}$$

With the above tool we can obtain a generalization error estimate for each pruned network. By selecting the network with the lowest estimated generalization error we obtain a stopping criterion for pruning.

Note that the estimated average generalization eq. (9) error is a function of the regularization parameters, hence, it is possible to vary these and search for minimal test error. In MacKay's Evidence framework a similar strategy was adopted, however, with the purpose of maximizing the so-called Evidence. We find it more natural to optimize the quantity that is our basic objective, namely the test error. It is at present not clear what the relation between the Evidence and the generalization error is. Empirically, they have been found to be related [2, 8].

We use a simple gradient descent procedure for minimization of the generalization error:

$$\alpha(n+1) = \alpha(n) - \rho \left. \frac{\partial E_{\text{test}}}{\partial \alpha} \right|_{\alpha = \alpha(n)}, \tag{12}$$

where ρ is a gradient descent parameter, and n is the iteration index (one epoch). The Designer Net approach is based on the diagonal approximation to the Hessian. In terms of the diagonal elements the recursion above reads,

$$\alpha_w(n+1) = \alpha_w(n) + \frac{4\rho}{p^2} \sum_{ij}^{N_w} \frac{(\sigma^2 - \alpha_w(n)(w_{ij}^*)^2)\lambda_{ij}^2}{(\lambda_{ij} + 2\alpha_w(n)/p)^3}. \tag{13}$$

A similar expression applies for the hidden-to-output weight decay parameter α_W, in fact an arbitrary set of weight decay parameters can be defined and estimated using this recipe[3]. Expression (13) contains two unknown quantities: the teacher weights w_{ij}^* and the noise variance σ^2. The teacher weights are replaced by the *current estimated weights* of the network (see [2] for a

[2]In fact the notion of an effective number of parameters is quite delicate see [6].

[3]In the derivative of the test error we have kept the dependence $\lambda^2/(\lambda + 2\alpha/p)^3$ (rather than $1/\lambda$) providing a stabilizing effect similar to the Moore-Penrose pseudo inverse discussed in [6].

discussion), while the noise variance is estimated from the training error in the same approximation as in eq. (11):

$$\widehat{\sigma}^2 = \left(1 - \frac{N_{\text{eff}}}{p}\right)^{-1} E_{\text{train}}. \tag{14}$$

EXPERIMENTS

We illustrate the virtues of the adaptive regularization scheme on two time series forecasting problems. The first experiment explores the functional dependence of the derivative of the estimated test error cf. equation (13). The forecasting problem is the sunspot benchmark involving estimation of the yearly sunspot activity from the past twelve years activity (see, e.g., [11] for a detailed description of the benchmark). To simplify we consider a linear model for which the parameters are uniquely determined when using the least squares cost function. The sunspot benchmark involves three data sets: A training set and two test sets. In figure 1 we show the weight decay dependence of the two test errors and of the derivative of the *estimated* test error. Note that both test sets have shallow minima at values just below $\alpha = 0.1$, and that the derivative of the estimated test error passes through zero at a compatible value. Also note that the particular functional form of the derivative implies that the iterative scheme will converge to the zero point of the gradient, hence, provide near-optimal regularization with improved generalization errors. To further illustrate the role of adaptive regularization in the Designer Net framework we present tentative results on a standard problem of nonlinear dynamics, viz. the Mackey-Glass chaotic time series. This forecasting problem was previously studied in [12]. The Mackey-Glass attractor is a non-linear chaotic system described by the following equation:

$$\frac{dz(t)}{dt} = -bz(t) + a\frac{z(t-\tau)}{1 + z(t-\tau)^{10}} \tag{15}$$

where the constants are $a = 0.2$, $b = 0.1$ and $\tau = 17$. The series is resampled with sampling period 1 according to standard practice. We aim at identifying the underlying dynamic model, from this chaotic time series. The network configuration is $L = 16$, $n_H = 10$, with a total of 181 parameters, and we train to implement a six step ahead prediction. That is, $\mathbf{x}(k) = [z(k-6), z(k-12), \cdots, z(k-6L)]$ and $y(k) = z(k)$.

The errors are computed as:

$$E_{\text{set}} = \frac{1}{\sigma^2_{\text{total}} \cdot p_{\text{set}}} \sum_{k=1}^{p_{\text{set}}} \left[y(k) - F_{\mathbf{u}}(\mathbf{x}(k))\right]^2, \tag{16}$$

where p_{set} is the number of examples in the data (train or test) set in question, and σ^2_{total} is the total variance of $y(k)$ on the training and test set.

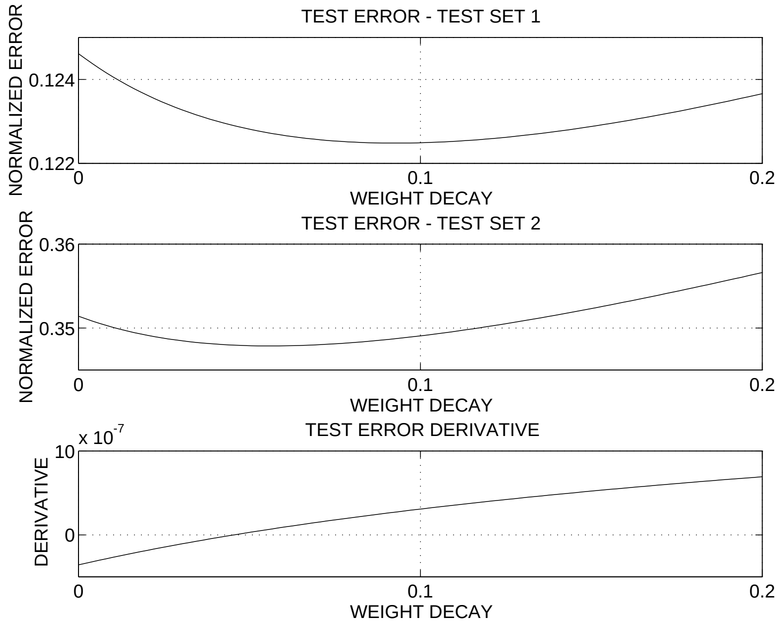

Figure 1: The role of weight decay regularization for a linear model on the sunspot benchmark series. The two upper figures show the errors (see [11] for a definition) on test sets both having shallow minima just below a weight decay of 0.1, while the bottom figure shows the derivative of the estimated test error as function of weight decay. Note that a gradient descent procedure will converge to the zero point.

There are several ways of implementing the adaptation scheme; here we initially set the weight decays to fixed values $\alpha_w = \alpha_W = 0.005$ for 100 epochs, then the network is trained with simultaneous adaptation of weight decay for 8000 epochs using eq. (13) with $\rho = 0.1$. After the initial training phase, further pruning and adaptation took place with pruning of 2% of the remaining weights per retraining round (400 epochs). In line with [12] it is seen that the stop criterion is able to select the optimal network. In figure 2 the normalized training errors, test errors cf. eq. (16), and the corresponding FPE error (after the initial training phase) are sketched for a training set size of 500 examples and the test set comprises 8500 examples. In [12] we

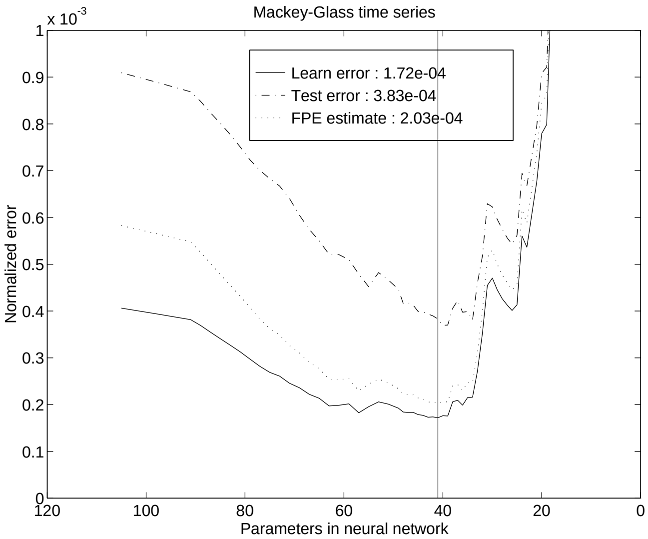

Figure 2: The evolution of training and test errors during pruning for the Mackey-Glass time series for a training set of size 500. The FPE estimate of the test error is based on eq. (11). The vertical line indicates the network for which the *estimated* test error is minimal.

compared the performances of pruned networks with those of fully connected nets, a linear model, and with a K-nearest-neighbor linear model. It was noted that the performance of the networks is similar to the nearest neighbor estimate. While the two weight decays previously were set manually we here adapt them according to equation (13). In figure 3 the development of the two weight decays is depicted as pruning proceeds. Note that the adaptive regularization scheme "chooses" relatively high regularization for the large network as should be expected. These networks have superfluous resources that could potentially harm generalization through overfitting. Eventually, at the end of the pruning session the test error estimates are rather biased (the network is underfitting) and the adaptive scheme does not provide reliable estimates. We have observed that the scheme in it present form has some

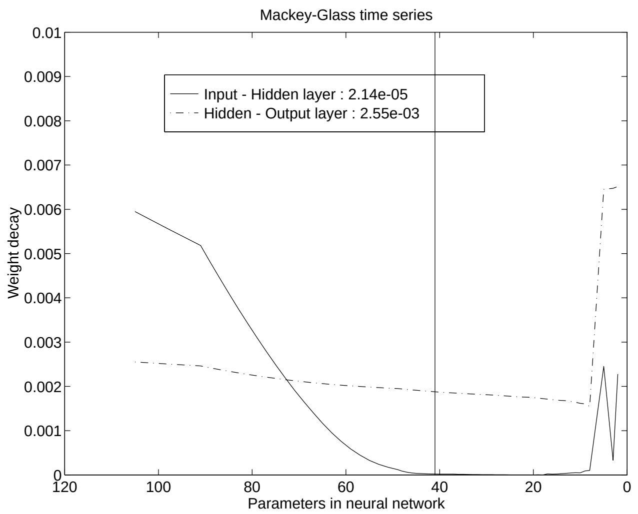

Figure 3: The evolution of weight decays during pruning for the Mackey-Glass time series. The vertical line indicates the network for which the *estimated* test error is minimal.

dependence on initialization of weights and weight decays; this is a topic for current research. The very low value of the input-to-hidden weight decay for the small networks is also in line with our earlier observations, namely that one can retrain the optimal architecture without weight decay and get slightly improved generalization [11, 12].

CONCLUSION

A scheme has been derived for adaptation of weight decay parameters. The scheme is based on asymptotic sampling theory. Two examples were given to illustrate the virtues of such adaptation. First, we showed that the functional form of the derivative of the estimated test error will provide convergence to near-optimal values for a linear model on the sunspot benchmark. Secondly, it was shown how the Designer Net framework can be applied with adaptive regularization, hence, relieving, manual tuning of these important parameters.

ACKNOWLEDGMENTS

This research was supported by the Danish Natural Science and Technical Research Councils through the Computational Neural Network Center (CONNECT).

REFERENCES

[1] H. Akaike, "Fitting Autoregressive Models for Prediction," Ann. Inst. Stat. Mat., **21**, 243-247, (1969).

[2] L.K. Hansen and C.E. Rasmussen, "Pruning from Adaptive Regularization," Accepted for Neural Computation, CONNECT Electronics Institute, Technical University of Denmark, preprint (1993).

[3] L.K. Hansen, "Stochastic Linear Learning: Exact Test and Training Error Averages," Neural Networks **6**, 393-396, (1993).

[4] J. Hertz, A. Krogh and R.G. Palmer, Introduction to the Theory of Neural Computation, Addison Wesley, New York, (1991).

[5] J. Larsen, Design of Neural Network Filters, Ph. D. Thesis, Electronics Institute, Technical University of Denmark, March (1993).

[6] J. Larsen and L.K. Hansen, "Generalization Performance of Regularized Neural Network Models," In proceedings of the IEEE Workshop on Neural Networks for Signal Processing NNSP'94.

[7] Y. Le Cun, J.S. Denker, and S.A. Solla, "Optimal Brain Damage," In Advances in Neural Information Processing Systems 2, 598-605, Morgan Kaufman, (1990).

[8] D. MacKay, "Bayesian interpolation". Neural Computation **4** 448-472, (1992).

[9] D. MacKay, "A practical framework for backpropagation networks". Neural Computation **4** 415-447 (1992).

[10] J.E. Moody, "Note on Generalization, Regularization and Architecture Selection in Nonlinear Systems," In Neural Networks For Signal Processing Proceedings of the 1991 IEEE-SP Workshop (Eds. B.H. Juang, S.Y. Kung, and C. Kamm), IEEE Service Center, 1-10, (1991).

[11] C. Svarer, L.K. Hansen, and J. Larsen, "On Design and Evaluation of Tapped Delay Line Networks," In Proceedings of the 1993 IEEE International Conference on Neural Networks San Francisco, 46-51, (1993).

[12] C. Svarer, L.K. Hansen, and J. Larsen, and C.E. Rasmussen, "Designer Networks for Time Series Processing," Proceedings of the 1993 IEEE Workshop on Neural Networks for Signal Processing (NNSP'93) Baltimore (Eds. C.A. Kamm et al.), 78-87, (1993).