1. Regularization of Neural Networks

Jan Larsen, Lars Kai Hansen

Department of Mathematical Modelling, Technical University of Denmark, Building 305, DK-2800 Lyngby, Denmark. Emails: jl,lkhansen@imm.dtu.dk.

Claus Svarer

Department of Neurology, National University Hospital, DK-2100 Copenhagen Ø, Denmark. Email: csvarer@pet.rh.dk.

1.1 Introduction

Neural networks are flexible tools for nonlinear function approximation and by expanding the network any relevant target function can be approximated [6]. The risk of overfitting on noisy data is of major concern in neural network design [2]. By using *regularization*, overfitting is reduced, thereby improving generalization ability on future data. In this contribution we present a scheme for estimation of regularization parameters using a simple validation set approach. Further work on empirical methods for optimization of neural networks models can be found in [10].

1.2 Training and Validation

The objective neural network modelling is to establish an estimate of the nonlinear relation among two vector variables: the input $\boldsymbol{x}$ and the output $\boldsymbol{y}$. The neural net is estimated (trained) from a dataset of related input-output examples.

The neural network implements a nonlinear function described by the vector function $\boldsymbol{f}(\boldsymbol{x};\boldsymbol{w})$ where $\boldsymbol{w}$ is the m-dimensional vector of all network weights. For a standard two-layer feed-forward neural net, the q'th component of $\boldsymbol{f}$ is given by (see e.g., [5]):

$$f_q(\boldsymbol{x},\boldsymbol{w}) = \sum_{i=1}^{m_H} w_{qi}^H \cdot \tanh\left(\sum_{j=1}^{m_I} x_j w_{qij}^I + w_{qi0}^I \right) + w_{q0}^H \tag{1.1}$$

where w_{qij}^I are input-to-hidden weights and w_{qi}^H hidden-to-output weights. m_I and m_H are the number of input and hidden neurons, respectively.

In order to train and validate, the available dataset, $\mathcal{D}$, of N collected input-output examples is split into two disjoint sets: a validation set, $\mathcal{V}$, with $N_v = \lceil \gamma N \rceil$ [1] examples

[1] $\lceil \cdot \rceil$ denotes rounding upwards to the nearest integer.

for architecture selection and estimation of regularization, and a training set, $\mathcal{T}$, with $N_t = N - N_v$ examples for estimation of network parameters. γ is referred to as the split-ratio.

The cost function for network training is supposed to be the sum of a loss function (or training error), $S_{\mathcal{T}}(\boldsymbol{w})$, and a regularization term $R(\boldsymbol{w})$, i.e.,

$$C(\boldsymbol{w}) = S_{\mathcal{T}}(\boldsymbol{w}) + R(\boldsymbol{w}) = \frac{1}{N_t}\sum_{k=1}^{N_t} \ell\left(\boldsymbol{y}(k), \widehat{\boldsymbol{y}}(k); \boldsymbol{w}\right) + R(\boldsymbol{w}) \tag{1.2}$$

where $\ell(\cdot)$ measures the distance between the output $\boldsymbol{y}(k)$ and the network prediction $\widehat{\boldsymbol{y}}(k) = \boldsymbol{f}(\boldsymbol{x}(k); \boldsymbol{w})$. Often the mean square error loss $\ell = |\boldsymbol{y} - \widehat{\boldsymbol{y}}|^2$ is chosen. $N_t \equiv |\mathcal{T}|$ defines the number of training examples, i.e., input-output pairs of the training set: $\mathcal{T} = \{(\boldsymbol{x}(k), \boldsymbol{y}(k))\}_{k=1}^{N_t}$. Training provides the estimated weight vector $\widehat{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}} C(\boldsymbol{w})$. The validation set consist of another $N_v \equiv |\mathcal{V}|$ examples and the validation error[2] of the trained network reads:

$$S_{\mathcal{V}}(\widehat{\boldsymbol{w}}) = \frac{1}{N_v}\sum_{k=1}^{N_v} \ell\left(\boldsymbol{y}(k), \widehat{\boldsymbol{y}}(k); \widehat{\boldsymbol{w}}\right) \tag{1.3}$$

where the sum runs over the N_v validation examples. $S_{\mathcal{V}}(\widehat{\boldsymbol{w}})$ is thus an estimate of the generalization error defined as the expected loss: $G(\widehat{\boldsymbol{w}}) = E_{\boldsymbol{x},\boldsymbol{y}}\{\ell(\boldsymbol{y}, \widehat{\boldsymbol{y}}; \widehat{\boldsymbol{w}})\}$, where $E_{\boldsymbol{x},\boldsymbol{y}}\{\cdot\}$ denotes the expectation w.r.t. to the joint input-output distribution.

A minimal necessary requirement for a procedure which estimates the network parameters on the training set and optimizes the amount of regularization from the validation set is the following: the generalization error of the regularized networks should be smaller than that of the non-regularized network trained on the *full data set*, $\mathcal{D}$. This is not always the case as discussed in [14, Chap. 6]: performing indirect regularization by stopping training when minimum validation error is reached is equivalent to training an non-regularized network on the full data set. There is consequently a risk of *overusing* the validation set. In order to avoid this situation prior constraints should be imposed, as will be demonstrated in various examples below.

There exist obviously an optimal choice of the split-ratio, γ, between training and validation set sizes. This choice is an interesting open and difficult problem. The choice depends 1. on the objective, 2. which scheme one for applies for minimizing the validation error, and 3. on the learning curve[3] [5] for the problem under consideration. The case of an optimal generalization error estimate (in mean square sense) was discussed in [9]. Without further ado, we would suggest to use the balanced split $\gamma = 0.5$.

1.3 Estimation of Regularization Parameters

The standard approach for estimation of regularization parameters, such as e.g., weight decays, is by more and less systematic search and evaluation of the validation set error; however, as will be shown, it is possible to derive a simple optimization algorithm based on gradient descent.

[2] The loss function on the validation set of data not used for training.

[3] The learning curve is a plot of the generalization error vs. the number of training examples.

Consider a regularization term $R(\boldsymbol{w}, \boldsymbol{\kappa})$ which depends on q regularization parameters contained in the vector $\boldsymbol{\kappa}$. Since the estimated weights $\widehat{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}} C(\boldsymbol{w})$ depend in a *definite* way on the regularization term, we may in fact consider the validation error as an *implicit function* of the regularization parameters,

$$S_{\mathcal{V}}(\widehat{\boldsymbol{w}}(\boldsymbol{\kappa})) = \frac{1}{N_v} \sum_{k=1}^{N_v} \ell\left(\boldsymbol{y}(k), \widehat{\boldsymbol{y}}(k); \widehat{\boldsymbol{w}}(\boldsymbol{\kappa})\right) \tag{1.4}$$

where $\widehat{\boldsymbol{w}}(\boldsymbol{\kappa})$ is the $\boldsymbol{\kappa}$-dependent vector of weights estimated from the training set. The optimal regularization can be found by e.g., gradient descent[4],

$$\boldsymbol{\kappa}(j+1) = \boldsymbol{\kappa}(j) - \eta \frac{\partial S_{\mathcal{V}}}{\partial \boldsymbol{\kappa}}(\widehat{\boldsymbol{w}}(\boldsymbol{\kappa}(j))) \tag{1.5}$$

where $\eta > 0$ is a gradient step-size and $\boldsymbol{\kappa}(j)$ is the estimate of the regularization parameters in iteration j. Using the chain rule the gradient vector can be rewritten as

$$\frac{\partial S_{\mathcal{V}}}{\partial \boldsymbol{\kappa}}(\widehat{\boldsymbol{w}}(\boldsymbol{\kappa}(j))) = \frac{\partial \boldsymbol{w}^\top}{\partial \boldsymbol{\kappa}}(\widehat{\boldsymbol{w}}(\boldsymbol{\kappa}(j))) \cdot \frac{\partial S_{\mathcal{V}}}{\partial \boldsymbol{w}}(\widehat{\boldsymbol{w}}(\boldsymbol{\kappa}(j))) \tag{1.6}$$

where $\partial \boldsymbol{w}^\top / \partial \boldsymbol{\kappa}$ is the $q \times m$ derivative matrix of the estimated weights w.r.t. the regularization parameters. In order to find this derivative matrix we use the following fact: if $\widehat{\boldsymbol{w}}(\boldsymbol{\kappa}(j))$ and $\widehat{\boldsymbol{w}}(\boldsymbol{\kappa}(j+1))$ are optimal weight vectors then

$$\frac{\partial C}{\partial \boldsymbol{w}}(\widehat{\boldsymbol{w}}(\boldsymbol{\kappa}(l))) = \boldsymbol{0}, \quad l = \{j, j+1\}. \tag{1.7}$$

This implies:

$$\frac{\partial^2 C}{\partial \boldsymbol{\kappa}^\top \partial \boldsymbol{w}}(\widehat{\boldsymbol{w}}(\boldsymbol{\kappa}(j))) = \boldsymbol{0}. \tag{1.8}$$

This equation can used for determining $\partial \boldsymbol{w}^\top / \partial \boldsymbol{\kappa}$. To be specific, assume that the regularization term is linear in the regularization parameters[5], i.e.,

$$R(\boldsymbol{w}, \boldsymbol{\kappa}) = \boldsymbol{\kappa}^\top \boldsymbol{r}(\boldsymbol{w}) = \sum_{i=1}^{q} \kappa_i r_i(\boldsymbol{w}) \tag{1.9}$$

where κ_i are the regularization parameters and $r_i(\boldsymbol{w})$ associated regularization functions. Using (1.9) in (1.8) gives after some algebra[6]:

$$\frac{\partial \boldsymbol{w}^\top}{\partial \boldsymbol{\kappa}}(\widehat{\boldsymbol{w}}) = -\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{w}^\top}(\widehat{\boldsymbol{w}}) \cdot \boldsymbol{J}^{-1}(\widehat{\boldsymbol{w}}) \tag{1.10}$$

[4] Moreover, one might consider suggesting second order schemes. However, these schemes work best for problems where the solution is an interior point somewhat away from the boundary of the domain over which the minimum is searched for. We tried a second order scheme for optimizing multiple weight decay parameters in a linear model without obtaining a convincing result.

[5] Although this regularization term is very general is does not include e.g., weight-elimination by Weigend *et al.* or Soft Weight Sharing by Nowlan *et al.*

[6] For convenience, here $\widehat{\boldsymbol{w}}$'s explicit $\boldsymbol{\kappa}$-dependence is omitted.

where $\boldsymbol{J} = \partial^2 C/\partial \boldsymbol{w}\partial \boldsymbol{w}^\top$ is the Hessian of the cost function which e.g., might be evaluated using the Gauss-Newton approximation [12]. Finally, substituting (1.10) into (1.6) gives

$$\frac{\partial S_{\mathcal{V}}}{\partial \boldsymbol{\kappa}}(\widehat{\boldsymbol{w}}) = -\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{w}^\top}(\widehat{\boldsymbol{w}}) \cdot \boldsymbol{J}^{-1}(\widehat{\boldsymbol{w}}) \cdot \frac{\partial S_{\mathcal{V}}}{\partial \boldsymbol{w}}(\widehat{\boldsymbol{w}}) \tag{1.11}$$

$\partial S_{\mathcal{V}}/\partial \boldsymbol{w}$ is found by ordinary back-propagation (see e.g., [5]) on the validation set while $\partial \boldsymbol{r}/\partial \boldsymbol{w}^\top$ is calculated from the specific assumptions on the regularizer. If the regularizer is assumed to be a weight decay regularizer with one regularization parameter per weight, i.e., $R(\boldsymbol{w}, \boldsymbol{\kappa}) = \sum_{i=1}^m \kappa_i w_i^2$, (1.11) reads:

$$\frac{\partial S_{\mathcal{V}}}{\partial \boldsymbol{\kappa}}(\widehat{\boldsymbol{w}}) = -2\left(\boldsymbol{J}^{-1}(\widehat{\boldsymbol{w}}) \cdot \frac{\partial S_{\mathcal{V}}}{\partial \boldsymbol{w}}(\widehat{\boldsymbol{w}})\right) \odot \widehat{\boldsymbol{w}} \tag{1.12}$$

where $\odot$ denotes the element-by-element vector product. In the case of one regularization parameter per weight there is an potential risk of overusing the validation set: it is frequently possible to choose $\boldsymbol{\kappa}$ such that $\partial S_{\mathcal{V}}/\partial \boldsymbol{w}|_{\widehat{\boldsymbol{w}}} = \boldsymbol{0}$ [7]. This corresponds to training on the validation set without regularization, which mostly is worse than training a non-regularized network on all data. Consequently, there is a need for constraining the search space. A natural choice is to impose $\kappa_i \geq 0$. Thus the regularization may be interpreted as a pruning regularizer since the weight is pruned by $\kappa_i \to \infty$ and kept in the model by having $\kappa_i = 0$. In practical implementations, the restriction $\kappa_i \geq 0$ is easily incorporated by enforcing κ_i the update in (1.5) to be positive even though the gradient step suggest it to be negative[8].

In the case of a simple weight decay, $R(\boldsymbol{w}, \kappa) = \kappa|\boldsymbol{w}|^2$, (1.11) obeys:

$$\frac{\partial S_{\mathcal{V}}}{\partial \kappa}(\widehat{\boldsymbol{w}}) = -2\widehat{\boldsymbol{w}}^\top \cdot \boldsymbol{J}^{-1}(\widehat{\boldsymbol{w}}) \cdot \frac{\partial S_{\mathcal{V}}}{\partial \boldsymbol{w}}(\widehat{\boldsymbol{w}}). \tag{1.13}$$

In this case there is no obvious risk of overusing the validation set. That is, it might be advantageous to have a negative weight decay even though the asymptotic ($N_t \to \infty$) optimal weight decay is positive [8].

1.4 Experiments

1.4.1 Linear Model

Consider the following data generating linear system: $y(k) = \boldsymbol{x}^\top(k)\boldsymbol{w}^\circ + \varepsilon(k)$. $\boldsymbol{x}(k)$ follows a $m = 10$ variate Gaussian distribution $\mathcal{N}(\boldsymbol{0}, \boldsymbol{H})$ where $\boldsymbol{H}$ is the covariance matrix. $\boldsymbol{x}(k)$ is an i.i.d. sequence, i.e., a time-independent series. The noise $\varepsilon(k) \sim \mathcal{N}(0, \sigma_\varepsilon^2)$ is i.i.d. and independent of $\boldsymbol{x}(k)$. $\sigma_\varepsilon^2 = \mathrm{NSR} \cdot V\{y\}$ where NSR $= 0.2$. The "true" weights are $\boldsymbol{w}^\circ = [0000000111]^\top$.

In order to evaluate the typical algorithm performance, generate, say Q, independent data sets of size $N = 60$ and train initially $m = 10$ dimensional linear models using

[7] This possibility is always present for a single output linear model, $\widehat{y} = \boldsymbol{w}^\top \boldsymbol{x}$, using the mean square loss function.

[8] An alternative solution is to reparameterize; e.g., $\beta_i = |\kappa_i|$ which gives $\partial S_{\mathcal{V}}/\partial \beta_i = \mathrm{sign}(\kappa_i) \cdot \partial S_{\mathcal{V}}/\partial \kappa_i$.

the standard mean square error cost to obtain the estimates $\widehat{\boldsymbol{w}}^{(i)}$, $i \in [1; Q]$. The true generalization error of each model is then given by $G(\widehat{\boldsymbol{w}}^{(i)}) = \sigma_\varepsilon^2 + (\widehat{\boldsymbol{w}}^{(i)} - \boldsymbol{w}^\circ)^\top \boldsymbol{H}(\widehat{\boldsymbol{w}}^{(i)} - \boldsymbol{w}^\circ)$.

Define $G(\boldsymbol{w}_\mathcal{D})$, the generalization error of the model trained on all data $\mathcal{D}$ without any architecture optimization. Further define $G(\widehat{\boldsymbol{w}})$, the generalization error of the model using architecture optimization – in the present case; weight decay optimization. We consider basically two measures of algorithm performance: First, the probability of improvement, defined as

$$P_{\text{imp}} = Q^{-1} \sum_{i=1}^{Q} \mu(G(\boldsymbol{w}_\mathcal{D}^{(i)}) - G(\widehat{\boldsymbol{w}}^{(i)})) \tag{1.14}$$

where $\mu(x) = 1$ for $x > 0$ and zero otherwise. Secondly, the relative improvement in generalization error, defined as

$$\text{RGI}^{(i)} = 100\% \cdot [G(\boldsymbol{w}_\mathcal{D}^{(i)}) - G(\widehat{\boldsymbol{w}}^{(i)})] \,/\, G(\boldsymbol{w}_\mathcal{D}^{(i)}). \tag{1.15}$$

In order to evaluate the performance of the gradient descent method for optimizing regularization parameters, we choose the individual weight decay regularizer $R(\boldsymbol{w}, \boldsymbol{\kappa}) = \sum_{i=1}^m \kappa_i w_i^2$. The κ_i's are forced to be positive in order to eliminate overuse of the validation data. First the model is trained with initial values of κ_i, then κ_i are updated by (1.12) and (1.5) using a gradient parameter η, finally retrain the weights, and repeat the scheme until the relative change in $S_\mathcal{V}$ is less than a prescribed small number. Subsequently, the weights are retrained on *all* data using the optimized regularization. We tried many different parameter settings and found that the algorithm is sufficiently robust. Figure 1.1 summaries a few results of running the algorithm.

1.4.2 Feed-Forward Neural Network

We tested the performance of the individual weight decay optimization using the gradient descent algorithm (1.5) on the Mackey-Glass chaotic time series prediction problem, see e.g., [7], [15]. The goal is to predict the series 100 steps ahead based on previous observations. The feed-forward net configuration is an input lag-space $\boldsymbol{x} = [x(k), x(k-6), x(k-12), x(k-18)]$ of 4 inputs, 15 hidden hyperbolic tangent neurons, and a single linear output unit $\widehat{y}(k)$ which predicts $y(k) = x(k+100)$. We used a data set, $\mathcal{D}$, of $N = 500$ examples and an independent test set of 8500 examples. The network was trained on $\mathcal{T}$ using a Gauss-Newton off-line training scheme with initial $\kappa_i = 4 \cdot 10^{-6}$. The gradient parameter η was initialized at 10^{-3}. If the $S_\mathcal{V}$ drops $\eta \leftarrow 1.2 \cdot \eta$; otherwise perform a successive bisection until a drop in $S_\mathcal{V}$ is achieved[9]. Initial experiments showed that the algorithm is fairly insensitive to the split-ratio in a region around $\gamma = 0.5$, thus $\gamma = 0.5$ is used in the following experiments. We made an ensemble of 5 networks with different random initial weights and splits of training and validation data and ran the algorithm. Afterwards, we retrained on all data using the optimized weight decays found at minimum validation error. The performance is measured by the test error normalized with the variance of $x(k)$. For comparison we made 11 repetitions of the standard OBD

[9] If η reaches 10^{-6} the update of κ is done irrespective of a an increase in $S_\mathcal{V}$.

γ	P_{imp}	avr(RGI)
0.5	0.964 ± 0.005	9.99
0.65	0.959 ± 0.006	10.2
0.7	0.939 ± 0.007	10.1

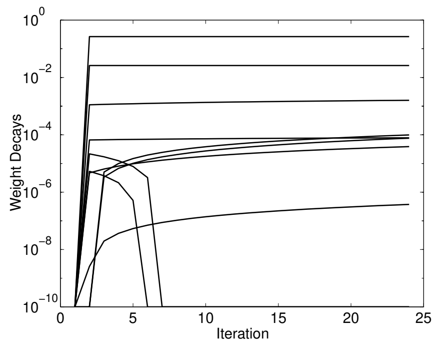


Figure 1.1: Performance when optimizing multiple weight decays on a linear model. γ is the split-ratio, P_{imp} the probability of improvement with associated 95% confidence error bars, and avr(RGI) is the average RGI over cases in which improvement was present. Experiment parameters: $Q = 5000$, $\boldsymbol{H} = \boldsymbol{A}\boldsymbol{A}^{\top}$ where $\boldsymbol{A}$ is a randomly chosen Vandermonde matrix, gradient parameter $\eta = 10^{-5}$, initial value of κ_i was 10^{-10}, algorithm stops when relative change in $S_{\mathcal{V}}$ is less than $2 \cdot 10^{-4}$. With this stopping, the average number of iterations was around 20. Note a relatively week dependency on the split-ratio. The plot in the right panel shows a typical run of the algorithm. Note that non-monotonic weights which start out to be regularized later on become non-regularized.

approach as described in [15]. The network employed 25 hidden neurons. Figure 1.2 shows the algorithm performance.

1.5 Conclusions

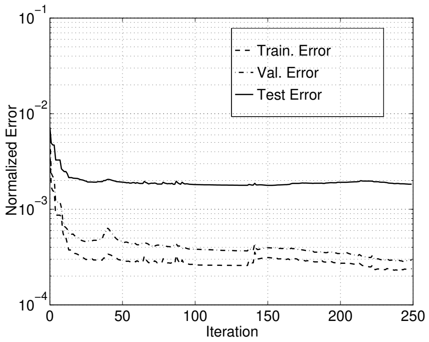
In this paper we suggested to improve neural network performance by optimizing the amount of regularization using a validation set based approach. The parameters of the regularization term are estimated by a gradient descent scheme. Numerical examples with linear models and neural networks demonstrated the potential of the framework.

1.6 Acknowledgments

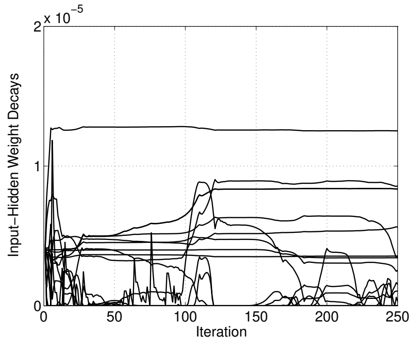
This research was supported by the Danish Natural Science and Technical Research Councils through the Computational Neural Network Center. JL furthermore acknowledge the Radio Parts Foundation for financial support.

References

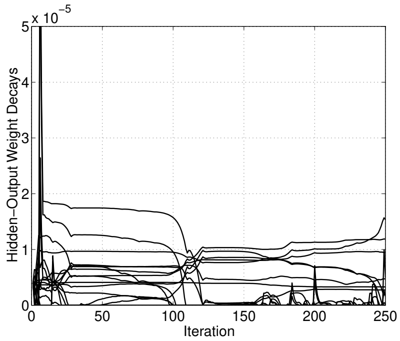
1. H. Akaike, *Fitting Autoregressive Models for Prediction*, Annals of the Institute of Statistical Mathematics, **21** (1969), 243–247.



(a)



(b)



(c)

Norm. Test Error	OBD	Indiv. Wgt. Decay
avr	$2.07 \cdot 10^{-3}$	$2.61 \cdot 10^{-3}$
min	$0.87 \cdot 10^{-3}$	$1.78 \cdot 10^{-3}$
max	$3.68 \cdot 10^{-3}$	$3.26 \cdot 10^{-3}$

(d)

Figure 1.2: (a) Evolution of training, validation and test errors. (b) Evolution of input-to-hidden weight decays. (c) Evolution of hidden-to-output weight decays. (d) Test error normalized with the variance of $x(n)$ for the algorithm compared to standard OBD. *avr*, *min*, *max* are the average, minimum and maximum performance over different runs, respectively. The effective number of parameters [8] was on the average $m_{\text{eff}} \approx 58$ for individual weight decay regularization and $m_{\text{eff}} \approx 83$ for OBD. The best normalized test error reported (see e.g., [7]) was around $2.5 \cdot 10^{-3}$.

2. S. Geman, E. Bienenstock & R. Doursat, *Neural Networks and the Bias/Variance Dilemma*, Neural Computation, **4** (1992), 1–58.

3. L.K. Hansen & M. With Pedersen, *Controlled Growth of Cascade Correlation Nets*, in M. Marinaro & P.G. Morasso (eds.), Proceedings of International Conference on Neural Networks, Sorrento, Italy, (1994), 797–800.

4. B. Hassibi & D.G. Stork, *Second Order Derivatives for Network Pruning: Optimal Brain Surgeon*, in S.J. Hanson *et al.* (eds.), Advances in Neural Information Processing Systems 5, Proceedings of the 1992 Conference, San Mateo, California: Morgan Kaufmann Publishers, (1993), 164–171.

5. J. Hertz, A. Krogh & R.G. Palmer, *Introduction to the Theory of Neural Computation*, Redwood City, California: Addison-Wesley Publishing Company, 1991.

6. K. Hornik, *Approximation Capabilities of Multilayer Feedforward Networks*, Neural Networks, **4** (1991), 251–257.

7. J. Larsen, *Design of Neural Network Filters*, Ph.D. Thesis, Electronics Institute, Technical University of Denmark, 1993. Available via ftp://eivind.imm.dtu.dk/dist/PhD_thesis/jlarsen.thesis.ps.Z

8. J. Larsen & L.K. Hansen, *Generalization Performance of Regularized Neural Network Models*, in J. Vlontzos *et al.* (eds.), Proceedings of the IEEE Workshop on Neural Networks for Signal Processing IV, Piscataway, New Jersey: IEEE, (1994), 42–51.

9. J. Larsen & L.K. Hansen, *Empirical Generalization Assessment of Neural Network Models*, in F. Girosi *et al.* (eds.), Proceedings of the IEEE Workshop on Neural Networks for Signal Processing V, Piscataway, New Jersey: IEEE, (1995), 30–39.

10. J. Larsen, L.K. Hansen, C. Svarer & M. Ohlsson: *Design and Regularization of Neural Networks: The Optimal Use of a Validation Set*, in S. Usui, Y. Tohkura, S. Katagiri & E. Wilson (eds.), Proceedings of the IEEE Workshop on Neural Networks for Signal Processing V, Piscataway, New Jersey: IEEE, (1996), 62–71.

11. Y. Le Cun, J.S. Denker & S.A. Solla, *Optimal Brain Damage*, in D.S. Touretzky (ed.), Advances in Neural Information Processing Systems 2, Proceedings of the 1989 Conference, San Mateo, California: Morgan Kaufmann Publishers, (1990), 598–605.

12. L. Ljung, *System Identification: Theory for the User*, Englewood Cliffs, New Jersey: Prentice-Hall, 1987.

13. M. With Pedersen, L.K. Hansen & J. Larsen, *Pruning with Generalization Based Weight Saliencies: γOBD, γOBS*, in D.S. Touretzky *et al.* (eds.), Advances in Neural Information Processing Systems 8, Proceedings of the 1995 Conference, Cambridge, Massachusetts: MIT Press, (1996), 521–527.

14. J. Sjöberg, *Non-Linear System Identification with Neural Networks*, Ph.D. Thesis no. 381, Department of Electrical Engineering, Linköping University, Sweden, 1995.

15. C. Svarer, L.K. Hansen, J. Larsen & C. E. Rasmussen, *Designer Networks for Time Series Processing*, in C.A. Kamm *et al.* (eds.), Proceedings of the IEEE Workshop on Neural Networks for Signal Processing 3, Piscataway, New Jersey: IEEE, (1993), 78–87.