DESIGN AND REGULARIZATION OF NEURAL NETWORKS: THE OPTIMAL USE OF A VALIDATION SET

J. Larsen[a], L.K. Hansen[a], C. Svarer[b] and M. Ohlsson[a]
[a]CONNECT, Department of Mathematical Modelling, Building 349
Technical University of Denmark
DK-2800 Lyngby, Denmark
emails: jl@imm.dtu.dk, lkhansen@ei.dtu.dk, mo@imm.dtu.dk

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

Abstract - In this paper we derive novel algorithms for estimation of regularization parameters and for optimization of neural net architectures based on a validation set. Regularization parameters are estimated using an iterative gradient descent scheme. Architecture optimization is performed by approximative combinatorial search among the relevant subsets of an initial neural network architecture by employing a validation set based Optimal Brain Damage/Surgeon (OBD/OBS) or a mean field combinatorial optimization approach. Numerical results with linear models and feed-forward neural networks demonstrate the viability of the methods.

INTRODUCTION

Neural networks are flexible tools for function approximation and by expanding the network any relevant target function can be approximated [6]. The associated risk of overfitting on noisy data is of major concern in neural network design [2]. The objective of architecture optimization is to minimize the generalization error. The literature suggest a variety of algebraic generalization error estimators e.g., FPE [1], FPER [9], GEN [7], GPE [13] and NIC [14]. These estimates, however, depend on a number of statistical assumptions which can be quite hard to justify. Hence, many neural net practitioners resort to empirical methods for design and validation, the simplest being to base the design on a single separate validation set. For further discussion on empirical generalization assessment see e.g., [10]. In this contribution we derive schemes for proper selection of neural net architecture and for estimation of regularization parameters using a simple validation set approach. In fact, one may view estimation of the optimal regularization as an alternative methodology for architecture selection.

TRAINING AND VALIDATION

In order to train and validate, the available dataset, $\mathcal{D}$, of N examples is split into two disjoint sets: a validation set, $\mathcal{V}$, with $N_v = \lceil \gamma N \rceil$ [1] examples 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 neural network is described by the vector function $\boldsymbol{f}(\boldsymbol{x};\boldsymbol{w})$ where $\boldsymbol{x}$ is the input vector and $\boldsymbol{w}$ is the vector of network weights and thresholds with dimensionality m. 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.,

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

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:

$$S_{\mathcal{V}}(\widehat{\boldsymbol{w}}) = \frac{1}{N_v}\sum_{k=1}^{N_v} \ell\left(\boldsymbol{y}(k), \widehat{\boldsymbol{y}}(k); \widehat{\boldsymbol{w}}\right) \qquad (2)$$

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.

An minimal necessary requirement for a procedure which estimates the network parameters on the training set and optimizes the architecture on the validation set is that the generalization error of an *non-optimized* network architecture trained on the full data set $\mathcal{D}$ is higher. This is not always the case as discussed in [17, 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

[1] $\lceil\cdot\rceil$ denotes rounding upwards to the nearest integer.
[2] The loss function on the validation set.

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 [10]. Without further ado, one would suggest to use the balanced split $\gamma = 0.5$

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 an optimization algorithm based on gradient descent.

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,

$$S_V(\widehat{\boldsymbol{w}}(\boldsymbol{\kappa})) = \frac{1}{N_v} \sum_{k=1}^{N_v} \ell\left(y(k), \widehat{y}(k); \widehat{\boldsymbol{w}}(\boldsymbol{\kappa})\right) \tag{3}$$

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],

$$\boldsymbol{\kappa}(j+1) = \boldsymbol{\kappa}(j) - \eta \frac{\partial S_V}{\partial \boldsymbol{\kappa}}(\widehat{\boldsymbol{w}}(\boldsymbol{\kappa}(j))) \tag{4}$$

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

$$\frac{\partial S_V}{\partial \boldsymbol{\kappa}}(\widehat{\boldsymbol{w}}(\boldsymbol{\kappa}(j))) = \frac{\partial \boldsymbol{w}^\top}{\partial \boldsymbol{\kappa}}(\widehat{\boldsymbol{w}}(\boldsymbol{\kappa}(j))) \cdot \frac{\partial S_V}{\partial \boldsymbol{w}}(\widehat{\boldsymbol{w}}(\boldsymbol{\kappa}(j))) \tag{5}$$

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

$$\frac{\partial C}{\partial \boldsymbol{w}}(\widehat{\boldsymbol{w}}(\boldsymbol{\kappa}(l))) = \boldsymbol{0}, \quad l = \{j, j+1\}. \tag{6}$$

This implies:

$$\frac{\partial^2 C}{\partial \boldsymbol{\kappa}^\top \partial \boldsymbol{w}}(\widehat{\boldsymbol{w}}(\boldsymbol{\kappa}(j))) = \boldsymbol{0}. \tag{7}$$

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

[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.

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.,

$$R(\boldsymbol{w}, \boldsymbol{\kappa}) = \boldsymbol{\kappa}^\top \boldsymbol{r}(\boldsymbol{w}) = \sum_{i=1}^{q} \kappa_i r_i(\boldsymbol{w}) \tag{8}$$

where κ_i are the regularization parameters and $r_i(\boldsymbol{w})$ associated regularization functions. Using Eq. (8) in (7) gives after some algebra[6]:

$$\frac{\partial \boldsymbol{w}^\top}{\partial \boldsymbol{\kappa}}(\widehat{\boldsymbol{w}}) = -\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{w}^\top}(\widehat{\boldsymbol{w}}) \cdot \boldsymbol{J}^{-1}(\widehat{\boldsymbol{w}}) \tag{9}$$

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 Eq. (9) into (5) gives

$$\frac{\partial S_{\mathcal{V}}}{\partial \boldsymbol{\kappa}}(\widehat{\boldsymbol{w}}) = -\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{w}^\top}(\widehat{\boldsymbol{w}}) \cdot \boldsymbol{J}^{-1}(\widehat{\boldsymbol{w}}) \cdot \frac{\partial S_{\mathcal{V}}}{\partial \boldsymbol{w}}(\widehat{\boldsymbol{w}}) \tag{10}$$

$\partial S_{\mathcal{V}} / \partial \boldsymbol{w}$ is found by ordinary back-propagation 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., $R(\boldsymbol{w}, \boldsymbol{\kappa}) = \sum_{i=1}^{m} \kappa_i w_i^2$, Eq. (10) reads:

$$\frac{\partial S_{\mathcal{V}}}{\partial \boldsymbol{\kappa}}(\widehat{\boldsymbol{w}}) = -2 \left(\boldsymbol{J}^{-1}(\widehat{\boldsymbol{w}}) \cdot \frac{\partial S_{\mathcal{V}}}{\partial \boldsymbol{w}}(\widehat{\boldsymbol{w}}) \right) \odot \widehat{\boldsymbol{w}} \tag{11}$$

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 Eq. (4) to be positive even though the gradient step suggest it to be negative[8].

In the case of a simple weight decay, $R(\boldsymbol{w}, \kappa) = \kappa |\boldsymbol{w}|^2$, Eq. (10) obeys:

$$\frac{\partial S_{\mathcal{V}}}{\partial \kappa}(\widehat{\boldsymbol{w}}) = -2\widehat{\boldsymbol{w}}^\top \cdot \boldsymbol{J}^{-1}(\widehat{\boldsymbol{w}}) \cdot \frac{\partial S_{\mathcal{V}}}{\partial \boldsymbol{w}}(\widehat{\boldsymbol{w}}). \tag{12}$$

[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 omtitted.

[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 $\partial S_{\mathcal{V}} / \partial \beta_i = \mathbf{sign}(\kappa_i) \cdot \partial S_{\mathcal{V}} / \partial \kappa_i$.

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 [9].

VALIDATION ERROR BASED OBD/OBS

A natural extension of the Optimal Brain Damage (OBD) [11] and Optimal Brain Surgeon (OBS) [4] pruning schemes is to base saliencies on generalization error rather than training error, as suggested in [15]. This enables deleting weights according to their influence on generalization error. The validation set based saliencies are based on the second order Taylor series expansion:

$$S_{\mathcal{V}}(\boldsymbol{w}) \approx S_{\mathcal{V}}(\widehat{\boldsymbol{w}}) + \frac{\partial S_{\mathcal{V}}}{\partial \boldsymbol{w}^{\top}}(\widehat{\boldsymbol{w}}) \cdot \Delta\boldsymbol{w} + \frac{1}{2}\Delta\boldsymbol{w}^{\top} \cdot \frac{\partial^2 S_{\mathcal{V}}}{\partial \boldsymbol{w} \partial \boldsymbol{w}^{\top}}(\widehat{\boldsymbol{w}}) \cdot \Delta\boldsymbol{w} \qquad (13)$$

where $\Delta\boldsymbol{w} = \boldsymbol{w} - \widehat{\boldsymbol{w}}$. For the OBD approximation: delete one weight and keep the remaining unchanged[9] results in the weight change $\Delta\boldsymbol{w} = [0 \cdots 0, -\widehat{w}_i, 0 \cdots 0]^{\top}$. Hence, the OBD validation saliencies[10] are:

$$\varrho_i = -\widehat{w}_i \frac{\partial S_{\mathcal{V}}}{\partial w_i}(\widehat{\boldsymbol{w}}) + \frac{1}{2}\widehat{w}_i^2 \frac{\partial^2 S_{\mathcal{V}}}{\partial w_i^2}(\widehat{\boldsymbol{w}}). \qquad (14)$$

The validation set based OBD procedure, vOBD, is as follows: 1. choose a sufficiently large network, 2. train the network on $\mathcal{T}$, 3. rank weights according to Eq. (14) and delete a small fraction of low saliency weights[11], 4. if $S_{\mathcal{V}}$ is minimal stop; otherwise go to step 2.

Similarly, vOBS saliencies can be obtanied from Eq. (13) using the fact that the remaining weights are retrained within a second order approximation of the cost function, see e.g., [3], [8].

COMBINATORIAL OPTIMIZATION OF NETWORK ARCHITECTURE

Architecture selection schemes are based on either network pruning or network growth. Preferably one wish for a hybrid method where deleted weights can reenter the model and vice versa. Exhaustive search would require evaluation of $2^m - 1$ subnetworks – m being the number of weights in the fully connected network. An alternative method is to use the computationally cost effective mean field annealing approach (see e.g., [5]) which have shown efficient for hard combinatorial optimization problems [16]. This approach is invoked by using a state vector of binary variables $\boldsymbol{s} \in \{0, 1\}^m$ indicating if the companion weights are pruned or not. That is, the network function

[9]This corresponds to assuming the Hessian, $\boldsymbol{J}$, to be diagonal.

[10]The saliencies $\varrho_i \approx S_{\mathcal{V}}(\widehat{\boldsymbol{w}} + \Delta\boldsymbol{w}) - S_{\mathcal{V}}(\widehat{\boldsymbol{w}})$.

[11]The saliencies will often be negative, indicating that the generalization actuaclly is improved by removing the particular weight.

becomes $\boldsymbol{f}(\boldsymbol{x}; \boldsymbol{w} \odot \boldsymbol{s})$. The weights are estimated on the training set and the state variables on the validation set by using a mean field approach. Inital experiments showed fair results; however, a crucial problem is that retraining is not invoked in the pruning scheme. We are currently pursuing this topic further.

EXPERIMENTS

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 $\mathrm{NSR} = 0.2$. 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 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 $G(\widehat{\boldsymbol{w}}^{(i)}) = \sigma_\varepsilon^2 + (\widehat{\boldsymbol{w}}^{(i)} - \boldsymbol{w}^\circ)^\top \boldsymbol{H} (\widehat{\boldsymbol{w}}^{(i)} - \boldsymbol{w}^\circ)$.

One Weight Decay. Dealing with the linear model and a single weight decay, i.e., $R(\boldsymbol{w}, \kappa) = \kappa|\boldsymbol{w}|^2$, it is possible to show that Eq. (12), for the optimal κ, becomes a 3rd order equation in κ. However, the coefficients of the equation depends in a very complicated way on the actual training and validation data; hence, we resort to numerical optimization[12]. 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

$$P_{\mathrm{imp}} = Q^{-1} \sum_{i=1}^{Q} \mu(G(\boldsymbol{w}_\mathcal{D}^{(i)}) - G(\widehat{\boldsymbol{w}}^{(i)})) \tag{15}$$

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

$$\mathrm{RGI}^{(i)} = 100\% \cdot [G(\boldsymbol{w}_\mathcal{D}^{(i)}) - G(\widehat{\boldsymbol{w}}^{(i)})] \,/\, G(\boldsymbol{w}_\mathcal{D}^{(i)}). \tag{16}$$

The results are summarized in Table 1.

Multiple Weight Decays and vOBD. In order to evaluate the performance of the gradient descent method for optimizing regularization parameters, we choose the individual weight decay regularizer $R(\boldsymbol{w}, \boldsymbol{\kappa}) = \sum_{i=1}^{m} \kappa_i w_i^2$.

[12]Here the MATLAB function fmin.m is used.

γ	P_{imp}	$avr(\text{RGI})$
0.4	0.846 ± 0.007	9.49
0.5	0.892 ± 0.006	9.61
0.6	0.911 ± 0.006	9.87
0.7	0.939 ± 0.005	9.88

Table 1: Performance when optimizing a single weight decay on a linear model. γ is the split-ratio, P_{imp} the probability of improvement with associated 95% confidence error bars, and $avr(\text{RGI})$ is the average RGI over cases in which improvement was present. Experiment parameters: $Q = 10000$, $\boldsymbol{H} = \boldsymbol{A}\boldsymbol{A}^{\top}$ where $\boldsymbol{A}$ is a randomly chosen Vandermonde matrix (large eigenvalue spread). Note that the results do only have a week dependency on the split-ratio.

The κ_i's are forced to be positive in order to eliminate overuse of the validation data. First the model is trained with inital values of κ_i, then κ_i are updated by Eq. (11) and (4) 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. Fig. 1 summaries a few results of running the algorithm. For comparison we ran a standard

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

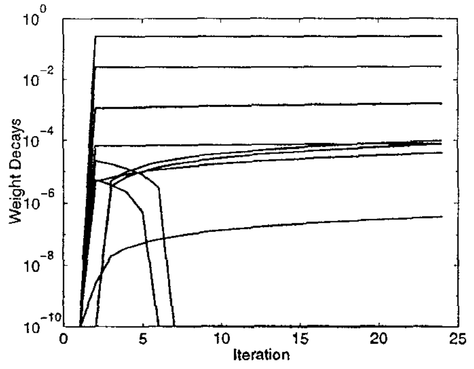


Figure 1: Performance when optimizing multiple weight decays on a linear model. γ is the split-ratio, P_{imp} the probability of improvement with associated 95% confidence error bars, and $avr(\text{RGI})$ is the average RGI over cases in which improvement was present. Experiment parameters: $Q = 5000$, $\boldsymbol{H} = \boldsymbol{A}\boldsymbol{A}^{\top}$ where $\boldsymbol{A}$ is a randomly chosen Vandermonde matrix (different from that used in Table 1), gradient parameter $\eta = 10^{-5}$, inital 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 – as in Table 1 – 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.

OBD[13] and a validation set based vOBD deleting one weight per iteration

[13]As in [18], the standard OBD is terminated when the FPE criterion [1] is minimal.

with the same covariance matrix $\boldsymbol{H}$ as used in Fig. 1. In $Q = 10000$ repetitions the vOBD result is: $P_{\text{imp}} = 0.863 \pm 0.007$ and $avr(RGI) = 4.57$ for $\gamma = 0.7$. OBD delivers: $P_{\text{imp}} = 0.835 \pm 0.007$ and $avr(RGI) = 3.65$. Thus inidividual weight decay regularization is superior to vOBD, which again is slightly better than standard OBD.

Feed-Forward Neural Network

We tested the performance of the individual weight decay optimization using the gradient descent algorithm Eq. (4) on the Mackey-Glass chaotic time series prediction problem, see e.g., [8], [18]. 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 exampels. 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[14]. 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 approach as described in [18]. The network employed 25 hidden neurons. Fig. 2 shows the algorithm performance.

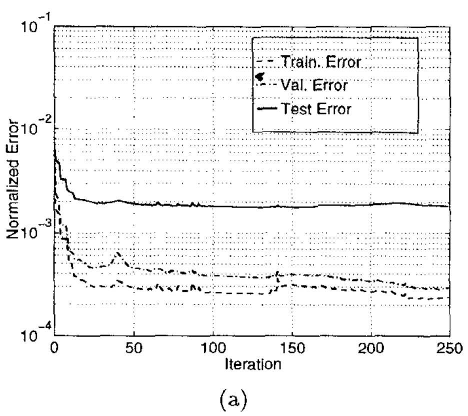
CONCLUSIONS

In this paper it was suggested to optimize the network architecture by minimizing the error on a validation set. We derived a gradient descent scheme for optimizing regularization parameters, suggested to use validation set based saliency estimates for OBD/OBS, and finally discussed the possibility of approximative combinatorial architecture search by using the mean field algorithm. Numerical examples with linear models and neural networks demonstrated the potential of the framework.

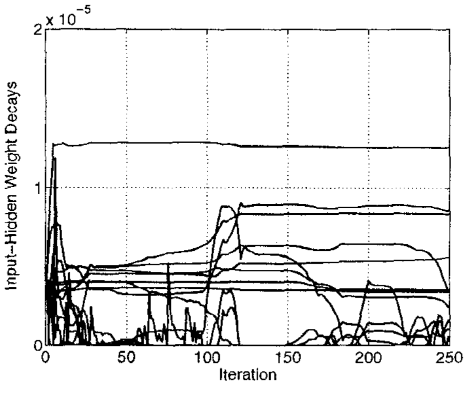
ACKNOWLEDGMENTS

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.

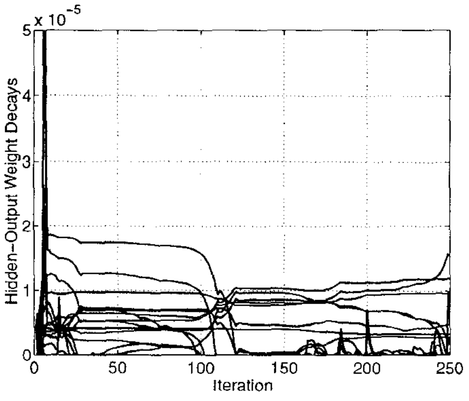
[14]If η reaches 10^{-6} the update of κ is done irrespective of a an increase in $S_{\mathcal{V}}$.



(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 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 [9] 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., [8]) was around $2.5 \cdot 10^{-3}$.

REFERENCES

[1] H. Akaike, "Fitting Autoregressive Models for Prediction," **Annals of the Institute of Statistical Mathematics**, vol. 21, pp. 243–247, 1969.

[2] S. Geman, E. Bienenstock & R. Doursat, "Neural Networks and the Bias/Variance Dilemma," **Neural Computation**, vol. 4, pp. 1–58, 1992.

[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, pp. 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, pp. 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**, vol. 4, pp. 251–257, 1991.

[7] J. Larsen, "A Generalization Error Estimate for Nonlinear Systems," in S.Y. Kung *et al.* (eds.), **Neural Networks for Signal Processing 2: Proceedings of the 1992 IEEE-SP Workshop**, Piscataway, New Jersey: IEEE, 1992, pp. 29–38.

[8] J. Larsen, **Design of Neural Network Filters**, Ph.D. Thesis, Electronics Institute, Technical University of Denamrk, 1993. Available via ftp://ei.dtu.dk/dist/PhD_thesis/jlarsen.thesis.ps.Z

[9] 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, pp. 42–51, 1994.

[10] 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, pp. 30–39.

[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, pp. 598–605.

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

[13] J. Moody, "Prediction Risk and Architecture Selection for Neural Networks," in V. Cherkassky *et al.* (eds.), **From Statistics to Neural Networks: Theory and Pattern Recognition Applications**, Series F, vol. 136, Berlin, Germany: Springer-Verlag, 1994.

[14] N. Murata, S. Yoshizawa & S. Amari, "Network Information Criterion — Determining the Number of Hidden Units for an Artificial Neural Network Model," **IEEE Transactions on Neural Networks**, vol. 5, no. 6, pp. 865–872, Nov. 1994.

[15] 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.

[16] C. Peterson & B. Söderberg, "A New Method for Mapping Optimization Problems onto Neural Networks," **International Journal of Neural Systems**, vol. 1, pp. 3–22, 1989.

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

[18] 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, pp. 78–87.