ADAPTIVE REGULARIZATION OF NEURAL CLASSIFIERS

L. Nonboe Andersen, J. Larsen, L.K. Hansen & M. Hintz-Madsen
CONNECT, Department of Mathematical Modelling, Building 321
Technical University of Denmark, DK-2800 Lyngby, Denmark
Phones: +45 4525+ext. 3899,3923,3889,3894
Fax: +45 45872599
emails: lna,jl,lkhansen,mhm@imm.dtu.dk
www: http://eivind.imm.dtu.dk

Abstract. In this paper we present a regularization scheme which iteratively adapts the regularization parameters by minimizing the validation error. It is suggested to use the adaptive regularization scheme in conjunction with Optimal Brain Damage pruning to optimize the architecture and to avoid overfitting. Furthermore, we propose an improved neural classification architecture eliminating an inherent redundancy in the widely used SoftMax classification network. Numerical results demonstrate the viability of the method.

INTRODUCTION

Neural networks are flexible tools for pattern recognition and by expanding the network architecture any relevant target function can be approximated [6]. In this contribution we present an improved version of the neural classifier architecture based on a feed-forward net with SoftMax [2] normalization presented in [7], [8] avoiding an inherent redundant parameterization. The outputs of the network estimate the class conditional posterior probabilities and the network is trained using a maximum a posteriori (MAP) framework.

The associated risk of overfitting on noisy data is of major concern in neural network design [4]. The objective of architecture optimization is to minimize the generalization error. The architecture can be optimized *directly* by e.g., pruning techniques or *indirectly* by using regularization. One might consider various regularization schemes: from adapting a single regularization parameter to individual regularization of the weights in the net. These subjects are further addressed in [9], [10]. We suggest a hybrid approach with Optimal Brain Damage [11] for pruning and an adaptive regularization scheme. The inevitable problem of adapting the *amount* of regularization is solved by minimizing the generalization error w.r.t. regularization parameters. Using the validation error calculated from a single validation set as an estimate of the generalization error, it is possible to formulate an iterative gradient descent scheme for adapting the regularization parameters [9]. The Bayesian way to adapt regularization parameters is to minimize the evidence [1, Ch. 10], [14]; however, the evidence does not, in a simple way, relate to the generalization error which is our primary object of interest.

NETWORK ARCHITECTURE

Suppose that the input (feature) vector is denoted by $\boldsymbol{x}$ with $\dim(\boldsymbol{x}) = n_I$. The aim is to model the posterior probabilities $p(\mathcal{C}_i|\boldsymbol{x})$, $i = 1, 2, \cdots, c$ where $\mathcal{C}_i$ denotes the i'th class. Then under a simple loss function the Bayes optimal[1] classifier assigns class label $\mathcal{C}_i$ to $\boldsymbol{x}$ if $i = \arg\max_j p(\mathcal{C}_j|\boldsymbol{x})$.

Following [8] (see also [1]), the outputs, $\widehat{y}_i$, of the neural network represent *estimates* of the posterior probabilities, i.e., $\widehat{y}_i = \widehat{p}(\mathcal{C}_i|\boldsymbol{x})$; hence, $\sum_{i=1}^{c} p(\mathcal{C}_i|\boldsymbol{x}) = 1$. That is, we need merely to estimate $c-1$ posterior probabilities, say $p(\mathcal{C}_i|\boldsymbol{x})$, $i = 1, 2, \cdots, c-1$, then the last is calculated as $p(\mathcal{C}_c|\boldsymbol{x}) = 1 - \sum_{i=1}^{c-1} p(\mathcal{C}_i|\boldsymbol{x})$.

Define a 2-layer feed-forward network with n_I inputs, n_H hidden neurons and $c-1$ outputs by:

$$h_j(\boldsymbol{x}) = \tanh\left(\sum_{\ell=1}^{n_I} w_{j\ell}^I x_\ell + w_{j0}^I\right), \quad \phi_i(\boldsymbol{x}) = \sum_{j=1}^{n_H} w_{ij}^H h_j(\boldsymbol{x}) + w_{i0}^H \tag{1}$$

where $w_{j\ell}^I$, w_{ij}^H are the input-to-hidden and hidden-to-output weights, respectively. All weights are assembled in the weight vector $\boldsymbol{w} = \{w_{j\ell}^I, w_{ij}^H\}$.

In order to interpret the network outputs as probabilities a *modified* normalized exponential transformation similar to SoftMax [2] is used,

$$\widehat{y}_i = \frac{\exp(\phi_i)}{\sum_{j=1}^{c-1} \exp(\phi_j) + 1}, \quad i = 1, 2, \cdots, c-1, \qquad \widehat{y}_c = 1 - \sum_{i=1}^{c-1} \widehat{y}_i. \tag{2}$$

The modification amounts to fixing $\exp(\phi_c)$ in the standard SoftMax at 1 eliminating the inherent redundancy of the output weights as also mentioned in [18, p. 150]. The redundancy implies that a particular set of outputs, $\widehat{y}_i$, $i = 1, 2, \cdots, c$ induces a one-dimensional sub-manifold in weight space. The network architecture is shown in Fig. 1.

TRAINING AND REGULARIZATION

Assume that we have a training set $\mathcal{T}$ of N_t related input-output pairs $\mathcal{T} = \{(\boldsymbol{x}(k), \boldsymbol{y}(k))\}_{k=1}^{N_t}$ where

$$y_i(k) = \begin{cases} 1 & \text{if } \boldsymbol{x}(k) \in \mathcal{C}_i \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

[1] That is, each misclassification is equally serious corresponding to minimal probability of misclassification.

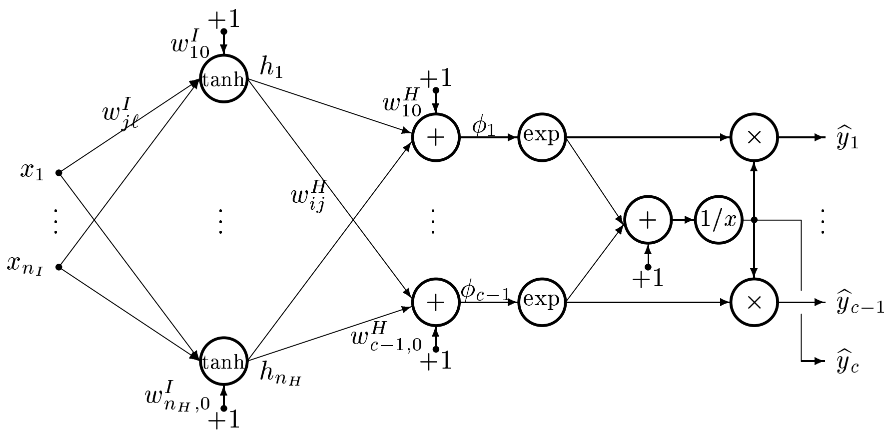

Figure 1: Neural network architecture.

The likelihood of the network parameters is given by (see e.g., [1], [8]),

$$p(\mathcal{T}|\boldsymbol{w}) = \prod_{k=1}^{N_t} p(\boldsymbol{y}(k)|\boldsymbol{x}(k), \boldsymbol{w}) = \prod_{k=1}^{N_t} \prod_{i=1}^{c} \left(\widehat{y}_i(k)\right)^{y_i(k)} \tag{4}$$

where $\widehat{\boldsymbol{y}}(k) = \widehat{\boldsymbol{y}}(\boldsymbol{x}(k), \boldsymbol{w})$ is a function of the input and weight vectors. The training error is the normalized negative log-likelihood

$$S_{\mathcal{T}}(\boldsymbol{w}) = -\frac{1}{N_t} \log p(\mathcal{T}|\boldsymbol{w}) \equiv \frac{1}{N_t} \sum_{k=1}^{N_t} \ell\left(\boldsymbol{y}(k), \widehat{\boldsymbol{y}}(k); \boldsymbol{w}\right) \tag{5}$$

with $\ell(\cdot)$ denoting the loss given by

$$\ell\left(\boldsymbol{y}(k), \widehat{\boldsymbol{y}}(k); \boldsymbol{w}\right) = \sum_{i=1}^{c} y_i(k) \log\left(1 + \sum_{j=1}^{c-1} \exp(\phi_j(\boldsymbol{x}(k)))\right) - \sum_{i=1}^{c-1} y_i(k)\phi_i(\boldsymbol{x}(k)). \tag{6}$$

The objective of training is minimization of the regularized cost function[2]

$$C(\boldsymbol{w}) = S_{\mathcal{T}}(\boldsymbol{w}) + R(\boldsymbol{w}, \boldsymbol{\kappa}) \tag{7}$$

where the regularization term $R(\boldsymbol{w}, \boldsymbol{\kappa})$ is parameterized by a set of regularization parameters $\boldsymbol{\kappa}$. Training provides the estimated weight vector $\widehat{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}} C(\boldsymbol{w})$ and is done using a Gauss-Newton scheme,

$$\boldsymbol{w}^{\text{new}} = \boldsymbol{w}^{\text{old}} - \eta \cdot \boldsymbol{J}^{-1}(\boldsymbol{w}^{\text{old}}) \boldsymbol{\nabla}(\boldsymbol{w}^{\text{old}}) \tag{8}$$

[2]This might be viewed as a maximum a posteriori (MAP) method.

where η is the step-size (line search parameter). For that purpose we require the gradient, $\boldsymbol{\nabla}(\boldsymbol{w}) = \partial C/\partial \boldsymbol{w}$, and the Hessian, $\boldsymbol{J}(\boldsymbol{w}) = \partial^2 C/\boldsymbol{w}\boldsymbol{w}^\top$ of the cost function given by,

$$\frac{\partial C}{\partial \boldsymbol{w}}(\boldsymbol{w}) = -\frac{1}{N_t}\sum_{k=1}^{N_t}\sum_{i=1}^{c-1}\left[y_i(k) - \widehat{y}_i(k)\right]\frac{\partial \phi_i(\boldsymbol{x}(k))}{\partial \boldsymbol{w}} + \frac{\partial R(\boldsymbol{w},\boldsymbol{\kappa})}{\partial \boldsymbol{w}}, \tag{9}$$

$$\boldsymbol{J}(\boldsymbol{w}) = \frac{1}{N_t}\sum_{k=1}^{N_t}\sum_{i=1}^{c-1}\sum_{j=1}^{c-1}\widehat{y}_i(k)\left[\delta_{ij} - \widehat{y}_j(k)\right]\frac{\partial \phi_i(\boldsymbol{x}(k))}{\partial \boldsymbol{w}}\frac{\partial \phi_j(\boldsymbol{x}(k))}{\partial \boldsymbol{w}^\top} + \frac{\partial^2 R(\boldsymbol{w},\boldsymbol{\kappa})}{\partial \boldsymbol{w}\partial \boldsymbol{w}^\top}. \tag{10}$$

Here δ_{ij} is the Kronecker delta and we have used the Gauss-Newton approximation to the Hessian.

ADAPTING REGULARIZATION PARAMETERS

The available data set, $\mathcal{D}$, of N examples is split into two disjoint sets: a validation set, $\mathcal{V}$, with $N_v = \lceil \gamma N \rceil$ 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 validation error of the trained network is given by

$$S_{\mathcal{V}}(\widehat{\boldsymbol{w}}) = \frac{1}{N_v}\sum_{k=1}^{N_v}\ell\left(\boldsymbol{y}(k), \widehat{\boldsymbol{y}}(k); \widehat{\boldsymbol{w}}\right) \tag{11}$$

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. the joint input-output distribution.

Aiming at adapting the regularization parameters $\boldsymbol{\kappa}$ so that the validation error is minimized we can apply the iterative scheme suggested in [9]:

$$\boldsymbol{\kappa}^{\mathrm{new}} = \boldsymbol{\kappa}^{\mathrm{old}} - \mu\frac{\partial S_{\mathcal{V}}}{\partial \boldsymbol{\kappa}}(\widehat{\boldsymbol{w}}(\boldsymbol{\kappa}^{\mathrm{old}})) \tag{12}$$

where μ is a step-size and $\widehat{\boldsymbol{w}}(\boldsymbol{\kappa}^{\mathrm{old}})$ is the estimated weight vector using the regularization parameter $\boldsymbol{\kappa}^{\mathrm{old}}$. Suppose the regularization term is linear in the regularization parameters, i.e.,

$$R(\boldsymbol{w},\boldsymbol{\kappa}) = \boldsymbol{\kappa}^\top \boldsymbol{r}(\boldsymbol{w}) = \sum_{i=1}^{q}\kappa_i r_i(\boldsymbol{w}) \tag{13}$$

where κ_i are the regularization parameters and $r_i(\boldsymbol{w})$ the associated regularization functions. The gradient of the validation error then equals [9]:

$$\frac{\partial S_{\mathcal{V}}}{\partial \boldsymbol{\kappa}}(\widehat{\boldsymbol{w}}) = -\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{w}^\top}(\widehat{\boldsymbol{w}}) \cdot \boldsymbol{J}^{-1}(\widehat{\boldsymbol{w}}) \cdot \frac{\partial S_{\mathcal{V}}}{\partial \boldsymbol{w}}(\widehat{\boldsymbol{w}}). \tag{14}$$

Consider the specific case of weight decay regularization with separate weight decays for input-to-hidden and hidden-to output layers, i.e.,

$$R(\boldsymbol{w}, \boldsymbol{\kappa}) = \kappa_I \cdot |\boldsymbol{w}^I|^2 + \kappa_H \cdot |\boldsymbol{w}^H|^2 \tag{15}$$

where $\boldsymbol{\kappa} = [\kappa_I, \kappa_H]^\top$ and $\boldsymbol{w} = [\boldsymbol{w}^I, \boldsymbol{w}^H]$ with $\dim(\boldsymbol{w}^I) = m_I$, $\dim(\boldsymbol{w}^H) = m_H$ and $\dim(\boldsymbol{w}) = m = m_I + m_H$.

The gradient then yields,

$$\frac{\partial S_{\mathcal{V}}}{\partial \kappa_I}(\widehat{\boldsymbol{w}}) = -2(\widehat{\boldsymbol{w}}^I)^\top \cdot \boldsymbol{g}_{m_I}, \quad \frac{\partial S_{\mathcal{V}}}{\partial \kappa_H}(\widehat{\boldsymbol{w}}) = -2(\widehat{\boldsymbol{w}}^H)^\top \cdot \boldsymbol{g}_{m_H} \tag{16}$$

where $\boldsymbol{g} = [\boldsymbol{g}_{m_I}, \boldsymbol{g}_{m_H}] = \boldsymbol{J}^{-1}(\widehat{\boldsymbol{w}}) \cdot \partial S_{\mathcal{V}}(\widehat{\boldsymbol{w}})/\partial \boldsymbol{w}$.

In summary the algorithm for adapting regularization parameters consists of the following 8 steps:

1. Choose the split ratio γ between training and validation set sizes.
2. Initialize $\boldsymbol{\kappa}$ and the weights of the network.
3. Train the network with fixed $\boldsymbol{\kappa}$ to achieve $\widehat{\boldsymbol{w}}(\boldsymbol{\kappa})$. Calculate the validation error $S_{\mathcal{V}}$.
4. Calculate the gradient $\partial S_{\mathcal{V}}/\partial \boldsymbol{\kappa}$ cf. Eq. (14). Initialize the step-size μ.
5. Update $\boldsymbol{\kappa}$ using Eq. (12).
6. Retrain the network from the previous weights and calculate the validation error $S_{\mathcal{V}}$.
7. If no decrease in validation error then perform a bisection of μ and goto step 5; otherwise, continue.
8. Repeat steps 4–7 until the relative change in validation error is below a small percentage or, e.g., the 2-norm of the gradient $\partial S_{\mathcal{V}}/\partial \boldsymbol{\kappa}$ is below a small number.

PRUNING

In order to reduce and optimize the network architecture we suggest to use a pruning scheme, e.g., Optimal Brain Damage (OBD) [11]. An alternative method is Optimal Brain Surgeon (OBS) [5]; however, in a series of experiments we noticed that extreme care is essential in order not to underestimate the saliencies [16]. Thus OBS is less robust than OBD.

OBD ranks the weights according to importance or *saliency*. Here we use the validation error based OBD proposed in [9]. The saliency for weight i is given by

$$\varrho_i = -\widehat{w}_i \frac{\partial S_{\mathcal{V}}}{\partial w_i}(\widehat{\boldsymbol{w}}) + \frac{1}{2}\widehat{w}_i^2 \frac{\partial^2 S_{\mathcal{V}}}{\partial w_i^2}(\widehat{\boldsymbol{w}}). \tag{17}$$

By repeatedly removing weights with small saliencies and retraining the resulting network, a nested family of network architectures is obtained. The validation error (or an alternative measure of generalization performance[3]) is then used for selecting the optimal architecture.

EXPERIMENTS

We test the performance of the adaptive regularization algorithm on a vowel classification problem. The data are based on the Peterson and Barney database [17]. The classes are vowel sounds characterized by the first four formant frequencies. 76 persons (33 male, 28 female and 15 children) have pronounced $c = 10$ different vowels (IY IH EH AE AH AA AO UH UW ER) two times. This results in a data base of totally 1520 examples. The database is the verified database described in [22] where all data[4] are used, including examples where utterance failed of unanimous identification in the listening test (26 listeners). All examples were included to make the task more difficult.

The examples were split into a data set, $\mathcal{D}$, consisting of $N = 760$ examples (16 male, 14 female and 8 children) and an independent test set of the remaining 760 examples. The regularization was adapted by splitting the data set $\mathcal{D}$ equally into a validation set of $N_v = 380$ examples and a training set of $N_t = 380$ examples (8 male, 7 female and 4 children in each set).

Suppose that the network weights are given by $\boldsymbol{w} = [\boldsymbol{w}^I, \boldsymbol{w}^I_{\text{bias}}, \boldsymbol{w}^H, \boldsymbol{w}^H_{\text{bias}}]$ where $\boldsymbol{w}^I$, $\boldsymbol{w}^H$ are input-to-hidden and hidden-to-output weights, respectively, and the bias weights are assembled in $\boldsymbol{w}^I_{\text{bias}}$ and $\boldsymbol{w}^H_{\text{bias}}$. In this example, we use the following weight decay regularization term:

$$R(\boldsymbol{w}, \boldsymbol{\kappa}) = \kappa^I \cdot |\boldsymbol{w}^I|^2 + \kappa^I_{\text{bias}} \cdot |\boldsymbol{w}^I_{\text{bias}}|^2 + \kappa^H \cdot |\boldsymbol{w}^H|^2 + \kappa^H_{\text{bias}} \cdot |\boldsymbol{w}^H_{\text{bias}}|^2. \quad (18)$$

where $\boldsymbol{\kappa} = [\kappa^I, \kappa^I_{\text{bias}}, \kappa^H, \kappa^H_{\text{bias}}]$. We further define the normalized weight decays as $\boldsymbol{\alpha} \equiv \boldsymbol{\kappa} \cdot N_t$. The simulation set-up was:

- Network: 4 inputs, 5 hidden neurons, 9 outputs[5].
- The training input data were normalized to zero mean and unit variance in order to facilitate training and weight initialization.
- Weights were initialized uniformly over $[-0.5, 0.5]$, regularization parameters were initialized at zero. 10 steps in a gradient descent training algorithm (see e.g., [12]) was performed and the weight decays, $\boldsymbol{\kappa}$, were re-initialized at $\lambda_{\max}/10^2$, where $\lambda_{\max}$ is the max. eigenvalue of the Hessian matrix of the cost function. This initialization scheme is motivated by the following observations:
  - Weight decays should be so small that they do not reduce the approximation capabilities of the network significantly.

[3]E.g., the previously suggested algebraic estimate [8], [15].

[4]The database can be retrieved from `ftp://eivind.imm.dtu.dk/dist/data/vowel/PetersonBarney.tar.Z`

[5]We only need 9 outputs since the posterior class probability of the 10th class is given by $1 - \sum_{j=1}^{9} p(\mathcal{C}_j|\boldsymbol{x})$.

	NNet	KNN ($K=9$)
Training	0.105 ±0.008	0.150
Validation	0.115 ± 0.005	0.158
Test	0.122 ± 0.005	0.199
Test after retrain.	0.119 ± 0.003	0.153

Table 1: Probability of misclassification, *pmc*. For the neural network the averages and standard deviations over 6 runs are reported.

  – They should be so large that the algorithm is prevented from being trapped in a local optimum and numerical instabilities are eliminated.

- Training is now done using a Gauss-Newton algorithm (see e.g., [12]). The Hessian is inverted using the Moore-Penrose pseudo inverse (see e.g., [19]) ensuring that the eigenvalue spread[6] is less than 10^8.
- The regularization step-size η is initialized at 1.
- When the adaptive regularization scheme has terminated we prune 3% of the weights using a validation set based version of the Optimal Brain Damage recipe [9], [11].
- We alternate between pruning and adaptive regularization until the validation error has reached a minimum.
- Finally, remaining weights are retrained on all data using the optimized weight decay parameters.

Table 1 reports the average and standard deviations of the probability of misclassification (*pmc*) over 6 runs for pruned networks using the optimal regularization parameters. Note that retraining on the full data set decreases the test *pmc* slightly on the average; improvement was found in 4 out of 6 runs. For comparison we used a K-nearest-neighbor (KNN) classification, see e.g., [1] and found that $K = 9$ was optimal on the validation set. Note that the neural network performed significantly better. Contrasting the obtained results to other work is difficult. In [20] results on the Peterson-Barney vowel problem are reported, but their data are not exactly the same; only the first 2 formant frequencies were used. Furthermore, different test sets have been used for the different methods presented. The best result reported [13] is obtained by using KNN and reach $pmc = 0.186$ which is somewhat higher than our results.

In Fig. 2 the evolution of the adaptive regularization as well as the pruning algorithm is demonstrated.

CONCLUSIONS

This paper presented a framework for design of neural classifiers which include architecture optimization by pruning and adaptation of regularization

[6]Eigenvalue spread should not be larger than the square root of the machine precision [3].

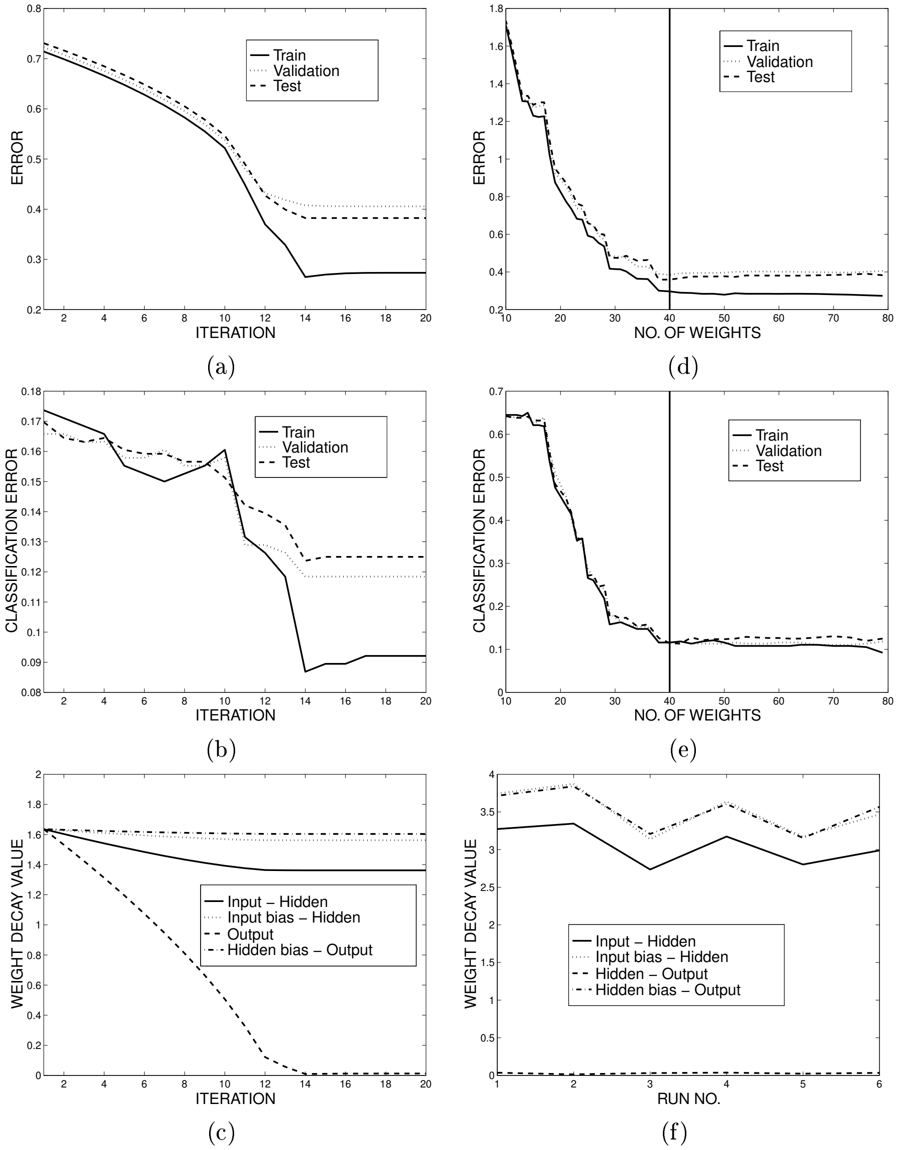

Figure 2: Panels (a), (b) and (c) show the evolution of the adaptive regularization algorithm in a typical run. Optimal weight decays are found by minimizing the validation error in (a). Note that also the test errors decreases. This tendency is also evident in (b) displaying pmc even though a small increase is noticed. In (c) the normalized weight decays, $\alpha = \kappa \cdot N_t$, are depicted. (d) and (e) show the evolution of errors and pmc during pruning. The optimal network having minimal validation error is indicated by the vertical line. There is only a marginal effect of pruning. Finally, the variation of the optimal normalized weight decays (before pruning) in different runs is shown in (f) and is seen to be relatively small.

parameters. Moreover, an improved neural net architecture was presented. Numerical examples 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 (CONNECT). JL furthermore acknowledge the Radio Parts Foundation for financial support. Allan René Rasmussen is thanked for programming assistance.

REFERENCES

[1] C.M. Bishop: **Neural Networks for Pattern Recognition**, Oxford, UK: Oxford University Press, 1995.

[2] J.S. Bridle: "Probabilistic Interpretation of Feedforward Classification Network Outputs with Relationships to Statistical Pattern Recognition," **Neurocomputing - Algorithms, Architectures and Applications**, Berlin Germany: Springer-Verlag, vol. 6, pp. 227–236, 1990.

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

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

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

[6] K. Hornik: "Approximation Capabilities of Multilayer Feedforward Networks," **Neural Networks**, vol. 4, pp. 251–257, 1991.

[7] M. Hintz-Madsen, L.K. Hansen, J. Larsen, E. Olesen & K.T. Drzewiecki: "Design and Evaluation of Neural Classifiers - Application to Skin Lesion Classification," in F. Girosi, J. Makhoul, E. Manolakos & E. Wilson (eds.), **Proceedings of the IEEE Workshop on Neural Networks for Signal Processing V**, Piscataway, New Jersey: IEEE, 1995, pp. 484–493.

[8] M. Hintz-Madsen, M. With Pedersen, L.K. Hansen, & J. Larsen: "Design and Evaluation of Neural Classifiers," in S. Usui, Y. Tohkura, S. Katagiri & E. Wilson (eds.), **Proceedings of the IEEE Workshop on Neural Networks for Signal Processing VI**, Piscataway, New Jersey: IEEE, 1996, pp. 223–232.

[9] 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 VI, Piscataway, New Jersey: IEEE, 1996, pp. 62–71.

[10] J. Larsen, C. Svarer, L. Nonboe Andersen & L.K. Hansen: "Adaptive Regularization in Neural Network Modelling," preprint IMM, DTU, submitted 1997. Available by `ftp://eivind.imm.dtu.dk/dist/1997/larsen.bot.ps.Z`.

[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] D. Lowe: "Adaptive Radial Basis Function Nonlinearities and the Problem of Generalisation," **Proceedings of IEE Conference on Artificial Neural Networks**, 1989, pp. 171–175.

[14] D.J.C. MacKay: "A Practical Bayesian Framework for Backprop Networks," **Neural Computation**, vol. 4, no. 3, pp. 448–472, 1992.

[15] N. Murata, S. Yoshizawaand & 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.

[16] M.W. Pedersen, L.K. Hansen & J. Larsen: "Pruning with Generalization Based Weight Saliencies: γOBD, γOBS," in D.S. Touretzky, M.C. Mozer & M.E. Hasselmo (eds.) **Advances in Neural Information Processing Systems 8, Proceedings of the 1995 Conference**, Cambridge, Massachusetts: MIT Press, pp. 521–528, 1996.

[17] G.E. Peterson & H.L. Barney: "Control Methods Used in a Study of the Wowels," **JASA** vol. 24, pp. 175–184, 1952.

[18] B.D. Ripley: **Pattern Recognition and Neural Networks**, Cambridge, UK: Cambridge University Press, 1996.

[19] G.A.F. Seber & C.J. Wild: **Nonlinear Regression**, New York, New York: John Wiley & Sons, 1989.

[20] R.S. Shadafan & M. Niranjan: "A Dynamic Neural Network Architecture by Sequential Partitioning of the Input Space," **Neural Computation**, vol. 6, no. 6, pp. 1202–1222, 1994.

[21] C. Svarer, L.K. Hansen & J. Larsen: "On Design and Evaluation of Tapped-Delay Neural Architectures," in **Proceedings of the IEEE International Conference on Neural Networks**, San Francisco, California, USA, vol. 1, pp. 46–51, 1993.

[22] R.L. Watrous: "Current Status of PetersonBarney Vowel Formant Data," **JASA**, vol. 89 pp. 2459–2460, 1991.