Neural Classifier Construction Using Regularization, Pruning and Test Error Estimation

Mads Hintz-Madsen, Lars Kai Hansen, Jan Larsen,
Morten With Pedersen, and Michael Larsen
CONNECT, Dept. of Mathematical Modelling, Build. 321,
Technical University of Denmark, DK-2800 Lyngby, Denmark,
Phone: (+45) 4525 3885, Fax: (+45) 4587 2599,
Email: mhm, lkhansen, jl@imm.dtu.dk[1]

[1]This research is supported by the Danish Research Councils for the Natural and Technical Sciences through the Danish Computational Neural Network Center. Jan Larsen thanks the Radio-Parts Foundation for financial support.

Abstract

In this paper we propose a method for construction of feed-forward neural classifiers based on regularization and adaptive architectures. Using a penalized maximum likelihood scheme, we derive a modified form of the entropic error measure and an algebraic estimate of the test error. In conjunction with Optimal Brain Damage pruning, a test error estimate is used to select the network architecture. The scheme is evaluated on four classification problems.

Keywords: Neural classifiers, Architecture optimization, Regularization, Generalization estimation.

1 INTRODUCTION

Pattern recognition is an important aspect of most scientific fields and indeed the objective of most neural network applications. Some of the classic applications of neural networks like Sejnowski and Rosenbergs "NetTalk" concern classification of patterns into a finite number of categories. In modern approaches to pattern recognition the objective is to produce class probabilities for a given pattern. Using Bayes decision theory, the "hard" classifier selects the class with the highest class probability, hence, minimizing the probability of error. If different costs are associated with the individual classes, a risk-based approach can be adopted (Bishop, 1995),(Ripley, 1996). The conventional approach to pattern recognition is statistical and concerns the modeling of stationary class-conditional probability distributions by a certain set of basis functions, e.g., Parzen windows or Gaussian mixtures (Duda & Hart, 1973),(Bishop, 1995),(Ripley, 1996).

In this paper we define and analyze a system for construction and evaluation of feed-forward neural classifiers based on regularization and adaptive architectures. The proposed scheme is a generalization of the approach we have suggested for time series processing (Svarer, Hansen, & Larsen, 1993; Svarer, Hansen, Larsen, & Rasmussen, 1993) and for binary classification in the context of a medical application (Hintz-Madsen, Hansen, Larsen, Olesen, & Drzewiecki, 1995). The key concept of the new methodology for optimization of neural classifiers is an asymptotic estimate of the test error of the classifier providing an algebraic expression in terms of the training error and a model complexity estimate. Our approach is a penalized maximum likelihood scheme. The likelihood is formulated using a simple stationary noise model of the pattern source. For any given input pattern there can be defined a probability distribution over a fixed finite set of classes. The training set involves simply labeled data, i.e., each input vector is associated with a single class label. The task of the network is to estimate the relative frequencies of class labels for a given pattern. In conjunction with SoftMax normalization of the outputs of a standard, computationally universal, feed-forward network we recover a slightly modified form of the so-called entropic error measure (Bridle, 1990). For a fixed architecture the neural network weights are estimated using a Gauss-Newton scheme (Seber & Wild, 1995), while the model architecture is optimized using Optimal Brain Damage (Cun, Denker, & Solla, 1990). The problem of proper selection of regularization parameters is also briefly discussed, see also (Larsen, L.K. Hansen, Svarer, & Ohlsson, 1996).

While most of the components of our approach have been described in brief conference papers, we have here aimed at a complete account of the computational aspects as well as thorough tests on practical examples.

2 NEURAL CLASSIFIERS

Assume we have a training set, D, consisting of q input-output pairs

$$D = \{(\mathbf{x}^\mu, y^\mu) | \mu = 1, ..., q\} \tag{1}$$

where $\mathbf{x}$ is an input vector consisting of n_I elements and y is the corresponding class label. In this presentation we will assume that the class label is of the definite form $y = 1, ..., n_O$, with n_O being the number of classes. An alternative soft target assignment might be relevant in some practical contexts where the target could be, e.g., an estimate of class probabilities for the given input (Ripley, 1996).

We aim to model the posterior probability distribution

$$p(y = i|\mathbf{x}), \qquad i = 1, \ldots, n_O. \tag{2}$$

In some applications it might be desirable to use a rejection threshold when classifying, that is if all of the posterior probabilities fall below this threshold then no classification decision is made, see e.g., (Duda & Hart, 1973), (Hintz-Madsen et al., 1995).

To represent these distributions we choose the following feed-forward network architecture:

$$h_j(\mathbf{x}^\mu) = \tanh\left(\sum_{k=1}^{n_I} w_{jk} x_k^\mu + w_{j0}\right) \tag{3}$$

$$\phi_i(\mathbf{x}^\mu) = \sum_{j=1}^{n_H} W_{ij} h_j(\mathbf{x}^\mu) + W_{i0} \tag{4}$$

with n_I input units, n_H hidden units, n_O output units, and parameters $\mathbf{u} = (\mathbf{w}, \mathbf{W})$, where w_{j0} and W_{i0} are thresholds. To ensure that the outputs can be interpreted as probabilities, we use the normalized exponential transformation known as SoftMax (Bridle, 1990):

$$\hat{p}(y^\mu = i|\mathbf{x}^\mu) \equiv \frac{\exp[\phi_i(\mathbf{x}^\mu)]}{\sum_{i'=1}^{n_O} \exp[\phi_{i'}(\mathbf{x}^\mu)]} \tag{5}$$

where $\hat{p}(y^\mu = i|\mathbf{x}^\mu)$ is the estimated probability, that $\mathbf{x}^\mu$ belongs to class i. Numerical aspects of equation (5) are discussed in appendix A.

Assuming that the training data are drawn independently, the likelihood of the model can be expressed as

$$P(D|\mathbf{u}) = \prod_{\mu=1}^{q} \prod_{i=1}^{n_O} \hat{p}(y^\mu = i|\mathbf{x}^\mu)^{\delta_{i,y^\mu}} \tag{6}$$

where the Kronecker delta is defined by: $\delta_{i,y^\mu} = 1$ if $i = y^\mu$, otherwise $\delta_{i,y^\mu} = 0$.

Training is based on minimization of the negative normalized log-likelihood

$$E(\mathbf{u}) = -\frac{1}{q} \log P(D|\mathbf{u}) = \frac{1}{q} \sum_{\mu=1}^{q} \epsilon(\mathbf{x}^\mu, y^\mu, \mathbf{u}) \tag{7}$$

where

$$\epsilon(\mathbf{x}^\mu, y^\mu, \mathbf{u}) = -\sum_{i=1}^{n_O} \delta_{i,y^\mu} \left[\phi_i(\mathbf{x}^\mu) - \log\left(\sum_{i'=1}^{n_O} \exp[\phi_{i'}(\mathbf{x}^\mu)] \right) \right]. \tag{8}$$

Numerical aspects of equation (8) are discussed in appendix A.

In order to eliminate overfitting and ensure numerical stability, we augment the cost function by a regularization term, e.g., a simple weight decay, to form a penalized log-likelihood,

$$C(\mathbf{u}) = E(\mathbf{u}) + \frac{1}{2}\mathbf{u}^{\mathrm{T}}\mathbf{R}\mathbf{u} \tag{9}$$

where $\mathbf{R}$ is a positive definite matrix. In this paper we consider a diagonal matrix with elements $2\alpha_j \delta_{j,k}/q$.

The gradient of (7) is

$$\frac{\partial E(\mathbf{u})}{\partial u_j} = -\frac{1}{q} \sum_{\mu=1}^{q} \sum_{i=1}^{n_O} \left[\delta_{i,y^\mu} - \hat{p}(y^\mu = i|\mathbf{x}^\mu)\right] \frac{\partial \phi_i(\mathbf{x}^\mu)}{\partial u_j}. \tag{10}$$

See appendix B for details.

The matrix of second derivatives (the Hessian) can be expressed as

$$H_{jk} \equiv \frac{\partial^2 E(\mathbf{u})}{\partial u_j \partial u_k} \approx \frac{1}{q} \sum_{\mu=1}^{q} \sum_{i=1}^{n_O} \sum_{i'=1}^{n_O} \hat{p}(y^\mu = i|\mathbf{x}^\mu) \left[\delta_{i,i'} - \hat{p}(y^\mu = i'|\mathbf{x}^\mu)\right] \frac{\partial \phi_{i'}(\mathbf{x}^\mu)}{\partial u_k} \frac{\partial \phi_i(\mathbf{x}^\mu)}{\partial u_j} \tag{11}$$

where we have used a Gauss-Newton like approximation. See appendix B for details.

It is important to notice that the Hessian in (11) is singular everywhere for the SoftMax network. This is due to the redundant output representation (5) which leaves the set of outputs invariant to certain linear transformations of the hidden-to-output weights. The use of regularization, however, ensures that the effects of this symmetry don't interfere with training or evaluation of the network. We are currently working on a modified implementation which explicitly removes the SoftMax redundancy (Andersen, Larsen, Hansen, & Hintz-Madsen, 1997).

Using matrix/vector notation the Gauss-Newton paradigm of updating the weights can now be computed as (Seber & Wild, 1995)

$$\mathbf{u}^{\mathrm{new}} = \mathbf{u} - \eta \left(\mathbf{H} + \mathbf{R}\right)^{-1} \left[\frac{\partial E}{\partial \mathbf{u}} + \mathbf{R}\mathbf{u}\right] \tag{12}$$

where $\mathbf{Ru}$ and $\mathbf{R}$ are the first and second derivatives of the regularization term, respectively, and η is a step-size, that may be used to ensure a decrease in the cost function, e.g., by line search.

The determination of regularization parameters is an issue of ongoing research. A natural approach is to minimize the test error with respect to the regularization parameters. Here one may use an estimate of the test error, as derived in the next section. This technique is demonstrated in section 3.1. Another approach which is based on minimization of a validation error has been implemented in (Larsen et al., 1996).

2.1 Test Error Estimate

One of the main objectives in our approach is to estimate a network model with a high generalization ability. In order to obtain this we need an estimate of the generalization ability of a model. The generalization, or test error, for a given network $\mathbf{u}$ may be defined as

$$E_{\text{test}}(\mathbf{u}) = \int \epsilon(\mathbf{x}, y, \mathbf{u}) P(\mathbf{x}, y)\, d\mathbf{x} dy \tag{13}$$

where $P(\mathbf{x}, y)$ is the true underlying distribution of examples and $\epsilon(\mathbf{x}, y, \mathbf{u})$ is the error on example $(\mathbf{x}, y)$. Since the test error involves an average over all possible examples, it is in general not accessible, but it can be estimated by using additional statistical assumptions, thus giving us the following estimate for the average test error of a network $\mathbf{u}$ estimated on a training set D (Murata, Yoshizawa, & Amari, 1994),

$$\widehat{\langle E_{\text{test}} \rangle} = E_{\text{train}}(\mathbf{u}(D)) + \frac{N_{\text{eff}}}{q} \tag{14}$$

where $E_{\text{train}}(\mathbf{u}(D))$ is the training error of the model. The effective number of parameters is given by $N_{\text{eff}} = \text{Tr}[\mathbf{H}(\mathbf{H}+\mathbf{R})^{-1}]$, where $\mathbf{R}$ is the second derivative of the regularization term.

In brief, the following assumptions enter the derivation of (14):

- Independence of input and error on output.
- Many examples per weight: $N_{\text{eff}}/q \to 0$.
- There exists a network, $\mathbf{u}^*$, that implements the true model.

For a detailed discussion on test error estimates and their assumptions, see, e.g., (Larsen, 1992), (Larsen, 1993). This estimate of the test error averaged over all possible training sets may be used to select the optimal network e.g., among a nested family of pruned networks; hence, be used as a pruning stop criterion similarly to our procedure for evaluation of function approximation networks (Svarer et al., 1993, 1993).

2.2 Pruning with Optimal Brain Damage

In order to reduce and optimize a networks architecture, we recommend to apply a pruning scheme such as *Optimal Brain Damage* (OBD) (Cun et al., 1990). The aim of OBD is to estimate the importance of the weights for the training error and rank the weights according to their importance. If the importance is estimated using a second order expansion of the training error around its minimum, the *saliency* for a weight u_i is (Svarer et al., 1993)

$$s_i = \left(R_{ii} + \frac{1}{2} H_{ii} \right) u_i^2 \tag{15}$$

where the Hessian H_{ii} is given by (11) and R_{ii} is the i'th diagonal element of $\mathbf{R}$.

The following assumptions enter the derivation of OBD:

- The training error is at a minimum.
- The terms of third and higher orders in the deleted weights can be neglected.

- The off-diagonal terms in the Hessian, $\frac{\partial^2 E(\mathbf{u})}{\partial u_j \partial u_k}$, can be neglected (if more than one weight is pruned).

By repeatedly removing weights with the smallest saliencies and retraining the resulting network, a nested family of networks is obtained. Here we may use the previously derived test error estimate to select the "optimal" network.

2.3 Recipe Overview

The algorithm can be summarized by the following:

1. Determine regularization parameters (e.g. by using the grid-sampling technique described below in section 3.1).
2. Train/retrain network using Gauss-Newton optimization.
3. Compute the estimated test error.
4. Compute OBD saliencies and remove a percentage of the weights with the smallest saliency. Goto 2, if # of remaining weights > 0.
5. Select the network with the smallest estimated test error as the optimal network.

2.4 Comments on Algorithm Complexity

When choosing an algorithm for solving a particular problem, it is necessary not only to ensure that the algorithm is theoretically well founded, but it should also be applicable in practice. One limiting factor is the available computational resources. Though a diminishing problem, it still needs consideration. In this section we'll briefly discuss the complexity of the proposed algorithm.

For each training iteration it is necessary to compute the regularized Hessian and its inverse. Computing the Hessian is an $O(q{n_O}^2 N^2)$ operation, where q is the number of training examples, n_O the number of classes and N the number of weights, while inverting the regularized Hessian is an $O(N^3)$ operation. Since the number of training examples is usually larger than the number of weights, it is the computation of the Hessian that can be a limiting factor.

As a guiding principle, this algorithm will on a computer with performance equivalent to a Pentium 200 MHz machine produce results in hours when dealing with network configurations of hundreds of weights, while using thousands of weights is less feasible and will take considerable longer.

Note, that after each pruning step with OBD, it is usually only necessary to retrain the network for just a few iterations. Because only weights with small saliencies are removed, the minimum of the cost function is only slightly changed.

3 EXPERIMENTS

The proposed methodology for constructing neural classifiers has been evaluated on several problems: the artificial *contiguity problem*, the real world problems of *glass classification*, *bacteria cell classification*, and *skin lesion classification*.

3.1 The Contiguity Problem

The contiguity problem has been used for evaluating optimization schemes, see e.g., (Denker et al., 1987), (Gorodkin, Hansen, Krogh, Svarer, & Winther, 1993). The boolean input vector (± 1) is interpreted as a one-dimensional image and connected clumps of $+1$'s are counted. Two classes are defined: those with two and three clumps. We consider the case, where $n_I = 10$. In this case there are 792 legal input patterns consisting of 432 patterns with three clumps and 360 with two clumps. We use a randomly

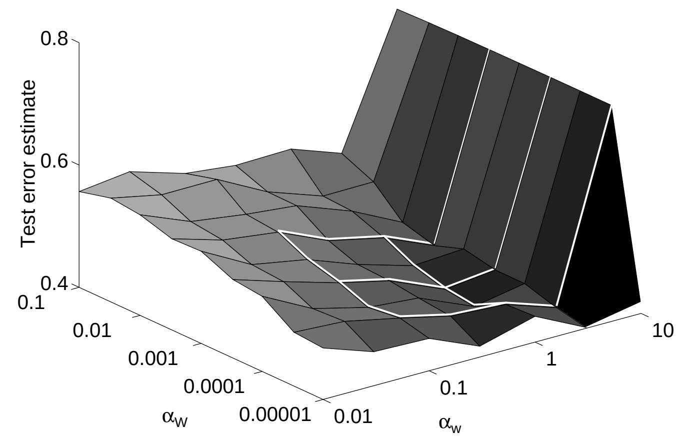

Figure 1: The estimated test error for the *contiguity problem* as function of the weight decay parameters. The grid indicates the points where the estimated test error is sampled.

selected training set with 150 patterns and a test set with 510 patterns both containing an even split of the two classes.

Initially a network architecture consisting of 10 input units, 8 hidden units and 2 output units was chosen. We only employ two different weight decays: α_w for the input-to-hidden weights and α_W for the hidden-to-output weights. By sampling the space spanned by α_w and α_W for non-pruned networks[1] with e.g., a 3×3 grid and computing the estimated test error , it is possible to fit e.g., a diagonal quadratic form[2] in a least-square sense to the sample points, locate the minimum[3] of the quadratic form and use the weight decays found for the design of the network. This is shown in figure 1 and 2. In order to cover a large range of values for the regularization parameters, it's appropriate to use a logarithmic scale for the sampling grid. The values for α_w and α_W should be chosen large enough to ensure numerical stability in equation 12, yet small enough in order not to impose too large a restriction on the number of degrees of freedom. This method is a quick and dirty way to determine the approximate order of magnitude for the regularization parameters and works best when the estimated test error surface is close to being convex. This is typically the case due to the nature of the test error estimate.

Next ten fully connected networks were trained[4] using the estimated weight decay parameters, subsequently pruned using the OBD saliency ranking, removing one weight per iteration. In figure 3 and 4 the distribution of the individual test errors is shown for fully connected networks and pruned networks, respectively. The error distribution shows that the mean error is predominantly driven by a few examples with a high error, thus suggesting that one should monitor the median[5] error as well in order to get a good indication of a network's performance. This problem arises due to the nature of the logarithm in the error function and one should be aware of this property when evaluating the performance.

Seven of the ten pruned networks had a classification[6] error on the test set between 0% and 3.3%, while three networks had an error of $16-19\%$. In (Gorodkin et al., 1993) seven of ten networks had an error of $8-38\%$ using the same size of training set, while three networks had errors around 0%. Compared with these results, our classifier design scheme has a significantly higher yield.

[1]To reduce the computational burden.
[2]Diagonal quadratic form: $(z-z_0)=(x-x_0)^2/a^2+(y-y_0)^2/b^2$.
[3]In case the minimum is located outside the sample-grid, one should relocate the grid and find a new minimum.
[4]Training was stopped when the 2-norm of the gradient vector was below 10^{-5}.
[5]$E_{median}(\mathbf{u}) = \text{median}\,\{\epsilon(\mathbf{x}^\mu, y^\mu, \mathbf{u})|\mu=1,...,q\}$, where ϵ is the error measure defined by equation (8).
[6]Following Bayes decision theory, the network output with the highest probability determines the class label.

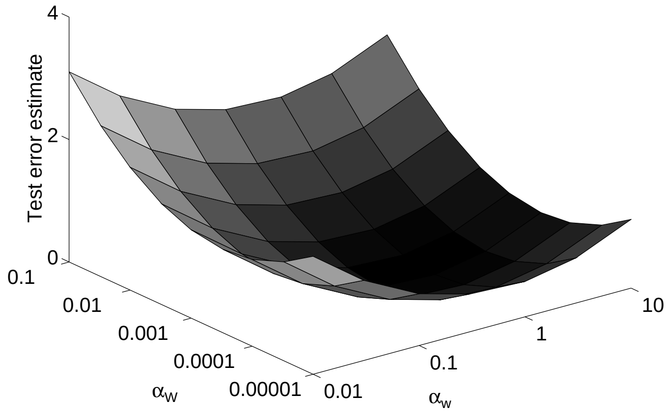

Figure 2: Quadratic form fitted to the 3×3 grid for the *contiguity problem* shown in figure 1. Minimum located at $(\alpha_w, \alpha_W) = (0.68, 2.8 \cdot 10^{-4})$.

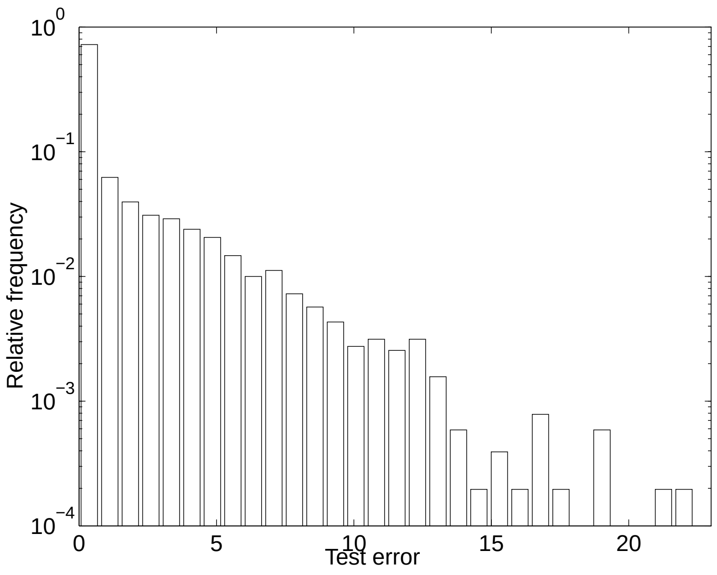

Figure 3: The distribution of the errors of the individual test examples for 10 fully connected *contiguity* networks combined in one pool. Notice the "long tail" of the distribution resulting in a high mean error (0.94) and a small median error (0.022) i.e., the mean is predominantly driven by a few examples with high error.

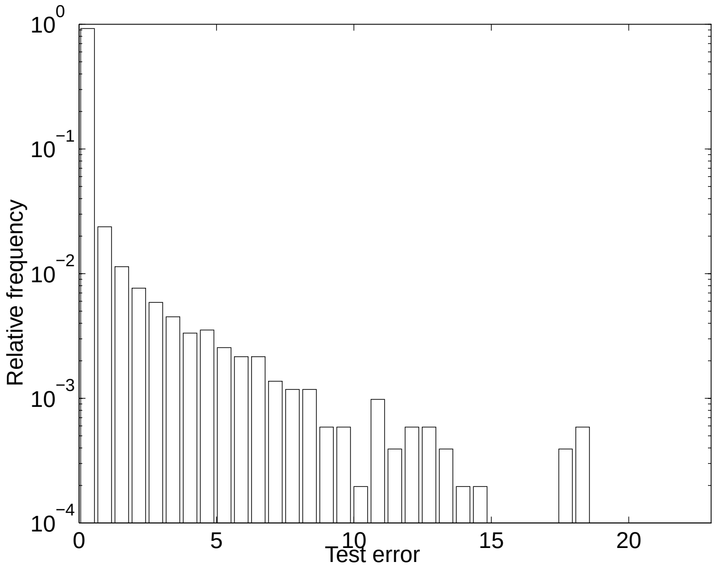

Figure 4: The distribution of the errors of the individual test examples for 10 pruned *contiguity* networks selected by the minimum of the estimated test error combined in one pool. The mean error is 0.37 and the median error is 0.0014 showing a significant performance improvement as result of pruning compared to the fully connected networks in figure 3.

3.2 The Glass Classification Problem

The real world glass classification problem is a part of the Proben1 neural network benchmark collection (Prechelt, 1994). The task is to classify glass splinters into six classes. The glass splinters have been chemical analyzed and nine different measures have been extracted from the analysis, see (Prechelt, 1994) for details. The original dataset (*glass1*) consists of 214 examples divided into a training set (107), a validation set (54) and a test set (53). Since our approach doesn't require a validation set, we have used two different training scenarios: one using the original training set and one using a new training set consisting of the original training and validation set. The initial network architecture chosen consisted of 9 input units, 6 hidden units and 6 output units. We estimated the regularization parameters using the sample-grid technique and the small training set. The parameters were found to be $\alpha_w = 2.2 \cdot 10^{-2}$ and $\alpha_W = 4.7 \cdot 10^{-4}$.

In figure 5-6 and figure 7-8 we show the pruning results of networks trained with the *small* and *large* training set, respectively, using the estimated regularization parameters. The "optimal" network found with the *small* training set had a classification error of 32% on the test set, while the "optimal" network found with the *large* training set had an error of 28%. In (Prechelt, 1994) Prechelt reports a test error of 32% for a fixed network architecture using the *small* training set. The validation set is used to stop training, thus he effectively uses both the training and validation set for training (Sjöberg, 1995). Our approach using the estimated test error for model selection eliminates the need for a validation set, thus allowing us to use more data for the actual training resulting in a better generalization performance. The problem of comparing the performance of neural network models is addressed in (Larsen & Hansen, 1995).

For comparison a standard *k-Nearest-Neighbor*[7] (k-N-N) classification (Duda & Hart, 1973) was performed using the *large* training set. The training error may be computed from the training set by including each training pattern in the majority vote. A *leave-one-out* "validation" error on the training set may be computed by excluding each training pattern from the vote. Finally, the test patterns may be classified by voting among the k nearest neighbors found among the training patterns. Using the *leave-one-out* validation error we found that $k = 2$ was optimal for this data set. The 2-N-N scheme

[7]Within k-N-N a pattern is classified according to a majority vote among its k nearest neighbors using the simple Euclidean metric.

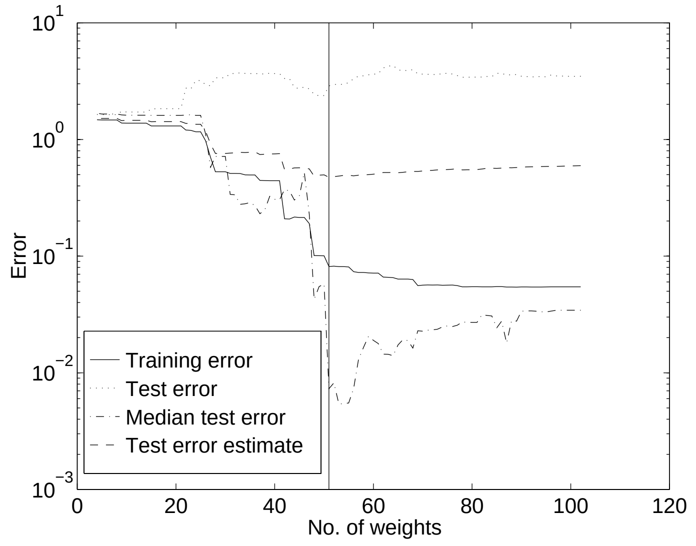

Figure 5: Pruning of a *glass classification* network using the *small* training set. The vertical line indicates the "optimal" network selected by the minimum of the estimated test error. Note that the test error is very high and evolves quite differently from the classification error on the test set shown in figure 6. The development of the median test error is more similar to the development of the classification error on the test set.

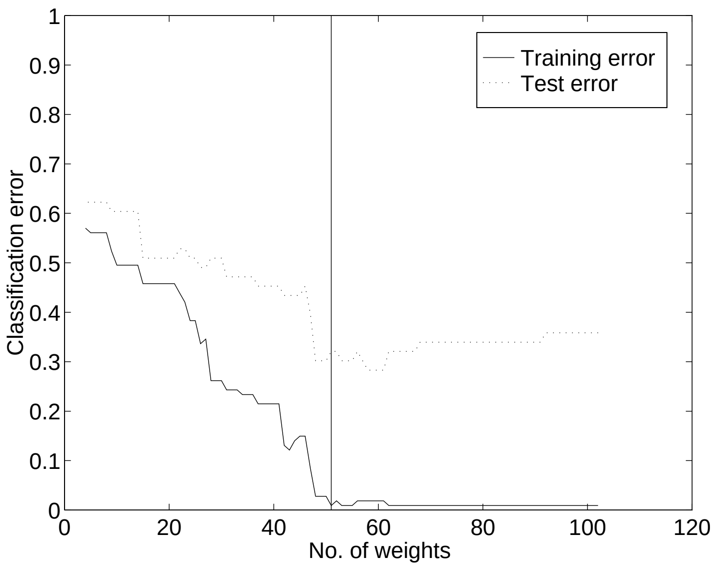

Figure 6: The classification error during pruning of a *glass classification* network using the *small* training set. The vertical line indicates the "optimal" network selected by the minimum of the estimated test error shown in figure 5.

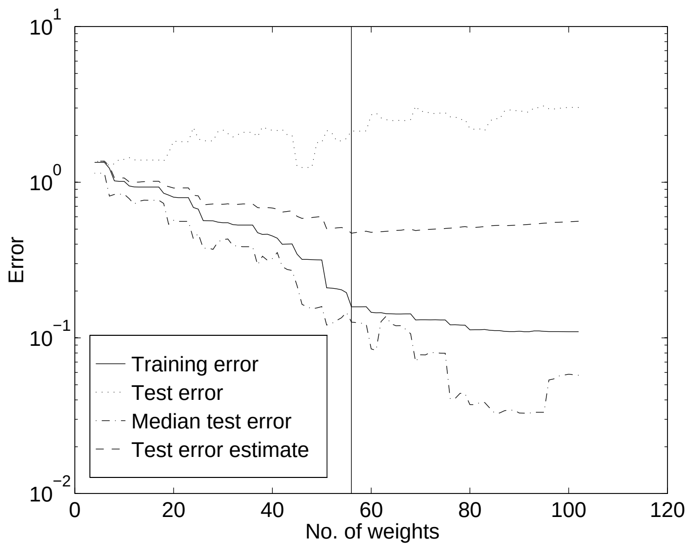

Figure 7: Pruning of a *glass classification* network using the *large* training set. The vertical line indicates the "optimal" network selected by the estimated test error.

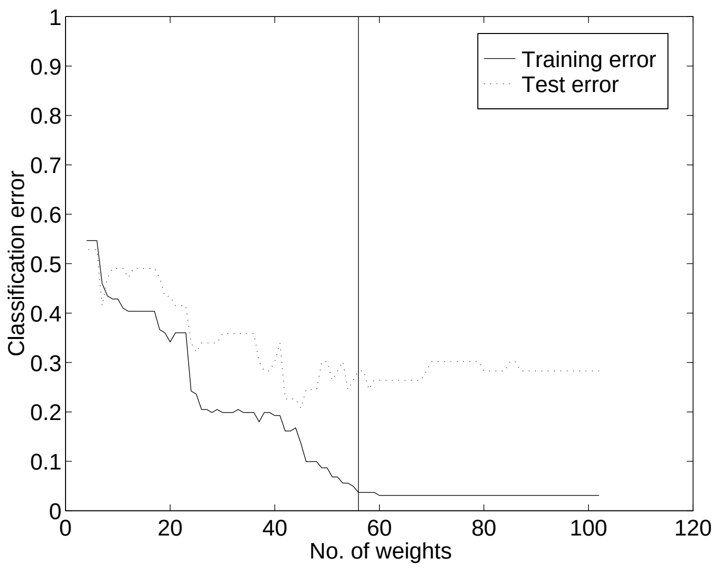

Figure 8: The classification error during pruning of a *glass classification* network using the *large* training set. The vertical line indicates the "optimal" network selected by the estimated test error shown in figure 7. Notice the overall lower classification error on the test set compared to figure 6.

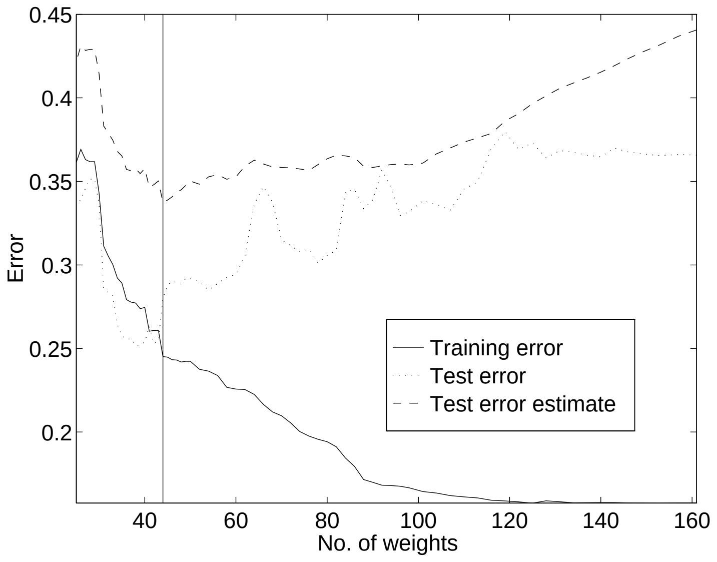

Figure 9: Pruning of a *bacteria cell* network. The vertical line indicates the "optimal" network selected by the minimum of the estimated test error.

had a classification error of 34% on the test set. Thus the performance of the optimized k-N-N scheme cannot match Prechelt's or our networks.

3.3 The Bacteria Cell Classification Problem

In order to evaluate the quality of water in e.g. oceans and lakes, it is desirable to know the type and extent of bacteria cells in water samples. Here we address the problem of classifying cells in microscopic images into five different morphological classes. 398 cells have been detected and their shapes are described by 10 complex *Fourier descriptors* (Granlund, 1972). The dataset is divided into a training set (320) and a test set (78).

Initially a network architecture consisting of 20 input units, 6 hidden units and 5 output units was chosen. The regularization parameters were after preliminary experiments set to $\alpha_w = 0.5$ and $\alpha_W = 0.5$.

In figure 9 a typical pruning scenario for a network is shown. Before pruning this particular network classified 96.9% of the training set and 85.9% of the test set correctly, while the "optimal" pruned network in figure 10 had a classification rate of 92.5% on the training set and 92.3% on the test set. Thus the pruning has increased the generalization ability of the network.

A k-N-N classification was performed for comparison. Using the leave-one-out validation error as described in section 3.2 we found that $k = 3$ was optimal for this data set. The 3-N-N scheme classified 91.0% of the test set correctly. Thus for this data set the performance of the neural and k-N-N classifier is similar.

3.4 The Skin Lesion Classification Problem

The incidence of malignant melanoma, the most lethal of skin cancers, has risen rapidly during the last 50 years. Fortunately patients can be saved from this life-threatening cancer, if it is detected at an early stage. Thus, in recent years, there has been an increased interest in schemes for automatic and early detection of melanoma. Digital imaging may assist and improve the possibility of such early detections. A review of digital imaging in this field was recently published in "Skin Research and Technology" (Stoecker, Moss, Ercal, & Umbaugh, 1995).

From a collection of color photographs of skin tumors at The National University Hospital of Denmark 21 statistical measurements describing color and texture properties have been acquired for each tumor and are used for classification into three groups: Benign nevi (non-cancer), dysplastic nevi (non-cancer, but increased risk of developing cancer) and melanoma.

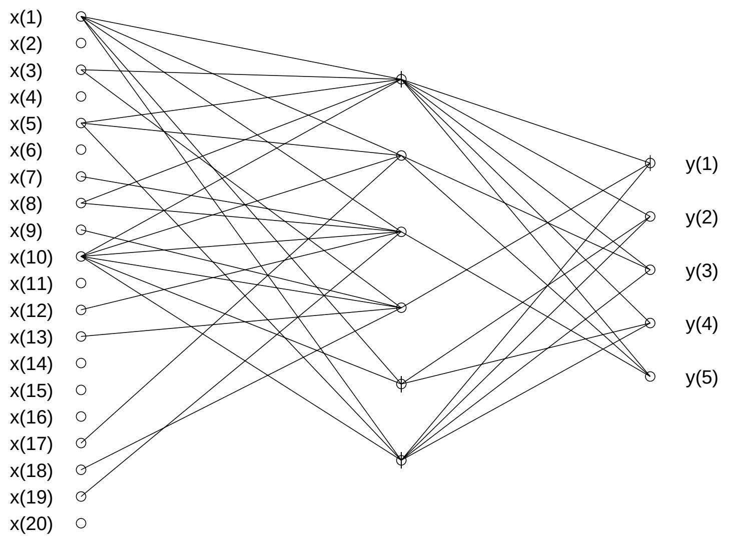

Figure 10: The "optimal" pruned *bacteria cell* network selected by the minimum estimated test error in figure 9. Note that a large number of inputs are not used. A vertical line thorugh a node indicates that a threshold is present.

Table 1: Confusion matrix for the test set using fully connected (103 weights) and pruned networks (41-62 weights). The mean and standard deviation of ten runs are reported. Note that the classifier performs best for the critical melanoma class. † indicates the estimated output classes.

	Fully connected ANN			**Pruned ANN**		
Conf. mat.	Benign nevi	Dyspl. nevi	Melanoma	Benign nevi	Dyspl. nevi	Melanoma
Benign nevi†	42.4±9.1%	29.0±6.2%	22.0±10.1%	47.6±9.3%	27.5±7.6%	18.5±8.8%
Dyspl. nevi†	24.3±5.2%	49.5±6.9%	5.5±3.7%	22.4±6.0%	53.5±5.8%	5.5±4.4%
Melanoma†	33.3±6.7%	21.5±6.7%	72.5±7.6%	30.0±8.4%	19.0±3.9%	76.0±7.8%

A total of 180 images with an even split of the three classes were used for training and 60 images were used for testing. Ten feed-forward networks with an initial architecture consisting of 21 inputs, 4 hidden units and 3 output units were trained and subsequently pruned; hence resulting in ten nested families of pruned networks. The estimate of the test error for each family was used to select the network with the lowest estimated generalization error. The weight decay parameters were after preliminary experiments set to $\alpha_w = 0.1$ and $\alpha_W = 1$.

In figure 11-13 a typical pruning scenario is shown. Table 1 shows the confusion matrix for the test set classified with the fully connected networks and the selected "optimal" networks. The mean and standard deviation of the ten runs are reported. Overall the fully connected networks classified $89.6 \pm 1.6\%$ of the training set and $54.6 \pm 4.1\%$ of the test set correctly, while the results for the pruned networks are $86.7 \pm 3.6\%$ for the training set and $58.9 \pm 2.1\%$ for the test set. Thus the pruning has increased the generalization ability of the networks. An important effect of the pruning approach is the selection of input features, that are salient for the classification; thus providing us with information that can be used in clinical dermatology. Of the ten pruned networks, four didn't use input 15 and three didn't use input 16. Such information can be valuable feedback for the design of future experiments. In the particular case, the two inputs that some networks discard are color variances, suggesting that these do not carry useful information for the classifier. Hence, we might e.g. investigate the color control of the illumination system in order to stabilize color variances.

A k-N-N classification was performed for comparison. Using the validation error as described in section 3.2 we found that $k = 6$ was optimal for this data set. The 6-N-N scheme classified 74.7% of the training set and 57.3% of the test set correctly. Thus the performance of the optimized k-N-N scheme fall in between the pruned and fully connected networks.

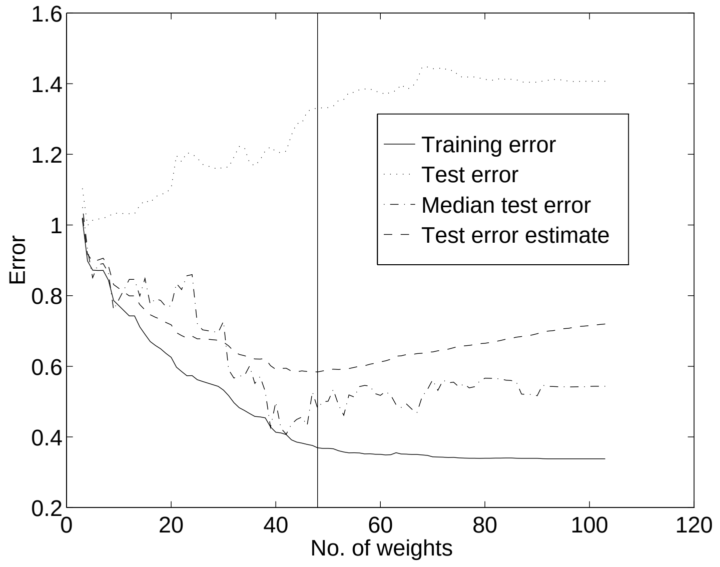

Figure 11: Pruning of a *skin lesion* network. The vertical line indicates the "optimal" network selected by the estimated test error. Notice the big difference between the test error and median test error.

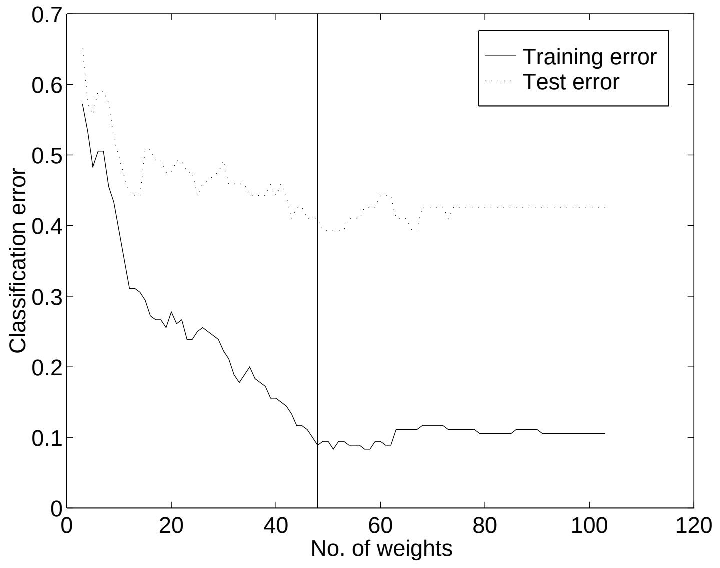

Figure 12: The classification error during pruning of a *skin lesion* network. Note again that the median test error in figure 11 follows the development of the classification test error better than the test error.

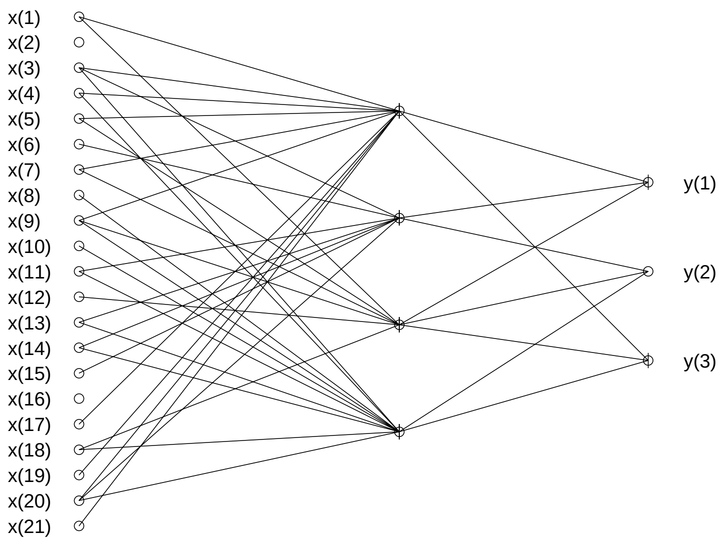

Figure 13: The "optimal" pruned *skin lesion* network (from 103 to 48 weights) selected by the minimum of the estimated test error in figure 11. Note that two inputs are not used.

4 CONCLUSION

We have developed a methodology for construction and evaluation of neural classifiers. Our aim was to present a *practical* approach dealing with the problems of overfitting and model selection without the use of a validation set. The approach was applied to one artificial and three real world problems. It was shown that the test error estimator for classifiers could be used to select optimal networks among families of pruned networks, thus increasing the generalization ability compared to non-pruned networks. Currently, the aim is to establish more empirical data for the validation of the neural classifier construction approach.

A NUMERICAL CONSIDERATIONS

When doing computer simulations, one has to consider the effects of calculations with finite word-length data. Here we rewrite the SoftMax equation (5) and the error measure in equation (8) to prevent overflow caused by the exponential function. The reformulations ensure that the argument to the exponential function is always smaller than or equal to zero. Equation (5) can be rewritten as

$$\begin{aligned}\hat{p}(y^\mu = i|\mathbf{x}^\mu) &= \frac{\exp[\phi_i(\mathbf{x}^\mu)]}{\sum_{i'} \exp[\phi_{i'}(\mathbf{x}^\mu)]} && (16)\\ &= \frac{\exp[\phi_i(\mathbf{x}^\mu) - \phi_{i_{\max}}(\mathbf{x}^\mu)]}{\sum_{i'} \exp[\phi_{i'}(\mathbf{x}^\mu) - \phi_{i_{\max}}(\mathbf{x}^\mu)]} && (17)\\ &= \frac{\exp[\phi_{i(\mathbf{x}^\mu)} - \phi_{i_{\max}}(\mathbf{x}^\mu)]}{1 + \sum_{i' \neq i_{\max}} \exp[\phi_{i'}(\mathbf{x}^\mu) - \phi_{i_{\max}}(\mathbf{x}^\mu)]} && (18)\end{aligned}$$

where

$$\phi_{i_{\max}}(\mathbf{x}^\mu) = \max_i \{\phi_i(\mathbf{x}^\mu)\}. \qquad (19)$$

Equation (8) is reformulated as

$$\begin{aligned}\epsilon(\mathbf{x}^\mu, y^\mu, \mathbf{u}) &= -\sum_i \delta_{i,y^\mu} \left[\phi_i(\mathbf{x}^\mu) - \log\left(\sum_{i'} \exp[\phi_{i'}(\mathbf{x}^\mu)]\right)\right] && (20)\\ &= -\sum_i \delta_{i,y^\mu} \left[\phi_i(\mathbf{x}^\mu) - \log\left(\exp[\phi_{i_{\max}}(\mathbf{x}^\mu)] \sum_{i'} \exp[\phi_{i'}(\mathbf{x}^\mu) - \phi_{i_{\max}}(\mathbf{x}^\mu)]\right)\right] && (21)\end{aligned}$$

$$= -\sum_{i} \delta_{i,y^\mu} \left[\phi_i(\mathbf{x}^\mu) - \phi_{i_{\max}}(\mathbf{x}^\mu) - \log\left(1 + \sum_{i' \neq i_{\max}} \exp[\phi_{i'}(\mathbf{x}^\mu) - \phi_{i_{\max}}(\mathbf{x}^\mu)]\right)\right] \quad (22)$$

where $\phi_{i_{\max}}(\mathbf{x}^\mu)$ is given by (19).

B COMPUTATION OF GRADIENT AND HESSIAN

The gradient of (7) is

$$\frac{\partial E(\mathbf{u})}{\partial u_j} = -\frac{1}{q}\sum_{\mu=1}^{q}\sum_{i=1}^{n_O} \delta_{i,y^\mu}\left[\frac{\partial \phi_i(\mathbf{x}^\mu)}{\partial u_j} - \frac{\sum_{i'=1}^{n_O}\left(\exp[\phi_{i'}(\mathbf{x}^\mu)]\frac{\partial \phi_{i'}(\mathbf{x}^\mu)}{\partial u_j}\right)}{\sum_{i''=1}^{n_O}\exp[\phi_{i''}(\mathbf{x}^\mu)]}\right] \quad (23)$$

$$= -\frac{1}{q}\sum_{\mu=1}^{q}\sum_{i=1}^{n_O}\sum_{i'=1}^{n_O} \delta_{i,y^\mu}\left[\delta_{i',i}\frac{\partial \phi_i(\mathbf{x}^\mu)}{\partial u_j} - \frac{\exp[\phi_{i'}(\mathbf{x}^\mu)]\frac{\partial \phi_{i'}(\mathbf{x}^\mu)}{\partial u_j}}{\sum_{i''=1}^{n_O}\exp[\phi_{i''}(\mathbf{x}^\mu)]}\right] \quad (24)$$

$$= -\frac{1}{q}\sum_{\mu=1}^{q}\sum_{i=1}^{n_O}\sum_{i'=1}^{n_O} \delta_{i,y^\mu}\left[\delta_{i',i} - \hat{p}(y^\mu = i'|\mathbf{x}^\mu)\right]\frac{\partial \phi_{i'}(\mathbf{x}^\mu)}{\partial u_j} \quad (25)$$

where $\delta_{i,y^\mu} = 1$ if $i = y^\mu$, otherwise $\delta_{i,y^\mu} = 0$.

Note that only when the index i equals the correct class for example $\mathbf{x}^\mu$ in (25), is the contribution to the gradient non-zero. This means that equation (25) can be reduced to

$$\frac{\partial E(\mathbf{u})}{\partial u_j} = -\frac{1}{q}\sum_{\mu=1}^{q}\sum_{i=1}^{n_O}\left[\delta_{i,y^\mu} - \hat{p}(y^\mu = i|\mathbf{x}^\mu)\right]\frac{\partial \phi_i(\mathbf{x}^\mu)}{\partial u_j} \quad (26)$$

The Hessian can be expressed as

$$\frac{\partial^2 E(\mathbf{u})}{\partial u_j \partial u_k} = -\frac{1}{q}\sum_{\mu=1}^{q}\sum_{i=1}^{n_O}\left(\left[\delta_{i,y^\mu} - \hat{p}(y^\mu = i|\mathbf{x}^\mu)\right]\frac{\partial^2 \phi_i(\mathbf{x}^\mu)}{\partial u_j \partial u_k} - \frac{\partial \hat{p}(y^\mu = i|\mathbf{x}^\mu)}{\partial u_k}\frac{\partial \phi_i(\mathbf{x}^\mu)}{\partial u_j}\right) \quad (27)$$

$$\approx \frac{1}{q}\sum_{\mu=1}^{q}\sum_{i=1}^{n_O}\frac{\partial \hat{p}(y^\mu = i|\mathbf{x}^\mu)}{\partial u_k}\frac{\partial \phi_i(\mathbf{x}^\mu)}{\partial u_j} \quad (28)$$

$$= \frac{1}{q}\sum_{\mu=1}^{q}\sum_{i=1}^{n_O}\sum_{i'=1}^{n_O}\hat{p}(y^\mu = i|\mathbf{x}^\mu)\left[\delta_{i,i'} - \hat{p}(y^\mu = i'|\mathbf{x}^\mu)\right]\frac{\partial \phi_{i'}(\mathbf{x}^\mu)}{\partial u_k}\frac{\partial \phi_i(\mathbf{x}^\mu)}{\partial u_j} \quad (29)$$

where we have used the Gauss-Newton approximation (Seber & Wild, 1995). This is motivated by Fisher's argument which is valid when using a log-likelihood cost function.

References

Andersen, L., Larsen, J., Hansen, L., & Hintz-Madsen, M. (1997). Adaptive Regularization of Neural Classifiers. In *Proceedings of the 1997 IEEE Workshop on Neural Networks for Signal Processing VII* (pp. 24–33). Piscataway, New Jersey.

Bishop, C. (1995). *Neural Networks for Pattern Recognition.* Oxford: Oxford University Press.

Bridle, J. (1990). Probabilistic Interpretation of Feedforward Classification Network Outputs with Relationships to Statistical Pattern Recognition. In *Neurocomputing - Algorithms, Architectures and Applications* (Vol. 6, pp. 227–236). Berlin: Springer-Verlag.

Cun, Y. L., Denker, J., & Solla, S. (1990). Optimal Brain Damage. *Advances in Neural Information Processing Systems*, *2*, 598–605.

Denker, J., Schwartz, D., Wittner, B., Solla, S., Howard, R., Jackel, L., & Hopfield, J. (1987). Large Automatic Learning, Rule Extraction and Generalization. *Complex Systems*, *1*, 877–922.

Duda, R., & Hart, P. (1973). *Pattern Classification and Scene Analysis.* New York: Wiley-Interscience.

Gorodkin, J., Hansen, L., Krogh, A., Svarer, C., & Winther, O. (1993). A Quantitative Study of Pruning by Optimal Brain Damage. *International Journal of Neural Systems*, *4*, 159–169.

Granlund, G. (1972). Fourier Preprocessing for Hand Print Character Recognition. *IEEE Transactions on Computers*, 195–201.

Hintz-Madsen, M., Hansen, L., Larsen, J., Olesen, E., & Drzewiecki, K. (1995). Design and Evaluation of Neural Classifiers - Application to Skin Lesion Classification. In *Proceedings of the 1995 IEEE Workshop on Neural Networks for Signal Processing V* (pp. 484–493). Piscataway, New Jersey.

Larsen, J. (1992). A Generalization Error Estimate for Nonlinear Systems. In *Proceedings of the 1992 IEEE Workshop on Neural Networks for Signal Processing II* (pp. 29–38). Piscataway, New Jersey.

Larsen, J. (1993). *Design of Neural Network Filters.* Unpublished doctoral dissertation, Electronics Institute, Technical University of Denmark.

Larsen, J., & Hansen, L. (1995). Empirical Generalization Assessment of Neural Network Models. In *Proceedings of the 1995 IEEE Workshop on Neural Networks for Signal Processing V* (pp. 30–39). Piscataway, New Jersey.

Larsen, J., L.K. Hansen, Svarer, C., & Ohlsson, M. (1996). Design and Regularization of Neural Networks: The Optimal Use of A Validation Set. In *Proceedings of the 1996 IEEE Workshop on Neural Networks for Signal Processing VI* (pp. 62–71). Piscataway, New Jersey.

Murata, N., Yoshizawa, S., & Amari, S. (1994). Network Information Criterion - Determining the Number of Hidden Units for an Artificial Neural Network Model. *IEEE Transactions on Neural Networks*, *5*, 865–872.

Prechelt, L. (1994). *PROBEN1 — A Set of Benchmarks and Benchmarking Rules for Neural Network Training Algorithms* (Tech. Rep. No. 21/94). Fakultät für Informatik, Universität Karlsruhe, Germany. (*Available via anonymous ftp* `ftp.ira.uka.de/pub/papers/techreports/1994/1994-21.ps.Z`)

Ripley, B. (1996). *Pattern Recognition and Neural Networks.* Cambridge: Cambridge University Press.

Seber, G., & Wild, C. (1995). *Nonlinear Regression.* New York, New York: John Wiley & Sons.

Sjöberg, J. (1995). *Non-Linear System Identification with Neural Networks.* Unpublished doctoral dissertation, Department of Electrical Engineering, Linköping University, Sweden.

Stoecker, W., Moss, R., Ercal, F., & Umbaugh, S. (1995). Nondermatoscopic Digital Imaging of Pigmented Lesion. *Skin Research and Technology*, *1*, 7–16.

Svarer, C., Hansen, L., & Larsen, J. (1993). On Design and Evaluation of Tapped-Delay Neural Network Architectures. In *Proceedings of the 1993 IEEE International Conference on Neural Networks* (pp. 46–51). Baltimore.

Svarer, C., Hansen, L., Larsen, J., & Rasmussen, C. (1993). Designer Networks for Time Series Processing. In *Proceedings of the 1993 IEEE Workshop on Neural Networks for Signal Processing III* (pp. 78–97). Piscataway, New Jersey.