DESIGN AND EVALUATION OF NEURAL CLASSIFIERS

Mads Hintz-Madsen, Morten With Pedersen,
Lars Kai Hansen, and Jan Larsen
CONNECT, Dept. of Mathematical Modeling, B. 349
Technical University of Denmark
DK-2800 Lyngby, Denmark

Phone: (+45) 4525 3885, Fax: (+45) 4588 0117
Email: hintz, with, lkhansen, jlarsen@ei.dtu.dk

Abstract - In this paper we propose a method for design 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 the test error estimate is used to optimize the network architecture. The scheme is evaluated on an artificial and a real world problem.

INTRODUCTION

Pattern recognition is an important aspect of most scientific fields and indeed the objective of most neural network applications. Some of the by now 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. The conventional approach to pattern recognition is statistical and concerns the modeling of class probability distributions for patterns produced by a stationary stochastic source by a certain set of basis functions, e.g., Parzen windows or Gaussian mixtures.

In this paper we define and analyze a system for design 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 [1, 2] and for binary classification in the context of a medical application [3]. 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 complexity estimate. Our approach is a penalized maximum likelihood scheme. The likelihood is formulated using a simple stationary noisy channel model of the pattern source. For any fixed input pattern there can be defined a fixed probability distribution over a fixed finite set of classes. The training set involves simple labelled data, i.e., for each input vector we are provided 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 [4]. For a fixed architecture the neural network weights are estimated using a Gauss-Newton method [5], while the model architecture is optimized using Optimal Brain Damage [6]. The problem of proper selection of regularization parameters is also briefly discussed, see also [7].

The salient features of the approach are: Efficient Newton optimization, pruning by Optimal Brain Damage and evaluation of network architectures by an algebraic test error estimate.

NEURAL CLASSIFIERS

Let us assume that 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.

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., [3].

To represent these distributions we choose the following network architecture:

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

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

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, $\phi_i(\mathbf{x}^\mu)$, can be interpreted as probabilities, we use the normalized exponential transformation known as SoftMax [4]:

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

where $\hat{p}(y^\mu = i|\mathbf{x}^\mu)$ is the estimated probability, that $\mathbf{x}^\mu$ belongs to class i.

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

Training is based on the minimization of the negative 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

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

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

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

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

The gradient of (7) is

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

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

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

where we have used a Gauss-Newton like approximation.

Using matrix/vector notation the Gauss-Newton paradigm of updating the weights can now be computed as [5]

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

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

A natural approach for determining the regularization parameters is by minimizing 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. Let us now consider the case with only 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 with e.g., a 3x3 grid and computing the estimated test error, it is possible to fit e.g., a paraboloid[1] in a least-square sense to the sample points, locate the minimum[2] of the paraboloid and use the weight decays found for the design of the network.

A different approach using a validation set for determining the regularization parameters is described in [7].

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

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

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, see e.g., [3] and [8], thus giving us the following estimate for the average test error of a network $\mathbf{u}$ estimated on a training set D [8],

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

[1] Paraboloid: $(z - z_0) = (x - x_0)^2/a^2 + (y - y_0)^2/b^2$.

[2] In case the minimum is located outside the sample-grid, one should relocate the grid and find a new minimum.

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. 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 [1, 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) [6]. 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 [1]

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

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

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.

EXPERIMENTS

The proposed methodology for designing neural classifiers has been evaluated on the artificial *contiguity problem* and the real world *glass classification problem.* The latter is a part of the Proben1 neural network benchmark collection [9].

The Contiguity Problem

The contiguity problem has in several cases been used for evaluating optimization schemes, see e.g., [10]. 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 selected training set with 150 patterns and a test set with 510 patterns both containing an even split of the two classes.

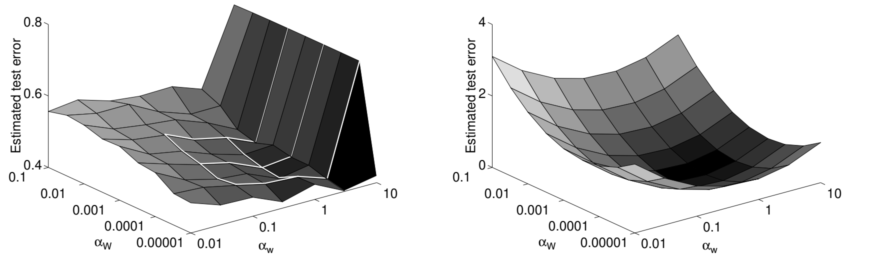

Figure 1: Left: The estimated test error for the *contiguity problem* as function of the weight decay parameters. Right: Paraboloid fitted to the 3x3 grid shown in the left panel. Minimum located at $(\alpha_w, \alpha_W) = (0.68, 2.8 \cdot 10^{-4})$.

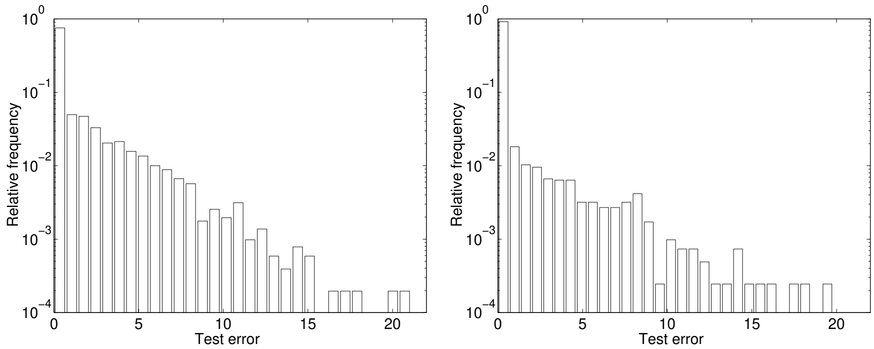

Figure 2: Left: The distribution of the test error for 10 fully connected *contiguity* networks combined. Notice the "long tail" of the distribution resulting in a high mean error (0.94) and a small median error (0.022) i.e., the mean is predominantly driven by a few examples with high error. Right: The distribution of the test error for 10 pruned networks selected by the minimum of the estimated test error combined. The mean error is 0.37 and the median error is 0.0014 showing a significant performance improvement as result of pruning.

Initially a network architecture consisting of 10 input units, 8 hidden units and 2 output units was chosen. The weight decay parameters were estimated by using the previously described sample-grid technique. This is shown in figure 1. Next ten fully connected networks were trained[3] using the estimated weight decay parameters, subsequently pruned using the OBD saliency ranking, removing one weight per iteration. In figure 2 the distribution of the test error is shown. 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 error as well in order to get a good indication of a network's performance. Seven of the ten pruned networks had a classification[4]

[3]Training was stopped when the 2-norm of the gradient vector was below 10^{-5}.

[4]Following Bayes decision theory, the network output with the highest probability determines the class label.

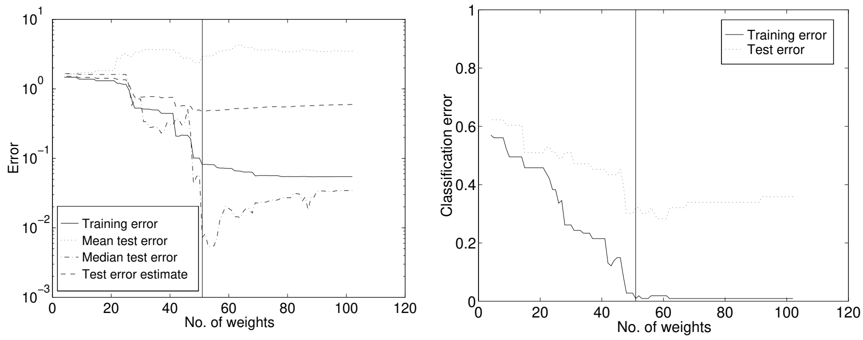

Figure 3: Pruning of a *glass classification* network using the small training set. The vertical line indicates the "optimal" network selected by the minimum of the estimated test error. Notice the similarity in the development of the median error and the classification error on the test set.

error on the test set between 0% and 3.3%, while three networks had an error of 16-19%. In [10] 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.

The Glass Classification Problem

The task in the glass classification problem 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 [9] 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 3 and 4 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 [9] 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 [11]. Our approach using the estimated test error for model selection eliminates the need for a validation

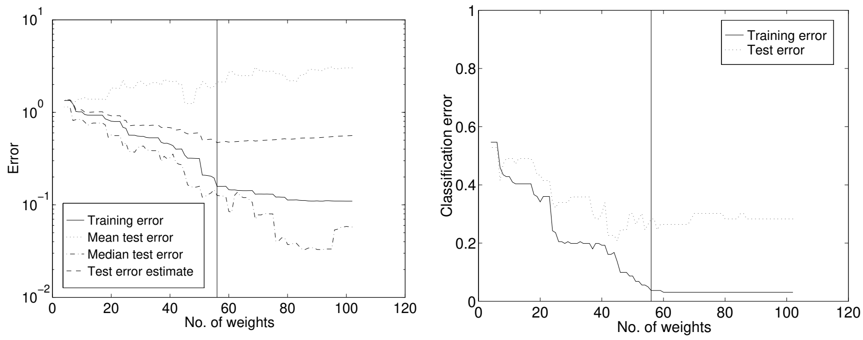

Figure 4: Pruning of a *glass classification* network using the large training set. The vertical line indicates the "optimal" network selected by the estimated test error. Notice the overall lower classification error on the test set compared to figure 3.

set, thus allowing us to use more data for the actual training resulting in a better generalization performance. In a forthcoming paper the problem of comparing the performance of neural network models is addressed [12].

For comparison a standard *k-Nearest-Neighbor*[5] (k-N-N) classification 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 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.

CONCLUSION

We have developed a methodology for design and evaluation of neural classifiers. The approach was applied to the *contiguity problem* and the *glass classification problem*. 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 design approach.

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

ACKNOWLEDGMENT

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.

REFERENCES

[1] C. Svarer, L.K. Hansen, and J. Larsen. "On Design and Evaluation of Tapped-Delay Neural Network Architectures". **Proceedings of the 1993 IEEE International Conference on Neural Networks**, pages 46–51, 1993.

[2] C. Svarer, L.K. Hansen, J. Larsen, and C.E. Rasmussen. "Designer Networks for Time Series Processing". **Proceedings of the 1993 IEEE Workshop on Neural Networks for Signal Processing III**, pages 78–97, 1993.

[3] M. Hintz-Madsen, L.K. Hansen, J.Larsen, E. Olesen, and K.T. Drzewiecki. "Design and Evaluation of Neural Classifiers - Application to Skin Lesion Classification". **Proceedings of the 1995 IEEE Workshop on Neural Networks for Signal Processing V**, pages 484–493, 1995.

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

[5] G.A.F. Seber and C.J. Wild. **Nonlinear Regression**. John Wiley & Sons, New York, New York, 1995.

[6] Y. Le Cun, J. Denker, and S. Solla. "Optimal Brain Damage". **Advances in Neural Information Processing Systems**, 2:598–605, 1990.

[7] J. Larsen, L.K. Hansen, C. Svarer, and M. Ohlsson. "Design and Regularization of Neural Networks: The Optimal Use of A Validation Set". **Proceedings of the 1996 IEEE Workshop on Neural Networks for Signal Processing VI**, 1996.

[8] S. Amari and N. Murata. "Statistical Theory of Learning Curves under Entropic Loss Criterion". **Neural Computation**, 5:140–153, 1993.

[9] Lutz Prechelt. "PROBEN1 — A set of benchmarks and benchmarking rules for neural network training algorithms". **Technical Report 21/94, Fakultät für Informatik, Universität Karlsruhe, Germany**, 1994. *Available via anonymous ftp* `ftp.ira.uka.de/pub/papers/techreports/1994/1994-21.ps.Z`.

[10] J. Gorodkin, L.K. Hansen, A. Krogh, C. Svarer, and O. Winther. "A Quantitative Study of Pruning by Optimal Brain Damage". **International Journal of Neural Systems**, 4:159–169, 1993.

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

[12] J. Larsen et al. "Empirical Comparison of Neural Network Models". **In preparation**, 1996.