A New Sensitivity-Based Feature Selection Technique for Feed-Forward Neural Networks That Improves Generalization

Iveta Mrázová and Zuzana Reitermanová
Faculty of Mathematics and Physics, Charles University in Prague
Malostranské náměstí 25, 118 00 Prague, Czech Republic
email: iveta.mrazova@mff.cuni.cz, zuzana.reitermanova@matfyz.cz

Abstract

Multi-layer neural networks of the back-propagation type became already a well-established tool used successfully in various application areas. Efficient solutions to complex tasks currently dealt with obviously require sufficient generalization capabilities of the formed networks and an easy interpretation of their function. For this reason, we will introduce here a new feature selection technique called SCGSIR inspired by the fast method of scaled conjugate gradients (SCG) and sensitivity analysis.

Enforced internal knowledge representation supports an easy interpretation of the formed network structure. Network sensitivity inhibited during training impacts successful pruning of input neurons and optimization of network structure, too. Experiments performed so far on the problem of binary addition and on real data obtained from the World Bank yield promising results outperforming reference techniques when considering both their ability to find networks with and optimum architectures and generalization capabilities of the trained networks. [1]

1 Introduction

Artificial neural networks represent a widely acknowledged means applicable to many emerging areas like data and web mining or multimedia information processing. Reliable solutions to such large-scale tasks require, however, adequate generalization of the extracted knowledge. Good generalization means a correct behavior also for previously unseen or noisy patterns.

From the literature, it is well known that smaller networks with smoother functions, lower curvature and a larger margin set along the separating hyper-planes are expected to reduce the VC-dimension of the final network and generalize better [1], [2], [3] [4], [8], [14], [16], [20]. Unfortunately, even if the optimum size of the network were known, it might be difficult to train such a network completely from scratch [15]. Namely, standard BP-networks usually do not tend to develop a transparent network structure. For such networks, it is extremely difficult to "guess" the real meaning of every particular hidden or even input neuron for a proper network output. Such networks often use small differences of neuron outputs to distinguish between the presented patterns. Pruning techniques applied to larger already trained networks might represent a viable option in such a case [12].

In this paper, we will therefore develop a new sensitivity-based feature selection technique inspired by the SCGIR-training algorithm using a fast optimization approach of scaled conjugate gradients (SCG). The following Section 2 discusses relevant pruning and feature selection techniques based on sensitivity analysis and enforced internal representation. In Section 3, we will outline the idea the new sensitivity-based feature selection technique inspired by the algorithm for learning internal representation with scaled conjugate gradients (SCGSIR). Section 4 is devoted to experimental results obtained so far for the problem of binary addition of two 3-bit numbers and for real data provided by the World Bank. The concluding Section 5 summarizes the achieved results.

[1] This research was partially supported by the Grant Agency of Charles University in Prague under Grant-No. 17608, by the Czech Science Foundation under Grant-No. P103/10/0783, Grant-No. P202/10/1333 and Grant-No. 201/09/H057, and by SVV project number 263 314.

2 Related works

To keep the paper self-contained, we will first describe briefly also the well-known back-propagation algorithm (BP) used to train fully-connected feed-forward neural networks (BP-networks). Although we will in this paper restrict our consideration to two-layer BP-networks the obtained results can be easily generalized also to networks with more hidden layers. For the BP-training algorithm, the training set T is a finite non-empty set of P ordered pairs of input/output patterns:

$$T \;=\; \{\,[\,\vec{x}_1\,,\,\vec{d}_1\,],\;\ldots\;,\,[\,\vec{x}_P\,,\,\vec{d}_P\,]\,\}\;\;. \tag{1}$$

A neuron with the weights $(w_1, \ldots, w_n)$, the threshold ϑ and the input vector $\vec{z} = (z_1, \ldots, z_n)$, computes its potential value ξ as $\xi \;=\; \sum_{i=0}^{n} z_i w_i$, where $w_0 \;=\; \vartheta$ and $z_0 \;=\; 1$. For the output neurons, we will consider the linear transfer function $y = f(\xi) = \xi$ with the derivative equal to one: $f'(\xi) = 1$. For the output of hidden neurons, we will consider the hyperbolic tangent transfer function of the form:

$$y \;=\; f(\xi) \;=\; \frac{1 \;-\; e^{-\,2\,\xi}}{1 \;+\; e^{-\,2\,\xi}}\;\;. \tag{2}$$

Its derivative $f'(\xi)$ equals to:

$$f'(\xi) \;=\; (\,1 \;+\; y\,)\,(\,1 \;-\; y\,) \;=\; 1 \;-\; y^2\;\;. \tag{3}$$

The aim of the standard BP-training algorithm [13] is to find a set of weights that ensure that for each input pattern the actual output produced by the network is the same as (or sufficiently close to) the output pattern. The desired behavior is evaluated by the objective function E:

$$E \;=\; \frac{1}{2}\;\sum_{p}\;\sum_{v}\,(\,y_{v,p} \;-\; d_{v,p}\,)^2\;\;, \tag{4}$$

where p is an index over all training patterns, v is an index over all output neurons, y is their actual and d is their desired output value. Omitting the index p for the desired and actual neuron output values d and y, the weights of the network are adjusted iteratively after presenting each respective training pattern by:

$$w_{ij}(t+1) \;\;=\;\; w_{ij}(t) + \alpha \delta_j y_i\;\;. \tag{5}$$

The terms for δ_j correspond to:

$$\delta_j \;\;=\;\; \begin{cases} d_j \;-\; y_j & \text{for an output neuron} \\ (\,1 \;+\; y_j\,)\,(\,1 \;-\; y_j\,)\;\sum_k\,\delta_k\,w_{jk} & \\ & \text{for a hidden neuron,} \end{cases} \tag{6}$$

i and k index neurons in the layers below and above the neuron j, respectively. $t+1$ and t index next and present weights, respectively, α stands for the learning rate.

2.1 Learning internal representation

In order to clarify the role of hidden neurons, the method enforcing the so-called condensed internal representation [11] groups the outputs of the hidden neurons around three possible values 1, -1 and 0 corresponding to active, passive and to the so-called silent states. In the AI-terminology, this corresponds to creating rules of an expert system, where active neuron states (output close to 1) indicate "yes", passive states (output close to -1) "no" and the so-called silent states stand for those cases where "no decision is possible." The criterion for developing a condensed internal representation will be formulated as an additional term of the objective function to be minimized during training:

$$F \;=\; \sum_{p}\;\sum_{h}\,(\,1 \;+\; y_{h,p}\,)^s\,(\,1 \;-\; y_{h,p}\,)^s\,y_{h,p}^2\;\;, \tag{7}$$

where p is an index over all training patterns and h an index over all hidden neurons, y represents their output value. The respective terms achieve their minima for one of the values 1, -1 and 0. s tunes the shape of the representation error function F (a recommended value is $s = 4$). Now, the new objective function has the form:

$$\hat{H} \;=\; E \;+\; c_F\, F \;\;, \tag{8}$$

where E represents the standard BP-error function and F stands for the above defined representation error function. c_F reflects the trade-off between the influence of E and F in $\hat{H}$. To minimize $\hat{H}$, gradient descent can be applied as well by adjusting the weights according to:

$$w_{ij}(t+1) \;=\; w_{ij}(t) \;+\; \alpha\, \delta_j\, y_i \;+\; \alpha_r\, \varrho_j\, y_i \;\;. \tag{9}$$

In this formula, δ_j is defined by the term (6) and

$$\varrho_j \;=\; \begin{cases} 0 & \text{for an output neuron} \\ 2\left[(s+1)y_j^2 - 1\right](1+y_j)^s(1-y_j)^s y_j & \\ & \text{for a hidden neuron,} \end{cases} \tag{10}$$

w stands for the weights. i indexes the neuron connected with neuron j via the weight w_{ij}. y_j is the actual output value of the neuron j. s tunes the shape of the representation error function. $t+1$ and t index next and present weights, respectively. α and α_r represent the particular learning rates.

In spite of the significant advantages provided by the enforced internal representation, its combination with the standard BP-algorithm can be relatively slow for large-scale real data. An extremely efficient alternative to standard BP-training represents, however, the method of scaled conjugate gradients (SCG) [9]. Although SCG uses second order information from the neural network, it requires only $O(N)$ memory usage with N being the number of weights in the network and shows super-linear convergence for most problems. Therefore, we implemented the described strategy of internal representation enforcement as an enhancement of the SCG-training algorithm [10].

2.2 Dimensionality reduction and sensitivity analysis

Very often, practical implementations of neural networks face a serious problem of an excessive task dimensionality. Ideally, the system should learn to ignore redundant and irrelevant inputs by itself. But in practice, superfluous inputs may lead to worse generalization. Intuitively, the intrinsic generalization of an input pattern $\vec{x}_p$ includes all the input patterns $\vec{x}_q, q \neq p$ that cannot be distinguished from $\vec{x}_p$ under the (low-dimensional) internal representation r: $r(\vec{x}_p) = r(\vec{x}_q)$. Quite naturally, one would like to group all the patterns from each class into a small number of equivalence classes, with each class having its cardinality as large as possible. A lower number of equivalence classes implies reduced dimensionality of trained networks resulting into a lower VC-dimension and improved generalization [18]. This general idea has inspired several cluster-based feature selection techniques as well.

Other techniques like PCA try to identify mutual correlations among the input features. Yet in principle, PCA can namely detect just linear dependencies among the data. A strategy capable of detecting also non-linear dependencies among the data is discussed in [6]. The method called Feature Subset Selection (FSS) is based on the main principles of sensitivity analysis. The sensitivity coefficients S_{ij} are computed as the mean absolute value of the derivatives of the j-th output with respect to the i-th input over all of the P input patterns:

$$S_{ij} \;=\; \frac{1}{P} \sum_{p=1}^{P} \left|\partial\, y_{j,p} / \partial x_{i,p}\,\right| \tag{11}$$

and express how and how much the solution to a given problem depends on the data. Input neurons with low sensitivity coefficients are considered to be less important and can be pruned from the network.

The concept of network sensitivities can be used also for learning. The Sensitivity-Based Linear Learning Method (SBLLM) [5] proposed for two-layer feed-forward neural networks, calculates the

weights by solving a system of linear equations. Assuming that the applied non-linear transfer functions are invertible, the method minimizes for each pattern the difference between the actual and desired neuron potentials. For both the hidden and output neurons, the desired potential value can be determined as the value of the inverse transfer function of their desired outputs. During training, the error Q of the network is evaluated as the sum of the error Q^{HID} (determined as the difference between the desired and actual potential values over all hidden neurons h) and Q^{OUT} (determined as the difference between the desired and actual potential values over all output neurons v):

$$Q = Q^{HID} + Q^{OUT} \ , \tag{12}$$

$$Q^{HID} = \sum_p \sum_h \left(\sum_u w_{uh} x_{u,p} - f_h^{-1}(y_{h,p}) \right)^2 \ , \tag{13}$$

$$Q^{OUT} = \sum_p \sum_v \left(\sum_h w_{hv} y_{h,p} - f_v^{-1}(y_{v,p}) \right)^2 \ , \tag{14}$$

This leads to a system of linear equations to be solved for both considered layers. Afterwards, the sensitivity terms for the sensitivity of the error Q^{HID} of the hidden neuron potentials to actual outputs of the hidden neurons $\frac{\partial Q^{HID}}{\partial y_{h,p}}$ and for the sensitivity of the error Q^{OUT} of the output neuron potentials to actual outputs of the hidden neurons $\frac{\partial Q^{OUT}}{\partial y_{h,p}}$ are computed and then also used to adjust the estimated "desired" output values for the hidden neurons:

$$\frac{\partial Q^{HID}}{\partial y_{h,p}} = -\frac{2\left(\sum_u w_{uh} x_{u,p} - f_h^{-1}(y_{h,p})\right)}{f_h'(y_{h,p})} \ ; \forall p, h \tag{15}$$

$$\frac{\partial Q^{OUT}}{\partial y_{h,p}} = 2 \sum_v \left(\sum_h w_{hv} y_{h,p} - f_v^{-1}(y_{v,p}) \right) w_{hv} \ ; \forall p, h \tag{16}$$

These sensitivities allow then for an efficient and extremely fast iterative gradient-based update of the "desired" hidden neuron states to be applied. This method reaches a minimum error in a few epochs of training. This behavior is very convenient when dealing with huge data sets and large networks. However, when the training set is not representative enough, the few iterations employed by the method make it very difficult to avoid overtraining with techniques like early stopping.

A usual technique to avoid over-fitting is regularization that consists in adding a penalty term to the loss function. Therefore, a generalization of the SBLLM method [7] uses a regularization term based on the well-known weight decay regularizer that is defined as the sum of squares of all the weights and thresholds in the network. As a result, the weights of both layers are calculated independently by minimizing a new objective function $\hat{Q}^{(l)}$ for each of the respective layers, l:

$$\hat{Q}^{(l)} = L^{(l)} + \alpha \sum_i \sum_j w_{ij}^2 \ , \tag{17}$$

where

$$L^{(l)} = \sum_p \sum_j \left(f_j'(y_{p,j}) \varepsilon_{p,j} \right)^2 = \sum_p \sum_j \left(f_j'(y_{p,j}) \left(\sum_i w_{ij} x_{p,i} - f_j^{-1}(y_{p,j}) \right) \right)^2 \ . \tag{18}$$

In the above equations (17) and (18), α is the regularization parameter, the second term on the right-hand side of (17) is the regularization term, and i, and j are the indexes over the inputs and outputs of the considered layer l, $y_{p,j}$ is the desired output for the neuron j. The term $L^{(l)}$ measures the training error also as the sum of squared errors before the non-linear transfer functions. However, as big differences of the potentials matter more around zero desired potentials, a scaling term to the sensitivity loss function has been introduced that multiplies it by the derivative of the transfer function applied to the desired layer output, $f_j'(y_{p,j})$. This equalizes the errors calculated before and after the non-linearities.

3 Sensitivity-based SCGSIR-training

Although the above-sketched sensitivity-based techniques exhibit fast convergence, they are primarily aimed at training instead of feature selection and pruning. Due to the character of the networks found by solving a system of linear equations, they might also tend to overtrain. However, our main goal is to develop a training algorithm likely to find an adequate network structure automatically during training. At the same time, the network should generalize well and support an easy interpretation of the extracted knowledge, e.g. by means of the formed internal representations. Moreover, the algorithm should optimize the sensitivity of the entire network and not just of its parts.

For the above reasons, the new method of the so-called SCGSIR-training is inspired also by the previously developed SCGIR-algorithm enforcing condensed internal representation [10]. The SCGSCIR-algorithm is, however, further enhanced by sensitivity control already during training. In essence, there are two main options for sensitivity control in the networks – sensitivity can be either inhibited or enforced. Both variants are assumed to increase the differences among the achieved sensitivity coefficients of the respective neurons and restrict the space of candidate hypotheses for the wanted network function. Anyway, we expect that BP-networks with an inhibited sensitivity might yield better results due to their smoother function.

In general, the sensitivity of network output towards its input can be characterized by means of the function G:

$$G \;=\; \frac{1}{2} \sum_{p} \sum_{u} \sum_{v} \left(\frac{\partial y_{v,p}}{\partial x_{u,p}} \right)^2 \quad . \tag{19}$$

For a given input pattern p, the term $\partial y_{v,p}/\partial x_{u,p}$ corresponding to the sensitivity $S_{uv,p}$ of the output value $y_{v,p}$ of the output neuron v to the u-th element of the input can be derived for the considered network architecture as:

$$\begin{aligned} S_{uv,p} \;&=\; \frac{\partial y_{v,p}}{\partial x_{u,p}} \;=\; \frac{\partial y_{v,p}}{\partial \xi_{v,p}} \; \frac{\partial \xi_{v,p}}{\partial x_{u,p}} \;= \\ &=\; \frac{\partial y_{v,p}}{\partial \xi_{v,p}} \sum_{h} \frac{\partial \xi_{v,p}}{\partial y_{h,p}} \; \frac{\partial y_{h,p}}{\partial \xi_{h,p}} \; \frac{\partial \xi_{h,p}}{\partial x_{u,p}} \;= \\ &=\; \sum_{h} \left(1 - y_{h,p}^2 \right) \; w_{uh} \; w_{hv} \quad , \end{aligned} \tag{20}$$

where h indexes the network's hidden neurons. As a result, the sensitivity criterion to be optimized simultaneously with the above-considered network performance already during training has the following form of:

$$\begin{aligned} G \;&=\; \frac{1}{2} \sum_{p} \sum_{u} \sum_{v} \left(\frac{\partial y_{v,p}}{\partial x_{u,p}} \right)^2 = \\ &=\; \frac{1}{2} \sum_{p} \sum_{u} \sum_{v} \left[\sum_{h} (1 - y_{h,p}^2) \, w_{uh} \, w_{hv} \right]^2 \quad , \end{aligned} \tag{21}$$

where p is an index over all training patterns, v is an index over all output neurons, u is an index over all input neurons and h is an index over all hidden neurons. y denotes the actual output value of the respective neuron while x corresponds to the considered element of the presented training pattern. In the SCGSIR-algorithm, we will thus use the following modification of the objective function H: $H(\vec{w}) = E(\vec{w}) + c_F F(\vec{w}) + c_G G(\vec{w})$ with $H'(\vec{w}) = E'(\vec{w}) + c_F F'(\vec{w}) + c_G G'(\vec{w})$. E represents here the standard BP-error function, F stands for the above defined representation error function and G corresponds to the new-proposed network sensitivity criterion to be optimized ($\sim$ usually minimized) during training. c_F and c_G are coefficients reflecting the trade-off between the influence of E, F and G in H. For the considered networks, $H(\vec{w})$ can be evaluated for the SCGSIR-training algorithm according

to:

$$H(\vec{w}) \;=\; \frac{1}{2}\sum_{p}\sum_{v}(\,y_{v,p}\;-\;d_{v,p}\,)^2\;+$$

$$+\,c_F\sum_{p}\sum_{h}(1+y_{h,p})^s\,(1-y_{h,p})^s\,y_{h,p}^2\;+$$

$$+\,c_G\,\frac{1}{2}\sum_{p}\sum_{u}\sum_{v}\left[\sum_{h}(1-y_{h,p}^2)\,w_{uh}\,w_{hv}\right]^2\quad, \tag{22}$$

where p goes over all training patterns, j and h are indexes over all output and hidden neurons, respectively. y denotes the actual output of a neuron while d is its desired output value. s is a parameter for tuning the shape of the representation error function. To minimize H, we will minimize E, F and G simultaneously. The terms used to adjust the weights with respect to the error function E and the representation error function F were already stated above in Equation (6) and (10). To minimize also the sensitivity control function G, we have to change each weight, w_{ij}, by an amount $\Delta_G w_{ij}$ proportional to the negative partial derivative of G with respect to this weight, $-\,\partial G/\partial w_{ij}$. By omitting the index p for both the actual neuron output values y and the elements of the feature vectors, x, we obtain:

$$\begin{aligned}\Delta_G w_{ij} \;&=\; -\frac{\partial}{\partial w_{ij}}\left(\frac{1}{2}\sum_{v,u}\left[\sum_{h}(1-y_h^2)w_{uh}w_{hv}\right]^2\right)=\\[2ex]
&=\; -\sum_{v}\sum_{u}\left[\sum_{h}(1-y_h^2)\,w_{uh}\,w_{hv}\right]\cdot\\[2ex]
&\quad\cdot\left[\frac{\partial}{\partial w_{ij}}\left(\sum_{h}(1-y_h^2)w_{uh}w_{hv}\right)\right]\quad,\end{aligned} \tag{23}$$

where u denotes the input neurons and v indicates the output neurons. For the output layer, $\Delta_G w_{ij}$ thus corresponds to:

$$\begin{aligned}\Delta_G w_{ij} \;&=\; -\sum_{v}\sum_{u}\left[\sum_{h}(1-y_h^2)w_{uh}w_{hv}\right]\cdot(1-y_i^2)\,w_{ui}\,\frac{\partial w_{iv}}{\partial w_{ij}}=\\[2ex]
&=\; -(1-y_i^2)\sum_{u}w_{ui}\left[\sum_{h}(1-y_h^2)w_{uh}w_{hj}\right]=\\[2ex]
&=\; -(1-y_i^2)\sum_{u}\,w_{ui}\,S_{uj}\;\;.\end{aligned} \tag{24}$$

And for the hidden layer we obtain:

$$\begin{aligned}\Delta_G w_{ij} \;&=\; -\sum_{v}\sum_{u}\left[\sum_{h}(1-y_h^2)w_{uh}w_{hv}\right]\cdot\\[2ex]
&\quad\cdot\left[\sum_{h}w_{hv}\,(\,\frac{\partial(1-y_h^2)}{\partial w_{ij}}\,w_{uh}\;+\;(1-y_h^2)\,\frac{\partial w_{uh}}{\partial w_{ij}}\,)\right]=\\[2ex]
&=\; -\sum_{v}\left[\sum_{h}(1-y_h^2)w_{ih}w_{hv}\right](1-y_j^2)w_{jv}\;+\\[2ex]
&\quad+2\sum_{v}\sum_{u}\left[\sum_{h}(1-y_h^2)\,w_{uh}\,w_{hv}\right]\cdot\,y_j\,(1-y_j^2)\,x_i\,w_{uj}\,w_{jv}\;=\\[2ex]
&=\; -(1-y_j^2)\sum_{v}\,w_{jv}\,S_{iv}\;+\;2y_j(1-y_j^2)x_i\sum_{v}\sum_{u}S_{uv}\,w_{uj}\,w_{jv}\;\;.\end{aligned} \tag{25}$$

In the above derivations, w stands for weights. i indexes the neuron connected with neuron j via the weight w_{ij}. h indexes the hidden neurons and y_j denotes the actual output value of the neuron j. The elements $\partial H/\partial w_{ij}$ of $H'(w) = (\partial H/\partial w_{11}, \ldots, \partial H/\partial w_{m_{last}m})$ correspond to:

$$\frac{\partial H}{\partial w_{ij}} = \begin{cases} -(d_j - y_j)y_i - c_G(1-y_i^2)\sum_u w_{ui}S_{uj} \\ \qquad\qquad \text{for output neurons} \\ -\{\ \sum_k \delta_k w_{jk} + c_F\left[2(s+1)y_j^2 - 1\right]\cdot(1+y_j)^{s-1}(1-y_j)^{s-1}y_j\ \}\cdot \\ \qquad \cdot(1+y_j)(1-y_j)y_i - \\ \qquad -c_G\ \left[(1-y_j^2)\sum_k\ w_{jk}\ S_{ik}\ +\ 2y_j(1-y_j^2)x_i\sum_k\ \sum_u\ S_{uk}\,w_{uj}\,w_{jk}\right] \\ \qquad\qquad \text{for hidden neurons} \end{cases}$$

w stands for weights. i indexes neurons connected with neuron j via the weight w_{ij}. k indexes the neurons from the layer above the neuron j, u indexes the neurons from the layer below the neuron j, m and m_{last} denote the number of output and hidden neurons, respectively. y_j is the actual output value of the neuron j. s is the parameter for tuning the shape of the representation error function. c_F and c_G are constants representing the influence of the respective error terms. δ_k can be determined according to (6).

The entire process of sensitivity-based training by means of scaled conjugate gradients with enforced internal representation and controlled sensitivity (SCGSIR) is stated in detail in the algorithm box below.

4 Supporting experiments

In our experiments, we wanted to assess the benefits of the new-proposed method (SCGSIR) when compared with other techniques – pure scaled conjugate gradients (SCG) and SCG with enforced condensed internal representation (SCGIR). More variants of the SCGSIR method were tested – SCGSIR-W (sensitivity is inhibited), SCGSIR-S (sensitivity is enforced), SCGS-W, SCGS-S (variants with $c_F = 0$).

In particular, we were interested in answering the following questions:

1. Can the cluster-based feature selection techniques provide features relevant for the processed data and what relevance measures do best?
2. What is the speed of sensitivity-based training and how well do the trained networks generalize?
3. Are the SCGS and SCGSIR-trained networks more likely to develop an optimum architecture than their SCG-trained counterparts?

The performance of the discussed methods has been evaluated on two kinds of problems. The first illustrative one is the binary addition of two 3-bit numbers. The second one deals with real-world data obtained from the *World Bank*. Several input features generated randomly with a uniform distribution were added to the data sets to examine, whether the methods are able to recognize the relevant input features.

In both kinds of tests, we used the bipartite model. During all the tests, the weights were randomly initialized from the interval $[-1; 1]$. When comparing the methods, we use the following notation: n_{corr} is the number of networks with no errors on the training, validation and testing sets, *epochs* is the number of training epochs, *t(s)* is the training time in seconds, n_H / n_I is the average number of hidden / input neurons after training. n_{cH} / n_{cI} is the number of networks that achieved the optimum number of hidden / input neurons. The tests were performed on a 2.8 GHz quad core processor, 12 GB RAM. Our system was implemented in Matlab 7.0.1 and used a single processor and 2 GB RAM.

The training process with pruning consists of two repetitive steps: *a)* train the network with early stopping (stop training, if the error on the validation set grows five times in a row), *b)* prune the network. Stop, if no further pruning is possible, and select the network with the lowest error on the test set to be the

Sensitivity-based training with scaled conjugate gradients and an enforced internal representation (SCGSIR):

1. Initialize the weight vector $\vec{w_1}$ with small random values, set scalars such that $0 < \sigma \leq 10^{-4}$, $0 < \lambda_1 \leq 10^{-8}$ and $\overline{\lambda}_k = 0$. Set $\vec{r}_1 = \vec{g}_1 = -H'(\vec{w}_1)$, $success = true$ and the discrete time variable k equal to 1.

2. If $success = true$, then calculate the second-order information:

$$\sigma_k = \frac{\sigma}{\| \vec{g_k} \|} \quad , \quad \vec{s}_k^{SCGSIR} = \frac{H'(\vec{w}_k + \sigma_k \vec{g}_k) - H'(\vec{w}_k)}{\sigma_k} \quad \text{and} \quad \delta_k^{SCGSIR} = \vec{g_k}^T \vec{s}_k^{SCGSIR}$$

3. Scale $\vec{s}_k^{SCGSIR}$, δ_k^{SCGSIR}:

$$\vec{s}_k^{SCGSIR} = \vec{s}_k^{SCGSIR} + (\lambda_k - \overline{\lambda_k})\,\vec{g}_k$$
$$\delta_k^{SCGSIR} = \delta_k^{SCGSIR} + (\lambda_k - \overline{\lambda_k}) \| \vec{g}_k \|^2$$

4. If $\delta_k^{SCGSIR} < 0$ then make the Hessian matrix positive definite by setting:

$$\vec{s}_k^{SCGSIR} = \vec{s}_k^{SCGSIR} + \left(\lambda_k - 2\frac{\delta_k^{SCGSIR}}{\| \vec{g}_k \|^2}\right)\vec{g}_k \quad , \quad \overline{\lambda}_k = 2\left(\lambda_k - \frac{\delta_k^{SCGSIR}}{\| \vec{g}_k \|^2}\right),$$
$$\delta_k^{SCGSIR} = -\delta_k^{SCGSIR} + \lambda_k \| \vec{g}_k \|^2 \quad \text{and} \quad \lambda_k = \overline{\lambda_k}$$

5. Calculate step size α_k^{SCGSIR}: $\mu_k = \vec{g}_k^T \vec{r}_k \quad ; \quad \alpha_k^{SCG} = \frac{\mu_k}{\delta_k^{SCGSIR}}$

6. Calculate the comparison parameter Δ_k^{SCGSIR}: $\Delta_k^{SCGSIR} = \frac{2\,\delta_k^{SCGSIR}\,[\,H(\vec{w}_k) - H(\vec{w}_k + \alpha_k^{SCGSIR}\vec{g}_k)\,]}{\mu_k^2}$

7. If $\Delta_k^{SCGSIR} \geq 0$ then a successful reduction in the value of the objective function H can be made:

$$\vec{w}_{k+1} = \vec{w}_k + \alpha_k^{SCGSIR}\vec{g}_k \,, \quad \vec{r}_{k+1} = -H'(\vec{w}_{k+1}) \,, \quad \overline{\lambda}_k = 0 \quad \text{and} \quad success = true$$

   (a) If $k \bmod N = 0$ then restart the algorithm by setting: $\vec{g}_{k+1} = \vec{r}_{k+1}$.

   else create a new conjugate direction: $\beta_k = \frac{\vec{g}_k^T \cdot (\vec{g}_k - \vec{g}_{k-1})}{\| \vec{g}_{k-1} \|^2}$ and $\vec{g}_{k+1} = \vec{r}_{k+1} + \beta_k \vec{g}_k$

   (b) If $\Delta_k^{SCG} > 0.75$ then reduce the scale parameter: $\lambda_k = \frac{1}{4}\lambda_k$.

   else reduction in error is not possible: $\overline{\lambda}_k = \lambda_k \quad ; \quad success = false.$

8. If $\Delta_k^{SCGSIR} < 0.25$ then increase the scale parameter: $\lambda_k = \lambda_k + \frac{\delta_k^{SCGSIR}\,(1 - \Delta_k^{SCGSIR})}{\| \vec{g}_k \|^2}$

9. If steepest descent direction $\vec{r}_k \neq 0$, set $k = k + 1$ and go to step 2)
   else terminate and return the weight vector $\vec{w}_k$ as the desired minimum for the objective function H.

final one. Two kinds of pruning were performed: *a)* pruning of the hidden neurons based on the internal representation [11], *b)* pruning of the hidden and input neurons based on the sensitivity analysis. For a neuron i from the hidden or input layer, let $S_{ij,p}$ be the sensitivity of the output neuron j to the activity of neuron i for the pattern p. If $\max_p mean_i S_{ij,p} < \beta mean_{ilp} S_{il,p}$ for a neuron j, then j is selected to be pruned. All neurons satisfying this condition are pruned at once. Parameter β was set to 0.7 in our experiments.

4.1 Binary addition of two 3-bit numbers

The data set consists of 320 examples with 18 bipolar input features and 4 bipolar output features. The data set is divided into the training set of the size 192, the validation set and the test set, both of the size 64. The first 6 input features are two three-bit binary numbers and the four bits of the output indicate the sum of the two binary numbers. Each of the 64 possible training patterns is present 3-times in the training set and once in the other two subsets. The other 12 input features are bipolar bits generated randomly with a uniform distribution. When adding $4(\sim (+1,-1,-1))$ and $7(\sim (+1,+1,+1))$ yielding the sum $11(\sim (+1,-1,+1,+1))$, the corresponding training pattern would have the form: $[[+1,-1,-1,+1,+1,+1,...],[+1,-1,+1,+1]]$.

All the trained networks had the topology 18-12-4. For each method, the algorithm was repeated 100-times with different network initializations. The parameter c_F was for the SCGIR and SCGSIR methods set experimentally to $2 * 10^{-5}$. c_G was for the SCGS-W and SCGSIR-W methods set to $2 * 10^{-5}$ and for the SCGS-S and SCGSIR-S methods to $-2 * 10^{-7}$. The average numbers of examples with incorrect outputs on the training, validation and test sets will be denoted by E_{tr}, E_v, E_t, respectively.

4.2 World Bank

The *World Bank* data set consists of 956 examples with 25 numerical input features encoding the WDI-indicators of 162 different countries from the years 2001-2006. The output features label the five classes of Income Groups. The other 10 input features were generated randomly with a uniform distribution. The results were obtained by the 10-fold cross-validation. All the networks had initial topology 35-50-4. The parameter c_F was for the SCGIR and SCGSIR methods set experimentally to $4 * 10^{-5}$. c_G was for the SCGS-W and SCGSIR-W methods set to $2 * 10^{-5}$. In this case, E_{tr}, E_v, E_t denotes the classification error on the training, validation and test sets, respectively. We also performed experiments, where we considered just 18 of the input features, which were selected by the *sens* method (see Section 4.3.4).

4.3 The first set of experiments

The goal of this experiment was to test the applicability of cluster based techniques to feature selection. Clustered data provide namely automatically an intrinsic equivalence class structure expected to yield improved generalization. In this context, various relevance measures have been tested.

In this experiment, the data set $\hat{T} = \{(\overrightarrow{x_1}, d_1), ..., (\overrightarrow{x_p}, d_P)\}$, $d_i \in \{1..cl\}$, where cl is the number of classes, is divided by the c-means clustering method into k clusters $C_1..C_k$. Let $\overrightarrow{c_1}..\overrightarrow{c_k}$ be the centroids of the clusters $C_1..C_k$. Let $O_l \in \{1..cl\}$ be the majority class for the examples in cluster C_l and vector $\overrightarrow{A_l} \in [0,1]^{cl}$, $A_{lj} = 1 \iff C_l = j$. For each example $\overrightarrow{x_i}, i \in \{1..P\}$ belonging to the cluster C_{l_i}, let $a_i = A_{l_i}$. Then $T = \{(\overrightarrow{x_1}, a_1), ..., (\overrightarrow{x_p}, a_P)\}$ is the new training set.

Several relevance measures may be used to select the subset $S_F \subset \{1..n\}$ of relevant input features. At first, the relevance R_{lj} of the feature j for the cluster C_l is computed for each input feature and each cluster:

$$R_{lj} = mean_{\{i, c_{k_i} = c_l\}} r_{ij},$$

where r_{ij} is for an example $\overrightarrow{x_i}$ and feature j one of the relevance measures listed below. Then, S_F is selected in the following way: For each cluster C_l, order the features according to R_{lj} in the descending order and select the first k_l features before a great fall in R_{lj}. If there occurr more possible splits, that one yielding less features is chosen.

4.3.1 Distance-relevance *(dist)*

For the example $\overrightarrow{x_i}$ belonging to the cluster C_{k_i} with the centroid $\overrightarrow{c_{k_i}}$ and the input feature j:

$$r_{ij} = -\frac{|x_{ij} - c_{k_i j}|}{\sum_{l \neq k_i} |x_{ij} - c_{lj}|}.$$

4.3.2 Minimum-relevance *(min)*

For the example $\overrightarrow{x_i}$ belonging to the cluster C_{k_i} with the centroid $\overrightarrow{c_{k_i}}$ and the input feature j:

$$r_{ij} = -min(1, \frac{|x_{ij} - c_{k_i j}|}{\min_{l \neq k_i} |x_{ij} - c_{lj}|})$$

for $\min_{l \neq k_i} |x_{ij} - c_{lj}| \neq 0$. Otherwise, $r_{ij} = -1$.

4.3.3 Maximum-relevance *(max)*

For the example $\overrightarrow{x_i}$ belonging to the cluster C_{k_i} with the centroid $\overrightarrow{c_{k_i}}$ and the input feature j:

$$r_{ij} = -min(1, \frac{\max_{l \neq k_i} (x_{ij} - c_{lj})^{-2}}{(x_{ij} - c_{k_i j})^{-2}})$$

for $\max_{l \neq k_i} (x_{ij} - c_{lj})^{-2} \neq \infty$. Otherwise, $r_{ij} = -1$.

4.3.4 Sensitivity relevance *(sens)*

A neural network is trained on T using the SCGIR algorithm with pruning based on the sensitivity analysis and condensed internal representation. For the input example $\overrightarrow{x_i}$ and input feature j, let $S_{jm,i}$ be the sensitivity of the output m on the input j. Then $r_{ij} = mean_m S_{jm,i}$.

4.3.5 Entropy-relevance *(entro)*

The relevance measure R_j does not depend on clustering. For examples $\overrightarrow{x_i}, \overrightarrow{x_j}$, let

$$D_{ij}(S_F) = \sqrt{\sum_{f \in S_F} \left(\frac{x_{if} - x_{jf}}{\max_h x_{hf} - \min_h x_{hf}} \right)},$$

$$\hat{D}_{ij}(S_F) = 2^{\frac{D_{ij}(S_F)}{mean_{i,j} D_{ij}(S_F)}},$$

$$\hat{R}(F) = -\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \Big[\hat{D}_{ij}(S_F) \log \hat{D}_{ij}(S_F) + (1 - \hat{D}_{ij}(S_F)) \log(1 - \hat{D}_{ij}(S_F)) \Big],$$

for the set of features $S_F \subset \{1..n\}$. For the feature j, the relevance measure is: $R_j = \hat{R}(\{1..n\}) - \hat{R}(\{1..n\} \setminus j)$.

Table 1 shows the input features detected by the tested feature selection techniques (based on k-means clustering of the *Binary addition* data into 8 clusters and of the *World Bank* data into 14 clusters) as the most relevant. The only method, that was for *Binary addition* able to identify reliably all the relevant input features $[1, 2, 3, 4, 5, 6]$, was *sens*. The other methods preferred irrelevant features instead.

Table 1: Features selected by the respective methods. The last two columns indicate the total number of selected features and the total number of selected random features for the World Bank data set.

	Binary adition	*World Bank*	
method	selected input features	total	random
sens	**[1 2 3 4 5 6]**	18	0
dist	[1 7 9 10 11 12 13 16]	17	0
min	[1 5 7 9 10 11 13 14 16]	18	1
max	[1 5 7 9 10 11 13 14 16 17 18]	18	1
entro	all	16	10

Table 2: Performance of the SCG, SCGIR, SCGS-S and SCGS-W, SCGSIR-S and SCGSIR-W methods on the Binary addition data set and 18-12-4 network architecture.

name	c_F	c_G	E_tr	E_v	E_t	n_{corr}	epochs	t(s)
SCG	–	–	3.05 ± 11.14	4.38 ± 9.92	4.60 ± 10.36	76	342.1	1.1
SCGIR	$2 * 10^{-5}$	–	3.04 ± 11.26	4.39 ± 10.06	4.31 ± 10.06	79	638.3	4.6
SCGIR	$5 * 10^{-5}$	–	3.15 ± 10.83	4.43 ± 9.35	4.77 ± 10.44	74	625.9	5.1
SCGIR	$5 * 10^{-4}$	–	6.6 ± 20.55	8.46 ± 12.4	8.15 ± 12.15	44	1271	10.4
SCGS-S	–	$-2 * 10^{-7}$	2.94 ± 11.09	4.20 ± 9.85	4.18 ± 10.04	77	409.5	141.6
SCGS-S	–	-10^{-7}	2.94 ± 11.09	4.20 ± 9.85	4.21 ± 10.04	75	333.9	107.2
SCGS-S	–	-10^{-6}	3.58 ± 11.99	5.46 ± 10.69	5.52 ± 11.14	64	329.8	105.5
SCGS-S	–	$-2 * 10^{-6}$	4.19 ± 13.22	6.83 ± 11.11	6.39 ± 11.38	27	181.4	65.0
SCGSIR-S	$2 * 10^{-5}$	$-2 * 10^{-7}$	3.15 ± 10.74	4.75 ± 9.47	5.03 ± 10.57	71	558.8	181.0
SCGSIR-S	$5 * 10^{-4}$	$-2 * 10^{-7}$	6.83 ± 20.58	8.91 ± 12.42	8.65 ± 12.42	42	826.8	269.8
SCGS-W	–	$2 * 10^{-5}$	$\mathbf{0.09 \pm 0.51}$	$\mathbf{0.69 \pm 2.54}$	$\mathbf{0.73 \pm 2.76}$	**86**	749.4	229.0
SCGS-W	–	$2 * 10^{-4}$	5.28 ± 5.00	6.91 ± 3.52	7.33 ± 3.93	1	220.9	55.3
SCGS-W	–	10^{-4}	0.94 ± 2.53	2.49 ± 3.21	3.02 ± 3.30	19	557.8	216.7
SCGS-W	–	$2 * 10^{-6}$	2.47 ± 9.25	3.04 ± 8.52	3.15 ± 8.72	84	432.6	127.4
SCGS-W	–	$2 * 10^{-7}$	2.96 ± 11.18	3.95 ± 9.55	4.02 ± 9.94	79	312.9	107.3
SCGSIR-W	$2 * 10^{-5}$	$2 * 10^{-5}$	0.15 ± 0.80	0.83 ± 2.71	0.89 ± 3.07	83	1248.8	328.2
SCGSIR-W	$5 * 10^{-4}$	$2 * 10^{-5}$	1.84 ± 7.15	3.52 ± 7.56	3.69 ± 7.83	57	954.4	372.1
SCG (6-12-4)	–	–	0 ± 0	0 ± 0	0 ± 0	100	1000	4.1
SCG (6-6-4)	–	–	20.76 ± 30.29	7.51 ± 10.65	7.59 ± 10.48	51	303.2	1.08
SCGS-W (6-6-4)	–	$2 * 10^{-5}$	19.20 ± 29.57	6.41 ± 9.87	6.43 ± 9.88	62	1804.4	423.9

4.4 The second set of experiments

This set of tests shall assess actual generalization capabilities of SCGS-trained networks in connection with the time and number of epochs required to converge. Extensive experiments performed both on the task of *Binary addition* and on the *World Bank* data set (Tables 2, 4 and 5) confirmed that the new technique of sensitivity inhibition improves generalization capabilities of trained networks while maintaining a relatively stable behavior – the average number of erroneously recalled input patterns over all of the 100 trained networks has been reduced 6-times while reducing its variance 4-times. The number of error-less networks raised 13%. Sensitivity enforcement showed in this respect only marginal improvements. The superior performance of the sensitivity inhibiting algorithms limits their rather slow convergence in comparison with the SCG-training algorithm.

4.5 The third set of experiments

Networks trained with the SCG-algorithm and an optimum architecture from scratch required a relatively high number of epochs (about five times as many) to converge while maintaining a much lower performance - only about 55% of the networks were trained without errors (Table 2). A natural question sounds then, if the SCGS and SCGSIR-trained networks should be more likely to develop an optimum architecture than their SCG-trained counterparts.

For both tasks tested, sensitivity inhibition increases significantly the chance to form networks with an optimum structure. From Table 3, 5 and 4 we can see that the average number of neurons is always lower for SCGSIR and its variants than for SCG. At the same time, the number of (error-less) networks with an optimum architecture is higher than for SCG-trained networks. Sensitivity-based training enables also easier pruning of redundant hidden neurons and does not affect much the number of epochs necessary to train/retrain the networks. Sensitivity inhibition with simultaneous enforcement of internal representation yields more transparent networks, in particular for the *World Bank* data.

Table 3: Performance of the SCG, SCGIR, SCGS-S and SCGS-W, SCGSIR-S and SCGSIR-W methods with pruning on the Binary adition data set and 18-12-4 network architecture.

name	n_H	n_I	n_{cH}	n_{cI}	E_{tr}	E_v	E_t	n_{corr}	epochs	t(s)
Pruning of hidden neurons										
SCG	8.17 ± 2.12	18	23	–	2.79 ± 11.99	2.71 ± 7.84	3.03 ± 8.52	80	647.2	2.1
SCGIR	7.78 ± 2.00	18	32	–	3.71 ± 13.51	2.64 ± 7.28	2.98 ± 8.42	85	948.3	7.3
SCGS-S	7.96 ± 2.07	18	28	–	2.66 ± 11.96	2.54 ± 7.78	2.75 ± 8.41	83	744.8	221.6
SCGSIR-S	7.97 ± 2.12	18	30	–	3.27 ± 12.60	2.49 ± 6.93	2.73 ± 7.87	84	985.0	251.3
SCGS-W	7.75 ± 2.15	18	**38**	–	$\mathbf{0.18 \pm 0.89}$	$\mathbf{0.49 \pm 1.92}$	$\mathbf{0.66 \pm 2.72}$	**91**	1138.0	327.1
SCGSIR-W	7.77 ± 2.15	18	37	–	0.37 ± 2.00	0.82 ± 3.13	0.79 ± 2.98	88	1522.6	419.2
Pruning of hidden neurons for SCGS-W and different values of c_G										
$2 * 10^{-4}$	7.73 ± 2.05	-	37	-	3.59 ± 3.80	3.72 ± 3.05	4.11 ± 3.33	13	436.9	96.6
$1 * 10^{-4}$	7.96 ± 2.25	-	34	-	0.67 ± 2.49	1.23 ± 2.1	1.59 ± 2.08	31	957.3	328.6
$2 * 10^{-6}$	7.86 ± 2.02	-	27	-	2.31 ± 10.73	1.87 ± 6.89	1.87 ± 6.85	86	758.6	206.1
$2 * 10^{-7}$	8.27 ± 2.21	-	23	-	2.62 ± 11.71	2.36 ± 7.44	2.65 ± 8.07	83	616.2	185.3
Pruning of input neurons										
SCG	12	6.67 ± 2.63	–	89	0 ± 0	0 ± 0	0 ± 0	100	1339.1	5.3
SCGIR	12	6.20 ± 1.26	–	94	0 ± 0	0 ± 0	0 ± 0	100	1029.2	8.7
SCGS-S	12	6.76 ± 2.86	–	91	0 ± 0	0 ± 0	0 ± 0	100	1114.4	215.1
SCGSIR-S	12	6.55 ± 2.37	–	91	0 ± 0	0 ± 0	0 ± 0	100	963.0	199.7
SCGS-W	12	$\mathbf{6 \pm 0}$	–	**100**	0 ± 0	0 ± 0	0 ± 0	100	1148.1	282.3
SCGSIR-W	12	$\mathbf{6 \pm 0}$	–	**100**	0 ± 0	0 ± 0	0 ± 0	100	1447.8	339.0
Pruning of input neurons for SCGS-W and different values of c_G										
$2 * 10^{-4}$	-	6.02 ± 0.2	-	99	3.24 ± 6.03	1.1 ± 2.06	1.13 ± 2.16	62	432.9	77.6
$1 * 10^{-4}$	-	6.0 ± 0.0	-	100	0.24 ± 1.1	0.08 ± 0.37	0.08 ± 0.37	95	745.9	227.8
$2 * 10^{-6}$	-	6.02 ± 0.2	-	99	0 ± 0	0 ± 0	0 ± 0	100	1043.6	197.1
$2 * 10^{-7}$	-	6.04 ± 0.24	-	97	0 ± 0	0 ± 0	0 ± 0	100	1173.2	200
Pruning of both input and hidden neurons										
SCG	7.74 ± 1.64	7.22 ± 3.49	23	85	0.01 ± 0.1	0.09 ± 0.90	0.13 ± 1.30	99	1343.1	4.5
SCGIR	7.40 ± 1.41	6.91 ± 3.08	29	88	0.03 ± 0.30	0.01 ± 0.10	0.01 ± 0.10	99	1583.5	11.4
SCGS-S	7.65 ± 1.56	7.42 ± 3.77	23	84	0 ± 0	0 ± 0	0 ± 0	100	1339.0	258.2
SCGSIR-S	7.56 ± 1.44	7.29 ± 3.61	23	84	0.15 ± 1.23	0.05 ± 0.41	0.05 ± 0.41	98	1506.4	284.9
SCGS-W	$\mathbf{6.98 \pm 1.24}$	$\mathbf{6.52 \pm 2.38}$	**43**	**94**	0 ± 0	0 ± 0	0 ± 0	100	1597.1	343.2
SCGSIR-W	$\mathbf{7.06 \pm 1.24}$	$\mathbf{6.50 \pm 2.37}$	**37**	**95**	0.07 ± 0.70	0.21 ± 2.10	0.19 ± 1.90	99	1850.3	451.5
Pruning of both input and hidden neurons for SCGS-W and different values of c_G										
$2 * 10^{-4}$	7.03 ± 1.11	6.14 ± 1.21	37	98	2.33 ± 8.24	0.77 ± 2.7	0.82 ± 2.79	82	558.3	108.3
$1 * 10^{-4}$	7.02 ± 1.26	6.61 ± 2.63	41	94	0.15 ± 1.23	0.09 ± 0.57	0.1 ± 0.64	97	1154.2	296.5
$2 * 10^{-6}$	7.36 ± 1.33	6.27 ± 1.33	29	91	0 ± 0	0 ± 0	0 ± 0	100	1384.9	252.9
$2 * 10^{-7}$	7.34 ± 1.23	6.07 ± 0.43	27	96	0 ± 0	0 ± 0	0 ± 0	100	1300	230.8
Minimal 6-6-4 architecture without pruning										
SCG	6	6	–	–	20.07 ± 27.74	6.69 ± 9.25	6.69 ± 9.25	55	4852.2	15.5
SCGS-W	6	6	–	–	13.68 ± 24.50	4.56 ± 8.17	4.56 ± 8.17	58	2329.9	215.8

Table 4: Performance of the SCG, SCGIR, SCGS-W, and SCGSIR-W methods with and without pruning on World bank data using the 35-50-5 network architecture.

name	n_{cH}	n_{cI}	E_tr	E_v	E_t	epochs	t(s)
Without pruning							
SCG	50	35	0.002 ± 0.002	0.019 ± 0.018	0.041 ± 0.020	115.3 ± 36.2	2.7 ± 1.1
SCGIR	50	35	0.002 ± 0.002	0.019 ± 0.018	0.044 ± 0.021	115.8 ± 36.7	8.0 ± 2.5
SCGS-W	50	35	0.001 ± 0.001	0.017 ± 0.014	$\mathbf{0.034 \pm 0.014}$	131.1 ± 29.8	452.5 ± 155.9
SCGSIR-W	50	35	0.001 ± 0.001	0.017 ± 0.014	$\mathbf{0.035 \pm 0.015}$	131.1 ± 30.6	399.1 ± 91.2
SCGS-S	50	35	0.003 ± 0.003	0.019 ± 0.015	0.044 ± 0.022	110.2 ± 36.9	349.6 ± 118.8
SCGSIR-S	50	35	0.002 ± 0.003	0.022 ± 0.020	0.043 ± 0.022	110.4 ± 37.6	340.9 ± 117.3
Pruning of hidden neurons							
SCG	41.9 ± 3.3	35	0.001 ± 0.001	0.026 ± 0.012	0.037 ± 0.021	280.5 ± 54.4	6.3 ± 1.2
SCGIR	$\mathbf{28.9 \pm 12.2}$	35	0.001 ± 0.001	0.025 ± 0.012	0.034 ± 0.021	387.9 ± 110.9	20.3 ± 4.6
SCGS-W	34.0 ± 10.5	35	0.001 ± 0.001	0.015 ± 0.009	$\mathbf{0.031 \pm 0.011}$	447.3 ± 187.4	1326.2 ± 512.4
SCGSIR-W	33.8 ± 11.5	35	0.003 ± 0.004	0.017 ± 0.012	$\mathbf{0.032 \pm 0.011}$	457.3 ± 161.8	1227.2 ± 403.8
SCGS-S	$\mathbf{33.8 \pm 10.5}$	35	0.002 ± 0.003	0.030 ± 0.016	0.033 ± 0.016	288.7 ± 84.0	792.9 ± 227.0
SCGSIR-S	$\mathbf{33.7 \pm 10.2}$	35	0.001 ± 0.001	0.032 ± 0.017	0.035 ± 0.018	302.6 ± 91.8	854.4 ± 253.3
Pruning of input neurons							
SCG	50	22.7 ± 5.1	0.009 ± 0.005	0.009 ± 0.009	0.028 ± 0.017	294.7 ± 79.0	9.9 ± 2.3
SCGIR	50	22.2 ± 2.0	0.009 ± 0.003	0.016 ± 0.010	0.035 ± 0.022	267.9 ± 97.4	20.3 ± 6.6
SCGS-W	50	$\mathbf{21.1 \pm 2.6}$	0.009 ± 0.003	0.015 ± 0.009	$\mathbf{0.024 \pm 0.020}$	308.7 ± 94.2	893.8 ± 286.3
SCGSIR-W	50	$\mathbf{21.2 \pm 2.1}$	0.008 ± 0.004	0.017 ± 0.010	$\mathbf{0.025 \pm 0.019}$	316.8 ± 105.8	787.1 ± 247.5
SCGS-S	50	21.6 ± 3.2	0.012 ± 0.003	0.016 ± 0.008	0.037 ± 0.023	257.1 ± 89.5	592.8 ± 198.5
SCGSIR-S	50	21.6 ± 3.2	0.013 ± 0.003	0.018 ± 0.010	0.035 ± 0.020	257.3 ± 88.7	590.3 ± 195.5
Pruning of both hidden and input neurons							
SCG	39.0 ± 3.8	22.3 ± 1.5	0.008 ± 0.005	0.013 ± 0.011	0.028 ± 0.015	493.1 ± 117.6	12.1 ± 2.4
SCGIR	27.7 ± 7.9	21.0 ± 2.4	0.011 ± 0.004	0.015 ± 0.013	0.028 ± 0.015	533.6 ± 154.9	27.2 ± 5.8
SCGS-W	$\mathbf{22.8 \pm 7.8}$	21.3 ± 1.9	0.011 ± 0.005	0.010 ± 0.011	$\mathbf{0.026 \pm 0.018}$	609.8 ± 167.2	1389.8 ± 470.0
SCGSIR-W	$\mathbf{23.7 \pm 11.9}$	21.5 ± 5.4	0.013 ± 0.009	0.015 ± 0.015	$\mathbf{0.027 \pm 0.018}$	620.6 ± 168.0	1272.6 ± 333.8
SCGS-S	31.9 ± 3.8	21.9 ± 1.8	0.013 ± 0.007	0.019 ± 0.012	0.035 ± 0.019	491.3 ± 127.7	956.8 ± 221.0
SCGSIR-S	29.5 ± 6.9	21.3 ± 2.1	0.019 ± 0.021	0.024 ± 0.015	0.033 ± 0.017	473.8 ± 115.7	974.3 ± 246.3

Table 5: Performance of the SCG, SCGIR, SCGS-W, and SCGSIR-W methods with and without pruning on World bank data using the 18-50-5 network architecture.

name	n_{cH}	n_{cI}	E_tr	E_v	E_t	epochs	t(s)
Without pruning							
SCG	50	18	0.013 ± 0.004	0.017 ± 0.008	0.034 ± 0.015	118.6 ± 23.2	2.1 ± 0.4
SCGIR	50	18	0.016 ± 0.009	0.027 ± 0.010	0.032 ± 0.010	122.5 ± 40.5	8.9 ± 2.6
SCGS-S	50	18	0.014 ± 0.004	0.019 ± 0.005	0.038 ± 0.018	113.9 ± 21.8	185.8 ± 36.3
SCGSIR-S	50	18	0.020 ± 0.009	0.032 ± 0.007	0.033 ± 0.019	114.3 ± 25.1	226.8 ± 53.5
SCGS-W	50	18	0.014 ± 0.005	0.023 ± 0.009	0.033 ± 0.019	157.4 ± 80.0	242.4 ± 122.8
SCGSIR-W	50	18	0.018 ± 0.009	0.023 ± 0.011	0.032 ± 0.012	126.5 ± 35.9	198.8 ± 97.5
Pruning hidden neurons							
SCG	36.9 ± 4.4	18	0.008 ± 0.003	0.032 ± 0.010	0.031 ± 0.018	320.5	7.5
SCGIR	27.6 ± 11.0	18	0.011 ± 0.004	0.031 ± 0.011	0.033 ± 0.019	342.0	19.4
SCGS-S	37.7 ± 5.8	18	0.010 ± 0.003	0.033 ± 0.015	0.031 ± 0.021	278.6	544.5
SCGSIR-S	26.9 ± 9.3	18	0.010 ± 0.004	0.026 ± 0.014	0.032 ± 0.023	319.0	625.6
SCGS-W	30.3 ± 7.4	18	0.006 ± 0.003	0.022 ± 0.011	0.029 ± 0.013	535.6	726.5
SCGSIR-W	25.2 ± 15.0	18	0.013 ± 0.004	0.016 ± 0.011	0.031 ± 0.009	497.3	610.2
Pruning of input neurons							
SCG	50	16.3 ± 1.6	0.018 ± 0.008	0.027 ± 0.017	0.034 ± 0.014	276.2	9.8
SCGIR	50	15.5 ± 2.5	0.018 ± 0.008	0.033 ± 0.006	0.042 ± 0.020	329.5	25.8
SCGS-S	50	15.9 ± 2.3	0.019 ± 0.008	0.030 ± 0.014	0.034 ± 0.017	322.4	449.1
SCGSIR-S	50	15.9 ± 2.0	0.021 ± 0.009	0.028 ± 0.011	0.033 ± 0.014	298.0	508.9
SCGS-W	50	16.3 ± 2.4	0.019 ± 0.010	0.024 ± 0.011	0.032 ± 0.012	294.0	405.8
SCGSIR-W	50	16.9 ± 1.3	0.013 ± 0.005	0.027 ± 0.013	0.032 ± 0.020	384.5	544.4
Pruning of both hidden and input neurons							
SCG	41.1 ± 4.6	18.0 ± 0	0.010 ± 0.003	0.027 ± 0.012	0.033 ± 0.016	1025.4	13.4
SCGIR	30.3 ± 11.3	17.1 ± 1.912	0.013 ± 0.007	0.033 ± 0.013	0.037 ± 0.021	570.7	27.4
SCGS-S	37.5 ± 6.2	16.4 ± 1.8	0.014 ± 0.007	0.032 ± 0.017	0.037 ± 0.022	471.2	786.5
SCGSIR-S	32.5 ± 8.7	17.5 ± 1.6	0.012 ± 0.007	0.031 ± 0.015	0.032 ± 0.014	572.3	842.6
SCGS-W	35.7 ± 4.0	17.7 ± 1.0	0.009 ± 0.004	0.024 ± 0.014	0.028 ± 0.012	662.3	821.2
SCGSIR-W	34.3 ± 7.7	18.0 ± 0	0.012 ± 0.003	0.022 ± 0.013	0.032 ± 0.017	577.4	791.6

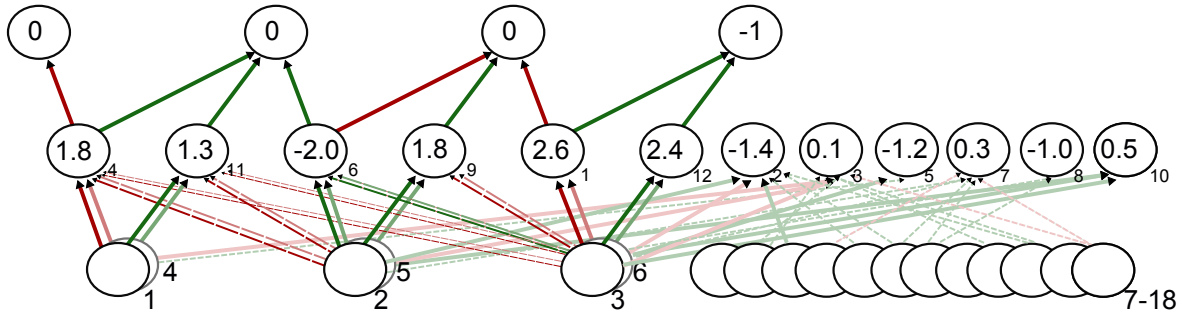

Figure 1: A typical network trained on Binary addition data by the SCGS-W method on the 18-12-4 network architecture (before pruning). The numbers inside the neurons correspond to thresholds.

In Figure 1 and 2 we can see a typical network formed using the sensitivity inhibition SCGS-W method (before pruning and after pruning). Red edges are negative, green edges are positive. In Fig. 1, the 6 hidden neurons on the right are redundant (with almost zero edges to all of the outputs) and can be pruned immediately. The 12 input neurons on the right are also redundant, as there are nearly zero edges from these inputs to the relevant hidden neurons, and can be pruned next. The remaining neurons form an optimal architecture that solves the given task of binary addition. For the standard SCG method, the difference in the sensitivity between relevant and redundant inputs /neurons is not always so clear and pruning is thus more difficult.

In Figure 2, we can clearly see that the pruned network has succeeded in finding the actual computing algorithm. Moreover, the weights between its input and hidden layer are pairwise equal for the corresponding input neurons 1 and 4, 2 and 5, and 3 and 6. The first, third and fifth hidden neurons compute the 'carry' for higher output bits. The second, fourth and sixth hidden neurons also compute similar functions for single output neurons.

When compared with SCG, sensitivity enforcement exhibited again only marginal improvements in all of the above-considered aspects.

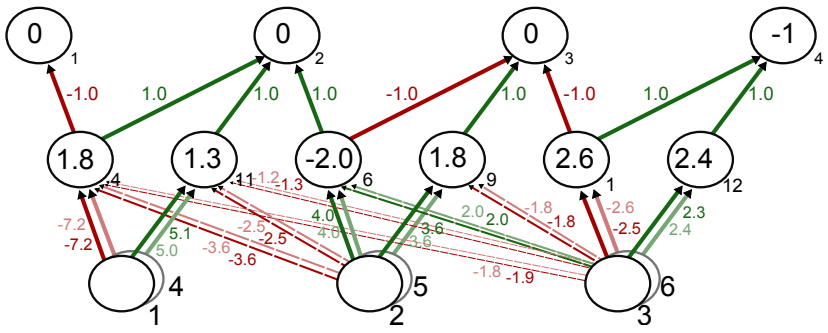

Figure 2: A typical network trained on Binary addition by the SCGS-W method on the 18-12-4 network architecture (after pruning). Each edge is weighted. The numbers inside the neurons correspond to thresholds.

5 Conclusions

Most 'classical' techniques used to train BP-networks usually achieve a good training accuracy by forming very complex decision boundaries involving time-consuming training and excessively large network architectures impacting worse generalization. For this reason, we have proposed a new sensitivity-based SCGSIR-training algorithm inspired by the general idea of scaled conjugate gradients.

In spite of its higher computational complexity, the algorithm usually converges within a reasonable amount of time. The main advantage of the SCGSIR-training method consist, nevertheless, in superior generalization and an outstanding capability to form transparent network structures with optimum architecture. In this respect, the new SCGSIR-training algorithm outperforms both the standard SCG-technique and the SCGIR-method. However, the proposed model should be tested more extensively also for networks with more hidden layers and on larger data sets.

The framework developed to enforce suitable internal representations and simultaneously control the sensitivity of the network to its inputs reflects the basic concept of learning with hints [1] and is expected to induce networks with lower VC-dimensions. As networks with inhibited sensitivities benefit also from smoother network functions, their behavior proved to be even more stable than the other tested algorithms. Anyway, the right choice of the trade-off coefficients applied during training can impact the quality of the solution obtained. Too large values of these coefficients might namely result into BP-networks with a perfectly formed condensed internal representation (in)sensitive to any inputs yet incapable of approximating the desired function because of saturated hidden neuron outputs.

Networks with fewer hidden and input neurons are characterized by a lower VC-dimension implying better generalization capabilities of the trained network [17], [18]. The greatest appeal of the new SCGSIR method consists thus in its ability to form networks with a transparent network structure that supports reliable pruning and improves generalization. We are, however, fully aware of the necessity to test the proposed model more extensively also for networks with more hidden layers and on larger data sets comprising several thousands of patterns, in order to provide statistically significant results. Further, we plan to investigate also the effects of the chosen trade-off parameters on the performance of trained networks.

References

[1] Y. S. Abu-Mostafa, "Hints and the VC Dimension," *Neural Computation*, vol. 5, pp. 278–288, 1993.

[2] Y. S. Abu-Mostafa, "Learning from Hints," *Journal of Complexity*, vol. 10, pp. 165–178, 1994.

[3] C. M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, Cambridge, UK, 1996.

[4] J. A. Bullinaria, "Evolving efficient learning algorithms for binary mappings," *Neural Networks*, vol. 16, pp. 793–800, 2003.

[5] E. Castillo, B. Guijarro-Berdiñas, O. Fontenla-Romero, and A. Alonso-Betanzos, "A Very Fast Learning Method for Neural Networks Based on Sensitivity Analysis," *Journal of Machine Learning Research*, vol. 7, pp. 1159–1182, 2006.

[6] J. N. Fidalgo, "Feature subset selection based on ANN sensitivity analysis - a practical study," *N. Mastorakis (ed.): Advances in Neural Networks and Applications*, WSES Press, pp. 206–211, 2001.

[7] B. Guijarro-Berdiñas, O. Fontenla-Romero, B. Pérez, and A. Alonso-Betanzos, "A Regularized Learning Method for Neural Networks Based on Sensitivity Analysis," *Proc. of ESANN 2008*, 289–294, 2008.

[8] K. Hara, M. Okada, "On-line learning through simple perceptron learning with a margin," *Neural Networks*, vol. 17, pp. 215–223, 2004.

[9] M. Møller, "A scaled conjugate gradient algorithm for fast supervised learning," *Neural Networks*, vol. 6, pp. 525–533, 1993.

[10] I. Mrázová, Z. Reitermanová, "Enforced knowledge extraction with BP-networks," *Intelligent Engineering Systems through Artificial Neural Networks*, vol. 17, ASME Press, New York, pp. 285–290, 2007.

[11] I. Mrázová, D. Wang, "Improved generalization of neural classifiers with enforced internal representation," *Neurocomputing*, vol. 70, no. 16-18, pp. 2940–2952, 2007.

[12] R. Reed, R. J. Marks II, "Neurosmithing: Improving Neural Network Learning," *M. A. Arbib (Ed.): The Handbook of Brain Theory and Neural Networks*, The MIT Press, pp. 639–644, 1998.

[13] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, (323), pp. 533–536, 1986.

[14] R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee, "Boosting the margin: A new explanation for the effectiveness of voting methods," *The Annals of Statistics*, vol. 26 (5), pp. 1651–1686, 1998.

[15] N. E. Sharkey, "Connectionist Representation Techniques," *AI Review*, vol. 5, pp. 143–167, 1990.

[16] J. Sietsma, R. J. F. Dow, "Creating artificial neural networks that generalize," *Neural Networks*, vol. 4, pp. 67–79, 1991.

[17] M. Solazzi, A. Uncini, "Regularising neural networks using flexible multivariate activation function," *Neural Networks*, vol. 17, pp. 247–260, 2004.

[18] V. N. Vapnik, *The Nature of Statistical Learning Theory*, Springer Verlag, Berlin, Germany, 2000.

[19] *World Development Report 2000/2001: Attacking Poverty*, The World Bank Group, Washington, D. C., 2001.

[20] L. Xu, "Data smoothing regularization, multi-sets-learning, and problem solving strategies," *Neural Networks*, vol. 16, pp. 817–825, 2003.