Chapter 5

Architecture selection

Architecture selection is critical to NN modeling where the objective is to find the smallest network that accurately maps the true function described by the training data. Architecture selection has to reduce network complexity while maintaining good generalization. A network that is too large may lead to overfitting of the training data resulting in poor generalization when presented with similar but slightly different data. If the network is too small, underfitting may occur that results in poor approximation of the function [Baum *et al* 1989, Le Cun 1989].

The objectives of pruning are usually motivated by two aims: to obtain networks of a *small size* and with a *good generalization performance*. The objective is to find a minimal network topology. It is usually not obvious what the smallest network with the best generalization is for a particular task. Different approaches have been devised to solve this problem.

Architecture selection approaches can be grouped in four categories, i.e. brute-force approaches, regularization, network growing (network construction) and pruning. While these topics have been introduced in section 2.17, this chapter focuses on pruning.

Pruning starts training with a neural network which is expected to be big enough to ensure successful training. At convergence of the oversized network, weights and/or units that are irrelevant or redundant are removed, upon which the pruned network is retrained [Thimm *et al* 1995]. If the retraining converges then the removal-retraining cycle is resumed. If the retraining fails, the smallest network that satisfied the convergence criterion is assumed to have the most suitable topology for the given data set. The decision to prune a network is based on some measure of parameter (i.e. a unit or weight) relevance. A relevance is computed for each parameter and a pruning heuristic is applied to determine whether a parameter is irrelevant or not. Numerous pruning algorithms have been proposed, including the following,

- The Smallest Variance, $(min(\sigma))$, method of Sietsma and Dow that removed connections with smallest contribution variance on the training set, where the contribution of a connection is the value available to the connection from the lower layer, multiplied by its weight. The mean output of the removed connection is then added to the corresponding bias [Sietsma *et al* 1991].

- Skeletonization, which is a weight removal method, defines a measure of the relevance of a unit as the error when the unit is removed from the network, minus the error when the unit is left in the network [Mozer *et al* 1989]. This is accomplished by multiplying the output of a unit j by a coefficient, α_j, that represents the attentional strength of the unit [Mozer *et al* 1989]. In the case of hidden units,

$$o_k = f(\sum_{j}^{J+1} w_{kj}\alpha_j y_j) \tag{5.1}$$

where $f(\cdot)$ is the activation function, o_k is the activation of output unit O_k, w_{kj} is the weight between hidden unit Y_i and output unit O_k and y_j is the activation of hidden unit Y_j. If $\alpha_j = 0$, unit Y_j has no influence on the rest of the network. If $\alpha_j = 1$, unit Y_j is a conventional unit. The units are then removed for which

the derivative of the error function to these attentional strengths, α_j's, are small [Thimm *et al* 1995]. Skeletonization can also be applied to prune input units.

- Karnin's method that estimates the sensitivity s of a weight by :

$$s = \sum_{n=1}^{N} (\Delta w(n)^2 \cdot \frac{w(n)}{\eta \cdot (w(n) - w(0))}) \tag{5.2}$$

where $w(n)$ is the weight in the current training epoch n, $w(0)$ the initial weight, and $\Delta w(n)$ the weight change in the n^{th} epoch [Karnin 1990]. The denominator in this formula can become zero, and experiments have indeed shown this to happen. This problem is not dealt with in Karnin's publication. It can, however, easily be solved by setting the whole fraction to zero. The calculation for s then becomes,

$$s = \begin{cases} \sum_{n=1}^{N} (\Delta w(n)^2 \cdot \frac{w(n)}{\eta \cdot (w(n) - w(0))}) & \text{if } w(n) \neq w(0) \\ 0 & \text{if } w(n) = w(0)) \end{cases} \tag{5.3}$$

- Autoprune developed by Finnof *et al*, where a test statistic is defined based on the probability that a weight becomes zero [Finnoff 1993b]. A weight is then removed if the probability that it will become zero is high. Prechelt extended Autoprune to $\lambda - prune$ to calculate the number of units to be pruned at each pruning step [Prechelt 1994].

- Genetic algorithms (GAs) also provide a biological plausible approach to pruning of NNs [Whitley *et al* 1990]. The GA is populated with several pruned versions of the original network. Each of these networks must be trained separately. In this type of pruning genetic operators such as mutation, reproduction and cross-over are applied to create differently pruned networks. These pruned networks 'compete' for survival, being awarded for using fewer parameters and for improving generalization. A drawback of the GA approach to pruning of neural networks is that it is time consuming.

- The Variance Nullity method developed by Engelbrecht is a computationally efficient pruning heuristic based on variance analysis of sensitivity information [Engelbrecht *et al* 1999c, Engelbrecht 2001]. This algorithm utilizes first-order derivatives of the NN output with respect to parameter perturbations, which are already calculated when gradient descent is used for neural network training.

- Optimal Brain Damage (OBD), Optimal Brain Surgeon (OBS) and Optimal Cell Damage (OCD) are all based on second-order derivatives of the 'objective function' with respect to parameter perturbations. In OBD and OCD complexity is reduced by assuming that (a) the function is well approximated by a second-order expansion around its minimum, (b) the off-diagonal elements of the Hessian matrix are zero and (c) all errors between the target and output values are zero. In OBS assumption (b) mentioned above is removed and also is retraining after pruning avoided by automatically adjusting the remaining weights. OBD, OBS and OCD all require differentiable activation functions. Criticism concerning assumption (c) is that outliers in the data nullify the assumption. The calculation of the Hessian in OBD, OBS and OCD increases the complexity of these pruning methods. In OBS the complexity is further increased since the inverse of the Hessian must also be calculated.

The only assumptions for variance nullity pruning method are,

1. that the network must be well trained and

2. that the activation functions must at least be once differentiable.

Also, this algorithm is not as computational intensive as other pruning algorithms. For this reason the 'Variance Nullity pruning Method' is the algorithm of choice to be implemented in this thesis. The next section provides an overview of sensitivity analysis.

5.1 Overview of Sensitivity Analysis

Research has shown that any continuous function can be approximated by a multilayer NN using a monotonically increasing differentiable activation function [Funahashi 1989, Hornik *et al* 1989a]. Gallant *et al* further showed that when a NN converges towards the underlying (target) function, then all the NN derivatives also converge towards the derivatives of the underlying function [Gallant 1992]. This property of NNs derivatives allows efficient use of the NN derivatives to compute sensitivity information. Sensitivity analysis of a system is the study of how the derivatives of a performance function can be used to quantify the response of the system to parameter perturbations [Holtzman 1992]. Thus, sensitivity analysis techniques quantify the relevance of a network parameter (i.e. an input unit, hidden unit or weight) as the influence that small parameter perturbations have on a performance function [Engelbrecht 2001]. Sensitivity analysis also provides a neural network tool to automatically identify all relevant parameters using the significance measures obtained from a sensitivity analysis tool.

There are two main approaches to sensitivity analysis for feed-forward neural networks (FFNNs). These approaches differ in the performance function used. In the one approach the *objective function* to be minimized serves as the performance function, in the second approach it is the neural network *output function*. Objective function sensitivity analysis has been used widely in pruning of NN parameters [Hassibi *et al* 1994, Le Cun *et al* 1990], to develop more sophisticated optimization techniques [Battiti 1992], and to study the robustness and stability of NNs [Oh *et al* 1995]. NN output sensitivity analysis on the other hand has been used to study the generalization abilities of FFNNs [Fu *et al* 1993], to assess the significance of input parameters [Engelbrecht *et al* 1995b], for selective learning and incremental

learning [Engelbrecht *et al* 1999d] and for pruning irrelevant network parameters [Engelbrecht 2001, Zurada *et al* 1997]. In OBD, OBS and OCD sensitivity analysis is performed with regards to the training error. Assuming gradient descent optimization and the sum-squared objective function, Engelbrecht has shown that output sensitivity analysis and OBD are conceptually the same under the assumptions of OBD [Engelbrecht 2001]. Output sensitivity analysis has the advantages that it is not based on assumptions to simplify complexity, as is the case with OBD, OBS and OCD. Also, output sensitivity analysis is less complex than objective function sensitivity analysis. Furthermore, objective function sensitivity analysis is dependent on the objective function and the optimization algorithm used to update the weights. while output sensitivity does not depend on the objective function or the optimization algorithm.

The next section describes the variance nullity pruning algorithm of Engelbrecht. The pruning algorithm is subsequently applied to prune oversized PUNNs used to learn the test functions of section 4.4.1 on page 99.

5.2 The Variance Nullity Pruning Approach

The variance nullity pruning algorithm of Engelbrecht is based on NN output sensitivity where the relevance of parameters is based on parameter sensitivity information [Engelbrecht 2001]. In this algorithm a variance nullity measure is computed for each parameter. The statistical nullity in parameter sensitivity variance is defined in equation (5.4). Thus, the variance nullity measure provides a statistically sound mechanism to decide whether or not a unit or weight is pruned. The objective of the variance nullity measure is to test whether the variance in parameter sensitivity for the different patterns is significantly different from zero [Engelbrecht *et al* 1999c]. If the latter is not the case, then it indicates that the corresponding parameter has little

or no effect on the output of the NN over the entire set of patterns presented to the network. A hypothesis testing step developed by Engelbrecht *et al* uses these variance nullity measures to statistically test if a parameter should be pruned, using the χ^2 distribution [Engelbrecht *et al* 1999c, Engelbrecht 2001].

Engelbrecht *et al* defines statistical nullity in parameter sensitivity variance, Υ_{θ_i}, of a NN parameter θ_i over patterns $p = 1, ..., P$ as follows:

$$\Upsilon_{\theta_i} = \frac{(P-1)\sigma^2_{\theta_i}}{\sigma^2_0} \tag{5.4}$$

where $\sigma^2_{\theta_i}$ is the variance of the sensitivity of the network to perturbations in parameter θ_i, σ^2_0 is a value close to zero and P the number of patterns in the pruning set.
The variance in parameter sensitivity, $\sigma^2_{\theta_i}$, is computed as

$$\sigma^2_{\theta_i} = \frac{\sum_{p=1}^{P}(\aleph^{(p)}_{\theta_i} - \overline{\aleph}_{\theta_i})^2}{P-1} \tag{5.5}$$

where

$$\aleph^{(p)}_{\theta_i} = \frac{\sum_{k=1}^{K} S^{(p)}_{O\theta,ki}}{K} \tag{5.6}$$

and $\overline{\aleph}_{\theta_i}$ is the average parameter sensitivity over all patterns $p = 1, ..., P$, i.e.

$$\overline{\aleph}_{\theta_i} = \frac{\sum_{p=1}^{P} \aleph^{(p)}_{\theta_i}}{P} \tag{5.7}$$

$S_{O\theta}$ refers to the sensitivity matrix of the output vector $\vec{o}$ with respect to the parameter vector $\vec{\theta}$, and individual elements $S_{O\theta,ki}$ refers to the sensitivity of output o_k to perturbations in parameter θ_i over all patterns; $S^{(p)}_{O\theta,ki}$ refers to the sensitivity of output o_k to changes in parameter θ_i for a single pattern p, defined as (assuming differentiable activation functions)

$$S^{(p)}_{O\theta,ki} = \frac{\partial o_k}{\partial \theta^{(p)}_i} \tag{5.8}$$

where $\theta_i^{(p)}$ is the activation value of unit θ_i for pattern p. Section 5.3 derives the sensitivity equations with respect to input and hidden units. In equation (5.6), $\aleph_{\theta_i}^{(p)}$ is the average sensitivity of the NN output to perturbations in parameter θ_i for pattern p over the K output units.

In 'autoprune', developed by Finnoff *et al*, the final weight test variables are based on significance tests for deviations from zero in the weight update process [Finnoff 1993b]. Weights are updated using,

$$\Delta w_h^p(w) = -\eta \cdot \left(\frac{\partial E_p(w)}{\partial w_h} \right) \tag{5.9}$$

where the above denotes the local gradient of the error with respect to pattern p and weight w_h. The results of further training were estimated using an average over the variables ξ_h^p, where

$$\xi_h^p = w_h + \Delta w_h^i(w) \tag{5.10}$$

for $(z_p, t_p) \in \mathcal{D}_t$. For the null hypothesis that the expected value of variable ξ_h^p is equal to zero; the significance of the deviation from zero was tested using the test variable,

$$T_h = \frac{\left| \sum_{p,(z_p,t_p)\in(D)_t} \xi_h^p \right|}{\sqrt{\sum_{p,(z_p,t_p)\in\mathcal{D}_t} (\xi_h^p - \overline{\xi}_h)^2}} \tag{5.11}$$

where $\overline{\xi}_h$ denotes the average over the set ξ_h^p and $(z_p, t_p) \in \mathcal{D}_t$. A large value for T_h indicates high importance of the connection with weight h_p. Connections with small weights can be pruned. In the analysis of means, as is done by Finnoff *et al* a problem may arise where large negative and positive values may cancel each other or produce a sum close to zero, thus incorrectly indicating that the parameter is insignificant. In variance analysis pruning, Engelbrecht *et al* adopted an analysis of variance instead of an analysis of means, as is done by Finnof *et al*, to address this problem.

Basically the statistical pruning heuristic of Engelbrecht is based on proving or disproving the null hypothesis that the variance in parameter sensitivity is approximately zero.

The null hypothesis is then defined as

$$\mathcal{H}_0 : \sigma^2_{\theta_i} = \sigma^2_0 \tag{5.12}$$

This hypothesis can however not be used, since equation (5.4) does not allow $\sigma^2_0 = 0$, and therefore it cannot be hypothesized that the variance in parameter sensitivity over all patterns is exactly zero. To alleviate this problem a small value close to zero is chosen for σ^2_0, and the alternative hypothesis,

$$\mathcal{H}_1 : \sigma^2_{\theta_i} < \sigma^2_0 \tag{5.13}$$

is tested. The variance nullity measure defined in equation (5.14) has a $\chi^2(P-1)$ distribution in the case of P patterns. The critical value, Υ_c, can therefor be obtained from χ^2 distribution tables, i.e.

$$\Upsilon_c = \chi^2_{v;1-\alpha} \tag{5.14}$$

where $v = P - 1$ is the number of degrees of freedom and α is the level of significance. A significance level $\alpha = 0.01$, for example, means that we are satisfied with incorrectly rejecting the hypothesis once out of 100 times. Using the critical value defined in equation (5.14), if $\Upsilon_{\theta_i} \leq \Upsilon_c$, the alternative hypothesis $\mathcal{H}$ is accepted and parameter θ_i is pruned. Engelbrecht *et al* pointed out that the success of this pruning heuristic depended on the value of σ^2_0. A too small value for σ^2_0 will result in no parameters to be pruned. If σ^2_0 is too large, then important parameters may be pruned. It was recommended that the algorithm should start off with a small value for σ^2_0 that is gradually increased if no parameter is pruned. The performance of the network is first tested after each step of pruning. If the performance of the network has not degraded too much, the pruned network is accepted, otherwise the original network is restored and pruning is stopped. Engelbrecht pointed out that the testing of the performance of the pruned network makes the validity of the algorithm insensitive to the value with

which σ_0^2 is increased: if relevant parameters are pruned due to the repetitive increase in σ_0^2, the performance of the network will degrade unacceptably, and the previous architecture will thus be restored.

Computational time during the hypothesis testing phase can be reduced by arranging the variance nullity measures Υ_{θ_i} in increasing order. Hypothesis tests start on the smallest Υ_{θ_i} and continue until no more parameters can be identified for pruning.

The statistical pruning heuristic based on variance nullity is summarized below:

1. Initialize the NN architecture and learning parameters
2. Repeat
   (a) train the NN until overfitting is observed
   (b) let $\sigma_0^2 = 0.0001$
   (c) for each θ_i
      i. for each $p = 1, ..., P$, calculate $\aleph_{\theta_i}^{(p)}$ using equation (5.6)
      ii. calculate the average $\overline{\aleph}_{\theta_i}$ using equation (5.7)
      iii. calculate the variance in parameter sensitivity using $\sigma_{\theta_i}^2$ from equation (5.5)
      iv. calculate test variable Υ_{θ_i} using equation (5.4)
   (d) apply the pruning heuristic
      i. arrange Υ_{θ_i} in increasing order
      ii. find Υ_c using equation (5.14)
      iii. for each θ_i, if $\Upsilon_{\theta_i} \leq \Upsilon_c$, then prune θ_i
      iv. if $\Upsilon_{\theta_i} > \Upsilon_c$ for all θ_i, let $\sigma_0^2 = \sigma_0^2 \times 10$

   until no θ_i is pruned, or the reduced network is not accepted due to an unacceptable deterioration in generalization performance

3. Train the final pruned NN architecture

The variance nullity algorithm starts pruning the hidden layer first, followed by the input layer. Weights can also be pruned once the irrelevant units have been removed. The calculation of the nullity measures can be done on any one of the training, test or validation sets. In this thesis a separate set consisting of 100 randomly generated values was used to calculate variance nullity measures. Pruning is initiated when overfitting is detected on the validation set, i.e. when $\xi_V > \overline{\xi}_V + \delta_{\xi_V}$ where ξ_V is the current error on the validation set, $\overline{\xi}_V$ is the average error on the validation set over the previous iterations and δ_{ξ_V} is the standard deviation in test error. After each pruning step, retraining starts on the reduced network on new initial random weights. The pruning process stops when no more parameters can be identified for pruning, or if the reduced network's performance has degraded too much.

The next section derives the sensitivity equations that are used to calculate the variance nullity measures.

5.3 Sensitivity equations

This section defines equations for the sensitivity analysis of output units with respect to hidden units and input units. It is assumed that the network consists of an input layer, a single hidden layer of product units and an output layer of summation units. Linear activation functions are assumed in both hidden and output layers.

5.3.1 Output-Hidden Layer Analysis

For the sake of notational convenience the superscript p, that refers to a specific pattern, is removed. Let $S_{OY,kj} = \frac{\partial o_k}{\partial y_j}$ be the sensitivity of output unit o_k to small perturbations in hidden hidden unit y_j for a single pattern (The first part of the subscript indicates the layer involved and the second part indicates the respective unit of each layer). Then,

$$\begin{aligned} S_{OY,kj} &= \frac{\partial o_k}{\partial y_j} \\ &= \frac{\partial o_k}{\partial net_{o_k}} \frac{\partial net_{o_k}}{\partial y_j} \\ &= f'(net_{o_k}) \cdot w_{kj} \\ &= w_{kj} \end{aligned} \tag{5.15}$$

where $f'(net_{o_k}) = 1$ for linear activation.

5.3.2 Output-Input Layer Analysis

The sensitivity of output unit O_k with respect to input unit Z_i is calculated as,

$$\begin{aligned} S_{OZ,ki} &= \frac{\partial o_k}{\partial z_i} \\ &= \frac{\partial o_k}{\partial net_{o_k}} \frac{\partial net_{o_k}}{\partial z_i} \end{aligned} \tag{5.16}$$

where, for linear activation,

$$\begin{aligned} o_k &= f(net_{o_k}) \\ &= net_{o_k} \end{aligned} \tag{5.17}$$

Then

$$\begin{aligned} \frac{\partial o_k}{\partial z_i} &= \frac{\partial o_k}{\partial net_{o_k}} \frac{\partial net_{o_k}}{\partial z_i} \\ &= f'(net_{o_k}) \cdot \frac{\partial net_{o_k}}{\partial z_i} \end{aligned}$$

$$
\begin{aligned}
&= 1 \cdot \frac{\partial net_{o_k}}{\partial z_i} \\
&\quad \sum_{j=1}^{J} \frac{\partial net_{o_k}}{\partial y_j} \cdot \frac{\partial y_j}{\partial z_i} \\
&= \sum_{j=1}^{J} w_{kj} \cdot \frac{\partial y_j}{\partial z_i} \qquad (5.18)
\end{aligned}
$$

For a PUNN with a distortion unit, using equation (A.45) on page 200, we have

$$y_j = e^{\rho} \cdot \cos(\pi\phi) \qquad (5.19)$$

where

$$\rho = \sum_{i=1}^{I+1} v_{ji} \ln |z_i|$$

$$\phi = \sum_{i=1}^{I+1} v_{ji} \mathcal{I}_i$$

Thus,

$$
\begin{aligned}
\frac{\partial y_j}{\partial z_i} &= \frac{\partial}{\partial z_i}(e^{\rho} \cdot \cos(\pi\phi)) \\
&= e^{\rho} \frac{\partial \rho}{\partial z_i} \cdot \cos(\pi\phi) + \frac{\partial \cos(\pi\phi)}{\partial z_i} \\
&= e^{\rho} \cdot \frac{\partial (\sum_{i=1}^{I+1} v_{ji} \ln |z_i|)}{\partial z_i} \cdot \cos(\pi\phi) + \frac{\partial \cos(\pi \sum_{i=1}^{I+1} v_{ji} \mathcal{I}_i)}{\partial z_i} \\
&= e^{\rho} \cdot \frac{v_{ji}}{|z_i|} \cdot \cos(\pi\phi) \\
&= \frac{v_{ji}}{|z_i|} \cdot (e^{\rho} \cdot \cos(\pi\phi)) \qquad (5.20)
\end{aligned}
$$

Substitution of (5.20) in (5.18) gives

$$\frac{\partial o_k}{\partial z_i} = \sum_{j=1}^{J} w_{kj} \frac{v_{ji}}{|z_i|} \cdot (e^{\rho} \cdot \cos(\pi\phi)) \qquad (5.21)$$

This concludes the derivation of the sensitivity equations for a PUNN with a distortion unit.

The next section applies the variance nullity pruning algorithm to PUNNs for selected function approximation problems.

5.4 Application of the Variance Nullity Pruning Algorithm to PUNNs

The variance nullity pruning algorithm was applied to the eight test functions described on page 99. The training and test sets of section 4.4.1 were used in training the network. The PUNNs were trained using particle swarm optimization. The optimal parameters for each of the eight test functions, as determined in chapter 4, were used for oversized initial networks. In these oversized networks, the number of hidden units were deliberately increased from the optimal architectures as determined in chapter 4. After each pruning step the weights were randomly re-initialized, as stipulated by the variance nullity pruning algorithm. In the case of PSO this implies re-initializing the positions and velocities for the particles before the next pruning step commenced. This random re-initialization of weights sometimes resulted in poorer performance of the re-initialized network by often producing larger MSEs on the training set and poorer generalization on the test set.

In this section a network's performance was measured by its MSE on the test set, in other words, its generalization. Tables 5.1 to 5.8 contain for both the oversized and pruned network the number of hidden units, MSE on training and test sets for 30 simulations, the average MSE on training and tests sets and the average number of hidden units. Unacceptable performance was defined as a reduction of 20% or more in the MSE on the test set on the subsequent pruning step, in which case the pruning process was stopped. This explains the entries in tables 5.1 to 5.8 where pruning ended with the same number of hidden weights as the initial oversized network. In these instances no parameters were identified for pruning and the pruning process was repeated with a smaller value for θ_0^2, with re-initialized particles that often resulted in larger MSE values. For each function, 30 pruning simulations were conducted as

reflected in tables 5.1 to 5.11. Tables 5.1 to 5.8 contain the results for the pruning of the hidden layer for all eight functions. Further, to show that the variance nullity can also be applied to prune the input units of PUNNs, pruning was applied to three of the eight test functions. In these cases extra input units were added to the architecture, with inputs for these units randomly generated. Tables 5.9 to 5.11 contain the results for pruning of the input layer for PUNNs. The tables also contain the averages for the number of hidden and input units calculated over the 30 simulations and the average MSEs calculated on the training and test sets together with a 95% confidence interval. Tables 5.1 to 5.8 show that the average number of hidden units for the various functions are close to the optimal number of hidden units as determined in chapter 4, bearing in mind that the average also includes the number of simulations where the initial oversized networks showed a degradation in performance due to re-initialization of weights.

Each training session started with random weights. Due to the stochastic search employed by PSO, the particle swarm optimizer is not always guaranteed to converge to a global optimum . PSO did therefor not always succeed in pruning all the irrelevant hidden units [Van den Bergh *et al* 2001c]. This explains why the average number of hidden units is slightly higher than the values obtained in chapter 4. Table 5.2 on page 157 shows that an initial network comprising 8 hidden units, for the cubic function, were pruned to 1 unit in 27 out of 30 simulations. The average number of hidden units for the pruned network as reflected in table 5.2 over 30 simulations is 2 (i.e. 1.5 rounded). This shows that the function $f(x) = x^3 - 0.04x$ can be represented by a PUNN containing two hidden units compared to an optimal SUNN that requires 3 hidden units. The tables also reflect a performance similar to the results contained in table 4.25 on page 121.

The variance nullity method can only remove irrelevant units; it cannot remove redun-

dant units from a network. This may explain why certain simulations ended in a higher number of units than the optimal number of units for the specific function. Similarly, the average number of input units are comparable to the optimal number of input units as determined in chapter 4. The performance of the oversized network, in all cases, is much poorer than the performance of the pruned network. Thus, a larger PUNN architecture does not necessarily translate to an increase in network performance, since the larger NNs overfitted the data.

5.5 Conclusion

In this chapter the variance nullity pruning algorithm developed by Engelbrecht was discussed and applied to oversized PUNN architectures of the eight test functions (as defined in chapter 4). The variance nullity pruning approach successfully pruned irrelevant hidden and input units of PUNNs. The variance nullity pruning algorithm produced averages for the number of hidden and input units that were comparable to the optimal number of units as determined by brute force in chapter 4. The results also indicate that the initial oversized PUNNs did not produce smaller MSEs than the pruned networks. This implies that in the case of PUNNs, a larger network does not necessarily translate into better performance. Re-initialization of oversized networks when no parameters were identified for pruning often resulted in a poorer performance that may lead to early termination of the pruning process. This could have been avoided, if all the unpruned weights were retained for the next pruning step, where a smaller value for θ_0^2 will subsequently be used by the pruning algorithm. Thus, an improvement for the variance nullity algorithm applied to PUNNs is to avoid re-initialization of weights in cases where no parameters were pruned by retaining the unpruned weights and to continue the pruning process by re-training only the bias.

$f(x) = x^2$						
	Oversized network				Pruned Network	
Simulation No	No of hidden units	MSE on Training set	MSE on Test set	No of hidden units	MSE on Training set	MSE on Test set
1	8	0.0031	0.0039	1	0.000377	0.000619
2	8	0.0134	0.0240	1	0.001376	0.001675
3	8	0.0062	0.0108	1	0.000059	0.000065
4	8	0.0292	0.0210	1	0.000219	0.000324
5	8	0.0290	0.0844	1	0.000448	0.000290
6	8	0.0041	0.0045	8	0.004073	0.004544
7	8	0.0102	0.0178	2	0.000643	0.003382
8	8	0.0157	0.0130	1	0.000205	0.000156
9	8	0.0077	0.0088	1	0.000172	0.000186
10	8	0.0018	0.0014	1	0.000051	0.000054
11	8	0.0244	0.0271	1	0.000100	0.000082
12	8	0.0193	0.0392	2	0.000563	0.000652
13	8	0.0171	0.0131	1	0.000238	0.000269
14	8	0.0032	0.0034	1	0.000061	0.000048
15	8	0.0265	0.0187	2	0.000777	0.000297
16	8	0.0042	0.0895	2	0.000434	0.000530
17	8	0.0246	0.0206	2	0.000195	0.000098
18	8	0.0252	0.5363	2	0.000906	0.000553
19	8	0.0203	0.0342	1	0.000270	0.000180
20	8	0.0083	0.0090	3	0.001427	0.001020
21	8	0.0056	0.0052	1	0.000126	0.000153
22	8	0.0369	0.0259	1	0.000101	0.000087
23	8	0.0177	0.0245	2	0.001014	0.000892
24	8	0.0172	0.0127	2	0.000189	0.000230
25	8	0.0121	0.0610	1	0.000140	0.000383
26	8	0.0124	0.0171	2	0.000359	0.000793
27	8	0.0051	0.0063	1	0.000131	0.000111
28	8	0.0046	0.0032	1	0.000648	0.000635
29	8	0.0224	0.0896	1	0.000632	0.000501
30	8	0.0075	0.0185	1	0.000253	0.000475
	Average no of hidden units	8		Average no of hidden units	1.6	
	Average	0.01450	0.04149	Average	0.00054	0.00064
	Confidence	0.00340	0.03512	Confidence	0.00028	0.00036

Table 5.1: Pruning of hidden units - function F1

$f(x) = x^3 - 0.04x$							
	Oversized network				Pruned Network		
Simulation No	No of hidden units	MSE on Training set	MSE on Test set		No of hidden units	MSE on Training set	MSE on Test set
1	8	0.0025	0.0075		1	0.000009	0.000009
2	8	0.0044	0.0106		1	0.000042	0.000035
3	8	0.0016	0.0008		8	0.001625	0.000844
4	8	0.0050	0.0043		1	0.000008	0.000006
5	8	0.0055	0.0037		2	0.000229	0.000132
6	8	0.0074	0.0142		1	0.000024	0.000026
7	8	0.0061	0.0072		1	0.000009	0.000008
8	8	0.0033	0.0034		1	0.000008	0.000007
9	8	0.0060	0.0065		1	0.000008	0.000009
10	8	0.0121	0.0165		1	0.000012	0.000007
11	8	0.0048	0.0042		1	0.000013	0.000013
12	8	0.0024	0.0039		1	0.000007	0.000008
13	8	0.0009	0.0013		1	0.000013	0.000010
14	8	0.0024	0.0027		1	0.000035	0.000035
15	8	0.0035	0.0022		1	0.000014	0.000010
16	8	0.0017	0.0016		1	0.000010	0.000008
17	8	0.0063	0.0052		1	0.000011	0.000009
18	8	0.0098	0.0063		1	0.000019	0.000016
19	8	0.0003	0.0003		1	0.000009	0.000007
20	8	0.0037	0.0051		1	0.000010	0.000009
21	8	0.0033	0.0038		1	0.000034	0.000035
22	8	0.0058	0.0076		1	0.000009	0.000010
23	8	0.0030	0.0034		1	0.000009	0.000008
24	8	0.0009	0.0008		1	0.000007	0.000006
25	8	0.0075	0.0114		1	0.000007	0.000016
26	8	0.0023	0.0040		8	0.002251	0.004021
27	8	0.0028	0.0039		1	0.000013	0.000008
28	8	0.0059	0.0049		1	0.000011	0.000008
29	8	0.0053	0.0061		1	0.000015	0.000011
30	8	0.0081	0.0092		1	0.000008	0.000008
	Average no of hidden units	8			Average no of hidden units	1.5	
	Average	0.00449	0.00542		Average	0.00015	0.00018
	Confidence	0.00099	0.00140		Confidence	0.00018	0.00027

Table 5.2: Pruning of hidden units - function F2

$z_t = 1 + 0.3z_{t-2} - 1.4z_{t-1}^2$						
	Oversized network				Pruned Network	
Simulation No	No of hidden units	MSE on Training set	MSE on Test set	No of hidden units	MSE on Training set	MSE on Test set
1	10	0.0004	0.0010	4	0.000102	0.006468
2	10	0.0022	0.0409	5	0.000039	0.000083
3	10	0.0003	0.0038	7	0.000220	0.000315
4	10	0.0010	0.0061	9	0.000036	0.000050
5	10	0.0002	0.0003	10	0.000175	0.000293
6	10	0.0007	0.0009	10	0.000658	0.000871
7	10	0.0117	0.0450	9	0.000069	0.000249
8	10	0.0008	0.0171	6	0.000017	0.000022
9	10	0.0003	0.0031	9	0.000015	0.000034
10	10	0.0006	0.0010	8	0.000109	0.000286
11	10	0.0025	0.0078	4	0.000011	0.000018
12	10	0.0092	0.0684	7	0.000189	0.000220
13	10	0.0002	0.0009	10	0.000233	0.000908
14	10	0.0003	0.0002	7	0.000062	0.000065
15	10	0.0011	0.1009	4	0.000041	0.000055
16	10	0.0001	0.0010	4	0.000088	0.000112
17	10	0.0623	0.0637	6	0.000007	0.000009
18	10	0.0009	0.0035	6	0.000082	0.000098
19	10	0.0001	0.0007	10	0.000092	0.000660
20	10	0.0002	0.0002	5	0.000011	0.000014
21	10	0.0032	0.0495	6	0.000081	0.000446
22	10	0.0094	0.0395	4	0.000129	0.001090
23	10	0.0010	0.0273	5	0.000027	0.000028
24	10	0.0004	0.0012	10	0.000432	0.001203
25	10	0.0010	0.0016	5	0.000064	0.000070
26	10	0.0002	0.0041	4	0.000001	0.000001
27	10	0.0032	0.0067	6	0.000005	0.000009
28	10	0.0010	0.0037	10	0.000196	0.000431
29	10	0.0008	0.0027	10	0.000829	0.002712
30	10	0.0020	0.0462	5	0.000090	0.000126
	Average no of hidden units	10		Average no of hidden units	6.8	
	Average	0.00391	0.03882	Average	0.00014	0.00056
	Confidence	0.00415	0.04519	Confidence	0.00007	0.00020

Table 5.3: Pruning of hidden units - function F3

$f(x,y) = y^7x^3 - 0.5x^6$						
	Oversized network				Pruned Network	
Simulation No	No of hidden units	MSE on Training set	MSE on Test set	No of hidden units	MSE on Training set	MSE on Test set
1	10	0.0015	0.0045	2	$2.15E-06$	$1.20E-03$
2	10	0.0037	0.0027	2	$1.02E-04$	$4.09E-05$
3	10	0.0007	0.0009	3	$1.22E-04$	$1.03E-04$
4	10	0.0009	0.0020	2	$1.52E-06$	$2.15E-03$
5	10	0.0013	0.0024	4	$7.04E-05$	$2.27E-04$
6	10	0.0017	0.0035	2	$4.50E-08$	$6.77E-03$
7	10	0.0002	0.0012	6	$1.17E-04$	$6.84E-04$
8	10	0.0008	0.0027	3	$4.57E-05$	$9.08E-05$
9	10	0.0023	0.0028	2	$1.32E-04$	$1.98E-03$
10	10	0.0011	0.0018	2	$1.76E-06$	$2.10E-04$
11	10	0.0028	0.0035	2	$3.43E-05$	$4.85E-04$
12	10	0.0009	0.0012	3	$3.49E-04$	$5.79E-04$
13	10	0.0007	0.0009	2	$7.24E-04$	$1.30E-04$
14	10	0.0002	0.0003	10	$2.22E-04$	$1.20E-04$
15	10	0.0025	0.0032	2	$7.34E-05$	$5.48E-05$
16	10	0.0011	0.0019	2	$4.00E-04$	$3.36E-03$
17	10	0.0002	0.0003	8	$1.85E-04$	$2.28E-03$
18	10	0.0003	0.0003	10	$2.80E-04$	$7.76E-04$
19	10	0.0013	0.0014	2	$1.94E-03$	$1.21E-03$
20	10	0.0064	0.0068	3	$1.24E-04$	$2.58E-04$
21	10	0.0028	0.0042	2	$2.07E-04$	$3.11E-05$
22	10	0.0027	0.0030	3	$1.77E-04$	$6.22E-05$
23	10	0.0014	0.0021	2	$6.76E-04$	$2.76E-03$
24	10	0.0001	0.0001	10	$8.91E-05$	$1.54E-04$
25	10	0.0003	0.0004	3	$3.21E-04$	$9.21E-04$
26	10	0.0005	0.0005	10	$5.17E-04$	$1.85E-04$
27	10	0.0045	0.0059	4	$4.13E-05$	$5.70E-03$
28	10	0.0045	0.0057	3	$7.12E-04$	$2.46E-04$
29	10	0.0012	0.0014	2	$4.32E-04$	$6.67E-03$
30	10	0.0012	0.0020	2	$1.57E-04$	$2.40E-04$
	Average no of hidden units	10		Average no of hidden units	3.8	
	Average	0.00166	0.00232	Average	0.00028	0.00132
	Confidence	0.00055	0.00062	Confidence	0.00014	0.00071

Table 5.4: Pruning of hidden units - function F4

$f(x,y) = x^2 + y^2$							
	Oversized network				Pruned Network		
Simulation No	No of hidden units	MSE on Training set	MSE on Test set		No of hidden units	MSE on Training set	MSE on Test set
1	10	0.0470	0.0491		2	0.00727	0.00777
2	10	0.0301	0.0357		2	0.01654	0.01864
3	10	0.0228	0.0262		2	0.00781	0.00875
4	10	0.0171	0.0174		2	0.01693	0.01674
5	10	0.0407	0.0774		4	0.02883	0.03137
6	10	0.0175	0.0186		10	0.01750	0.01859
7	10	0.0490	0.0433		2	0.00598	0.00596
8	10	0.1183	0.1338		2	0.00214	0.00212
9	10	0.0158	0.0152		10	0.01576	0.01517
10	10	0.0354	0.0512		3	0.00688	0.00699
11	10	0.0309	0.0341		3	0.00714	0.00785
12	10	0.0215	0.0248		2	0.00402	0.00393
13	10	0.0304	0.0345		2	0.01887	0.02020
14	10	0.0679	0.0774		2	0.00567	0.00508
15	10	0.0325	0.0434		4	0.00037	0.00043
16	10	0.0096	0.0113		10	0.00959	0.01129
17	10	0.0320	0.0328		3	0.02237	0.02253
18	10	0.0300	0.0297		3	0.00002	0.00002
19	10	0.0260	0.0359		2	0.01418	0.01467
20	10	0.0358	0.0368		2	0.01639	0.01590
21	10	0.0303	0.0457		6	0.01378	0.01374
22	10	0.0273	0.0328		3	0.00067	0.00069
23	10	0.0110	0.0162		3	0.00605	0.00633
24	10	0.0185	0.0233		2	0.00588	0.00591
25	10	0.0220	0.0291		3	0.01463	0.01780
26	10	0.0346	0.0309		2	0.01591	0.01385
27	10	0.0129	0.0169		2	0.00112	0.00133
28	10	0.0512	0.0586		2	0.02316	0.02118
29	10	0.0265	0.0359		2	0.01582	0.01483
30	10	0.0362	0.0537		3	0.00693	0.00687
	Average no of hidden units	10			Average no of hidden units	3.3	
	Average	0.03269	0.03501		Average	0.01024	0.01122
	Confidence	0.00744	0.00653		Confidence	0.00299	0.00281

Table 5.5: Pruning of hidden units - function F5

$f(x,y) = sin(x^2) + sin(y^2)$							
	Oversized network				Pruned Network		
Simulation No	No of hidden units	MSE on Training set	MSE on Test set		No of hidden units	MSE on Training set	MSE on Test set
1	10	0.0068	0.0210		6	0.00019	0.00021
2	10	0.0015	0.0020		8	0.00020	0.00048
3	10	0.0016	0.0025		4	0.00020	0.00021
4	10	0.0163	0.0440		2	0.00224	0.00343
5	10	0.0017	0.0142		2	0.00007	0.00007
6	10	0.0071	0.0112		2	0.00007	0.00008
7	10	0.0013	0.0035		7	0.00029	0.00043
8	10	0.0014	0.0038		10	0.00136	0.00377
9	10	0.0002	0.0017		6	0.00020	0.00038
10	10	0.0018	0.0021		7	0.00042	0.00069
11	10	0.0063	0.0091		4	0.00012	0.00021
12	10	0.0084	0.0141		6	0.00073	0.00134
13	10	0.0009	0.0010		10	0.00092	0.00105
14	10	0.0039	0.0043		3	0.00019	0.00023
15	10	0.0046	0.0069		4	0.00017	0.00025
16	10	0.0061	0.0111		7	0.00013	0.00017
17	10	0.0039	0.0047		5	0.00064	0.00119
18	10	0.0006	0.0007		10	0.00061	0.00071
19	10	0.0007	0.0026		4	0.00018	0.00020
20	10	0.0036	0.0046		5	0.00030	0.00045
21	10	0.0082	0.0437		2	0.00010	0.00013
22	10	0.0025	0.0102		3	0.00008	0.00011
23	10	0.0037	0.0079		5	0.00009	0.00013
24	10	0.0011	0.0014		4	0.00027	0.00035
25	10	0.0023	0.0037		2	0.00006	0.00008
26	10	0.0008	0.0015		6	0.00041	0.00106
27	10	0.0053	0.1544		5	0.00020	0.00028
28	10	0.0019	0.0121		6	0.00007	0.00036
29	10	0.0097	0.0163		7	0.00108	0.00182
30	10	0.0046	0.0130		2	0.00250	0.00280
	Average no of hidden units	10			Average no of hidden units	5.1	
	Average	0.00396	0.01431		Average	0.00047	0.00076
	Confidence	0.00127	0.01039		Confidence	0.00022	0.00036

Table 5.6: Pruning of hidden units - function F6

$f(x,y) = (4 - 2.1x^2 + (\frac{x^3}{3}))x^2 + xy + (4y^2 - 4)y^2$						
	Oversized network				Pruned Network	
Simulation No	No of hidden units	MSE on Training set	MSE on Test set	No of hidden units	MSE on Training set	MSE on Test set
1	15	0.0381	0.0459	12	0.03840	0.04219
2	15	0.0311	0.0471	5	0.02066	0.02591
3	15	0.0244	0.0317	10	0.03526	0.04204
4	15	0.0329	0.0421	2	0.06240	0.06875
5	15	0.0343	0.0436	15	0.03427	0.04364
6	15	0.0328	0.0362	6	0.03735	0.03649
7	15	0.0240	0.0322	4	0.02728	0.03045
8	15	0.0356	0.0570	4	0.03397	0.04387
9	15	0.0327	0.0407	3	0.02985	0.03440
10	15	0.0253	0.0411	5	0.02566	0.03724
11	15	0.0290	0.0512	3	0.04249	0.05062
12	15	0.0273	0.0399	5	0.01284	0.01518
13	15	0.0293	0.0351	4	0.02905	0.03501
14	15	0.0462	0.0145	7	0.02749	0.03669
15	15	0.0268	0.0772	4	0.02502	0.03588
16	15	0.0245	0.0394	4	0.02544	0.03403
17	15	0.0250	0.0302	15	0.02503	0.03022
18	15	0.0333	0.0383	5	0.02383	0.02318
19	15	0.0292	0.0351	3	0.03048	0.03614
20	15	0.0308	0.0394	14	0.02389	0.02961
21	15	0.0210	0.0276	4	0.02909	0.02626
22	15	0.0324	0.0783	4	0.02561	0.02977
23	15	0.0284	0.0403	4	0.03356	0.03624
24	15	0.0292	0.0388	4	0.03355	0.03580
25	15	0.0226	0.0384	3	0.07735	0.08508
26	15	0.0298	0.0546	3	0.04043	0.04510
27	15	0.0271	0.0301	14	0.02730	0.02504
28	15	0.0372	0.0611	3	0.02646	0.05175
29	15	0.0451	0.0110	5	0.05121	0.06213
30	15	0.0319	0.0459	3	0.02680	0.03542
	Average no of hidden units	15		Average no of hidden units	5.9	
	Average	0.03058	0.04252	Average	0.03273	0.03880
	Confidence	0.00208	0.00498	Confidence	0.00457	0.00512

Table 5.7: Pruning of hidden units - function F7

$f(x,y) = sin(x) \cdot sin(y) \cdot \sqrt{xy}$						
	Oversized network				Pruned Network	
Simulation No	No of hidden units	MSE on Training set	MSE on Test set	No of hidden units	MSE on Training set	MSE on Test set
1	12	0.0001	0.0001	8	0.000048	0.000056
2	12	0.0006	0.0010	7	0.000010	0.000011
3	12	0.0021	0.0038	8	0.000116	0.001136
4	12	0.0001	0.0002	6	0.000107	0.000102
5	12	0.0001	0.0001	12	0.000129	0.000131
6	12	0.0013	0.0064	4	0.000005	0.000005
7	12	0.0001	0.0001	7	0.000060	0.000065
8	12	0.0003	0.0005	5	0.000016	0.000013
9	12	0.0002	0.0006	6	0.000084	0.000137
10	12	0.0001	0.0017	12	0.000096	0.001705
11	12	0.0007	0.0159	8	0.000168	0.000421
12	12	0.0011	0.0021	6	0.000041	0.000055
13	12	0.0038	0.0198	8	0.000089	0.000071
14	12	0.0005	0.0009	6	0.000025	0.000069
15	12	0.0001	0.0003	12	0.000108	0.000348
16	12	0.0002	0.0331	4	0.000060	0.000065
17	12	0.0003	0.0168	4	0.000043	0.000167
18	12	0.0002	0.0002	5	0.000014	0.000019
19	12	0.0003	0.0006	5	0.000010	0.000012
20	12	0.0007	0.0009	6	0.000016	0.000017
21	12	0.0001	0.0001	7	0.000050	0.000061
22	12	0.0002	0.0003	12	0.000178	0.000298
23	12	0.0001	0.0002	12	0.000102	0.000192
24	12	0.0004	0.0005	4	0.000010	0.000008
25	12	0.0004	0.0008	5	0.000039	0.000039
26	12	0.0003	0.0007	4	0.000076	0.000220
27	12	0.0001	0.0014	8	0.000333	0.001303
28	12	0.0002	0.0286	12	0.000608	0.003324
29	12	0.0006	0.0138	4	0.000047	0.000062
30	12	0.0003	0.0003	5	0.000042	0.000035
	Average no of hidden units	12		Average no of hidden units	7.1	
	Average	0.00052	0.00506	Average	0.00009	0.00068
	Confidence	0.00027	0.00325	Confidence	0.00004	0.00077

Table 5.8: Pruning of hidden units - function F8

$f(x) = x^2$						
	Oversized network			Pruned Network		
Simulation No	No of input units	MSE on Training set	MSE on Test set	No of input units	MSE on Training set	MSE on Test set
1	4	0.0239	0.0365	1	0.00063	0.00054
2	4	0.0231	0.0224	1	0.00019	0.00016
3	4	0.0354	0.0313	4	0.04229	0.03902
4	4	0.0118	0.0229	1	0.00020	0.00028
5	4	0.0431	0.0711	1	0.00060	0.00087
6	4	0.0103	0.0100	1	0.00019	0.00020
7	4	0.0167	0.0166	2	0.00157	0.00246
8	4	0.0177	0.0318	2	0.00135	0.00129
9	4	0.0225	0.0284	1	0.00046	0.00055
10	4	0.0093	0.0359	1	0.00029	0.00031
11	4	0.0227	0.0242	1	0.00028	0.00017
12	4	0.0151	0.0122	1	0.00026	0.00021
13	4	0.0176	0.0176	2	0.00021	0.00019
14	4	0.0219	0.0142	1	0.00054	0.00023
15	4	0.0161	0.0170	2	0.00135	0.00139
16	4	0.0187	0.0214	1	0.00021	0.00016
17	4	0.0191	0.0078	1	0.00053	0.00050
18	4	0.0333	0.0275	1	0.00045	0.00061
19	4	0.0185	0.0160	1	0.00030	0.00033
20	4	0.0132	0.0113	2	0.00107	0.00095
21	4	0.0164	0.0179	1	0.00025	0.00022
22	4	0.0404	0.0363	1	0.00053	0.00039
23	4	0.0219	0.0257	1	0.00009	0.00009
24	4	0.0243	0.0385	1	0.00049	0.00045
25	4	0.0082	0.0101	1	0.00007	0.00009
26	4	0.0275	0.0248	1	0.00011	0.00009
27	4	0.0209	0.0096	2	0.00034	0.00025
28	4	0.0139	0.0091	1	0.00051	0.00032
29	4	0.0036	0.0026	1	0.00002	0.00007
30	4	0.0189	0.0161	1	0.00048	0.00031
	Average no of hidden units	4		Average no of hidden units	1.3	
	Average	0.01970	0.02223	Average	0.00186	0.00171
	Confidence	0.00327	0.00477	Confidence	0.00278	0.00257

Table 5.9: Pruning of input units - function F1

$f(x) = x^3 - 0.04x$							
	Oversized network				Pruned Network		
Simulation No	No of input units	MSE on Training set	MSE on Test set		No of input units	MSE on Training set	MSE on Test set
1	4	0.00201	0.00370		1	0.000011	0.000038
2	4	0.00624	0.00605		1	0.000012	0.000010
3	4	0.00011	0.00009		1	0.000040	0.000036
4	4	0.00038	0.00065		1	0.000008	0.000007
5	4	0.00033	0.00054		1	0.000011	0.000009
6	4	0.00002	0.00001		4	0.000015	0.000014
7	4	0.00228	0.00522		1	0.000008	0.000007
8	4	0.00024	0.00085		1	0.000009	0.000007
9	4	0.00003	0.00003		4	0.000026	0.000026
10	4	0.00004	0.00003		1	0.000007	0.000007
11	4	0.01621	0.02198		1	0.000007	0.000006
12	4	0.00015	0.00008		1	0.000041	0.000048
13	4	0.00007	0.00004		4	0.000072	0.000044
14	4	0.00010	0.00015		4	0.000059	0.000047
15	4	0.00622	0.00465		1	0.000008	0.000006
16	4	0.00041	0.00020		1	0.000016	0.000025
17	4	0.00034	0.00047		1	0.000036	0.000068
18	4	0.00007	0.00005		4	0.000155	0.000152
19	4	0.00238	0.00376		1	0.000007	0.000006
20	4	0.00572	0.00484		1	0.000009	0.000008
21	4	0.01302	0.01765		1	0.000011	0.000015
22	4	0.00243	0.00337		1	0.000007	0.000006
23	4	0.00022	0.00045		4	0.000216	0.000450
24	4	0.00001	0.00001		1	0.000013	0.000009
25	4	0.00025	0.00021		1	0.000038	0.000040
26	4	0.00011	0.00006		1	0.000007	0.000007
27	4	0.00041	0.00052		1	0.000008	0.000006
28	4	0.00005	0.00003		1	0.000008	0.000006
29	4	0.00013	0.00013		4	0.000130	0.000132
30	4	0.00074	0.00108		1	0.000009	0.000007
	Average no of hidden units	4			Average no of hidden units	1.7	
	Average	0.00199	0.00256		Average	0.00003	0.00004
	Confidence	0.00142	0.00185		Confidence	0.00002	0.00003

Table 5.10: Pruning of input units - function F2

$f(x,y) = y^7x^3 - x^6$						
	Oversized network				Pruned Network	
Simulation No	No of hidden units	MSE on Training set	MSE on Test set	No of hidden units	MSE on Training set	MSE on Test set
1	4	0.0046	0.0067	2	0.00038	0.00063
2	4	0.0087	0.0109	2	0.00003	0.00006
3	4	0.0031	0.0066	2	0.00033	0.00058
4	4	0.0059	0.0072	2	0.00056	0.00071
5	4	0.0040	0.0077	2	0.00041	0.00073
6	4	0.0089	0.0160	4	0.00887	0.01596
7	4	0.0077	0.0094	2	0.00038	0.00063
8	4	0.0100	0.0125	2	0.00055	0.00095
9	4	0.0071	0.0083	2	0.00005	0.00006
10	4	0.0057	0.0096	2	0.00032	0.00062
11	4	0.0045	0.0074	2	0.00046	0.00067
12	4	0.0030	0.0060	2	0.00030	0.00059
13	4	0.0092	0.0119	2	0.00036	0.00060
14	4	0.0046	0.0062	2	0.00042	0.00060
15	4	0.0092	0.0187	4	0.00921	0.01865
16	4	0.0100	0.0118	2	0.00041	0.00065
17	4	0.0093	0.0109	3	0.00108	0.00158
18	4	0.0033	0.0060	2	0.00036	0.00062
19	4	0.0083	0.0113	2	0.00036	0.00062
20	4	0.0077	0.0086	2	0.00049	0.00064
21	4	0.0078	0.0116	2	0.00370	0.00600
22	4	0.0067	0.0093	4	0.00665	0.00932
23	4	0.0059	0.0062	2	0.00064	0.00071
24	4	0.0087	0.0105	2	0.00053	0.00059
25	4	0.0096	0.0119	3	0.00044	0.00058
26	4	0.0067	0.0079	2	0.00477	0.00622
27	4	0.0050	0.0074	2	0.00042	0.00062
28	4	0.0100	0.0107	2	0.00005	0.00006
29	4	0.0029	0.0061	2	0.00030	0.00060
30	4	0.0060	0.0097	4	0.00599	0.00973
	Average no of hidden units	4		Average no of hidden units	2.3	
	Average	0.00680	0.00950	Average	0.00163	0.00269
	Confidence	0.00084	0.00108	Confidence	0.00097	0.00172

Table 5.11: Pruning of input units - function F4