Chapter 6

Conclusions

One of the objectives of this thesis was to show that the back-propagation algorithm, that provides a computationally efficient method for the training of multilayer summation neural networks, fails to train PUNNs. The reason for its failure can be ascribed to the search space for PUNNs that is usually extremely convoluted [Durbin *et al* 1989, Leerink *et al* 1995]. The main reasons for the failure of gradient descent in the convoluted search space of PUNNs are (a) incorrect weight initialization and (b) the presence of an increased number of local minima. Generally, gradient descent only manages to train PUNNs when the weights are initialized in close proximity of the optimal weight values. Usually, the optimal weight values are often not available resulting in bad choices for weight initialization, which in turn causes gradient descent to get stuck in one of the numerous local minima that occur on the error surface or become paralyzed (which occurs when the gradient of the error with respect to the current weight is close to zero). In chapter 3 an inspection of the error surfaces of $f(z) = z^3$, with $z \in [-1, 1]$, and $f(z_1, z_2) = z_1^2 + z_2^2$, with $z_1, z_2 \in [-1, 1]$, indicated that weight initialization greatly influenced the convergence of gradient descent when applied to PUNNs. It illustrated that initial weights chosen such that the direction of the negative of its gradient points to a local rather than a global minimum,

often resulted in gradient descent to converge to and become trapped by this bad local minimum. Also, it was shown that if initial weights are chosen along a steep incline of the error surface, where the derivative of the error surface with respect to the weight is extremely large, then weight updates will be large which may cause jumping over the global minimum. The neural network then oscillates between extreme points of the error surface overshooting the global minimum each time. The results in chapter 4 with respect to functions F1 and F2 indicate that gradient descent using PUNNs were trapped in local minima resulting in much larger MSEs than achieved by particle swarm optimization and genetic algorithms.

Another objective was to show that global optimization algorithms such as genetic algorithms, particle swarm optimization and leapfrog optimization could be used to avoid the numerous local minima that occur on the error surface of PUNNs in training PUNNs successfully. The results in chapter 4 in table 4.25 on page 121 show that the various optimization algorithms applied to PUNNs produced much lower MSEs on the training and test sets for each function than gradient descent applied to PUNNs, indicating that the PSO, GA and LFOP are more successful in training PUNNs than gradient descent. In functions F3, F4, F5, F6, F7 and F8 gradient descent was unsuccessful in training the corresponding PUNNs. In a comparison of the global optimization algorithms applied to SUNNs, it is evident that LFOP:SUs managed to produce smaller training errors and generalized much better than BP:SUs, except for functions $f(x) = x^2$ and $f(x, y) = x^2 + y^2$, where BP:SUs outperformed PSO:SUs and LFOP:SUs. The global optimization algorithms applied to SUNNs in all eight functions did not perform better than BP:SUs, however, they did manage to reach lower generalization levels using much fewer iterations than BP:SUs and a corresponding higher percentage of convergence for the various generalization levels. In the case of function F5, although BP:SUs achieved a smaller training error than

PSO:PUs, it did not manage to reach the low generalization level of 0.0001, that was achieved by PSO:PUs. It can be concluded that the global optimization algorithms appeared to find the global minimum on the error surface faster than BP:SUs.

Another objective was to determine the global optimization algorithm which is more efficient and robust in training PUNNs. Results in chapter 4 indicated that PUNNs performed the best with respect to functions F1, F2, F6 and F8, while SUNNs outperformed the PUNNs in functions F3, F4, F5 and F7 achieving lower MSEs on the training sets and improved generalization. In the case of the global algorithms applied to PUNNs, PSO was the only algorithm that managed to reach a low generalization level of 0.0001 for all functions, except for function F7. LFOP applied to PUNNs also managed to reach low generalization levels of 0.00001 in functions F1, F2, F3 and F5. GAs only managed to achieve a low generalization level in functions F1 and F2. In choosing a global algorithm applied to PUNNs that is the most robust, it appears that PSO is more robust than LFOP with respect to functions F1 and F2 from tables 4.30 and 4.32, also taking fewer iterations than PSO to reach convergence, whereas PSO appears to be more robust with respect to functions F4 and F5 since they have a larger percentage of simulations that converged to a generalization level of 0.00001. PSO tends to be more robust than LFOP if one takes into account the instances of functions F6, F7 and F8 where not a single simulation of LFOP could train the PUNNs successfully as reflected in tables 4.25 and 4.32; only overflows were produced in these cases due to the large weight adjustments caused by gradient descent.

The optimal architectures for PUNNs were initially determined using brute force pruning in chapter 4 which resulted in much smaller architectures. The number of hidden units that occurred in optimal PUNNs expressed as a percentage of the number of hidden units that occurred in the equivalent optimal SUNNs, are for

functions F1 50%, F2 33.3%, F3 80%, F4 33.3%, F5 50%, F6 66.7%, F7 75% and F8 77.8%. This shows that the optimal PUNNs were smaller than the equivalent SUNNs for the eight test functions. The variance nullity pruning algorithm applied in chapter 5 produced similar PUNN architectures as the brute force pruning approach of section 4.8 on page 114. The results of chapter 4 show that PUNNs, with their much smaller optimal architectures compared to the corresponding larger optimal SUNNs, did not always result in an improvement with respect to performance of the neural networks. These smaller PUNNs networks did not always produce good training errors and generalization compared to the larger architectures of SUNNs. However, global optimization using SUs showed an improvement in performance compared to gradient descent using SUs. In certain instances the PUNNs outperformed the SUNNs.

In general, PUNNs did not show a remarkable gain in performance, other than reaching lower generalization levels faster than gradient descent applied to SUNNs. One has to consider the trade-off between (a) added complexity when using PUNNs due to exponentiation and (b) the larger architecture required by SUNNs, before deciding on implementing a neural network using either PUs or SUs.

6.1 Possible Improvements and Future Research

The following aspects are suggested for future research:

1. The learning profiles in chapter 4 reflected that PSO and GA applied to PUNNs had larger reductions in error early in training, reaching low errors using substantially less training epochs than gradient descent applied to SUNNs. This suggests using the global optimization algorithms for initial training to produce weights close to the global minima on the error surface. Once the area where the global

minimum occurs is reached, gradient descent can then be applied to further train the PUNN to completion, an approach which has been used successfully in the past using different optimization algorithms.

2. PSO can be enhanced by incorporating constriction coefficients in the algorithm, which have lead to improved performance as reported by Eberhart *et al* [Eberhart *et al* 2000].

3. The variance nullity algorithm applied to PUNNs can be improved by avoiding 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.

4. An investigation into the overfitting tendencies of PSO, GA and LFOP to identify the algorithm that exhibits the smallest degree of overfitting and consequently the best generalization. These results can then be compared to results obtained by Lawrence and Giles that showed that the Scaled Conjugate Gradient algorithm tended to overfit more than gradient descent [Lawrence *et al* 2000].