5th Slovakian-Hungarian Joint Symposium on Applied Machine Intelligence and Informatics
January 25-26, 2007 ▒ Poprad, Slovakia

Combinations of Gradient and Evolutionary Methods for Neural Network Weights Adaptation

Pavol Maliňák, Rudolf Jakša

Center for Intelligent Technologies, Department of Cybernetics and Artificial Intelligence, Technical University of Košice, Letná 9, 042 00 Košice, pavol.malinak@tuke.sk, jaksa@neuron.tuke.sk

Abstract: In this paper, the use of evolutionary computation for feedforward neural network learning is discussed. The aim is to combine benefits of evolutionary and gradient learning into two methods: BP/ES and ES/LMS. We compared experimental results obtained on XOR data by back-propagation algorithm, evolution strategies, and combined approach.

Keywords: evolutionary computation, evolution strategies, neural networks

1 Introduction

There are several different approaches with various properties, which can be used for neural network learning [1]. Knowing and using these properties, potentially avoiding them, is necessary for practical problem solving. However, not every method is applicable to solve every problem. A method, which is appropriate to be used for some problems, may not have expected results for another class of tasks.

In this paper there is an attempt to compare two completely different approaches – back-propagation of error algorithm (BP) and evolution strategies (ESs) – and to deduce some pieces of knowledge which can facilitate combining their advantages. The emphasis is placed on ESs – a class of evolutionary computation methods that use normally distributed mutations, recombination, deterministic selection of offspring individuals, and self-adaptation of strategy parameters [2]. The aim is to find an optimal or a near-optimal set of weights for the neural network with fixed architecture. Considering strong computational requirements, relatively 'elementary' data – notably the XOR problem – are used in this paper.

EANNs are a special class of artificial neural networks, in which evolution is another fundamental form of adaptation in addition to learning [1]. As the most considerable possibilities of evolutionary computation application in neural

network domain seems to be the stochastic optimization, which deals with 4 main problems [3], [1].

- network topology optimization (evolution of architectures; especially GAs)
- network weights optimization (evolution of connection weights, especially GAs and simulated annealing)
- both network topology and weights optimization
- learning rules extraction (evolution of learning rules; GAs)

Since nearly exclusive using genetic algorithms for weights optimization (formulated as minimization of an error function), this paper explores application possibilities of a bit different method – multimembered evolution strategy $(\mu+\lambda)-ES$ and $(\mu,\lambda)-ES$.

1.1 Evolution Strategies (ESs)

Evolution Strategies are algorithms which imitate the principles of natural (organic) evolution. They were developed as methods for numerical optimization ([4], [5]). Numerical optimization problem can be described in following way:

If $f : M \subseteq \Re^n \to \Re, M \neq \varnothing$ is an objective function (i.e. function, which is going to be optimized), then optimization problem is to find a vector $\bar{x}^* \in M$ such that

$$\forall \bar{x} \in M : f(\bar{x}) \geq f(\bar{x}^*) = f^*$$

where f^* is a global minimum, $M = \{\bar{x} \in \Re^n \mid g_j(\bar{x}) \geq 0 \ \forall j \in \{1,...,q\}\}$ is the set of feasible points and $g_j : \Re^n \to \Re$ are inequality constraints.

The '2 main qualities', which distinguish ES from other stochastic techniques, are real-valued representation[1] and self-adaptation[2]. 'Classical' evolution strategy can be implemented in the following way [6]:

1 Generate the initial population of μ individuals
2 Evaluate the fitness value for each individual of the population
3 Create λ offspring
4 Evaluate the fitness of each offspring

[1] E.g. Bäck ([2]) says, that for continuous parameter optimization problems the binary search space representation might make the search more difficult.

[2] The main purpose behind the self-adaptation is to allow the 'strategic' (control) parameters to self-adapt (so we can search the space of solutions and strategy parameters in parallel).

5 Sort offspring (or parents and offspring) and select μ best individuals to be parents of the next generation

6 Stop if stopping criterion is satisfied; otherwise go to the step 3

In the context of evolution strategies can be distinguished:

- $(1+1)-ES$ - so-called 'two-membered' strategy, where offspring competes with its parent and better of both individuals then serves as the ancestor of the following generation
- $(\mu+1)-ES$ - where μ parents can participate in the generation of one offspring individual
- $(\mu+\lambda)-ES$ - where μ parents produce λ offspring and the selection process selects a new population of μ individuals from (μ + λ) set
- $(\mu,\lambda)-ES$ - where μ parents produce λ offspring and the selection process selects a new population from the set of λ offspring only

In the following text, the multimembered (e.g. population size > 1) ES will be considered.

1.1.1 Uncorrelated Mutations in ESs

The genetic representation of individual can be described as follows:

$$(\bar{\mathbf{x}},\bar{\boldsymbol{\sigma}}) = ((x_1,...,x_n),(\sigma_1,...\sigma_n)),$$

where $\bar{\mathbf{x}}$ is optimized real-valued vector and $\bar{\boldsymbol{\sigma}}$ is a vector of standard deviations.

The operators in multimembered ESs incorporate two-level learning – also 'strategic' (control) parameter $\bar{\boldsymbol{\sigma}}$ undergoes evolution process.

Crossover operator can be implemented either as *discrete* crossover (for vector $\bar{\mathbf{x}}$), where each offspring's component comes from the first or second parent with identical probability or as *intermediate* crossover (for standard deviations $\bar{\boldsymbol{\sigma}}$), where offspring becomes a linear combination of its parents.

Mutation operator works as follows:

$$\bar{\boldsymbol{\sigma}}' = \bar{\boldsymbol{\sigma}}.\exp(N(0,\Delta\sigma_0))$$

$$\bar{\mathbf{x}}' = \bar{\mathbf{x}} + N(0,\bar{\boldsymbol{\sigma}}'),$$

where $N(0,\Delta\sigma_0)$ is random Gaussian number with a mean of zero and constant standard deviation[3] $\Delta\sigma_0$.

[3] $\Delta\sigma_0$ is a parameter of the method in this case.

1.1.2 Correlated Mutations in ESs

The main aim of establishing correlated mutation is to improve convergence rate of ESs [4]. The individual is very similar to previous version. The first vector is optimized real-valued one; the second vector is formed by responding standard deviations (they determine how much the optimized vector values will be changed). The new one is a vector of rotation angles. They affect the mutation hyperellipsoid's main axes inclination to find 'right' (preferred) direction of the search.

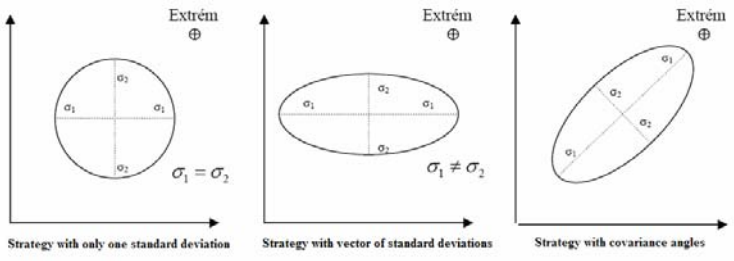


Figure 1
Mutations used in ESs (only one standard deviation; vector of standard deviations; vector of standard deviations and rotation angles)

The individual is now represented as a triple of vectors. The genetic representation (in the most general case) can be described as follows:

$$(\bar{\mathbf{x}}, \bar{\boldsymbol{\sigma}}, \bar{\boldsymbol{\alpha}}) = \left((x_1, \ldots, x_n), (\sigma_1, \ldots \sigma_n), (\alpha_1, \ldots, \alpha_{n(n-1)/2})\right)$$

where $\bar{\mathbf{x}}$ is optimized real-valued vector, $\bar{\boldsymbol{\sigma}}$ is a vector of standard deviations (dedicated σ_i for each x_i) and $\bar{\boldsymbol{\alpha}}$ is a vector of rotation angles.

In this version both operators, e.g. crossover and mutation, were used again. Also different types of crossover for different vectors were considered.

Crossover operator works analogically as in previous case (*discrete* crossover for vector $\bar{\mathbf{x}}$ and *intermediate* crossover for standard deviations $\bar{\boldsymbol{\sigma}}$ and rotation angles $\bar{\boldsymbol{\alpha}}$).

Mutation is applied in following steps:

1 mutation of standard deviations

$$\sigma_i' = \sigma_i . \exp\left(\tau_0 . N(0,1) + \tau . N_i(0,1)\right)^4$$

2 subsequently is applied a mutation of covariance angles

[4] The values for τ_0 and τ can be found in the appropriate literature, e.g. [7], [8].

$$\alpha'_j = \alpha_j + \beta.N_j(0,1),$$

where $\beta = 0.0873 = 5^\circ rad$

3 then $\Delta x'_i$ values are computed

$$\Delta x_i^* = N(0,\sigma'_i)$$

4 now successive rotations are being calculated:

$$(\Delta x'_1, \Delta x'_2, ..., \Delta x'_n) = \left(\prod_{i=1}^{n-1}\prod_{j=i+1}^{n} R(\alpha_{ij})\right).(\Delta x_1^*, \Delta x_2^*, ..., \Delta x_n^*)$$

5 finally, the new $\overline{\mathbf{x}}$ values are computed

$$x'_i = x_i + \Delta x'_i$$

2 Combinations of Error – Backpropagation Algorithm (BP) and Evolution Strategies (ESs)

The main aim of ESs and BP combinations is to raise the full advantages of both considered methods – not so high computational requirements, typical for gradient methods, and high 'robustness' typical for evolutionary computation. In the designed methods the direct encoding scheme is used – each connection is represented directly by its real-valued representation.

2.1 BP/ES Algorithm

The main idea of this approach is to learn neural network output layer weights and thresholds with evolution strategy and the other layers with gradient method. The main idea was to investigate the significance of output layer parameters impact on learning properties, and also was important attempt to reduce computational requirements of ES. One cycle of BP/ES algorithm can be described:

1 Run ESs computing the fitness over all examples in the training set on the weights from output layer.

2 Decode the best individual found above into weights in output layer.

3 Compute weights changes of output neurons and update (only) the hidden layer(s) weights iteratively for all examples in the training set.

2.2 ES/LMS[5] Algorithm

ES/LMS combination works analogically as one above. In this case output layer weights and thresholds are changed by LMS and the other layers ESs.

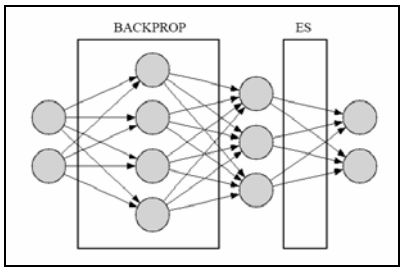


Figure 2
BP/ES algorithm (output layer weights and threshold are changed by ES, the other layers by gradient method)

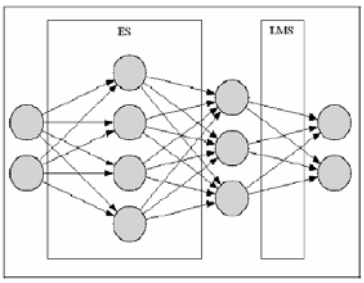


Figure 3
ES/LMS algorithm (output layer weights and thresholds are changed by LMS, the other layers by ES)

One cycle of ES/LMS algorithm can be described as follows:

1. Run ESs computing the fitness over all examples in the training set on the weights from hidden layer(s).
2. Decode the best individual found above into weights in hidden layer(s).
3. Compute weights changes of output neurons and update output layer weights iteratively for all examples in the training set.

3 Experiments

As mentioned above, main experiments were conducted on XOR data (classification of Boolean XOR function), due to their 'simplicity' and not so high computation requirements. Some illustrative experiments were conducted also on 'circle' data (classification of the points inside and outside the circle).

Feedforward neural network with one hidden layer in the experiments was used. There were 2 neurons in input layer, 3 neurons in first hidden layer and 1 neuron in output layer (so 2-3-1 topology was used). Parameter α (which determines the slop of transfer function) was equal to 1.0 and bias was equal to -1.0. In the beginning network weights were initialized from the interval <-0.5; 0.5>.

[5] LMS – Least Mean Square method

3.1 Back-Propagation of Error Algorithm Performance

First of all, the experiments with BP were conducted. Learning process was stopped, when error (mean square error between target and actual outputs averaged over all examples) was smaller than 0.01 or iterations count was greater then 20 000. Learning rate was 0.1, 0.2, 0.5, 0.8, 1.5, 3.0, 6.0, 20.0 and for each value 10 program runs were done.

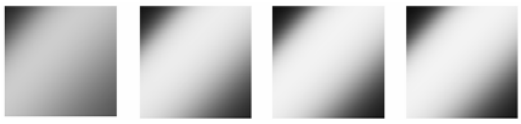


Figure 4
Visualisation of the network response for XOR data ($\gamma = 0.8$, iterations 1000, 1200, 1400 and 1600)

In the tests, the well-known fact of BP algorithm sensitivity to the learning parameter was confirmed. Smaller learning rate implies longer program runs. However, too great values of this parameter can cause instability of system during the learning process.

γ	Avg. iter. count	Avg. error	Avg. time (s)
0.1	15082.83	0.009999	4.474
0.2	5576,667	0,009996	2.13
0.5	2288,667	0,009991	0.928
0.8	1472.000	0,009988	0.526
1.5	719,3333	0,009985	0.324

Table 1
Some results obtained by BP for different γ values

3.2 Evolution Strategies without Covariances

The experiments were conducted on both 'comma' and 'plus' versions ($\mu = 15; \lambda = 100$) of multimembered strategies. In the beginning weights were initialized from <-0.5; 0.5>; during the evolution they could be from <-10; 10>. Minimum sigma value during the evolution process was 0.01. Evolution was stopped, when training error was smaller than 0.01 or when generations count was greater than 1000.

As mentioned above, the individual in ESs without covariances is made of two vectors. The first one consists of evolved network weights and thresholds and the second one is formed by vector of responding standard deviations used for mutation operation. In this case, the impact of $\Delta\sigma_0$ changes (0.1, 0.2, 0.5, 0.8,

1.5, 3.0, 6.0, 20.0, and for each 10 program runs) was investigated. The aim was to find optimal value of $\Delta\sigma_0$ and later apply this value in BP/ES and ES/LMS combinations.

$\Delta\sigma_0$	**Avg. gener. count**	**Avg. err.**	**Avg. time**
0.1	114.5714	0.009442	0.53
0.2	57.42857	0.008631	0.265714
0.5	37.0	0.005506	0.187143

Table 2
Results obtained by (15, 100) – ES (15 parents produce 100 offspring; parents necessarily die)

$\Delta\sigma_0$ greater than 0.5 makes the strategy ineffective – so 0.8, 1.5, …, 20.0 cases are not shown in the table.

$\Delta\sigma_0$	**Avg. gener. count**	**Avg. err.**	**Avg. time**
0.1	100,8571	0,009341	0,477143
0.2	55.0	0,008597	0,268571
0.5	32,71429	0,008098	0,168571
0.8	51,28571	0,005724	0,261429
1.5	229,2857	0,004927	13,40571

Table 3
Results obtained by (15 + 100) – ES (15 parents produce 100 offspring; parents don't die necessarily)

Results, we got, are quite comparable with results obtained by BP algorithm when 'correct' parameter settings are used.

3.3 Evolution Strategies with Covariances

In this version, both vectors mentioned above remain unchanged. The only new one is the vector of rotation angles. Sigma values were initialized from <0.01; 5.5>, minimum sigma value during the evolution process was 0.01.

Figure 5
Visualisation of the network response for xor data during (15, 100) – ES (generations 5, 10, 17)

Obtained results were consonant with expectations. The covariance version of ES works better but a bit slower than ESs without covariances.

3.4 BP/ES Algorithm

Due to need of comparability, settings used in the experiments with BP/ES version were identical as those used in experiments described above. One BP/ES cycle consisted of 20 ES generations and 50 BP iterations; maximal count of 'combined' cycles was 200. For each γ value 10 program runs were done.

γ	Avg. iter. c.	Avg. err.	Last cycle (G/I)	Avg. time
0.1	43.2	0.009996	20/29.8	2.31
0.2	31.1	0.009993	20/26.3	1.601
0.5	13.8	0.009978	16.1/18.6	0.725
0.8	7.9	0.009955	18.1/23.4	0.426
1.5	6.8	0.009726	10.2/7.7	0.398
3.0	4.4	0.008373	13.3/5.1	0.22
6.0	5.6	0.007857	6.4/1.0	0.276
20.0	10.1	0.007895	12.6/1.3	0.522

Table 4
Results of BP/ES – COMMA

Figure 6
Visualisation of the network response during BP/ES – COMMA, iterations 2, 4 and 6, γ = 0.8

Analogical tests were done for PLUS strategy:

γ	Avg. iter. c.	Avg. err.	Last cycle (G/I)	Avg. time
0.1	38.88889	0.009996	20/24.2	2.051
0.2	17.7	0.009993	20/27.6	0.954
0.5	9.7	0.00998	20/27.1	0.532
0.8	9.4	0.009953	18.2/15	0.512
1.5	4.6	0.009682	13.2/5.3	0.235
3.0	3.3	0.00884	12.6/13.2	0.169
6.0	3.2	0.007629	13/8.7	0.159
20.0	11.5	0.007673	13.1/2.8	0.625

Table 5
Results for BP/ES – PLUS

Comparison between **Table 4** and **Table 5** shows, that PLUS strategy overcomes the COMMA variant nearly for all γ values. In general, we can say that 'tuning' function of BP algorithm, due to great error signals propagated backward through the network, probably wasn't effective.

3.5 ES/LMS Algorithm

The settings used in this case were the same as settings used in BP/ES combination: One ES/LMS cycle consisted from 20 ES generations and 50 BP iterations; maximal count of 'combined' cycles was 200. For each γ value 10 program runs were done.

γ	Avg. iter. c.	Avg. err.	Last cycle (G/I)	Avg. time
0.1	67	0.009998	20/22.6	4.089
0.2	38.9	0.009996	20/21.4	2.557
0.5	13.6	0.00999	20/24.9	0.861
0.8	7.7	0.009984	20/18.1	0.48
1.5	6.7	0.009969	20/24.8	0.423
3.0	5.1	0.009969	20/25.9	0.33
6.0	1.7	0.009719	16.2/35.9	0.104
20.0	1.9	0.006088	20/11.1	0.121

Table 6
Results of ES – COMMA/LMS

γ	Avg. iter. c.	Avg. err.	Last cycle (G/I)	Avg. time
0.1	55.7	0.009998	20/32.6	3.554
0.2	31.4	0.009996	20/25.9	1.965
0.5	10.7	0.009987	20/29.4	0.685
0.8	7.5	0.009979	20/32.1	0.484
1.5	4.5	0.00997	20/29.2	0.291
3.0	3.6	0.009942	18.8/26	0.229
6.0	1.3	0.009711	18/34.6	0.088
20.0	1.4	0.00677	20/7.1	0.095

Table 7
Results of ES – PLUS/LMS

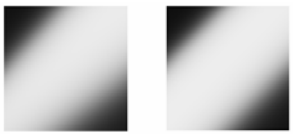


Figure 7
ES – PLUS/LMS. Algorithm was complete already after two combined cycles

The obtained results show PLUS strategy better than COMMA again. However, this situation is probably caused by 'simplicity' of the XOR data. In this case the 'tuning' function of BP algorithm arose – ES (hidden layer) was able to quickly provide the primary separation of the input space, so desirable network performance can be easy 'tuned' by BP.

3.6 Experiments with the 'Circle' Data

The main purpose of these experiments was not to investigate and analyze the possibilities of considered algorithms, but present their outcomes also from other than XOR data – they can be considered as illustrative.

In the experiments were used parent population with 10 individuals and offspring population with 200 individuals. One cycle consisted from 20 ES generations and 40 BP iterations; maximal count of 'combined' cycles was 30.

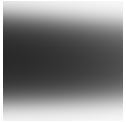
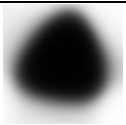
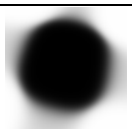
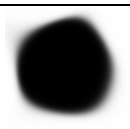
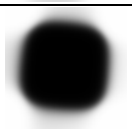
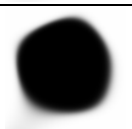
Iter.	BP/ES	ES/LMS	Iter.	BP/ES	ES/LMS
5			20		
10			25		
15			30		

Table 8
Visualisation of the network response

The results obtained from considered experiments in essence confirmed outcomes from XOR problem – the ES/LMS combination seems to be a bit better than BP/ES.

Conclusions

The main aim of conducted experiments was to implement gradient, evolutionary and 'combined' neural network learning approach and conduct experiments and comparisons on XOR data using these methods. For this purpose subsequent methods were implemented: BP/ES and ES/LMS, back-propagation of error algorithm and multimembered evolution strategies with/without covariances.

Evolutionary strategies are in their both versions – with or without covariance angles (this version is better, but rather smaller) – very powerful. They don't suffer from a convergence problem, although they can be extremely computationally and time consuming. This was the purpose for designing BP/ES and ES/LMS algorithms, which in several cases in our experiments even overcame BP algorithm.

It is also important to note that EC methods aren't designed to 'compete' with BP, but rather for use in situations, when other (gradient) methods aren't able to deal with. We can say, that efficient combination of ES and gradient methods advantages (not so high computational requirements of BP and 'local extreme trap resistance' of ES) can lead up to more efficient solving of many practical problems in different fields of study. Certainly, experiments on "more complicated" data would be more interesting.

References

[1] Yao, X.: Evolutionary Artificial Neural Networks. International Journal of Neural Systems, 1993, Vol. 4, No. 3, pp. 203-222

[2] Bäck, T.: Evolution Strategies: An Alternative Evolutionary Algorithm. Artificial Evolution, 1995, pp. 3-20

[3] Kvasnička, V. et al.: Úvod to teórie neurónových sietí. 1. vyd. Bratislava: IRIS, 1997. 265 s. ISBN 80-88778-30-1

[4] Bäck, T., Hoffmeister, F., Schwefel, H. P.: A Survey of Evolution Strategies. ICGA, 1991, pp. 2-9

[5] Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. Second, Extended Edition. Berlin: Springer-Verlag, 1994, ISBN 3-540-58090-5

[6] Yao, X., Liu, Y.: Fast Evolution Strategies, In: Proc. of the 6th Annual Conference on Evolutionary Programming

[7] Kvasnička, V. et al.: Evolučné algoritmy. 1. vyd. Bratislava: STU, 2000. 223 s. ISBN 80-227-1377-5

[8] Bäck, T, Schwefel, H. P.: An Overview of Evolutionary Algorithms for Parameter Optimization. Evolutionary Computation, 1993, Vol. 1, no. 1, pp. 1-23