Mixing Different Search Biases in Evolutionary Learning Algorithms

Kristina Davoian and Wolfram-M. Lippe

Department of Mathematics and Computer Science,
University of Münster, Einsteinstr. 62, 48149 Münster, Germany
{kristina.davoian,lippe}@uni-muenster.de

Abstract. This work investigates the benefits of using different distribution functions in the evolutionary learning algorithms with respect to Artificial Neural Networks' (ANNs) generalization ability. We examine two modification of the recently proposed network weight-based evolutionary algorithm (NWEA), by mixing mutation strategies based on three distribution functions at the chromosome and the gene levels. The utilization of combined search strategies in the ANNs training implies that different step sizes determined by mixed distributions will direct the evolution towards good generalized ANNs.

Keywords: Artificial Neural Networks, Learning, Evolutionary Algorithms.

1 Introduction

Evolutionary Algorithms (EAs) have found wide application in optimization of Artificial Neural Networks parameters, since they outperform the originally proposed gradient-descent learning approaches in terms of computations speed, simplicity and resistance to local minima trapping. However, mutation-based EAs, i.e. Evolutionary Programming (EP) and Evolutionary Strategies (ES) have proven to be more efficient in ANNs' learning than Genetic Algorithms, which due to their primary search operator (crossover) often face the permutation problem [1], [2], [14].

The key aspects that EP and ES concentrate on are the self-adaptive methods for changing the strategy parameters and the distribution used in mutation. The classical EP and ES algorithms utilize the standard normal distribution and similar self-adaptive methods, introduced by Rechenberg [3] and Schwefel [4] for ES and independently, by Fogel [5], [6] for meta-EP (widely known as classical EP). Later, Yao et al. [7] established that the distribution in the mutation strategy is crucial in the determination of the mutation step size, and proposed a novel EP technique, called the Fast Evolutionary Programming (FEP) [8], [9], which adopts the self-adaptation strategy of the classical EP (CEP), but uses the Cauchy distribution instead of the Gaussian one. Further, Yao investigated the impact of utilizing both Gaussian and Cauchy distributions at the chromosome and gene level and introduced modifications of FEP, referred to

C. Alippi et al. (Eds.): ICANN 2009, Part I, LNCS 5768, pp. 111–120, 2009.

as the improved Fast Evolutionary Programming (IFEP) and the mixed Fast Evolutionary Programming (MEP), respectively [13].

This work is concerned with the improvement of the evolutionary learning in ANNs. The goal for this work is to investigate the impact of mixing search biases of mutations based on three different distributions, on the generalization in ANNs. More specifically, we study how different step sizes determined by mixed distributions improve the quality affect the algorithm's convergence speed and the quality of evolved ANNs. We introduce two modifications of the recently proposed network weight (NW)-based EA (NWEA) strategy [10], which combine Gaussian, Cauchy and uniform distributions at the chromosome and gene levels in the learning strategy (the original NWEA strategy is based on the uniform distribution). In contrast to the CEP and FEP approaches, which are independent search methods, the NWEA algorithm was developed specially for ANNs' learning. The main feature of NWEA consists in the special approach to the adjustment of the random values. The strategy parameter in the classical techniques is evolved during the evolution alongside with the object parameters. In comparison to that, the adaptation strategy in NWEA consists of two components, which bear the information about the position of an individual in the search space and based of this knowledge, bias the improvement towards the perspective regions of the search space. The first modification of NWEA, referred further to as combined NWEA (CNWEA) produces three offspring as a result of mutation by using Gaussian, Cauchy and uniform random values, respectively (i.e. provides mutation at the chromosome level). The second modification uses three distributions to generate one offspring (i.e. carries out mutation at the gene level). The utilization of particular type of distribution is defined by a certain probability.

In order to evaluate the ANNs produced by CNWEA and MNWEA, the preliminary experimental studies on the breast cancer and heart disease diagnosis data sets were provided. The generalization results of ANNs, evolved by CNWEA and MNWEA were compared with those, evolved by NWEA, which is shown to be more efficient than classical searching techniques [11].

The rest of the paper is organized as follows: Section 2 discusses the advantages of using Gaussian and Cauchy distributions in terms of step size. Section 3 described the features of the NWEA adaptation strategy and main steps of the learning algorithm. Sections 4 and 5 introduce the combined NWEA (CNWEA) and the mixed NWEA (MNWEA), respectively. Following that, Section 6 presents the experiments and analyses the obtained results. Finally, Section 7 concludes this paper.

2 Length of Gaussian and Cauchy Jumps

Both CEP and FEP use the same mutation scheme to modify individuals and the same self-adaptation strategy to correct mutation step size. The only difference is the distribution used to generate random numbers. Therefore, it is reasonable to claim that the features of algorithms' performances are caused by a type of used distribution. Let us discuss the advantages of using both distributions.

The expected length of Gaussian (with $\mu = 0$ and $\sigma^2 = 1$) and Cauchy (with $\gamma = 1$) jumps can be calculated by integrating their probability density functions:

$$E_G(x) = \int_0^{+\infty} x \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx = \frac{1}{\sqrt{2\pi}} = 0.399$$

$$E_C(x) = \int_0^{+\infty} x \frac{1}{\pi\,(1+x^2)} dx = +\infty$$

Apparently, the Cauchy distribution enables longer jumps than the Gaussian one. At first sight it seems that longer jumps in the search space induce quicker convergence, and so the Cauchy distribution is preferable in the searching strategy. However, this assumption is wrong. The analytical studies in [3] provided to investigate when large jumps are beneficial, showed that long jumps are advantageous only when the global optimum is far away from the current search point. In other words, long jumps are effective when the distance between the global optimum and the current point is larger than mutation's step size. On the other hand, the Cauchy distribution will no longer be beneficial when the distance between the neighbourhood of the global optimum and the current point is smaller than the step size of the mutation. This implies that the use of small jumps is more effective near the neighbourhood of the global optimum. Hence, the Gaussian distribution increases the probability of finding the optimum when the distance between the current point and the neighbourhood of the global optimum is small.

3 Network Weight-Based Evolutionary Algorithm (NWEA)

The basic step of the self-adaptation mechanism in CEP and FEP consists of *a mutation of mutation parameters themselves*, i.e. the evolution of strategy parameters alongside with the object parameters. The modification of the control parameters is realized by multiplication with a random variable.

In contrast to the classical evolutionary algorithms, the NWEA algorithm [10], designed to evolve ANNs' parameters, uses different self-adaptation approach to find an optimum. The self-adaptation in NWEA comprises two control parameters, which *incorporate genotype and phenotype information about the position of an individual in search space*. The first component includes information about worth of a chromosome according to its fitness. The second component adds information about the current ANN topology, the genotype encodes, i.e. information about position of an individual in the ANN architecture space. Alike ES and EP, the NWEP approach relies on mutation and does not utilize crossover at all.

The evolution with the NWEA algorithm is implemented as follows:

1. Create an initial population consisting of randomly generated chromosomes. Each chromosome $x_i = (x_i^{(1)}, x_i^{(2)}, ..., x_i^{(k)})$, $\forall i \in \{1, ..., \mu\}$, represents one

possible set of connection weights[1], where is a population size, k is a total number of connections between neurons and $x_i^{(j)} \in [-1.0; 1.0]$, $j \in \{1, ..., k\}$ is a connection weight.

2. Evaluate the fitness of each individual from a population according to the objective function.
3. In contrast to other EP approaches, which apply probabilistic selection methods for choosing parents for reproduction, NWEA "allows" all parental chromosomes to take part in the creation of new individuals. It is worth noting that during mutation only one gene in the parental chromosome changes its value. Offspring chromosomes are created by application of the following equation to every chromosome in the population:

$$x_i^{(j)'} = x_i^{(j)} \left(1.0 + N_W(l, \bar{n}) \cdot N_E \cdot N_{Rand}^D\right), \tag{1}$$

where $x_i^{(j)}$ is a gene randomly chosen out of a chromosome x_i and mutated, $N_W(l, \bar{n})$ is a value, called *network weight*, that implicitly describes an ANN's internal structure, N_E represents an error, determined by the error function (MSE or other) of x_i, and N_{Rand}^D is a uniformly distributed random value.

The components N_E and $N_W(l, \bar{n})$ represent the adjustment components in the mutation strategy. They add knowledge about position and worth of a chromosome in search space in order to achieve the optimal improvement of chromosomes at each stage of evolution. The component N_E in Eq. (1) represents the genotype information, i.e. error of the mutated chromosome, which changes dynamically for every mutated chromosome. It enables the control of a randomly generated value and the adjustment of the mutation strength to an individual depending on its fitness, i.e. the higher the error of a chromosome, the higher the step size.

The value $N_W(l, \bar{n})$ depends on the number of hidden layers l and the average number of neurons in hidden layers $\bar{n}$ in the given ANN and is defined by Eq. (2). This value is distributed by the Fermi-Dirac-like function and is calculated according to the following formula:

$$N_w(l, \bar{n}) = A_1 + \frac{l}{2} + \frac{B_1 - \frac{l}{2}}{1 + \exp\left(\frac{\bar{n} - \mu}{T_1}\right)} \tag{2}$$

The value μ is similar to the chemical potential in the original Fermi-Dirac function (as cited in [10]) and depends on the number of hidden layers:

$$\mu = A_2 + \frac{B_2}{1 + \exp\left(\frac{l - B_2}{T_2}\right)} \tag{3}$$

The coefficients $A_1 = 3.0$, $B_1 = 2.0$, $T_1 = 0.4$, $A_2 = 1.2$, $B_2 = 3.2$, $T_2 = 0.6$ were obtained by the approximation of the results in [10] so that the N_W values never become negative (the coefficients T_1 and T_2 correspond to

[1] In case of the simultaneous evolution of ANN's weights and architectures, each chromosome consists of a set of vectors according to the ANN's connectivity matrix.

the temperature in the original Fermi-Dirac function). For each ANN the quantity N_W is calculated only once and does not change its value during the evolution, if we consider the evolution of connection weights in the environment determined by an ANN architecture; in case of simultaneous evolution of architectures and connection weights it becomes a new value every time when ANN's architecture is changed.

The main advantage of incorporating phenotype information in mutation is that it comprises detailed knowledge about the ANN's topology[2] and thus enables to improve the values of connection weights in respect to a given phenotype. It is known from the theory of evolution that individuals with favorable traits, determined by the genotype, are more likely to survive and reproduce ("survival of the fittest") and the fitness of every individual is defined by the individual's ability to adapt to the environment. From such point of view, the NWEA approach is an abstraction, which involves the knowledge of an environment (phenotype) and adapts genotype of every individual to it, and thus, increases the fitness of chromosomes of every next generation.

4. Evaluate the fitness of a new individual based on the objective function.
5. Repeat the process from point (3) until λ ($\lambda \geq \mu$) new chromosomes are created.
6. Create new population of m individuals: new population is created according to the $(\mu + \lambda)$-ES elitist method, which choosesμbest individuals from both parental and offspring chromosomes based on their fitness. This is accomplished by applying 2-tournament selection method that selects a group of individuals (usually four) from both parental and offspring populations, and compares their fitness. The individual with the higher fitness reaches the offspring population.
7. Repeat the process from point (2) until some halting criteria are satisfied.

Thus, by creating offspring population our greedy modification of the NWEA strategy selects the current best individuals. On the other hand, the risk of trapping is local optima is minimal, since the random values initially have long and short step sizes.

4 Combined NWEA (CNWEA)

The main idea behind combined NWEA (CNWEA) is to mix different search biases of mutations utilize Gaussian, Cauchy and uniform distributions, at the chromosome level. The benefits of using Gaussian and Cauchy distributions were described in section 3. The utilization of the uniformly distributed random values does not have strong motivation; however it adds an additional randomness in the evolution process.

[2] Although mutation strategy in (1) incorporates the knowledge about ANN's internal structure, it gives detailed information about considering topology, since the number of neurons in input and output layers is determined by a solving problem.

The implementation of CNWEA is simple and differs from NWEA only in point 3 of the algorithm described in section 4.2. Each parental chromosome undergo mutation is modified three times and thus, produces three different offspring by using different values of N^{D}_{Rand} in Eq. (1): the first offspring is created using normally distributed values$N^{D_G}_{Rand}$ with mean $\mu = 0$ and variance $\sigma^2 = 1$, i.e.

$$x_i^{(j)'} = x_i^{(j)} \left(1.0 + N_W(l, \bar{n}) \cdot N_F \cdot N^{D_G}_{Rand}\right), \tag{4}$$

the second offspring – by using Cauchy random numbers $N^{D_C}_{Rand}$ with a scale parameter $\gamma = 1$, i.e.

$$x_i^{(j)'} = x_i^{(j)} \left(1.0 + N_W(l, \bar{n}) \cdot N_F \cdot N^{D_C}_{Rand}\right), \tag{5}$$

and the third offspring – by utilizing uniformly distributed random values, $N^{D_U}_{Rand} \in$ [-1.0, 1.0]:

$$x_i^{(j)'} = x_i^{(j)} \left(1.0 + N_W(l, \bar{n}) \cdot N_F \cdot N^{D_U}_{Rand}\right) \tag{6}$$

The best offspring is selected as a survivor. The rest of CNWEA is the same as NWEA.

5 Mixed NWEA (MNWEA)

An alternative way to mixing different biases is to combine mutation operators, based on Gaussian, Cauchy and uniform distributions at the gene rather than chromosome level. In our second modification of NWEA, called mixed NWEA (MNWEA) we define certain probabilities to apply Gaussian, Cauchy or uniform random numbers in the mutation strategy. Thus, some genes in the chromosome will be modified with the probability p_G according to Eq. (4), others will be mutated with the probability p_C according to Eq. (5) and rest – with the probability p_U according to Eq. (6).

$$x_i^{(j)'} = \begin{cases} x_i^{(j)} \left(1.0 + N_W(l, \bar{n}) \cdot N_F \cdot N^{D_G}_{Rand}\right), \text{with } p_G \\ x_i^{(j)} \left(1.0 + N_W(l, \bar{n}) \cdot N_F \cdot N^{D_C}_{Rand}\right), \text{with } p_C \\ x_i^{(j)} \left(1.0 + N_W(l, \bar{n}) \cdot N_F \cdot N^{D_U}_{Rand}\right), \text{with } p_U \end{cases}$$

where p_G, $p_C$and $p_U$are the probabilities of applying mutations according to the Eq.(4), Eq. (5) and Eq. (6), respectively, and $p_G + p_C + p_U = 1$. In our experiments we set the values p_G, $p_C$and p_U to 0.4, 0.4 and 0.2, respectively, as we aim at exploring the impact of small and large jumps provided by Gaussian and Cauchy distributions (see Section 2).

6 Experiments

The experimental studies of CNWEA and MNWEA were provided for breast cancer and heart disease diagnosis for the purpose of studying generalization ability of ANNs. In order to reduce the noise in the fitness evaluation, the ANNs' weights and architectures have been optimized simultaneously during the evolution. For each problem 50 runs of both algorithms were provided. Following initial parameters were used for these experiments: the population size 30 for MNWEA and 10 for CNWEA (since CNWEA produces three offspring from each parent), the maximum number of generations 300, and the number of hidden nodes for each individual was chosen uniformly at random between 1 and 3. The algorithms stopped when the maximal generation was reached.

6.1 Breast Cancer Diagnosis

The breast cancer data set was originally obtained from Dr. William H. Wolberg at the University of Wisconsin Hospitals, Madison. The data set consists of 699 examples of which 458 (65.5%) are benign examples and 241 (34.5%) are malignant examples. Each example contains nine attributes: clump thickness, uniformity of cell size, uniformity of cell shape, marginal adhesion, single epithelial cell size, bare nuclei, bland chromatin, normal nucleoli, mitoses. The goal of the data set is to classify a tumour as either benign or malignant based on these attributes.

In our experiments, the whole data set was divided into three subsets, as suggested by Prechelt (1994): a training set, a validation set, and a testing set. The first set was used to train EANNs. The validation set was explored as a pseudo-testing set in order to evaluate the fitness of networks during evolution. This prevents overtraining of the network and improves its generalization ability. During this process ANN's learning is carried out until the minimal error on the validation set (and not on the training set) is achieved. Finally, the testing data were considered to evaluate the performance of the evolved ANNs. 349 examples of the given breast cancer data set were used as training data, the following 175 examples as validation data, and the final 175 patterns as training data.

The error function (fitness) was calculated according to the equation, proposed by Prechelt [12]:

$$E = 100 \cdot \frac{o_{\max} - o_{\min}}{N \cdot P} \sum_{p=1}^{P} \sum_{i=1}^{N} (o_{pi} - t_{pi})^2$$

where $o_{\min}$and $o_{\max}$are the minimum and maximum values of output coefficients in the problem representation. N is the number of output nodes, P is the number of patterns, o_{pi} and t_{pi} are the actual and desired outputs of node i for pattern t correspondingly.

Table 1 shows the architecture of evolved ANNs as well as generation numbers (min and mean) at which the optimal results were obtained. Table 2 presents the classification results for breast cancer diagnosis. The value "rate" in the Table 2 shows the percentage of incorrect classified samples.

Table 1. ANN architectures for breast cancer diagnosis

	NWEA	CNWEA	MNWEA
Connections (min)	14	14	14
Connections (max)	78	82	86
Connections (mean)	36	29	36
Hidden nodes (min)	0	0	0
Hidden nodes (max)	4	4	4
Hidden nodes (mean)	1.3	1.3	1.4
Generation (min)	101	87	104
Generation (mean)	139.1	118	123.7

Table 2. Comparative results of the prediction accuracy for breast cancer diagnosis

	NWEA			CWEA			MNWEA		
	min	max	mean	min	max	mean	min	max	mean
Error, training	1.418	3.650	2.722	1.183	2.659	2.329	1.377	3.247	2.798
Rate, training	0.01698	0.04672	0.03921	0.00744	0.02451	0.02246	0.00954	0.0313	0.03422
Error, validation	0.052	1.018	0.557	0.037	0.637	0.349	0.034	0.924	0.503
Rate, validation	0.00000	0.01072	0.00547	0.00000	0.00902	0.00562	0.00000	0.01091	0.00536
Error, testing	0.178	3.546	1.413	0.054	2.899	1.217	0.117	3.487	1.406
Rate, testing	0.00000	0.03397	0.01384	0.00000	0.02687	0.00467	0.00000	0.03455	0.01424

6.2 Heart Disease Diagnosis

The heart disease data set was obtained from Cleveland Clinic Foundation and was supplied by Robert Detrano of the V.A. Medical Center, Long Beach, CA. The data set consists of 270 examples. The heart disease original data set consisted of 303 examples, but 6 of them contained missing class values and were excluded from the database. Other 27 examples of the remained data were eliminated as they retained in case of dispute.

Each example in the database contains 13 attributes, which present results of medical tests provided on patients: age, sex, chest pain type, resting blood pressure, cholesterol, fasting blood sugar < 120 (true or false), resting electrocardiogram (norm, abnormal or hyper), max heart rate, exercise induced angina, oldpeak, slope, number of vessels colored and thal (normal, fixed, rever). These attributes have been extracted from a larger set of 75. The goal of diagnosis is to recognize the presence or absence of heart disease given the attributes. Initially, the data set considered four different degrees of the heart disease to classify the predicted results. Later, modification in the problem definition suggested reducing the number of predicted values on two and categorizing results into two classes: presence or absence of illness.

For this set of experiments we applied the same equation to calculate fitness values as for the breast cancer diagnosis problem. Table 3 presents the ANN architectures for heart disease problem. Table 4 reports the generalization accuracy of ANNs, evolved by NWEA, CNWEA and MNWEA.

Table 3. ANN architectures for heart disease diagnosis

	NWEA	CNWEA	MNWEA
Connections (min)	28	26	26
Connections (max)	202	192	200
Connections (mean)	88.4	78.3	82,6
Hidden nodes (min)	2	2	2
Hidden nodes (max)	8	8	8
Hidden nodes (mean)	4.3	3.7	4.1
Generation (min)	127	106	113
Generation (mean)	172.2	157.9	169.6

Table 4. Comparative results of the prediction accuracy for heart disease diagnosis

	NWEA			CNWEA			MNWEA		
	min	max	mean	min	max	mean	min	max	mean
Error, training	7.489	12.166	11.007	5.763	12.005	8.963	5.893	12.137	7.774
Rate, training	0.07879	0.15184	0.12477	0.04831	0.12069	0.09446	0.05271	0.15040	0.10645
Error, validation	11.746	14.301	12.450	9.271	12.158	9.677	9.814	12.816	10.240
Rate, validation	0.12124	0.19706	0.15935	0.08113	0.13812	0.10543	0.09276	0.13762	0.12418
Error, testing	10.126	13.842	12.266	7.009	12.932	10.633	7.112	13.004	10.458
Rate, testing	0.13195	0.17997	0.15165	0.09385	0.14889	0.11676	0.11243	0.17002	0.12276

7 Conclusions

In this paper we have investigated mixing mutation strategies based on different distributions and proposed two modifications of the NWEA learning strategy for ANNs training. A combined NWEA (CNWEA) uses mutation strategies based on Gaussian, Cauchy and uniform distributions at the chromosome level. The mixed NWEA (MNWEA) uses the same mutation strategies as CNWEA, but combines them at the gene level.

The evolution process has been observed under different step sizes determined by mixed distributions. We have compared generalization accuracy of ANNs using the suggested mutation strategies on two simple benchmark data sets. According to our preliminary experiments, both CNWEA and MNWEA evolved compact ANNs with high training and generalization accuracy. The difference in evolved ANN architectures on both modifications was insignificant compared to NWEA; however both of them demonstrated higher generalization accuracy, especially CNWEA. Statistical analysis of data reported in Tables 2 and 4 with t-test showed that the differences between the error accuracies on both training and testing sets are not significant for breast cancer diagnosis (Table 2) and extremely significant for heart disease problem (Table 4). Both modifications have demonstrated higher convergence speed compared to NWEA as measured by the number of iterations before an optimal network is obtained. The preliminary results obtained for CNEWA and MNWEA are promising and encourage further studies on data sets with large number of attributes.

References

1. Yao, X.: A review of evolutionary artificial neural networks. International Journal of Intelligent Systems 8(4), 539–567 (1993)
2. Yao, X.: Evolving artificial neural networks. In: Proceedings of the IEEE, pp. 1423–1447. IEEE Press, Los Alamitos (1999)
3. Rechenberg, I.: Cybernetic solution path of an experimental problem. In: Royal Aircraft Establishment, Farnborough, page Library Translation 1122 (1965)
4. Schwefel, H.-P.: Kybernetische Evolution als Strategie der experimentellen Forschung in der Strömungstechnik. Diplomarbeit, Technische Universität Berlin (1965)
5. Fogel, L.J.: Autonomous automata. Industrial Research 4, 14–19 (1962)
6. Fogel, L.J., Owens, A.J., Walsh, M.J.: Artificial intelligence through simulated evolution. Wiley, New York (1966)
7. Yao, X., Liu, Y.: An Analysis of evolutionary algorithms based on neighborhood and step size. In: Angeline, P.J., McDonnell, J.R., Reynolds, R.G., Eberhart, R. (eds.) EP 1997. LNCS, vol. 1213, pp. 297–307. Springer, Heidelberg (1997)
8. Yao, X., Liu, Y.: Fast Evolutionary Programming. In: Proc. of the Fifth Annual Conference on Evolutionary Programming, pp. 451–460. MIT Press, Cambridge (1996)
9. Yao, X., Liu, Y.: Evolutionary programming made faster. In: IEEE Transactions on Evolutionary Computation, vol. 3, pp. 82–102. IEEE Press, Los Alamitos (1999)
10. Davoian, K., Lippe, W.-M.: Including phenotype information in mutation to evolve artificial neural networks. In: Proc. of the IEEE International Joint Conference on Neural Networks (IJCNN 2007), Orlando, USA (2007)
11. Davoian, K., Lippe, W.-M.: Exploring the role of activation function type in evolutionary artificial neural networks. In: Proc. of the 2008 Int. Conference on Data Mining (DMIN 2008), pp. 443–449. CSREA Press, Las Vegas (2008)
12. Prechelt, L.: Proben1-A set of neural network benchmark problems and benchmarking rules. Fakultät für Informatik, Universät Karlsruhe, Germany, Tech. Rep. 21/94 (1994)
13. Yao, X., Liu, Y.: Scaling up evolutionary programming algorithms. In: Porto, V.W., Waagen, D. (eds.) EP 1998. LNCS, vol. 1447, pp. 103–112. Springer, Heidelberg (1998)
14. Lippe, W.-M.: Soft-Computing mit Neuronalen Netzen, Fuzzy-Logic und Evolutionären Algorithmen. Springer, Heidelberg (2006)