METHODS OF PARAMETRIC IDENTIFICATION BASED ON MONKEY BEHAVIOR

Anait Karapetyan, Eugene Fedorov, Kostiantyn Rudakov and Anatolii Chepynoha

Cherkasy State Technological University

Introduction

In modern conditions, the development of methods for the parametric identification of artificial neural network models used in intelligent computer systems is an urgent task.

However, existing parametric identification methods that obtain approximate solutions through global search do not guarantee convergence, while methods that obtain approximate solutions through local search have a high probability of getting trapped in local optima. Furthermore, methods that obtain exact solutions have high computational complexity. Thus, there is a problem of insufficient efficiency in existing parametric identification methods.

To reduce the probability of getting stuck in a local extremum and to speed up parametric identification, modern heuristics (or metaheuristics) are used. Metaheuristics expand the possibilities of heuristics by combining heuristic methods based on a high-level strategy. Metaheuristics often use the behavior of one or a group of animals. Metaheuristics are an approximate and usually non-deterministic method. Advanced metaheuristics use experience accumulated during the search, represented in memory, to manage the search process.

Object, subject and goal of the research

The object of the research: The process of parametric identification of an artificial neural network model.

The subject of the research: Methods of parametric identification of an artificial neural network model based on metaheuristics.

The goal to enhance the efficiency of the parametric identification of an artificial neural network model using metaheuristic methods with dynamic parameters based on the behavior of monkeys.

To achieve this goal, it is necessary to solve the following tasks:

1. To develop a numerical optimization method based on the synthesis of the monkey search algorithm.
2. To develop a numerical optimization method based on the synthesis of the monkey algorithm.
3. To develop a numerical optimization method based on the synthesis of the spider monkey optimization algorithm.
4. Conduct a numerical investigation of the proposed optimization methods.

Monkey search with dynamic parameters for optimizing numerical functions

1. Initialization.

1.1. Specifying the minimum distance between vertex positions (solutions) in memory ε, the parameter δ that controls the step size for generating a new solution, subject to the condition that, $\varepsilon > 0$, $0 < \delta < 1$.

1.2. Specifying the maximum number of trees N, maximum number of levels for each tree H, maximum number of vertices for each tree H at a given level N^H, memory size L, length of the vertex position vector M, minimum and maximum values for the solution $x_j^{\min}, x_j^{\max}$, $j \in \overline{1,M}$.

1.3. Specifying the cost function (objective function)

$$F(x) \to \min_x,$$ where x is the vector position of the tree vertex.

1.4. Creating the optimal solution randomly as a real vector

$$x^* = (x_1^*, \ldots, x_M^*),\ x_j^* = x_j^{\min} + (x_j^{\max} - x_j^{\min})U(0,1),$$

where $U(0,1)$ - is the function returns a standard uniformly distributed random number.

1.5. Memory $P^M = \varnothing$.

2. Tree number $n = 1$.

3. Creation of the n-th tree.

3.1. Set of explored node positions $V_n = \varnothing$.

3.2. Set of edges $A_n = \varnothing$.

3.3. Set of node positions H at the level $V_n^H = \varnothing$.

3.4. Exploring the root of the tree.

3.4.1. If $\left|P^M\right| < L$, then randomly create a root position, then the root position is randomly created.

$x^{root} = (x_1^{root}, \ldots, x_M^{root})$, $x_j^{root} = x_j^{\min} + (x_j^{\max} - x_j^{\min})U(0,1)$,

3.4.2. If $\left|P^M\right| = L$, then from memory, P^M a vertex with a number is randomly selected $v = round(1 + (L-1)U(0,1))$, which becomes a root x^{root}, where $round()$ – is a function rounding a number to the nearest integer.

3.4.3. Add the root position to the set of explored node positions

$V_n = V_n \cup \{x^{root}\}$

3.5. Set the position of the current tree vertex $x^{cur} = x^{root}$

3.6. Ascending to explored tree vertices.

3.6.1. Tree level number $m = 1$.

3.6.2. If $\neg\exists x^{left} \in V_n : (x^{cur}, x^{left}) \in A_n$ and $\neg\exists x^{right} \in V_n : (x^{cur}, x^{right}) \in A_n$, then go to step 3.7.

3.6.3. Calculate the probability of ascending to the left vertex

$$p = \frac{1/F(x^{left})}{1/F(x^{left}) + 1/F(x^{right})}$$

3.6.4. Select the position of the current vertex

$$\lambda = U(0,1),\ x^{cur} = \begin{cases} x^{left}, & p < \lambda \\ x^{right} & p \geq \lambda \end{cases}$$

3.6.5. If $m < H$, then $m = m + 1$, go to step 3.6.2, otherwise 3.5.

3.7. Ascending to unexplored tree vertices.

3.7.1. Create the position of the left vertex x^{left} by perturbation (e.g., mutation).

3.7.1.1. $\sigma_j(m) = \left(1 - \frac{m}{H}\right)\delta(x_j^{\max} - x_j^{\min})$,

$x_j^{left} = x_j^{cur} + \sigma_j(m)N(0,1),\ j \in \overline{1,M}$,

where $N(0,1)$ – is a function returning a standard normally distributed random number.

3.7.1.2. $x_j^{left} = \max\{x_j^{\min}, x_j^{left}\},\ x_j^{left} = \min\{x_j^{\max}, x_j^{left}\},\ j \in \overline{1,M}$

3.7.2. Create the position of the right vertex x^{right} by perturbation (e.g., mutation).

3.7.2.1. $\sigma_j(m) = \left(1 - \frac{m}{H}\right)\delta(x_j^{\max} - x_j^{\min})$,

$x_j^{right} = x_j^{cur} + \sigma_j(m)N(0,1),\ j \in \overline{1,M}$

3.7.2.2. $x_j^{right} = \max\{x_j^{\min}, x_j^{right}\},\ x_j^{right} = \min\{x_j^{\max}, x_j^{right}\},\ j \in \overline{1,M}$

3.7.3. If $F(x^{cur}) < \min\{F(x^{left}), F(x^{right})\}$, then go to step 3.7.1.

3.7.4. Add the vertex positions to the set of explored vertex positions

$V_n = V_n \cup \{x^{left}, x^{right}\}$

3.7.5. Add the edges to the set of edges

$A_n = A_n \cup \{(x^{cur}, x^{left}), (x^{cur}, x^{right})\}$

3.7.6. If $m < H$, then $m = m + 1$, go to step 3.5.

3.7.7. Add the vertex position to the set of vertex positions H at the level

$V_n^H = V_n^H \cup \{x^{left}, x^{right}\}$

3.7.8. If $\left|V_n^H\right| < N^H$, then go to step 3.5.

4. Determine the best vertex position of n-th tree

$$x_n^{best} = \arg\min_{x \in V_n^H} F(x)$$

5. Add the best vertex position of n-th tree to the memory.

5.1. If $\left|P^M\right| = 0$, then $P^M = P^M \cup \{x_n^{best}\}$, go to step 6.

5.2. If $\left|P^M\right| > 0 \wedge \min\limits_{x \in P^M} \left\| x - x_n^{best} \right\| < \varepsilon$, then go to step 6.

5.3. If $\left|P^M\right| < L$, then $P^M = P^M \cup \{x_n^{best}\}$, go to step 6.

5.4. Determine the worst position in the memory.

$$x^{worst} = \arg \max_{x \in P^M} F(x)$$

5.5. If $F(x^{worst}) \geq F(x_n^{best})$, then $P^M = P^M \setminus \{x^{worst}\}$, $P^M = P^M \cup \{x_n^{best}\}$.

6. Stop condition

If $n < N$, then $n = n + 1$, go to step 3, otherwise – stop.

7. Identification of the best global position

$$x^* = \arg \min_{x \in P^M} F(x)$$

The solution is x^*.

Monkey Algorithm with Dynamic Parameters for optimizing numerical functions

1. Initialization

1.1. Set the minimum and maximum step lengths $a^{\min}, a^{\max}$ for generating the position vector during the ascent process, and the minimum and maximum step lengths $b^{\min}, b^{\max}$ for generating the position vector during the observation-jump process, meanwhile $a^{\min} > 0, a^{\max} > 0, b^{\min} > 0, b^{\max} > 0$.

1.2. Set the maximum number of iterations N, the maximum number of iterations for local search N^L, the population size K, the length of the monkey's position vector M, and the minimum and maximum values for the position vector $x_j^{\min}, x_j^{\max}$, $j \in \overline{1, M}$.

1.3. Define the cost function (objective function):

$$F(x) \to \min_x,$$

where x is the monkey's position vector.

1.4. Create the initial population P.

1.4.1. Monkey number $k = 1$.

1.4.2. Generate a random position vector x_k

$$x_k = (x_{k1},..., x_{kM}),\ x_{kj} = x_j^{\min} + (x_j^{\max} - x_j^{\min})U(0,1),$$

where $U(0,1)$ – is a function that returns a standard uniformly distributed random number.

1.4.3. If $x_k \notin P$, then $P = P \cup \{x_k\}$, $k = k + 1$.

1.4.4. If $k \le K$, then go to step 1.4.2.

1.5. Determine the monkey with the best objective function value:

$$k^* = \arg \min_{k \in \overline{1,K}} F(x_k),\ x^* = x_{k^*}$$

1.6. Set the initial step lengths

$$a(1) = a^{\max},\ b(1) = b^{\max}.$$

2. Number of iterations $n = 1$.

3. Climbing process.

3.1. Monkey number $k = 1$.

3.2. $m = 1$.

3.3. Create a position vector increment Δx_k

$$\Delta x_k = (\Delta x_{k1}, \ldots, \Delta x_{kM}),\ \Delta x_{kj} = round(1 - 2U(0,1))a(n),$$

where $round()$ – is a function that rounds a number to the nearest integer.

3.4. Calculate the pseudo-gradient of the cost function

$$F_{kj}(x_k) = \frac{F(x_k + \Delta x_k) - F(x_k - \Delta x_k)}{2\Delta x_{kj}},\ j \in \overline{1, M}.$$

3.5. Create a position vector based on mutation

$$y = (y_1, \ldots, y_M),\ y_j = x_{kj} + \operatorname{sgn}(F_{kj}(x_k))a(n)$$

3.6. If $\forall j \in \overline{1, M}\, x_j^{\min} \le y_j \le x_j^{\max}$, then $x_k = y$.

3.7. If $m < N^L$, then $m = m + 1$, go to step 3.3.

3.8. If $k \le K$, then $k = k + 1$, go to step 3.2.

4. Observation-jump process.

4.1. Monkey number $k=1$.

4.2. Create a position vector y based on mutation

$y=(y_1,\dots,y_M)$, $y_j = x_{kj} + (1-2U(0,1))b(n)$

4.3. If $(F(y)<F(x_k)) \wedge \left(\forall j \in \overline{1,M}\, x_j^{\min} \le y_j \le x_j^{\max}\right)$, then $x_k = y$, otherwise go to step 4.2.

4.4. If $k \le K$, then $k=k+1$, go to step 4.2.

5. Climbing process.

5.1. Monkey number $k=1$.

5.2. $m=1$.

5.3. Create a position vector increment Δx_k

$\Delta x_k = (\Delta x_{k1},\dots,\Delta x_{kM})$, $\Delta x_{kj} = round(1-2U(0,1))a(n)$

5.4. Calculate the pseudo-gradient of the cost function

$$F_{kj}(x_k) = \frac{F(x_k + \Delta x_k) - F(x_k - \Delta x_k)}{2\Delta x_{kj}},\ j \in \overline{1,M}.$$

5.5. Create a position vector based on mutation

$y = (y_1, \dots, y_M)$, $y_j = x_{kj} + \delta(n) \operatorname{sgn}(F_{kj}(x_k)) a(n)$

5.6. If $\forall j \in \overline{1,M} x_j^{\min} \le y_j \le x_j^{\max}$, then $x_k = y$.

5.7. If $m < N^L$, then $m = m + 1$, go to step 5.3.

5.8. If $k \le K$, then $k = k + 1$, go to step 5.2.

6. Somersault process.

6.1. Monkey number $k = 1$.

6.2. Create a position vector based on the crossover

$$\mu = (\mu_1, \dots, \mu_M),\ \mu_j = \frac{1}{K} \sum_{l=1}^{K} x_{lj}$$

6.3. Create a position vector y based on the crossover

$y = (y_1, \dots, y_M)$, $y_j = x_{kj} + (1 - 2U(0,1))(\mu_j - x_{kj})$,

6.4. If $\forall j \in \overline{1,M} x_j^{\min} \le y_j \le x_j^{\max}$, then $x_k = y$.

6.5. If $k \le K$, then $k = k + 1$, go to step 6.2.

7. Determine the best monkey based on the cost function

$$k^* = \arg\min_{k \in \overline{1,K}} F(x_k)$$

8. Determine the global best position.

If $F(x_{k^*}) < F(x^*)$, then $x^* = x_{k^*}$.

9. Stop condition

If $n < N$, then $n = n + 1$, $a(n) = a^{\min} + (a^{\max} - a^{\min})\left(1 - \frac{n}{N}\right)$, $b(n) = b^{\min} + (b^{\max} - b^{\min})\left(1 - \frac{n}{N}\right)$, go to step 3.

The result is x^*.

Spider Monkey Optimization with Dynamic Parameters for optimizing numerical functions

1. Initialization.

1.1. Setting the minimum and maximum probabilities $p^{\min}, p^{\max}$.

1.2. Setting the maximum number of iterations N, the maximum number of groups $G^{\max}$, the maximum number of iterations without updating the position of the local leader N^{LL} (usually $N^{LL} = M \cdot K$), the maximum number of iterations without updating the position of the global leader N^{G} (usually $N^{G} \in [K/2, 2K]$), the population size K (for integer division of the population into groups we assume $K = 60$), the length of the monkey position vector M, and the minimum and maximum values for the position vector $x_j^{\min}, x_j^{\max}$, $j \in \overline{1, M}$.

1.3. Setting the cost function (objective function)

$$F(x) \to \min_{x},$$

where x – is the monkey position vector.

1.4. Number of groups $G = 1$.

1.5. Creation of one group.

1.5.1. Monkey number in the group $k=1$.

1.5.2. Random creation of a position vector x_{gk}

$$x_{1k}=(x_{1k1},...,x_{1kM}),\ x_{1kj}=x_j^{\min}+(x_j^{\max}-x_j^{\min})U(0,1),$$

where $U(0,1)$ – is a function returning a standard uniformly distributed random number.

1.5.3. If $x_{1k}\notin P_1$, then $P_1=P_1\cup\{x_{1k}\}$, $k=k+1$.

1.5.4. If $k\le|P_1|$, then go to step 1.5.2.

1.6. Determine the local leader in the group

$$k^*=\arg\min_{k\in\overline{1,|P_1|}}F(x_{1k}),\ x_1^*=x_{1k^*}$$

1.7. Determine the global leader $x^*=x_1^*$

1.8. Number of iterations without updating the local leader position $m_1=0$

1.9. Number of iterations without updating the global leader position $m=0$

1.10. Probability of preserving the position $p^s(1)=p^{\min}$.

1.11. Probability of a random position $p^r(1)=p^{\max}$.

2. Iteration number $n=1$.

3. Local leader phase.

3.1. Group number $g=1$.

3.2. Monkey number в g-group $k=1$.

3.3. Component number $j=1$.

3.4. Randomly select the number of the second monkey

$$s = round\left(1+\left(\left|P_g\right|-1\right)U(0,1)\right),$$

where $round()$ – is a function that rounds a number to the nearest integer.

3.5. If $s=k$, then go to step 3.4.

3.6. Save the position vector or create a position vector based on the crossover

$$\lambda = U(0,1),$$

$$\tilde{x}_{gkj} = \begin{cases} x_{gkj}, & \lambda < p^s(n) \\ x_{gkj} + U(0,1)(x^*_{gj} - x_{gkj}) + (1-2U(0,1))(x_{gsj} - x_{gkj}), & \lambda \ge p^s(n) \end{cases}$$

3.7. If $j<M$, then $j=j+1$, go to step 3.4.

3.8. If $k<\left|P_g\right|$, then $k=k+1$, go to step 3.3.

3.9. If $g<G$, then $g=g+1$, go to step 3.2.

4. Modify the position of group members.

4.1. Group number $g = 1$.

4.2. Monkey number в g-group $k = 1$.

4.3. If $F(\tilde{x}_{gk}) < F(x_{gk})$, then $x_{gk} = \tilde{x}_{gk}$.

4.4. If $k < \left|P_g\right|$, then $k = k + 1$, go to step 4.3.

4.5. If $g < G$, then $g = g + 1$, go to step 4.2.

5. Calculate probabilities for group members.

5.1. Group number $g = 1$.

5.2. Monkey number в g-group $k = 1$

5.3. $p_{gk} = 0.1 + 0.9 \dfrac{1/F(x_{gk})}{\max\limits_{s \in \overline{1,\left|P_g\right|}} 1/F(x_{gs})}$

5.4. If $k < \left|P_g\right|$, then $k = k + 1$, go to step 5.3.

5.5. If $g < G$, then $g = g + 1$, go to step 5.2.

6. Global leader phase.

6.1. Group number $g = 1$.

6.2. $c = 1$.

6.3. If $c \geq |P_g|$, then go to step 6.13.

6.4. Monkey number в g-group $k = 1$.

6.5. If $U(0,1) \geq p_{gk}$, then go to step 6.11.

6.6. $c = c + 1$.

6.7. Randomly select the component number

$$j = round\left(1 + (M - 1)U(0,1)\right)$$

6.8. Randomly select the number of the second monkey

$$s = round\left(1 + \left(|P_g| - 1\right)U(0,1)\right)$$

6.9. If $s = k$, then go to step 6.8.

6.10. Create a position vector based on the crossover

$$\tilde{x}_{gkj} = x_{gkj} + U(0,1)(x^*_{gj} - x_{gkj}) + \delta(n)(1 - 2U(0,1))(x_{gsj} - x_{gkj})$$

6.11. If $k < |P_g|$, then $k = k + 1$, go to step 6.5.

6.12. Modify the position of group members

$$x_{gk} = \tilde{x}_{gk},\ k \in \overline{1, |P_g|},\ g \in \overline{1, G}$$

6.13. If $g < G$, then $g = g + 1$, go to step 6.2.

7. Local leader learning phase.

7.1. Group number $g = 1$.

7.2. Determine the best monkey in the g-group based on the objective function

$$k^* = \arg \min_{k \in \overline{1, |P_g|}} F(x_{gk})$$

7.3. Determine the local leader in the g-group.

If $F(x_{gk^*}) < F(x_g^*)$, then $x_g^* = x_{gk^*}$, $m_g = m_g + 1$.

7.4. If $g < G$, then $g = g + 1$, go to step 7.2.

8. Global leader training phase.

8.1. Determine the best local leader $g^* = \arg \min_{g \in \overline{1, G}} F(x_g^*)$

8.2. Determine the global leader.

If $F(x_{g^*}) < F(x^*)$, then $x^* = x_{g^*}$, otherwise $m = m + 1$.

9. Local leader decision-making phase.

9.1. Group number $g = 1$.

9.2. If $m_g \le N^{LL}$, then go to step 9.9.

9.3. $m_g = 0$.

9.4. Monkey number в g-group $k = 1$.

9.5. Component number $j = 1$.

9.6. Create a position vector randomly or based on the crossover

$$\lambda = U(0,1),\ x_{gkj} = \begin{cases} x_j^{\min} + (x_j^{\max} - x_j^{\min})U(0,1), & \lambda < p^r(n) \\ x_{gkj} + U(0,1)(x_j^* - x_{gkj}) + U(0,1)(x_{gkj} - x_{gj}^*), & \lambda \ge p^r(n) \end{cases}$$

9.7. If $j < M$, then $j = j + 1$, go to step 9.6.

9.8. If $k < |P_g|$, then $k = k + 1$, go to step 9.5.

9.9. If $g < G$, then $g = g + 1$, go to step 9.2.

10. Global leader decision-making phase.

10.1. If $m \le N^G$, then go to step 11.

10.2. Number of iterations without updating the global leader position $m = 0$.

10.3. Fuse the group into the population $P = \bigcup_{g=1}^{G} P_g$

10.4. If $G < G^{\max}$, then $G = G + 1$, fusion the population P into G groups, go to step 10.6.

10.5. If $G \ge G^{\max}$, then $G = 1$.

10.6. Determine local leaders in groups

$x_g^* = \arg \min\limits_{k \in \overline{1,|P_g|}} F(x_{gk}),\ g \in \overline{1,G}$

10.7. Number of iterations without updating the positions of local leaders

$m_g = 0,\ g \in \overline{1,G}$

11. Stop condition

If $n < N$, then $n = n + 1$, $p^r(n) = p^{\min} + (p^{\max} - p^{\min})\left(1 - \dfrac{n}{N}\right)$, $p^r(n) = p^{\min} + (p^{\max} - p^{\min})\left(\dfrac{n}{N}\right)$, go to step 3.

The result is x^*.

Numerical study of the proposed methods

The numerical study was conducted on the Ackley function

$$f(x) = -a\exp\left(-b\sqrt{\frac{1}{M}\sum_{j=1}^{M} x_j^2}\right) - \exp\left(\frac{1}{M}\sum_{j=1}^{M}\cos(cx_j)\right) + a + \exp(1),$$

where $a = 20, b = 0.2, c = 2\pi$, $x_j^{\min} = -32.768, x_j^{\max} = 32.768$, $M = 30$.

In the study, the population size is $K = 60$, the maximum number of iterations or trees $N = 100$ (for monkey search and monkey algorithm) or $N = 100000$ (for spider monkey optimization) were used, the minimum distance between node positions in memory $\varepsilon = \dfrac{\| x^{\max} - x^{\min} \|}{8}$, a parameter controlling the step size for generating a new solution $\delta = 0.1$, the maximum number of levels for each tree $H = 100$, the maximum number of vertices on H level $N^H = 100$, memory size $L = 10$, minimum and maximum step lengths $a^{\min} = 0.1, a^{\max} = 0.001$ for generating the position vector during the ascent process, minimum and maximum step lengths $b^{\min} = 1, b^{\max} = 10$ for generating the position vector during the observation-jump process, the maximum number of local search iterations $N^L = 1000$, minimum and maximum probabilities $p^{\min} = 0.1, p^{\max} = 0.9$, the maximum number of groups $G^{\max} = 6$, the max number of iterations without updating the position of the local leader $N^{LL} = 1800$, the maximum number of iterations without updating the position of the global leader $N^G = 60$.

The results of comparing the proposed methods with classical methods are presented in Table 1.

Table 1

Comparison of proposed metaheuristic methods for parametric identification with existing ones based on mean squared error criterion

Method	Mean squared error	
	of the proposed method with dynamic parameters	of existing method
Monkey search	0.04	0.08
Monkey algorithm	0.03	0.07
Spider monkey optimization	0.02	0.06

Discussion

The selected parameter values of the proposed parametric identification methods ensure a high probability of mutation and random solution generation in the initial iterations and a low probability of mutation and random solution generation in the final iterations.

Classical methods based on the behavior of monkeys do not take into account the iteration number in the mutation operator and the random solution generation operator, which reduces the accuracy of the solution search (Table 1).

The proposed methods allow for the elimination of these shortcomings.

Conclusions

In the work, the problem of insufficient efficiency of parameter identification methods for artificial neural network models used in intelligent computer systems is considered. To improve the efficiency of parameter identification methods, metaheuristic methods based on the behavior of monkeys with dynamic parameters were proposed, which improve upon classical methods (monkey search, monkey algorithm, spider monkey optimization).. The proposed methods of parameter identification, due to global search at initial iterations and local search at final iterations, allow to increase the accuracy of search and does not require the transformation of the target function. The proposed metaheuristics can expand the scope of application of parameter identification methods based on monkey behavior and contribute to the increase of efficiency of intelligent computer systems. The prospects of further research include the investigation of the proposed methods for a wide range of artificial intelligence tasks.