README: NK Ensemble Neural Networks (experiments with double pole balancing problem)

Renato Tinós

November 19, 2019

1 Background

Consider an artificial neural network with one or more hidden layers. The artificial neural network is employed to reproduce a function with output $\psi \in \mathbb{R}$ and input vector $\mathbf{u} \in \mathbb{R}^p$. Instead of using only one output for the artificial neural network, we will use N outputs, each one used to reproduce the output of the mapped real function. Alternatively, we can see the artificial neural network as a committee with N artificial neural networks that differ only in the weights of the output layer. The last hidden layer has N neurons.

In the artificial neural network used here, all the weights of the network are randomly created and kept fixed during its training. During the training, only a vector $\mathbf{x} \in \mathbb{B}^N$ specifying if the neurons in the last hidden layer are activated or not is optimized. If $x_i = 1$, the i-th neuron of the last hidden layer is turned on, i.e., its activation is used as input for neurons in the output layer connected to it. If $x_i = 0$, the i-th neuron of the last hidden layer is turned off. While the connectivity in the previous layers can be sparse or full, the connectivity of the output layer is always sparse. Each neuron of the output layer receives inputs from the outputs of $K + 1$ neurons of the last hidden layer. The i-th neuron of the last hidden layer is always connected to the i-th output neuron. The other K connections for the output layer are defined according to a neighbourhood model. In the adjacent model, the i-th output neuron is connected to neurons i, $i + 1$, ..., $i + K$ of the last hidden layer. In the random model, the i-th output neuron is connected to neuron i and other K random neurons of the last hidden layer. Figure 1 illustrates an example with 2 hidden layers, where the first hidden layer is composed by reservoirs.

During the training, the i-th output of the neural network at iteration t can be written as:

$$y_i(\mathbf{u}(t), \mathbf{x}, t) = \phi\left(\sum_{j=1}^{K+1} \left(W_{i,k_j}^{out}\ s_{k_j}(\mathbf{u}(t), t)\ x_{k_j}\right)\right) \quad (1)$$

where $\phi(.)$ is the activation function of the neurons in the output layer, W_{i,k_j}^{out} is the weight between the k_j-th neuron of the last hidden layer and the i-th output neuron, $s_{k_j}(\mathbf{u}, t)$ is the output of the k_j-th neuron of the last hidden layer and x_{k_j} is the k_j-th element of binary vector $\mathbf{x}$. The connectivity between the neurons in the last hidden layer and the i-th output neuron can be defined by a binary mask $\mathbf{m}_i \in \mathbb{B}^N$. The number of 1's in $\mathbf{m}_i$ is $K+1$.

In the neural network investigated here, each output neuron is evaluated independently. As weights of the neural network are fixed, the evaluation of the i-th output neuron for a given problem depends on binary vector $\mathbf{x}$ and on binary mask $\mathbf{m}_i$. We denote the evaluation of the i-th output neuron as $f_i(\mathbf{x}, \mathbf{m}_i)$.

We propose to optimize vector $\mathbf{x}$ in order to maximize the mean $f_i(\mathbf{x}, \mathbf{m}_i)$. Thus, the evaluation function for the optimization process is:

$$f(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^{N} f_i(\mathbf{x}, \mathbf{m}_i) \tag{2}$$

The evaluation function given by Eq. 2 is exactly the same evaluation function used in the NK landscapes. In this way, optimizing the artificial neural network investigated here is equivalent to optimize the NK landscapes. In the NK landscapes, the individual fitness contributions $f_i(\mathbf{x}, \mathbf{m}_i)$ are random.

The algorithms for training the artificial neural network using NK landscapes algorithms are given in the following.

Algorithm 1:

i. Create an artificial neural network $ANN(\mathbf{M})$ with at least one hidden layer. The matrix $\mathbf{M} = [\mathbf{m}_1 \dots \mathbf{m}_N]$ defines the connectivity between the neurons of the last hidden layer and the neurons in the output layer. The mask $\mathbf{m}_i$, $i = 1, \dots, N$, has exactly $K+1$ ones defined according to the neighbourhood model (adjacent or random). All the weights and bias of $ANN(\mathbf{M})$ are random. A vector $\mathbf{x} \in \mathbb{B}^N$ is used to turn on or off the neurons in the last hidden layer.

ii. Evaluate each output neuron of $ANN(\mathbf{M})$ according to all combinations of the elements of vector $\mathbf{x}$ used by the output neuron. We will have $N2^{K+1}$ individual fitness contributions $f_i(\mathbf{x}, \mathbf{m}_i)$. For example, if $N = 5$, $K = 2$ and the neighbourhood adjacent model is used, $\mathbf{m}_1 = [1, 1, 1, 0, 0]^{\mathrm{T}}$. Thus, the first output is evaluated for all eight combinations of $[x_1, x_2, x_3]^{\mathrm{T}}$.

iii. Save $ANN(\mathbf{M})$. Also, save $\mathbf{M}$ and the individual fitness contributions $f_i(\mathbf{x}, \mathbf{m}_i)$ for all combinations of the elements of vector $\mathbf{x}$ used by the output neuron.

Algorithm 2:

i. Load $\mathbf{M}$ and the individual fitness contributions $f_i(\mathbf{x}, \mathbf{m}_i)$.

ii. Create an NK model with $\mathbf{M}$ and the individual fitness contributions $f_i(\mathbf{x}, \mathbf{m}_i)$.

iii. Optimize the NK model.

iv. Save the best decision vector $\mathbf{x}^*$.

Algorithm 3:

i. Load the artificial neural network $ANN(\mathbf{M})$ created in Algorithm 1 and the best decision vector $\mathbf{x}^*$ found in Algorithm 2.

ii. Evaluate $ANN(\mathbf{M})$ with vector $\mathbf{x}^*$. The signal employed to reproduce the output of the mapped function is composed by a weighted combination of the outputs $y_i(\mathbf{u}(t), \mathbf{x}^*, t)$, $i = 1, \ldots, N$, of the neural network.

2 Experimental Results

In order to test the methodology proposed in this work, the artificial neural network is applied to the double pole balancing problem without velocity information [5]. In this problem, the inputs of the artificial neural network at step t are composed by scaled cart position and angles of the two poles at step t, i.e., $\mathbf{u}(t) = [x_c(t)/x_c^{max}, \theta_1(t)/\theta_1^{max}, \theta_2(t)/\theta_2^{max}]^{\mathrm{T}}$ where $x_c(t)$ is the actual cart position, $\theta_i(t)$ is the actual angle of the i-th pole, and x_c^{max} and θ_i^{max} are the maximum allowed values used to scale the inputs between -1 and +1. In Algorithm 1, each output of the network is independently evaluated. The action (in Newtons) applied to the cart at iteration t when evaluating the i-th output for solution $\mathbf{x}$ is given by:

$$action(t) = 10 y_i(\mathbf{u}(t), \mathbf{x}, t) \tag{3}$$

In Algorithm 1, the following fitness function [1, 2] is used:

$$f = 10^{-4} t + 0.9 f_{stable} \tag{4}$$

where t is the number of steps inside the success domain until a limit of 5000 steps and

$$f_{stable} = \begin{cases} 0, & \text{if } t < 100 \\ \dfrac{0.75}{\sum_{i=t-100}^{t} (|x_c(i)| + |\dot{x_c}(i)| + |\theta_1(i)| + |\dot{\theta_1}(i)|)}, & \text{otherwise,} \end{cases} \tag{5}$$

The success domain is given by $x_c \in [-2.4, 2.4]$ meters and $\theta_i \in [-36, 36]$ degrees. The function f_{stable} indicates the stability of the system during the last 100 time steps if $t \geq 100$. For the evaluation of f in Algorithm 1, the systems always starts from the state $x_c(0) = \theta_2(0) = \dot{x_c}(0) = \dot{\theta}_1(0) = \dot{\theta}_2(0) = 0$ and $\theta_1(0) = 4.5$ degrees. The parameters of the double pole system used here are [1]: mass of cart equal to 1 kg, mass of pole 1 equal to 0.1 kg, mass of pole 2 equal to 0.01 kg, length of pole 1 equal to 1 m, length of pole 1 equal to 0.1 m, coefficient of friction of the cart on the track equal to 0.0005, coefficient of friction of the poles equal to 0.000002. The 4th order Runge-Kutta method with integration step equal to 0.01 was used.

Here, we present the results for the neural network presented in Figure 1 with $N = 20$, i.e., the number of neurons in the output layer (and also in the last hidden layer) is equal to 20. We tested five values for the connectivity degree between the last hidden layer and the output layer: $K = 2, 5, 6, 7, 8$. The adjacent neighbourhood model was considered. The first hidden layer is composed by $r = 1$ reservoir [3] with 80 neurons. The connectivity density for the reservoir is 25%, i.e., each neuron of the reservoir has recurrent connections to 25% of the neurons in the reservoir. The spectral radius for the reservoir is 0.95. All weights of the artificial neural network are fixed, being randomly generated between [-1.0,1.0]. All the neurons use the hyperbolic tangent function as the activation function.

The number of runs is 30. In each run, Algorithm 1 is used to compute the individual fitness contributions $f_i(\mathbf{x}, \mathbf{m}_i)$. Thus, the neural network is generated and each output y_i is evaluated using Eq. 4 for all possible combinations of the elements of vector $\mathbf{x}$ used by the i-th output neuron. Then, Algorithm 2 is used to obtain the best solution $\mathbf{x}^*$ for the evaluation given by Eq. 2. Here, as N is small (N=20), the exhaustive method is used to optimize the NK landscapes.

Finally, Algorithm 3 is employed to evaluate the artificial neural network with the combined outputs (committee). The output of the committee at step t is given by:

$$y_{committee}(\mathbf{u}(t), \mathbf{x}^*, t) = \sum_{i=1}^{N} a_i(\mathbf{x}^*) y_i(\mathbf{u}(t), \mathbf{x}^*, t) \tag{6}$$

where $\mathbf{u}(t)$ is the input vector of the neural network at step t, $\mathbf{x}^*$ is the solution vector obtained by Algorithm 2, y_i is the i-th output of the neural network and the weight a_i is given by

$$a_i(\mathbf{x}^*) = \frac{f_i(\mathbf{x}^*, \mathbf{m}_i)}{\sum_{i=1}^{N} f_i(\mathbf{x}^*, \mathbf{m}_i)} \tag{7}$$

i.e., the outputs with better results have higher weights. We tested also the uniform case where $a_i(\mathbf{x}) = \frac{1}{N}$ and the case where only a percentage of the

best outputs are used in Eq. 6, but the results were similar or worse. The action to the cart at step t is:

$$action(t) = 10 y_{committee}(\mathbf{u}(t), \mathbf{x}^*, t) \quad (8)$$

In Algorithm 3, the committee is evaluated using the generalization test proposed in [4]. In this generalization test, the committee is evaluated 625 times, each time with different initial settings for cart position, cart velocity, pole 1 angle, and pole 1 velocity. The initial pole 2 angle and velocity are always zero. The combination of five different initial settings for each variable is considered: 5, 25, 50, 75, and 95% of a reduced range of the variables. The reduced range is between -2.14 and 2.14 m for cart position, -1.35 and 1.35 m/s for cart velocity, -3.6 and 3.6 degrees for pole 1 angle, -8.6 and 8.6 degrees/s for pole 1 velocity. The evaluation of the generalization test is the number of times that the system remains in the success domain after 1000 steps. In order to compare the committee with the best individual output obtained by Algorithm 1, the same generalization test was applied to the best individual output obtained by Algorithm 1.

Tables 1-3 show the results for all runs in the experiments. The results for the results of the generalization test for $y_{committee}(\mathbf{u}(t), \mathbf{x}^*, t)$ are given in the column indicated as "Committee", while "Best Output" indicates the results for the best individual output obtained by Algorithm 1. The best results of $f_i(\mathbf{x}, \mathbf{m}_i)$ among the $N2^{K+1}$ possible fitness contributions obtained in Algorithm 1, i.e., the best evaluation of Eq. 4 for the individual outputs of the neural network using the best combination of the respective inputs for the output neuron, are also shown in the Table (this result is denoted $max(f_i(\mathbf{x}, \mathbf{m}_i))$. It is important to observe that evaluations of f (Eq. 4) higher than 0.5 indicate that the double pole system was in the success domain for all 5000 steps.

References

[1] P. Dürr, C. Mattiussi, and D. Floreano. Neuroevolution with analog genetic encoding. In *Parallel Problem Solving from Nature-PPSN iX*, pages 671–680. Springer, 2006.

[2] F. Jiang, H. Berry, and M. Schoenauer. Supervised and evolutionary learning of echo state networks. In *Parallel Problem Solving from Nature–PPSN X*, pages 215–224. Springer, 2008.

[3] M. Lukoševičius and H. Jaeger. Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3):127–149, 2009.

[4] D. Whitley, S. Dominic, R. Das, and C. W. Anderson. Genetic reinforcement learning for neurocontrol problems. In J. J. Grefenstette, editor,

Genetic Algorithms for Machine Learning, pages 103–128. Springer US, 1994.

[5] A. P. Wieland. Evolving neural network controllers for unstable systems. In *Proc. of the 1991 International Joint Conference on Neural Networks (IJCNN)*, volume 2, pages 667–673. IEEE, 1991.

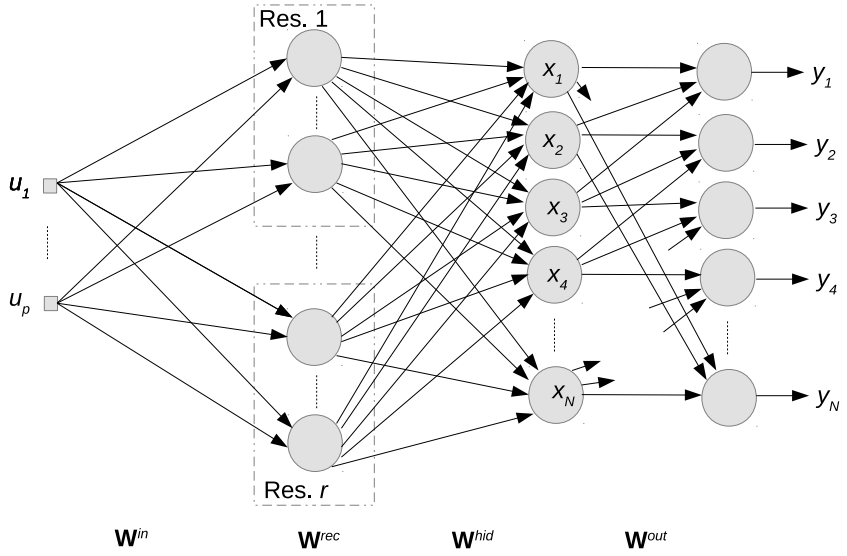

Figure 1: Artificial neural network with 2 hidden layers. The first hidden layer is composed by r reservoirs. The recurrent neurons of each reservoir are sparsely connected among them with density α. Each neuron of the output layer is connected to $K+1$ neurons of hidden layer 2. In the figure, $K=2$ and the adjacent neighbourhood model is adopted, i.e., each neuron i of the output layer is connected to neurons i, $i+1$, and $i+2$ of the hidden layer 2. All neurons of hidden layer 1 are connected to all neurons of input layer and hidden layer 2. The number of neurons in the output layer and hidden layer 2 is N. The neurons of hidden layer 2 are turned on or off according to a binary vector $\mathbf{x}$. If $x_i = 1$, the i-th neuron is turned on, i.e., its activation is used as input for neurons in the output layer connected to it. If $x_i = 0$, the i-th neuron is turned off. All weights are randomly created and kept fixed during the optimization of $\mathbf{x}$.

Table 1: Results of the Generalization Test (Algorithm 3) for the best artificial neural network output and for the committee in the experiments. The Wilcoxon signed rank test was used to compare the results of the Generalization Test. When the distribution of results from the Best Output and the Committee are statistically different, a symbol "s" is shown inside the parenthesis. When the mean result for the Committee is better, a symbol "+" is shown, while a "-" is shown when the mean result for the Best Output is better. The results of $max(f_i(\mathbf{x}, \mathbf{m}_i)$ (Algorithm 1) are also shown.

		Evaluation - Best Output	Generalization Test	
K	Run	$max(f_i(\mathbf{x}, \mathbf{m}_i)$	Best Output	Committee
2	0	0.50519	299	50
	1	0.51061	250	50
	2	0.5158	125	173
	3	0.090938	75	75
	4	0.054191	0	98
	5	0.50499	325	238
	6	0.12045	27	50
	7	0.044068	0	168
	8	0.072252	39	71
	9	0.093576	25	25
	10	0.035448	0	0
	11	0.072031	75	0
	12	0.039699	0	0
	13	0.13666	250	0
	14	0.51938	100	170
	15	0.098996	13	0
	16	0.14409	100	42
	17	0.51823	161	372
	18	0.51743	80	200
	19	0.3216	10	141
	20	0.41016	175	200
	21	0.043517	5	66
	22	0.10321	81	0
	23	0.24785	123	72
	24	0.055012	64	25
	25	0.05664	0	270
	26	0.50698	345	274
	27	0.10806	11	356
	28	0.13423	54	329
	29	0.072836	25	146
	mean	0.2218	94.5667	122.0333 (+)

Table 2: (continuation).

		Evaluation - Best Output	Generalization Test	
K	Run	$max(f_i(\mathbf{x}, \mathbf{m}_i)$	Best Output	Committee
5	0	0.21312	102	247
	1	0.36482	205	74
	2	0.4233	76	452
	3	0.50462	173	250
	4	0.51104	311	351
	5	0.55001	75	386
	6	0.50994	150	150
	7	0.50637	25	200
	8	0.52122	144	459
	9	0.51489	150	200
	10	0.50388	177	35
	11	0.51749	75	175
	12	0.52305	306	131
	13	0.54491	25	94
	14	0.51224	125	200
	15	0.50714	50	25
	16	0.50381	156	50
	17	0.50396	75	25
	18	0.51855	58	28
	19	0.52247	253	204
	20	0.50235	231	300
	21	0.52736	97	417
	22	0.50972	121	450
	23	0.081039	24	237
	24	0.51073	290	312
	25	0.51233	66	329
	26	0.10547	359	0
	27	0.52202	330	262
	28	0.5118	442	459
	29	0.51288	207	175
	mean	0.46908	162.6	222.5667 (+)
6	0	0.51549	90	61
	1	0.51212	103	265
	2	0.50777	42	79
	3	0.69794	92	100
	4	0.50541	285	193
	5	0.5086	138	197
	6	0.55391	253	230
	7	0.50982	25	75
	8	0.52122	144	455
	9	0.52148	175	241
	10	0.18847	15	55
	11	0.50442	200	140
	12	0.53161	25	27
	13	0.50532	54	132
	14	0.50902	96	234
	15	0.50353	25	0
	16	0.50861	210	57
	17	0.52558	50	75
	18	0.51855	58	395
	19	0.51438	45	143
	20	0.51269	176	184
	21	0.52495	70	75
	22	0.59109	26	317
	23	0.51575	205	350
	24	0.42341	195	132
	25	0.54454	78	257
	26	0.51265	75	192
	27	0.51303	122	147
	28	0.54011	317	414
	29	0.56978	100	152
	mean	0.51371	116.3	179.1333 (s+)

Table 3: (continuation).

		Evaluation - Best Output	Generalization Test	
K	Run	$max(f_i(\mathbf{x}, \mathbf{m}_i)$	Best Output	Committee
7	0	0.52339	175	32
	1	0.50999	130	43
	2	0.5367	25	239
	3	0.5467	25	333
	4	0.50904	254	336
	5	0.51762	175	200
	6	0.51402	239	245
	7	0.51112	86	230
	8	0.52264	175	428
	9	0.51489	150	25
	10	0.51721	34	122
	11	0.59499	37	100
	12	0.53383	25	58
	13	0.50612	122	50
	14	0.50491	400	286
	15	0.53738	106	100
	16	0.50998	122	302
	17	0.51466	50	25
	18	0.51855	58	50
	19	0.55919	102	208
	20	0.51933	106	50
	21	0.50776	31	170
	22	0.52832	498	434
	23	0.51045	170	429
	24	0.51906	154	174
	25	0.53875	218	201
	26	0.50392	55	33
	27	0.5176	189	25
	28	0.71921	61	294
	29	0.53667	101	400
	mean	0.53013	135.7667	187.4 (+)
8	0	0.52852	29	366
	1	0.51161	26	125
	2	0.52059	25	4
	3	0.55081	82	563
	4	0.53837	32	425
	5	0.54369	24	75
	6	0.51047	218	97
	7	0.51886	43	28
	8	0.52547	25	263
	9	0.55634	50	422
	10	0.52286	287	33
	11	0.61933	73	465
	12	0.51854	25	139
	13	0.52406	100	58
	14	0.50871	172	150
	15	1.1508	66	82
	16	0.51426	175	379
	17	0.55407	105	27
	18	0.51855	58	61
	19	0.63103	125	52
	20	0.53206	60	244
	21	0.51134	253	417
	22	0.58359	158	224
	23	0.51227	150	101
	24	0.53617	160	364
	25	0.54149	25	318
	26	0.53546	25	49
	27	0.62011	118	362
	28	0.56895	64	485
	29	0.54446	105	277
	mean	0.56176	95.2667	221.8333 (s+)