Neural-Gas for Function Approximation: A Heuristic for Minimizing the Local Estimation Error.

M. Winter, G. Metta and G. Sandini.
LIRA-Lab, DIST, University of Genoa, viale Causa 13, I-16145 Genoa, ITALY.
sandini@lira.dist.unige.it

In the classical neural-gas method, the neurons are located in the input space according to the input density. But if we want to use this kind of approach as a function approximator, it can be interesting to change the spatial distribution of the neurons, so that they concentrate in regions where the unknown function appeared to be more complex. To achieve this goal, we modified the update rule of the neurons so that they move towards regions where the estimated local error is high. In this article, we first show how to estimate this error locally to each neuron, and then we detail the modification of the algorithm. Finally, we present some simulations results that allow us to compare the modified approach with the classical one.

1. Introduction

The basic idea of the neural-gas type approaches is to divide the input space into small regions in accordance with the input density of probability of the function to be estimated: each of these regions are defined by a neuron in their centre, and the density of the neurons approximates the density of probability of the input space of the function.

In other words, the neurons concentrate in regions of the input space where most of the inputs occur, even if only one neuron would be enough to model the function in the whole region. It could be interesting to change the way neurons are moved in the input space in order to place them in regions where the function is hard to model, *i.e.* where the estimated error is large. We investigated in that direction and we found an efficient heuristic modification of the original neural-gas algorithm.

In the next section we present the neural-gas algorithm used in the context of function approximation. We explain our heuristic in section 3 and we show in section 4 some simulations results. Finally, section 5 is dedicated to concluding remarks.

2. The Neural-Gas approach to function approximation

We want the neural network to approximate a function $Y = F(X)$. Each neuron *i* contains a weight vector W_i which corresponds to its position in the input space, and two vectors Y_i^0 and A_i that are parameters used in the model of the function F. The neural-gas learning algorithm [1,4] divides the input space into Voronoi polyhedric regions, each of these subregions being defined as the neighbourhood of a particular neuron. If we denotes by $\tilde{F}$ the approximated function and V_i the region defined around neuron *i*, the approximation $\tilde{F}(X)$ in the region V_i is given by:

$$\tilde{F}(X) = Y_i^0 + A_i(X - W_i) \tag{1}$$

and a gradient descent method [2] is used for adjusting the parameters Y_i^0 and A_i each time the net receive a new input-output pair. Here we chose a linear basis function because of its simplicity but, of course, it is possible to use another one.

We defined the neighbourhood relationship as in [3]; hence, the online algorithm is as follows:

{1} initialise randomly Y_i^0, A_i, and W_i for all neurons. Initialise the age of all the connections: $a_{ij} = -1 \ \ \forall i, j \in D$, where D is the set of the indices of all the neurons.

{2} when an input X is received, find the indices of the two closest neurons, *i.e.*:

$$i^* = Arg \min_{i \in D} \|X - W_i\| \tag{2}$$

$$i^{**} = Arg \min_{i \in D / \{i^*\}} \|X - W_i\| \tag{3}$$

{3} generate an approximated output $\tilde{F}(X)$:

$$\tilde{F}(X) = Y_{i^*}^0 + A_{i^*}(X - W_{i^*}) \tag{4}$$

{4} update of the position of the neurons in the input space:

- the closest neuron is moved towards the input X by a fraction ε_w:

$$\Delta W_{i^*} = \varepsilon_w (X - W_{i^*}) \tag{5}$$

- the cells in the neighbourhood N_{i^*} of the closest cell are also moved towards the input by a fraction ε_n:

$$\Delta W_i = \varepsilon_n (X - W_i) \qquad \forall i \in N_{i^*} \tag{6}$$

where the neighbourhood is defined as:

$$N_i = \{ j / a_{ij} \geq 0, j \neq i \} \tag{7}$$

{5} update the output of the neuron i^*:

$$\Delta Y_{i^*}^0 = \lambda_y (Y - \tilde{F}(X)) \tag{8}$$

$$\Delta A_{i^*} = \lambda_A (Y - \tilde{F}(X) - A_{i^*}(X - W_{i^*}))(X - W_{i^*}) \tag{9}$$

where λ_y and λ_A are parameters that allow to tune the adaptation.

{6} update the ages of the connections:

$$\begin{aligned} & a_{i^* i^{**}} := 0 \\ & a_{i^* j} := a_{i^* j} + 1 && \forall j \in N_{i^*} \\ & a_{i^* j} := -1 \ \text{ if } \ a_{i^* j} > a_{\max} && \forall j \in N_{i^*} \end{aligned} \tag{10}$$

{7} if a stopping criterion is not fulfilled continue with step {2}.

3. The proposed heuristic

As we said previously, our goal is to change the way the neurons are distributed in the input space. More precisely, we want them to be denser in regions where the estimation error is high. The solution we propose in order to implement this principle consists in adding a term to equation (5) that updates the position of the winning neuron: it is moved towards its neighbour that has the highest error, if this error is higher than its own.

As a first step, we must estimate the local error. Hence, for each neuron *i*, we define a new scalar parameter r_i that represents the estimation of the local approximation error. At the beginning of the algorithm, this parameter is initialised to zero; and each time a new input-output pair is received, the parameter r_i of the closest neuron is updated using the following delta-rule:

$$\Delta r_{i^*} = \varepsilon_w \left(\left\| Y - \tilde{F}(X) \right\| - r_{i^*} \right) \tag{11}$$

The closest neuron is then moved towards its neighbour that has the higher error r_i, if this error is larger than its own:

$$\Delta W_{i^*} = \varepsilon_w (X - W_{i^*}) + \varepsilon_r \left(W_{j^*} - W_{i^*} \right) \Delta \bar{E} \tag{12}$$

where ε_r is a tuning parameter, and with:

$$j^* = \underset{j \in N_{i^*}}{Arg \max} \left(r_j - r_{i^*} \right) \tag{13}$$

$$\Delta \bar{E} = \begin{cases} \dfrac{r_{j^*} - r_{i^*}}{r_{j^*}} & if \quad r_{j^*} - r_{i^*} > 0 \\ 0 & otherwise \end{cases} \tag{14}$$

In order to illustrate the behaviour of the modified neural net, we present in the next section some simple simulations we made using the classical and the modified neural-gas network.

4. Simulation results

We want the networks to approximate the function plotted in figure 1. We used $\varepsilon_w = 0.05$, $\varepsilon_n = 0.0005$, $\lambda_y = 0.05$, $\lambda_A = 1$, $a_{\max} = 90$ and $\varepsilon_r = 0.05$. The evolution of the mean error between the expected and the estimated value is given in figure 2 for the two approaches.

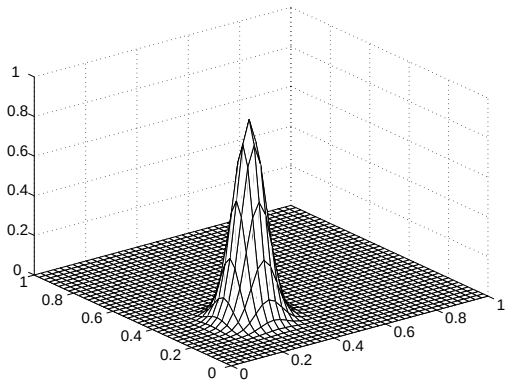

Figure 1: the function to be approximated.

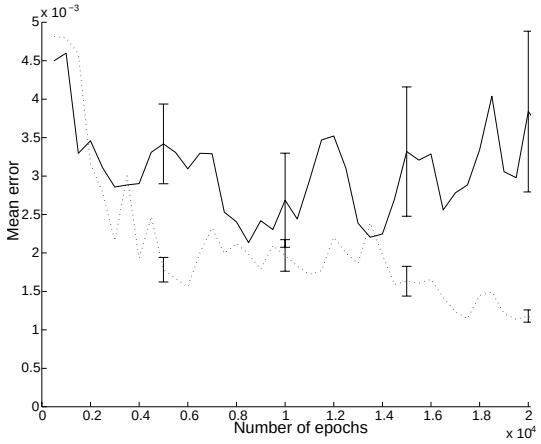

Figure 2: estimation error for the classical approach (solid line) and the modified approach (dashed line).

Figure 3 illustrates the position of the neurons in the input space, after 20000 cycles. Finally, we show in figure 4 the approximated functions.

As we can see, the proposed modification makes the neural network react as expected: the neurons move towards areas where the error is large. We can verify that this leads to a better estimation of the unknown function: after 20000 cycles, the mean estimation error equals 1.2×10^{-3} with our method, to be compared to 3.7×10^{-3} with the classical approach.

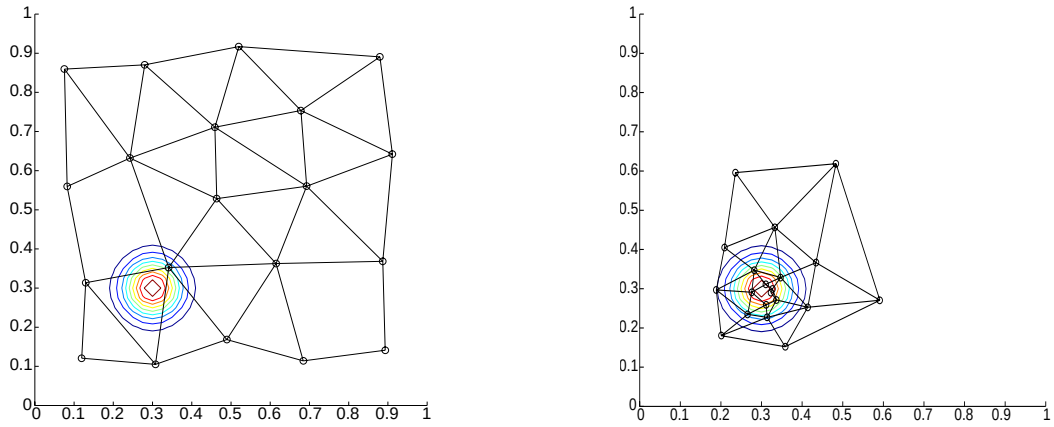

Figure 3: position of the neurons in the input space for the classical approach (left) and the modified approach (right).

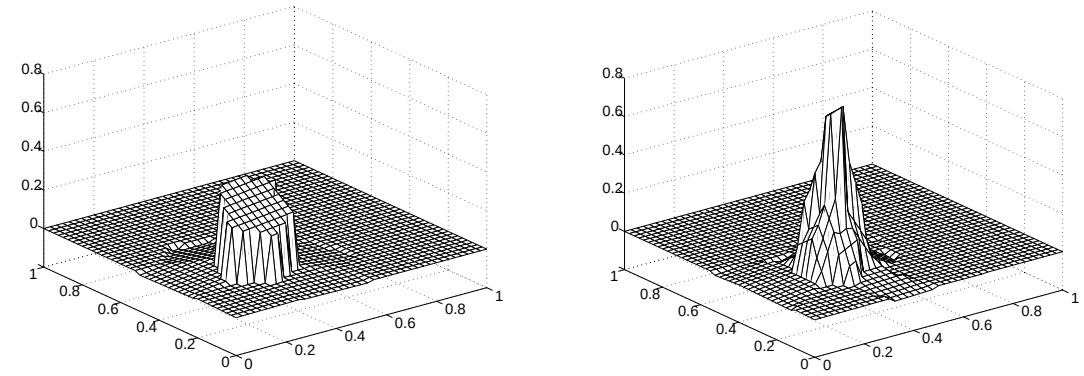

Figure 4: approximated function with the classical approach (left) and the modified approach (right).

5. Conclusion

We proposed a modification of the neural-gas approach for function approximation. The modification is a heuristic that moves the neurons towards regions in the input space where the error is large. As a result, the neurons are located in regions where the function is more complicated.

We illustrated the behaviour of the modified neural approach with simple simulations, and a comparison with the classical neural-gas approach showed that our heuristic leads to a better estimation of the unknown function.

[1] T. M. Martinetz and K. J. Schulten. A "neural-gas" network learns topologies. In T. Kohonen, K. Makisara, O. Simula, and J. Kangas, editors, *Artificial Neural Networks*, pages 397-402. North-Holland, Amsterdam, 1991.

[2] T. M. Martinetz, S. G. Berkovich, and K. J. Schulten. "Neural-Gas" Network for Vector Quantization and its Application to Time-Series Prediction, *IEEE Transactions on Neural Networks*, vol. 4, No. 4, pages 558-569, July 1993.

[3] B. Fritzke. Some competitive learning methods, *Technical Report from the Systems Biophysics Institute for Neural Computation*, Ruhr-Universitat Bochum, April 1997.

[4] J. A. Walter and K. J. Schulten. Implementation of self organizing neural networks for visuo-motor control of an industrial robot, *IEEE Transactions on Neural Networks*, vol. 4, No. 1, pages 86-95, 1993.