An Incremental Growing Neural Network and its Application to Robot Control

A. Carlevarino, R. Martinotti, G. Metta and G. Sandini
Lira Lab – DIST – University of Genova
Via Opera Pia, 13 – 16145 Genova, Italy
E-mail: pasa@lira.dist.unige.it

Abstract

This paper describes a novel network model, which is able to control its growth on the basis of the approximation requests. Two classes of self-tuning neural models are considered; namely Growing Neural Gas (GNG) and SoftMax function networks. We combined the two models into a new one: hence the name GNG-Soft networks. The resulting model is characterized by the effectiveness of the GNG in distributing the units within the input space and the approximation properties of SoftMax functions. We devised a method to estimate the approximation error in an incremental fashion. This measure has been used to tune the network growth rate. Results showing the performance of the network in a real-world robotic experiment are reported.

Introduction

This paper describes an incremental algorithm that includes aspects of both the Growing Neural Gas (GNG) (Fritzke, 1994), and the SoftMax (Morasso & Sanguineti, 1995) networks. The proposed model retains the effectiveness of the GNG in distributing the units within the input space, and the “good” approximation properties of the SoftMax basis functions. Furthermore, a mechanism to control the growth rate, based on the estimate of the approximation error, has been included. In the following sections, we briefly describe the peculiarities of the proposed algorithm, and present some experimental results in the context of sensori-motor mapping, where the network has been used to approximate an inverse model. The inverse model is employed in a robot control schema.

Growing Neural Gas and SoftMax Function Networks

Growing Neural Gas is an unsupervised network model, which learns topologies (Fritzke, 1995). A set of units connected by edges is distributed in the input space (a subset of $\Re^N$) with an incremental mechanism, which tends to minimize the mean distortion error. GNG networks can be effectively combined with different types of basis functions in order to approximate a given input-output distribution. As cited above, Morasso and coworkers, for instance, proposed a normalized Gaussian function (SoftMax) for their self-organizing network model. The SoftMax functions have the following analytic expression:

$$U_i(x,c_i) = \frac{G(\|x-c_i\|)}{\sum_j G(\|x-c_j\|)} \qquad G(x) = e^{-\frac{x^2}{2\sigma^2}} \tag{1}$$

c_i is the center of the activation function, where U_i has its maximum. Benaim and Tomasini (Benaim & Tomasini, 1991) proposed a Hebbian learning rule for the optimal placement of units:

$$\Delta c_i = \eta_1 (x - c_i) U_i(x,c_i) \tag{2}$$

where $x \in \Re^N$ is an input pattern and η_1 the learning rate. Indeed, a SoftMax network can learn a smooth non-linear mapping $z=z(x)$. The "reconstruction" formula is:

$$z(x) \cong \sum_i v_i U(x,c_i) \tag{3}$$

where the parameters v_i are the weights of the output layer and $z \in \Re^M$. In particular, considering an approximation case, this formula can be interpreted as a minimum variance estimator (Specht, 1990). The learning rule is:

$$\Delta v_i = \eta_2 (z - v_i) U_i(x,c_i) \tag{4}$$

The normalizing factor in U_i can be assimilated to a lateral inhibition mechanism (Morasso & Sanguineti, 1995).

GngSoft Algorithm

It is now evident how to combine the two previously illustrated self-organizing models to obtain an incremental and plastic network. Moreover, as mentioned before, we inserted a control mechanism based on an online estimate of the approximation error. The global error is estimated from the "sample error" as follows:

$$e_t = \psi_1 e_{t-1} + (1-\psi_1)\|\zeta - g(\xi)\| \tag{5}$$

and,

$$\dot{e}_t = \psi_2 \dot{e}_{t-1} + (1-\psi_2)\cdot(e_t - e_{t-1}) \tag{6}$$

The behavior of e_t resembles that of the instantaneous error $\|\zeta - g(\xi)\|$, although with a sort of memory. In fact, it is a low pass filtered version of the raw error signal. $\psi_{i=1,2}$ are positive constants, which determine the filters cut-off frequency. $\dot{e}_t$ is an estimate of the derivative of the error. The online measures have been used to block the insertion of new units according to the following conditions:

$$e_t < threshold_1 \tag{7}$$

And,

$$\|\dot{e}_t\| > threshold_2 \tag{8}$$

The derivative of the error was checked, and insertion was allowed if, and only if, it was approximately zero – i.e. the error itself does not decrease anymore. Figure 1 shows a typical result using the proposed estimation formulas.

Experimental results

The proposed algorithm (GngSoft netwrok) has been used in two different test-beds. The first experiment concerned a typical vector quantization task, whose goal was to build an appropriate code in a 16- and 64-dimensional space. The second experiment regarded the acquisition of an inverse model in the context of sensori-motor coordination.

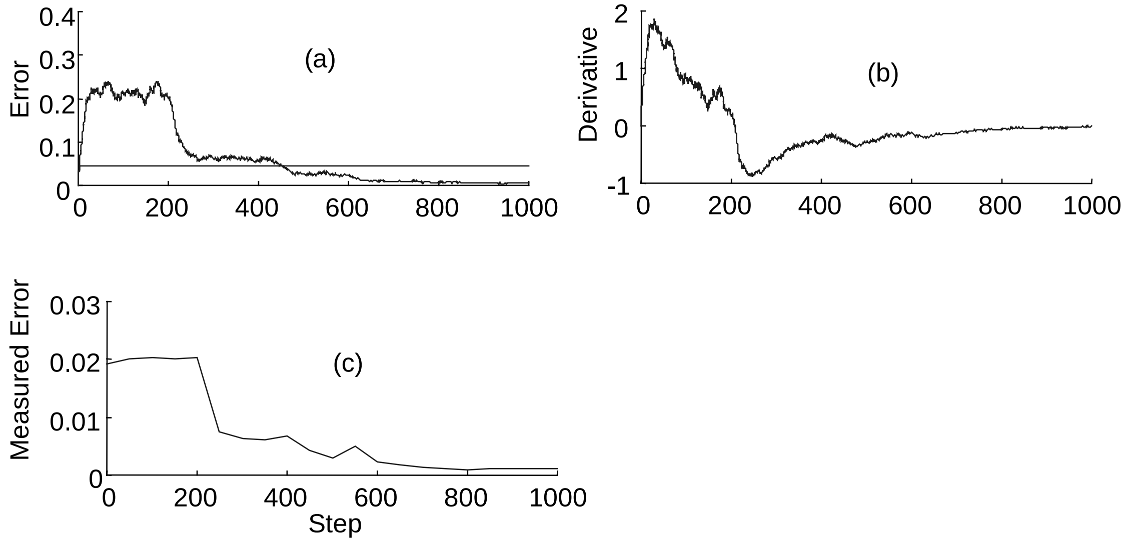


Figure 1: Some experimental results concerning the proposed error estimation formula. The network was trained for 10000 steps. **Upper row**: (a) the estimated error during the training (the solid horizontal line indicates the threshold), (b) the estimated derivative (according to equation (6)). **Lower row**: (c) the error estimated independently over a test set four times denser than the training set. ψ_1 and ψ_2 were 0.999 and 0.99 respectively.

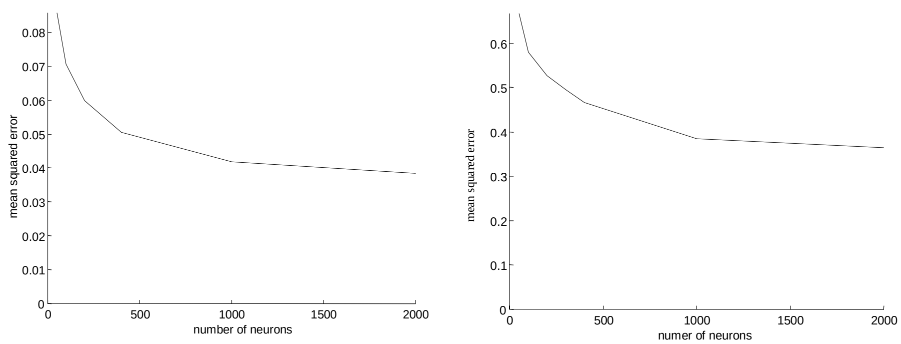


Figure 2: Mean square error vs. the number of neurons for the 16- (left) and 64-dimensional (right) case respectively.

A classical vector quantization experiment has been performed in order to assess the performance of the network in high dimensional spaces – i.e. the ability to map a big input space. The training data were built from images, which were divided in 4×4 and 8×8 blocks and eventually generated a 16- and 64-dimensional training space respectively. The training set was created from one image only (512×512 pixels), while the network testing was carried out on three different images (see (Ridella, Rovetta, & Zunino, 1998)). Figures 2 shows the mean square error versus the number of units for the 16- (left) and 64-dimensional (right) case. In figure 3, it is important to note that the CPU time grows only linearly with respect to the number of neurons.

The setup employed for the second experiment consists of the robot head shown in figure 4. The head kinematics allows independent vergence (both cameras can move independently along a vertical axis) and a common tilt motion. Furthermore the neck is capable of a tilt and pan

independent motion. The robotic setup is equipped with two space-variant color cameras (Sandini & Tagliasco, 1980), (Sandini, Dario, DeMicheli, & Tistarelli, 1993). The visual processing allows the robot to recover object position through a color segmentation algorithm.
The goal of the learning algorithm, in this case, is that of acquiring an inverse model, which maps image based target positions into the proper motor commands. The inverse map can be used within the control cycle to allow fast foveation of a spotted object location.

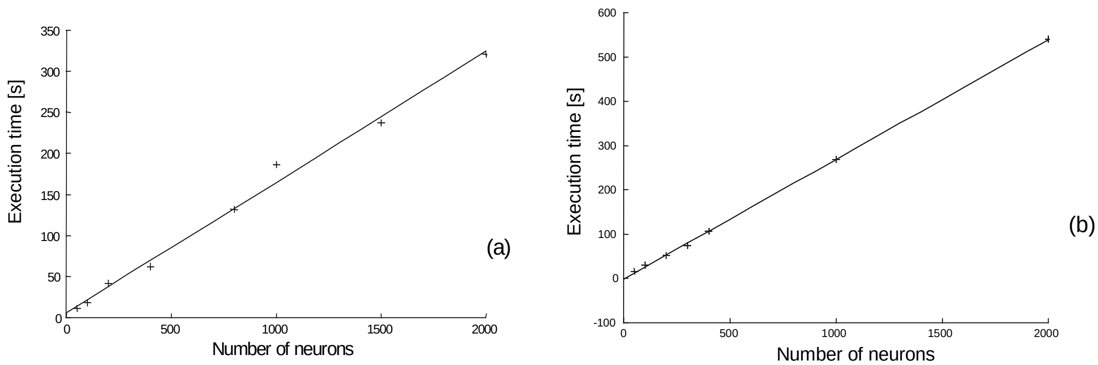


Figure 3: Execution time vs. the number of neurons. **Left**: 16-dimensional space. **Right**: 64-dimensional input space. Note that, the execution time grows only linearly as the number of units grow; this might be a crucial factor for some applications. We plotted the time required to execute 10000 training steps versus the number of neurons in the network.

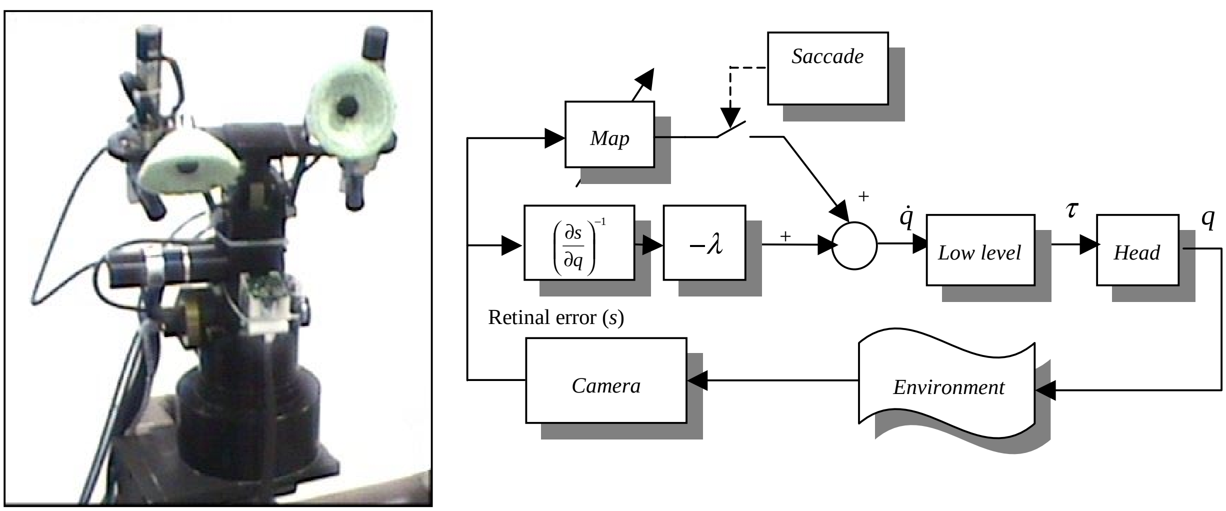
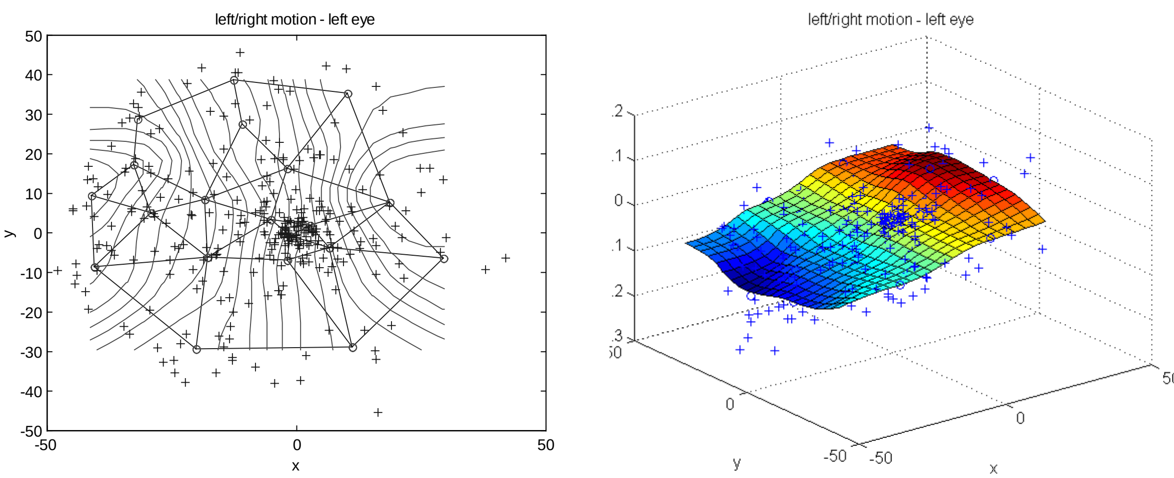


Figure 4. **Left**: the experimental set-up. **Right**: the robot control schema. It consists of a closed loop and a feed-forward secondary loop. The block identified by "Map" represents the GNG-Soft network.

The loop using the inverse image Jacobian is derived from classical visual servoing literature (Espiau, Chaumette, & Rives, 1992), and it was tuned beforehand in order to get stability. The secondary loop consists of an inverse model, which maps the retinal error into the joint rotations required to fixate a spotted target.
Under the hypothesis of a stationary target and a closed loop control in place as described above, the gathering of training pairs (each of them has the form $(\Delta\mathbf{q},\mathbf{s}) \in \Re^2 \times \Re^2$) is much simplified. The retinal error **s** is acquired at the beginning of the motion, while the required motor command $\Delta\mathbf{q}$ can be measured when the retinal error is zeroed. The fact that the error decreases to zero is

guaranteed by the stable closed loop. This schema is roughly equivalent to Kawato's feedback error learning (Kawato, Furukawa, & Suzuki, 1987), although we do not learn a velocity but a position command.
The map itself embeds all the parameters concerning the transformation from visual to motor space (i.e. camera intrinsic parameters, head kinematics, etc). Motor commands, as shown in figure 4 (right), are fed into the low-level PID controller. It accepts, in the present configuration, only velocity commands. Thus, the output of the network is eventually converted into a velocity pulse command lasting a control cycle (40*ms*).
Finally, figure 5 shows an evaluation of the robot performance. We measured the retinal error **s** at the end of each "saccade" (i.e. in analogy with humans, we called the fast eye movements, saccades). The rationale was that if the robot were improving its performance the retinal error would (on average) decrease over time. We recorded from the robot for 300 trials in noisy conditions; that is, cluttered background, moving targets, etc, and applied a 100-samples wide moving window to the raw data. We estimated the mean and standard deviation inside such a window. They are reported figure 6, left and right plot respectively. It can be observed a clear trend toward the reduction of the average retinal error, as well as, its standard deviation. Note also that the 300 samples were collected together with the network training set. From time to time the robot was stopped and the network performed some more "learning only steps". As mentioned above the network executed 90000 learning steps overall.



Figures 5. **Left**: the input space (in image space coordinates). The '+' signs represents training examples, and the circles are the position of unit's centers; contour lines are also shown. **Right**: one of the components of the output vector for the same training set. The plot has been obtained after about 90000 steps performed using a 300 samples wide training set.

Conclusion

This paper presented a new growing network model, where an estimate of the approximation error is used to tune the network growth dynamically.
We tested the network in two different contexts, namely: a vector quantization experiment, and learning of an inverse model within the control loop of a real robot. The latter experiment showed that the model could tolerate a certain amount of noise in the training data. However, it is fair to say that in noisy conditions the mentioned stopping criteria must be chosen carefully. If the thresholds were too low, the network might not achieve the required precision because of the noise. Nonetheless, it would continue to grow. A better strategy, beside the developmental mechanism, might apply a pruning schema; the two of them together could take care of

controlling the model complexity using some sort of performance criterion; the latter could act as

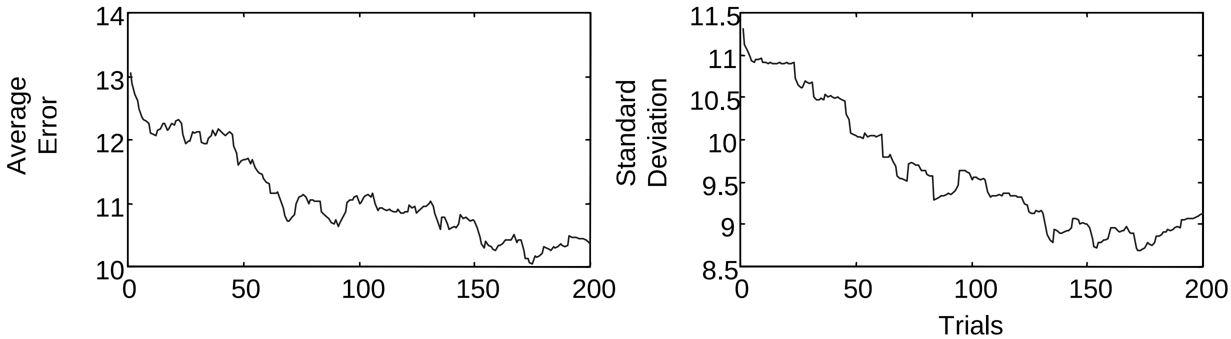


Figure 6: Robot motor performance. **Left**: moving window average of the residual retinal error (i.e. at the end of a fast motion). **Right**: the standard deviation of the same 300 samples. Abscissas represent the number of trials.

a reinforcement signal optimizing the model size. On-going investigation is directed at the theoretical analysis of the network capabilities, and toward the improvement of the insertion/removal mechanisms as outlined above.

Bibliography

Benaim, M., & Tomasini, L. (1991). Competitive and Self-organinzing Algorithms Based on the Minimization of an Information Criterion. In T. Kohonen, K. Makisara, O. Simula, & J. Kangas (Eds.), Artificial Neural Networks (pp. 391-396). Amsterdam, North-Holland: Elsevier Science Publisher.

Espiau, B., Chaumette, F., & Rives, P. (1992). A New Approach to Visual Servoing in Robotics. IEEE Transactions on Robotics and Automation,, 8(3), 313--326.

Fritzke, B. (1994). Fast learning with incremental RBF Networks. Neural Processing Letters, 1(1), 2-5.

Fritzke, B. (1995). A growing neural gas network learns topologies. In G. Tesauro, D. S. Touretzky, & T. K. Leen (Eds.), Advances in Neural Information Processing Systems (Vol. 7,). Cambridge MA, USA: MIT Press.

Kawato, M., Furukawa, K., & Suzuki, R. (1987). A hierarchical neural network model for control and learning of voluntary movement. Biological Cybernetics(57), 169-185.

Morasso, P., & Sanguineti, V. (1995). Self-Organizing Body Schema for Motor Planning. Journal of Motor Behavior, 27(1), 52-66.

Ridella, S., Rovetta, S., & Zunino, R. (1998). Plastic Algorithm for Adaptive Vector Quantisation. Neural Computing & Applications(7), 37-51.

Sandini, G., Dario, P., DeMicheli, M., & Tistarelli, M. (1993,). Retina-like CCD Sensor for Active Vision. Computer and Systems Sciences, Series F, 102.

Sandini, G., & Tagliasco, V. (1980). An Anthropomorphic Retina-like Structure for Scene Analysis. CVGIP, 14(3), 365-372.

Specht, D. F. (1990). Probabilistic Neural Networks. Neural Networks(3), 109-118.