In : Proceedings of ICANN-97, International Conference on Artificial Neural Networks, Springer, 1997, p. 613-618

A self-organizing network that can follow non-stationary distributions

Bernd Fritzke

Systembiophysik, Institut für Neuroinformatik
Ruhr-Universität Bochum
44780 Bochum, Germany

Abstract. A new on-line criterion for identifying "useless" neurons of a self-organizing network is proposed. When this criterion is used in the context of the (formerly developed) growing neural gas model to guide deletions of units, the resulting method is able to closely track non-stationary distributions. Slow changes of the distribution are handled by adaptation of existing units. Rapid changes are handled by removal of "useless" neurons and subsequent insertions of new units in other places.

1 Non-stationary data is difficult to handle ...

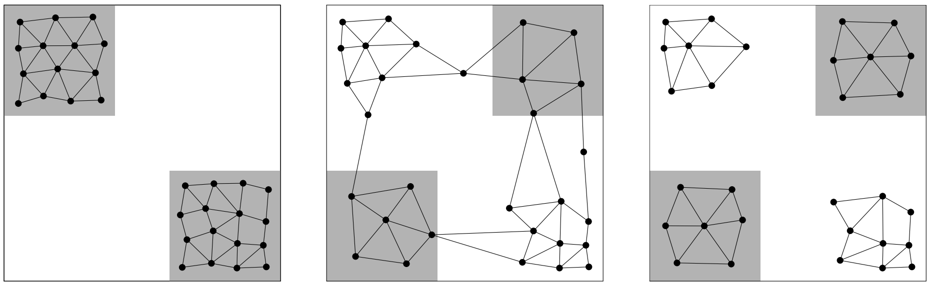
Non-stationary data distributions can be found in many technical, biological or economical processes. Self-organizing neural networks have rarely been considered for tracking those distributions since many of the models, e.g. the self-organizing map [6], neural gas [8], or the hypercubical map [1], use decaying adaptation parameters[1]. Once the adaptation strength has decayed, the network is "frozen" and thus unable to react to subsequent changes in the signal distribution.

2 ... even for incremental networks

Models with small constant parameters such as the incremental networks developed by the author [2–4] are in a somewhat better position for handling non-stationary distributions. The non-decreasing adaptation rate enables the networks to follow slowly changing probability distributions like e.g. a normal distribution with a slowly drifting mean. Rapid changes in the distribution, however, can in general not be handled properly as is illustrated for the growing neural gas (GNG) [3] model in figure 1.

In the case of GNG it is easy to see why considerably many units may get stuck in former regions of high probability density and become so-called *dead units*. The network topology is updated – apart from the characteristic insertions – by two mechanisms (as described in detail in [3]). "Competitive Hebbian Learning" [7] is used to create new connections by always inserting a connection between the units s_1 and s_2 nearest and second-nearest to the current input

[1] ... as does the classical k-means algorithm.



a) 20000 signals   b) 25000 signals   c) 40000 signals

Fig. 1. A growing neural gas (GNG) network tries (and fails) to track a non-stationary signal distribution $p(\boldsymbol{\xi})$. The maximum network size is set to 30. a) Initial state of $p(\boldsymbol{\xi})$ with the GNG network already grown to its maximum size. At this point the distribution was changed. b) State of the network 5000 adaptation steps after the distribution had changed. c) After 50000 adaptation steps some units have been adapted to the new regions of high probability density, but some have become dead units which are not adapted anymore. Parameters used (see [3] for details):$\lambda = 500$, $\epsilon_b = 0.05$, $\epsilon_n = 0.0006$, $\beta = 0.0005$, $a_{max} = 120$.

signal $\boldsymbol{\xi}$. When such a connection does exist already, its *age* parameter is set to zero. Moreover, the age of all edges adjacent to the winning unit s_1 is increased. If the age of an edge surpasses a maximum parameter a_{max}, the edge is removed and so are any units without any emanating edges.

One should note, that the described aging is a local process, since it happens only in the vicinity of the winning unit. Now consider the situation where a network has grown to a certain maximum size and has well adapted to a given probability distribution (figure 1a). When the probability density changes, some units are adapted rather quickly towards the new regions of high probability density (figure 1b). The edges connecting those units and the units remaining in the former regions of high probability density are rather quickly removed from the aging process (figure 1c).

Thereafter, however, those units and connections which remain in the former regions of high probability density are not changed at all anymore. The units are not winner since they are too far from the current input signals. Consequently, the connections do not age since aging is only performed for edges adjacent to the winner. The result is a waste of network resources which may get even worse if the distribution changes several times. One could perhaps interpret the dead units as a kind of memory which may be useful again, when the probability distribution takes on a previously held shape[2], but here we like to discuss ways of using a network of limited size to track the distribution in each moment as good as possible.

[2] a suggestion made by Christoph von der Malsburg

3 Objective function

As objective for the network we define the minimization of the expected quantization error

$$E = \int \|\boldsymbol{\xi} - \mathbf{w}_{s(\boldsymbol{\xi})}\|^2 p(\boldsymbol{\xi}) d\boldsymbol{\xi}. \tag{1}$$

Thereby $s(\boldsymbol{\xi})$, also denoted as s_1, is the winning unit for the current input signal $\boldsymbol{\xi}$ and $\mathbf{w}_{s(\boldsymbol{\xi})}$ is the reference vector associate with that unit. With $p(\boldsymbol{\xi})$ we denote the probability distribution of the input signals which is assumed to be unknown and non-stationary.

4 Local error measure

In a GNG network each unit c in the network has an error variable E_c which contains accumulated error information. For each input signal $\boldsymbol{\xi}$ the error variable of the winner s_1 for this signal is updated according to

$$E_{s_1} := E_{s_1} + \|\boldsymbol{\xi} - \mathbf{w}_{s_1}\|^2. \tag{2}$$

Moreover, after each adaptation step (small change of the reference vectors of s_1 and its topological neighbors, see [3]) the error variables of all units are subjected to exponential decay:

$$E_c := E_c - \beta E_c \quad \text{(for each unit } c\text{)} \tag{3}$$

whereby β is a suitable decay constant. The decay stresses the influence of more recently measured error and is needed to account for the adaptation of the reference vectors and for the non-stationarity of the distribution. The GNG algorithm performs insertions of new units always close to the unit q with maximum accumulated error.

5 Local utility measure

Removal of units would enable us to get rid of dead units. The GNG algorithm would in this case insert a new unit at a better location in input space to again reach its allowed maximum size. But how can we decide, when a unit is to be removed? On first thought one might have the idea to remove the units with low accumulated error values. However, this may not be a good choice at all. Consider a unit positioned at a point-like peak of the probability density function. It would have very small accumulated error since most input signals for which it is winner coincide with its reference vector and, therefore, generate no error. If we remove this unit, however, all these signals would be mapped to the unit previously being second-nearest to them. This unit, however, could be very distant.

A better approach is, to directly compute, how much the error for the given input signal $\boldsymbol{\xi}$ would increase if the winner s_1 would not be present. In this case the signal would be mapped to the second-nearest unit s_2 and, therefore, the increase in error is simply the difference of $\|\boldsymbol{\xi} - \mathbf{w}_{s_2}\|^2$ and $\|\boldsymbol{\xi} - \mathbf{w}_{s_1}\|^2$. This gives us – for the current input signal – a direct measure of the utility of the winner unit s_1. To sum up this utility measure over all input signals we introduce a local variable U_c for each unit c. For each input signal the utility variable of the winner s_1 is updated according to

$$U_{s_1} := U_{s_1} + \|\boldsymbol{\xi} - \mathbf{w}_{s_2}\|^2 - \|\boldsymbol{\xi} - \mathbf{w}_{s_1}\|^2. \tag{4}$$

As done for the local error values we subject all utility variables to exponential decay after each adaptation step:

$$U_c := U_c - \beta U_c \quad (\text{ for each unit } c). \tag{5}$$

This utility measure is an on-line variant of a measure recently proposed by the author for batch vector quantization methods [5]. The utility of a unit becomes small either when other units are in close vicinity (then $\mathbf{w}_{s_2}$ approaches $\mathbf{w}_{s_1}$) or when the unit is not anymore (or only rarely) winner.

6 Removal criterion

How can the utility measure be used to remove neurons? We can not simply remove the unit with minimum value of U since there always will be one and this strategy would cause a constant change of the network structure even for perfectly stationary distributions. Let us recall, that the set goal is error reduction. When we remove a unit the GNG algorithm immediately inserts a new unit near the unit q with maximum accumulated error, i.e., the unit such that

$$q := \arg\max_c E_c. \tag{6}$$

We can expect the newly inserted unit to reduce this error in the future, since it reduces the size of the Voronoi region of q. When the mean distance of the signals in q's Voronoi region is reduced by a certain factor, the (square) quantization error is reduced by an even larger factor. A cautious estimate is that the mean distance is reduced only by a factor two. More precise estimates of the error reduction would require unavailable knowledge of the local structure of $p(\boldsymbol{\xi})$. However, these general considerations are enough to formulate a removal criterion. An estimate of the increase in quantization error by removing a unit i is its current utility value U_i. The expected decrease caused by the insertion of a new unit near the unit q with maximum error E_q is some fraction of E_q.

A suitable strategy seems to remove a unit when its utility value falls below a certain fraction of the error variable E_q. Let i be the unit with minimum utility

$$i := \arg\min_c U_c. \tag{7}$$

Then, i should be removed if

$$E_q/U_i > k \tag{8}$$

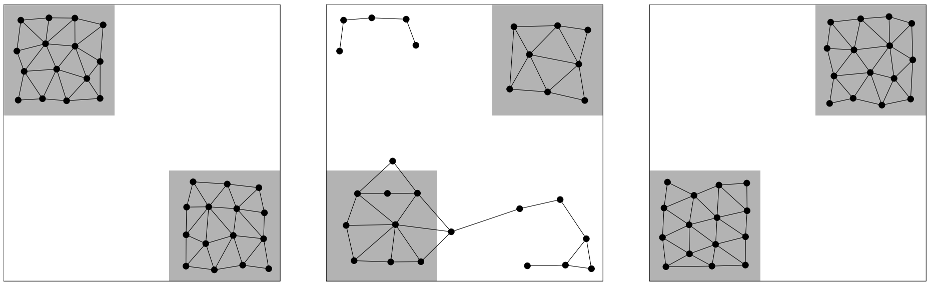

a) 20000 signals b) 25000 signals c) 40000 signals

Fig. 2. A GNG-U network successfully tracks a non-stationary signal distribution $p(\boldsymbol{\xi})$. a) situation before the distribution changes. b) State of the network 5000 adaptation steps after the distribution had changed. c) After 50000 adaptation steps all units have been adapted or relocated to regions of high probability density. The value used for k is 3. The other parameters are as in figure 1.

for some value of k. Large values of k require a large ratio of maximum error and minimum utility and will cause less frequent deletions of units. Smaller values will cause accordingly more and faster deletions which leads to a faster tracking of non-stationary distributions.

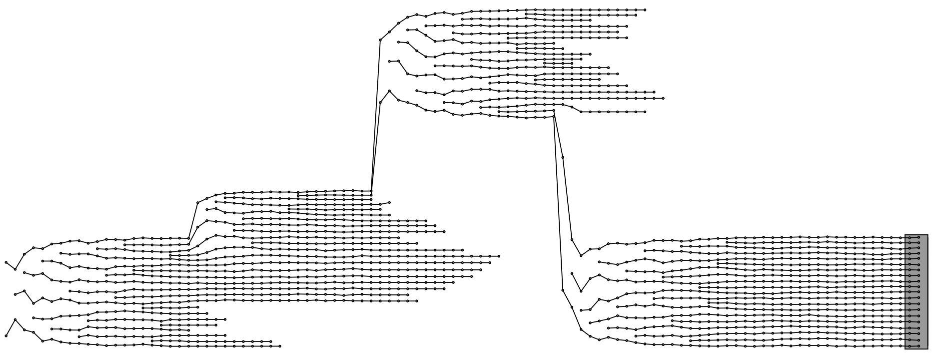
7 Simulation examples

The proposed utility criterion for removal of units was combined with the GNG algorithm and the resulting method, which we call GNG-U (growing neural gas with utility criterion) in the following, was successfully applied to various non-stationary distributions. Due to lack of space we report here only on two experiments with GNG-U. In figure 2 a simulation with the same non-stationary distribution as already used in figure 1 is shown. The GNG-U algorithm successfully relocates all units after the sudden change in the distribution. In figure 3 a non-stationary one-dimensional distribution is used to illustrate the behavior of GNG-U throughout a complete simulation.

8 Discussion

In this paper a novel on-line criterion for removal of units in self-organizing networks is proposed. The combination of this criterion and an existing incremental network model (growing neural gas) results in a new method, called GNG-U, which is able to closely track non-stationary distributions. This is a feature not found in known related models such as various variants of the self-organizing map or the neural gas algorithm and it could open up completely new application domains for this kind of networks.

For the new criterion a parameter k has to be chosen which indicates how large the ratio between accumulated error of a unit q and accumulated utility of



↑ $t = 0$ ↑ $t = 50000$

Fig. 3. A GNG-U network with maximum size set to 20 tracks a one-dimensional non-stationary probability distribution which is indicated by the shaded box on the right. After 10000, 20000, and 30000 signals the mean of the distribution was suddenly changed. It can be seen that moderate changes of the mean (like the change after 10000 signals) lead to a re-use of many units. More drastic changes, however, lead to deletions and subsequent insertions of units. Parameters are the same as in figure 2.

a unit i must be, before unit i is removed. In our experience the choice of this parameter is not at all difficult and the value used in our simulations is just one value which gave very satisfactory results.

References

1. H.-U. Bauer and Th. Villmann. Growing a hypercubical output space in a self-organizing feature map. TR-95-030, International Computer Science Institute, Berkeley, 1995.
2. B. Fritzke. Growing cell structures – a self-organizing network for unsupervised and supervised learning. *Neural Networks*, 7(9):1441–1460, 1994.
3. B. Fritzke. A growing neural gas network learns topologies. In G. Tesauro, D. S. Touretzky, and T. K. Leen, editors, *Advances in Neural Information Processing Systems 7*, pages 625–632. MIT Press, Cambridge MA, 1995.
4. B. Fritzke. Growing grid – a self-organizing network with constant neighborhood range and adaptation strength. *Neural Processing Letters*, 2(5):9–13, 1995.
5. B. Fritzke. The LBG-U method for vector quantization – an improvement over LBG inspired from neural networks. *Neural Processing Letters*, 5(1):35–45, 1997.
6. T. Kohonen. Analysis of a simple self-organizing process. *Biological Cybernetics*, 44:135–140, 1982.
7. T. M. Martinetz. Competitive Hebbian learning rule forms perfectly topology preserving maps. In *ICANN'93: International Conference on Artificial Neural Networks*, pages 427–434, Amsterdam, 1993. Springer.
8. 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*, 4(4):558–569, 1993.