Robustness Analysis of the Neural Gas Learning Algorithm[⋆]

Carolina Saavedra[1], Sebastián Moreno[1], Rodrigo Salas[1,2], and Héctor Allende[1,3]

[1] Universidad Técnica Federico Santa María;
Dept. de Informática; Casilla 110-V; Valparaíso-Chile
{smoreno, saavedra, hallende}@inf.utfsm.cl
[2] Universidad de Valparaíso; Departamento de Computación
rodrigo.salas@uv.cl
[3] Universidad Adolfo Ibañez; Facultad de Ciencia y Tecnología

Abstract. The Neural Gas (NG) is a Vector Quantization technique where a set of prototypes self organize to represent the topology structure of the data. The learning algorithm of the Neural Gas consists in the estimation of the prototypes location in the feature space based in the stochastic gradient descent of an Energy function. In this paper we show that when deviations from idealized distribution function assumptions occur, the behavior of the Neural Gas model can be drastically affected and will not preserve the topology of the feature space as desired. In particular, we show that the learning algorithm of the NG is sensitive to the presence of outliers due to their influence over the adaptation step.

We incorporate a robust strategy to the learning algorithm based on M-estimators where the influence of outlying observations are bounded. Finally we make a comparative study of several estimators where we show the superior performance of our proposed method over the original NG, in static data clustering tasks on both synthetic and real data sets.

Keywords: Neural Gas, Robust Learning Algorithm, M-estimators.

1 Introduction

The Neural Gas (NG), introduced by Martinetz et. al. [6], is a vector quantization technique that has been successfully applied in several areas as pattern recognition and data mining (see [9]). The NG is a variant of the Kohonen Self-Organizing Map [5] where the neighborhood relation is adaptively defined by the ranking order of the distance between the prototypes and the sample data. The NG has the advantage of being flexible and capable of both quantizing topologically heterogeneously structured manifolds and learning the similarity relationships among the input signals without the necessity of specifying a network topology.

The learning algorithm for the parameter estimation of neural networks models rely on the data. In real engineering and scientific applications, data are noisy with the presence of outlying observations. Assumptions of the underlying data generation process

[⋆] This work was supported in part by Research Grant Fondecyt 1061201, 1040365 and 7060040; and in part by DGIP-UTFSM.

no longer holds and the model estimates are badly affected obtaining a poor performance (see for example [1] and [8]).

In [2] and [7] the authors empirically show that the Neural Gas lacks of robustness and they incorporated several robust strategies such as outlier resistant scheme. In this paper we show that when deviations from idealized distribution function assumptions occurs, the behavior of the Neural Gas model can be drastically affected and will not preserve the topology of the feature space as desired. In particular, we show that the learning algorithm of the NG is sensitive to the presence of outliers due to their influence in the adaptation step. We incorporate a robust strategy to the learning algorithm based on M-estimators where the influence of outlying observations is bounded.

The remainder of this paper is organized as follows. In the next section we briefly introduce the Neural Gas model. In section 3 we review some concepts of Robust M-estimator applied to the learning process. In section 4 we investigate the robustness properties of the NG by casting the learning algorithm as a statistical estimation problem, furthermore, we introduce the M-estimators as a robust scheme for the parameter estimation process. In section 5 we provide a comparative study of several estimators where we show the superior performance of the robust methods over the original NG, in static data clustering tasks on both synthetic and real data sets. Conclusions and further work are given in section 6.

2 Neural Gas

The "Neural-Gas" (NG) model consists of an ordered set $\mathbf{m} = \{\mathbf{m}_1, ..., \mathbf{m}_M\}$ of M prototypes, neurons or "codebooks" vectors $\mathbf{m}_j \in \mathcal{M} \subseteq \mathbb{R}^d$, $j = 1, .., M$ arranged according to a neighborhood ranking relation between the units.

When the data vector $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^d$ is presented to the NG model, it is projected to a neuron position by searching the best matching unit (bmu), i.e., the prototype that is closest to the input, and it is obtained as $c(\mathbf{x}) = \arg\min_{j=1..M} \{\|\mathbf{x} - \mathbf{m}_j\|\}$, where $\|\cdot\|$ is the classical Euclidean norm. This procedure divides the manifold $\mathcal{X}$ into a number of subregions $V_j = \{\mathbf{x} \in \mathcal{X} | \ \|\mathbf{x} - \mathbf{m}_j\| \leq \|\mathbf{x} - \mathbf{m}_i\| \ \forall i\}$, called Voronoi polygons or Voronoi polyhedra, where each data vector $\mathbf{x}$ is described by its corresponding reference vector $\mathbf{m}_j$.

The neighborhood relation of the prototypes in the NG model is defined by the ranking order of the distance of the codebook vectors to the given sample. When a data vector $\mathbf{x}$ is presented, the "neighborhood-ranking" $(\mathbf{m}_{i_0}, \mathbf{m}_{i_1}, ..., \mathbf{m}_{i_{M-1}})$ is determined, with $\mathbf{m}_{i_0}$ being the closest to $\mathbf{x}$, $\mathbf{m}_{i_1}$ being second closest to $\mathbf{x}$, and $\mathbf{m}_{i_k}$, $k = 0, .., M-1$, being the reference vector for which there are k vectors $\mathbf{m}_j$ with $\|\mathbf{x} - \mathbf{m}_j\| < \|\mathbf{x} - \mathbf{m}_{i_k}\|$. If $k_j(\mathbf{x}, \mathbf{m})$ denotes the number k associated with each vector $\mathbf{m}_j$, which depends on $\mathbf{x}$ and the whole set $\mathbf{m}$ of reference vectors, then the adaptation step for adjusting the $\mathbf{m}_j$'s is given by:

$$\mathbf{m}_j(t+1) = \mathbf{m}_j(t) + \alpha h_\lambda(k_j(\mathbf{x}, \mathbf{m}))(\mathbf{x} - \mathbf{m}_j) \qquad j = 1, .., M \tag{1}$$

with both the learning parameter function $\alpha = \alpha(t) \in [0, 1]$ and the characteristic decay function $\lambda = \lambda(t)$ are monotonically decreasing functions with respect to time. For example for α the function could be linear $\alpha(t) = \alpha_0 + (\alpha_f - \alpha_0)t/t_\alpha$ or exponential

$\alpha(t) = \alpha_0(\alpha_f/\alpha_0)^{t/t_\alpha}$, where α_0 is the initial learning rate (< 1.0), α_f is the final rate (≈ 0.01) and t_α is the maximum number of iteration steps to arrive α_f. Analogously for λ (See [10] for further details).

The neighborhood kernel $h_\lambda(k_j(\mathbf{x}, \mathbf{m}))$ is unity for $k_j = 0$ and decays to zero for increasing k_j. In this paper we use $h_\lambda(k_i(\mathbf{x}, \mathbf{m})) = \exp^{k_i(\mathbf{x},\mathbf{m})/\lambda}$. Note that if $\lambda \to 0$ then (1) is the K-means adaptations rule, whereas for $\lambda \neq 0$ not only the "winner" (bmu) $\mathbf{m}_{i_0}$ but the second closest reference vector $\mathbf{m}_{i_1}$, third closest vector $\mathbf{m}_{i_2}$, etc., are also updated.

Martinetz et. al. [6] showed that the dynamics of the $\mathbf{m}_j$'s obeys a stochastic gradient descent on the cost function:

$$E_{ng}(\mathbf{m}, \lambda) = \frac{1}{2C(\lambda)} \sum_{i=1}^{M} \int h_\lambda(k_i(\mathbf{x}, \mathbf{m}))(\mathbf{x} - \mathbf{m}_i)^2 dF(\mathbf{x}) \tag{2}$$

where $C(\lambda) = \sum_{i=1}^{M} h_\lambda(k_i) = \sum_{k=0}^{M-1} h_\lambda(k)$ is a normalization factor that only depends on λ. $F(\mathbf{x})$ is the probability distribution measure of the data generating process.

3 Robust M-Estimators for the Learning Process

The learning process of the NG can be seen as a parameter estimation process, and their inference relies on the data [2]. When observations substantially different from the bulk of data exist, they can influence badly the model structure bringing degradation in the estimates. In this work we seek for a robust estimator of the NG parameters based on M-estimators (see [4]).

Let the data set $\{\mathbf{x}_1, ..., \mathbf{x}_n\}$ consists of an independent and identically distributed (i.i.d.) sample of size n obtained from the input space $\mathcal{X} \subseteq \mathbb{R}^d$ of dimension d, i.e., $\mathbf{x}_i \in \mathcal{X}$. An M-estimator $\hat{\theta}_n^M$ is defined as

$$\hat{\theta}_n^M = \arg\min\{RL_n(\theta) : \theta \in \Theta\} \quad \text{with} \quad RL_n(\theta) = \frac{1}{n} \sum_{i=1}^{n} \rho(\mathbf{x}_i, \theta)$$

where $\Theta \subseteq \mathbb{R}^D$ is the parametric space, $RL_n(\theta)$ is a functional cost and $\rho : \mathcal{X} \times \Theta \to \mathbb{R}$ is the function that we will name it robust when it introduces a bound to the influence of outliers data during the training process. By assuming that the function ρ is differentiable with respect the parameter $\theta = (\theta_1, ..., \theta_D)$, we obtain the score function $\psi(\mathbf{x}, \theta) = (\psi_1, ..., \psi_D)'$ whose components are the partial derivatives $\psi_j(\mathbf{x}, \theta) = \frac{\partial \rho(\mathbf{x}_i, \theta)}{\partial \theta_j}$, $j = 1..D$. Then the M-estimator can be defined implicitly as the solution of the vector equations $\frac{1}{n} \sum_{i=1}^{n} \psi_j(\mathbf{x}_i, \theta) = \mathbf{0}, j = 1..D$. Table 1 shows the Least Square (LS), Huber (H) and Tukey's biweight (B) methods as examples of M-estimators. Please refer to [4] for further examples of robust functions.

We consider estimators T which are functionals of the distribution functions, i.e., $T = T(F)$, and estimators that are Fisher consistent $T(F) = \theta$. The *influence function* (IF) of the functional T at the distribution function $F = F(\mathbf{x})$ is defined as

$$IF(\mathbf{x}; T, F) = \lim_{t \to 0} \frac{T((1-t)F + t\Delta_{\mathbf{x}}) - T(F)}{t}$$

where $(1-t)F + t\Delta_{\mathbf{x}}$ is the t-contaminated model of the distribution $F(\mathbf{x})$, where $\Delta_{\mathbf{x}}$ is the probability measure which puts mass 1 at the point $\mathbf{x}$. The influence function is a local measure introduced by Hampel [4] that describes the effect of an infinitesimal contamination at the point $\mathbf{x}$ on the estimate.

Table 1. Examples of M-estimators

Name	$\rho(x)$	$\psi(x)$
Least square	$x^2/2$	x
Huber	$\begin{cases} x^2/2 & \lvert x\rvert \le \kappa \\ \kappa\,\lvert x\rvert - \kappa^2/2 & \lvert x\rvert > \kappa \end{cases}$	$\begin{cases} x & \lvert x\rvert \le \kappa \\ \kappa\, sign(x) & \lvert x\rvert > \kappa \end{cases}$
Tukey's biweight	$\begin{cases} \kappa^2(1-[1-x^2/\kappa^2]^3)/ & \lvert x\rvert \le \kappa \\ \kappa^2/6 & \lvert x\rvert > \kappa \end{cases}$	$\begin{cases} x[1-x^2/\kappa^2]^2 & \lvert x\rvert \le \kappa \\ 0 & \lvert x\rvert > \kappa \end{cases}$

An important summary value based on the IF is the gross error sensitivity that measure the worst (approximate) influence which a small amount of contamination of fixed sized can have on the value of estimator. The *gross error sensitivity* of the estimator T at the distribution F is defined as $\gamma(T,F) := \sup_{\mathbf{x}}\{\|IF(\mathbf{x},T,F)\|\}$. It is a desirable feature that $\gamma(T,F)$ be finite and in such case we say that T is B-Robust at F.

4 Robustness Analysis of the Learning Algorithm

Let the data set $\{\mathbf{x}_1,...,\mathbf{x}_n\}$, $\mathbf{x}_i \in \mathcal{X} \subseteq \mathbb{R}^d$, consists of an independent and identically distributed (i.i.d) sample of size n with common probability distribution $F(\mathbf{x})$. The cost function of equation (2) is generalized to the following form:

$$\begin{aligned} E_{ng}(\mathbf{m},\lambda) &= \frac{1}{C(\lambda)}\sum_{i=1}^{M}\int_{\mathcal{X}} h_\lambda(k_i(\mathbf{x},\mathbf{m}))\rho(\mathbf{x}-m_i)dF(\mathbf{x}) \\ &= \frac{1}{C(\lambda)}\int_{\mathcal{X}}\sum_{i=1}^{M} h_\lambda(k_i(\mathbf{x},\mathbf{m}))\rho(\mathbf{x}-m_i)dF(\mathbf{x}) \\ &= \frac{1}{C(\lambda)}\int_{\mathcal{X}} \eta(\mathbf{x},\mathbf{m})dF(\mathbf{x}) \end{aligned} \tag{3}$$

where the second equation is obtained by interchanging the integral with the summand. Let the function $\eta : \mathcal{X}\times\mathcal{M} \longrightarrow \mathbb{R}$ be defined as $\eta(\mathbf{x},\mathbf{m}) = \sum_{i=1}^{M} h_\lambda(k_i(\mathbf{x},\mathbf{m}))\rho(\mathbf{x}-m_i)$ and $\rho : \mathcal{X}\times\mathcal{M} \longrightarrow \mathbb{R}$ is the function of the M-estimator functional cost (see table 1 for examples).

An M-estimator is defined as the value $\hat{\mathbf{m}} = \{\hat{\mathbf{m}}_1,...,\hat{\mathbf{m}}_M\}$ such that

$$\hat{\mathbf{m}} = arg\min_{\mathbf{m}}\{E_{ng}(\mathbf{m},\lambda)\}$$

Assuming that ρ is differentiable whose derivative is given by $\psi(r) = \frac{\partial \rho(r)}{\partial r}$ then

$$\varphi_j(\mathbf{x}, \mathbf{m}) = \frac{\partial \eta(\mathbf{x}, \mathbf{m})}{\partial \mathbf{m}_j} = \text{-}h_\lambda(k_j(\mathbf{x}, \mathbf{m}))\psi(\mathbf{x} - \mathbf{m}_j) + R_j \qquad j = 1, ..., M$$

with $R_j = \sum_{i=1}^{M} h'_\lambda(k_j(\mathbf{x}, \mathbf{m}))\rho(\mathbf{x} - \mathbf{m}_j)\frac{\partial k_i(\mathbf{x},\mathbf{m})}{\partial \mathbf{m}_j}$. Martinez et al. [6] demonstrated that $\int R_j dF(x) = 0$ for each $j = 1, .., M$. An M-estimator $\hat{\mathbf{m}}$ can be defined implicitly by the solution of the vector equation $\varphi_j(\mathbf{x}, \mathbf{m}) = 0, \forall j = 1, ..., M$.

The influence function $IF_j(\mathbf{x}, \mathbf{m}, F)$ associated to the M-estimator $\hat{\mathbf{m}}_j$ is given by the following equation:

$$IF_j(\mathbf{x}, \mathbf{m}, F) = h_\lambda(k_j(\mathbf{x}, \mathbf{m}))\psi(\mathbf{x} - \mathbf{m}_j)H^{-1} \qquad \forall j = 1, ..., M$$

where $H = \text{-} \int \frac{\partial}{\partial \mathbf{m}} \left[\varphi_j(\mathbf{x}, \mathbf{m})\right]_{\hat{\mathbf{m}}} dF(\mathbf{x})$. Note that if the classical Least Square estimator is used, then the gross error sensitivity is $\gamma(\hat{\mathbf{m}}^{LS}, F) = \infty$, this reflect the fact that, for any samples size, even a single outlier can carry the estimates over all bounds (if it is far enough).

4.1 Robust Learning Algorithm

The updating rule (1) employed in the original NG algorithm lacks of robustness as was shown in the previous section. The gross error sensitivity is infinity implying that the learning step is biased toward the location of the outlying observations. To overcome this problem the following updating rule that obeys a stochastic gradient descent on the cost function (3) is used instead:

$$\mathbf{m}_j(t+1) = \mathbf{m}_j(t) + \alpha h_\lambda(k_j(\mathbf{x}, \mathbf{m}))\psi(\mathbf{x} - \mathbf{m}_j) \qquad j = 1, ..., M \tag{4}$$

where the $\psi(\cdot)$ function diminish the influence of the outliers (see table 1 for examples). Unfortunately, the learning rule of equation (4) is not invariant with respect to scale, which is often a nuisance parameter. To overcome this problem we can standardized each data sample $\mathbf{x}_i = (x_1, ..., x_d)'$ as $\hat{S}_j^{-1/2}(\mathbf{x} - \mathbf{m}_j)$, where $\hat{S}_j$ is the robust estimation of the covariance matrix of the difference $\mathbf{x} - \mathbf{m}_j$ of all the data that belong to the Voronoi polygon V_j, and $\hat{S}_j^{-1/2} = \sqrt{\hat{S}_j^{-1}}$ is the square root of the inverse of the covariance matrix. Now we can redefine the robust learning rule as follows

$$\mathbf{m}_j(t+1) = \mathbf{m}_j(t) + \alpha h_\lambda(k_j(\mathbf{x}, \mathbf{m}))\psi\left(\hat{S}_j^{-1/2}(\mathbf{x} - \mathbf{m}_j)\right) \qquad j = 1, .., M \tag{5}$$

One could compute $\mathbf{m}$ and $\hat{S}$ simultaneously as M-estimators of location and scale respectively. However, simulations have shown the superiority of M-estimators with scale estimated iteratively during the learning process by the scaled version of the *median of the absolute deviations from the median* (sMAD):

$$sMAD(x_1, ..x_n) = \frac{1}{\Phi^{-1}(\frac{3}{4})} \underset{l=1..n}{median} \left\{ \left| x_l - \underset{k=1..n}{median}(x_k)) \right| \right\}$$

where $\Phi^{-1}(p)$ is the inverse of the standard Gaussian cumulative distribution function at the probability p. The constant $\frac{1}{\Phi^{-1}(\frac{3}{4})} \approx 1.483$ is needed to make the $sMAD$ scale estimator Fisher consistent when the data behave as Gaussian distribution. We apply the sMAD function for each dimension component of the sample $\{(\mathbf{x_i} - \mathbf{m}_j)\}_{i=1}^{n}$ and for each prototype $j = 1..M$.

The robust M-estimators of the learning update rule have initialization problems. To overcome this situation the constant κ of the $\psi(\cdot)$ function is considered as a function of time and the standard deviation, i.e., $\kappa = \kappa(t)\sigma$, where $\kappa(t)$ monotonically decreases with time. In this paper we use a linear decreasing function $\kappa(t) = \kappa_0 + (\kappa_f - \kappa_0)t/t_f$, where κ_0 is the initial value (bigger than 6); and κ_f is the final value (between 3 and 6) and t_f is the maximum number of iteration steps to arrive κ_f.

5 Simulation Results

In this section we provide a comparative study of the Least Square (LS), Huber (H) and Biweight (B) M-estimators applied to the learning process of the NG, in static data clustering tasks on both synthetic and real data sets, the latter were obtain from the UCI benchmark [3].

In the experiments, all the dimensions of the training data set were scaled to interval $[-1, 1]$, and the test data set were scaled using the same scale applied to the training data set. The test data set will not necessarily fall in the same interval. The number of epochs utilized for all experiments is $t_f = 500$ and the α and λ decay in exponential form with values: $\alpha_0 = 0.9$, $\alpha_f = 0.05$, $\lambda_0 = M/3$, $\lambda_f = 0.01$, $\kappa_0^H = 30$, $\kappa_0^B = 40$ and $\kappa_f = 3$.

5.1 Performance Evaluation of the NG

In the experiments we use the following metrics to evaluate the performance of the NG model. The *Classification Accuracy* (CA) is the right classification percentage

$$CA = \frac{1}{n}\sum_{i=1}^{n} \mathcal{I}(\mathbf{x}_i, \mathbf{m}_{c(\mathbf{x}_i)}) \cdot 100\% \tag{6}$$

where we label the neuron $\mathbf{m}_j$ with the class that has the majority in its Voronoi polygon V_j. $\mathcal{I}(\mathbf{x}_i, \mathbf{m}_{c(\mathbf{x}_i)})$ takes the value of one if the label of the data $\mathbf{x}_i$ is equal to the label of its best matching unit $\mathbf{m}_{c(\mathbf{x}_i)}$, and takes the value of zero in the other case.

The *Mean Square Quantization Error* is given by

$$MSQE = \frac{1}{n}\sum_{i=1}^{n} \left\|\mathbf{x}_i, \mathbf{m}_{c(\mathbf{x}_i)}\right\|^2 \tag{7}$$

The *Mean Distance from the Neurons to the closest cluster Center* measures the average Euclidean distance between the neurons $\mathbf{m}_j$ and its closest cluster center $\mu(\mathbf{m}_j)$:

$$MDNC = \frac{1}{M}\sum_{j=1}^{M} \|\mathbf{m}_j - \mu(\mathbf{m}_j)\|^2 \tag{8}$$

Finally, the *Numbers of clusters Center in the Voronoi polygon* is given by

$$NC = \sum_{\mathbf{m}_j \in \mathcal{M}} |\tau_j - 1| \tag{9}$$

where τ_j correspond to the number of clusters center that are inside the Voronoi polygon V_j. A desirable feature is that each neuron model no more and no less than one cluster center.

5.2 Experiment #1: Computer Generated Data

The synthetic experiments were constructed by generating a square grid of twenty five cluster drawn from two-dimensional Gaussian distributions $\mathbf{X}_l \sim \mathcal{N}(\mu_l, \Sigma_l)$, $l = 1, ..., L = 25$, where μ_l is the mean vector of the cluster l and $\Sigma_l = \Sigma$ is its co-variance matrix. A total of 2500 samples for the training and the same quantity for the test were generated, where each cluster has an expected size of 100 samples.

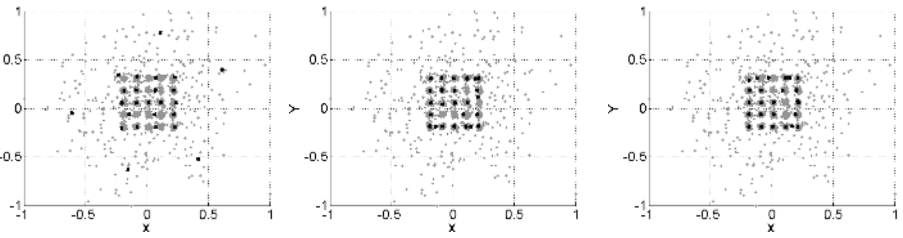

Fig. 1. Topology adaptation of the NG. Comparative results of the NG model to the synthetic data with 15% of outliers for the Least Square (left), Huber (middle) and Biweight (right) estimators.

The observational process is obtained by adding additive outliers: $\mathbf{Z}_l = \mathbf{X}_l + V_l\, U_l$, where V_l is zero-one process with $P(V_l \neq 0) = \varepsilon$, $0 < \varepsilon \ll 1$ and U_l has distribution $\mathcal{N}(\mathbf{0}, \Sigma_U)$ with $|\Sigma_U| \gg |\Sigma|$. The generating process was affected with $\varepsilon = 5\%$, 10%, 15% and 20% of outliers with $\Sigma_U = 27\Sigma$.

Figure 1 shows the adaptation of the NG to the data with 15% of outliers for the three estimators: Least Square (left), Huber (middle) and Biweight (right). The LS estimator was badly affected by locating five prototypes far from the square grid. Nevertheless, the robust estimators (Huber and Biweight) were much less affected and their respective neurons were located inside the grid of 25 clusters, with each neuron close to the some of the clusters center. The robust estimators diminished the influence of the outlying observations and they had a better topology preservation than the LS-estimator.

Table 2 shows the summary results of the performance evaluation of the NG with the three estimators.The LS estimator obtained the lowest $MSQE$ evaluated in the data with outlier for all the experiments (columns $E1$ and $E2$), this result was expected because the LS-estimator minimizes the cost function (3) while the robust estimators minimize the cost function (3). However, if we compute the MSQE in the data without outliers (columns $E3$ and $E4$), the robust estimators outperforms the LS-estimator.

The $CA1$ and $CA2$ show the classification accuracy of the training and test set respectively, as was expected, the robust estimators obtained better performance than the LS-estimator when the percentage of outliers are increased. The column NC computed with equation (9) show how the robust estimators remains stable with an increasing percentage of outliers, while the LS-estimator got worse. Finally, the last column show that the robust estimators outperforms the LS-estimator, meaning that the former locate the prototypes closer to the clusters center. Note that the last two columns measures the quality of the topology preservation of the square grid.

Table 2. Summary results of the performance evaluation of the NG model with 25 prototypes with the LS, H and B estimators used in the training process. The second column is the percentage of outliers in the data. From the columns $E1$ to $E4$ are the value of Mean Square Quantization Error (7), column $E1$ and $E2$ correspond to the error evaluation of training and test set respectively by considering the outliers, while columns $E3$ and $E4$ are the error evaluation of training and test data sets without considering the outliers. The columns $AC1$ and $AC2$ are the value of the Classification Accuracy (6) of the training and test sets respectively by considering the outliers. Finally the column NC and $MDNC$ are the values obtained with equations (9) and (8) respectively.

Estimators	% Outliers	$E1$	$E2$	$E3$	$E4$	$CA1(\%)$	$CA2(\%)$	NC	$MDNC$
LS	5	**0.0039**	**0.0045**	**0.0009**	**0.0009**	**88.7**	**88.0**	**3.8**	0.0171
H	5	0.0063	0.0066	0.0012	0.0013	86.9	85.5	4.8	**0.0004**
B	5	0.0064	0.0067	0.0016	0.0016	82.5	81.6	6.8	0.0005
LS	10	**0.0054**	**0.0065**	0.0013	0.0013	78.4	77.1	7.6	0.0263
H	10	0.0102	0.0112	**0.0011**	**0.0012**	**82.4**	**81.0**	**5.2**	**0.0004**
B	10	0.0103	0.0113	0.0015	0.0016	80.4	79.1	6.0	0.0005
LS	15	**0.0056**	**0.0069**	0.0016	0.0017	69.2	66.4	11.4	0.0388
H	15	0.0134	0.0141	**0.0010**	**0.0011**	76.4	75.9	6.0	**0.0003**
B	15	0.0134	0.0141	0.0013	0.0013	**76.7**	**76.0**	**5.2**	0.0004
LS	20	**0.0065**	**0.0072**	0.0019	0.0019	62.5	59.6	13.6	0.0432
H	20	0.0171	0.0170	**0.0010**	**0.0010**	**74.7**	**73.9**	**5.0**	0.0003
B	20	0.0175	0.0174	0.0013	0.0014	72.5	71.9	6.2	**0.0002**

5.3 Experiment #2: Real Data Sets

In the second experiment we test the algorithm with four real datasets obtained from the UCI Machine Learning repository [3]. The *Wine recognition*, *Glass Identification*, *Cancer* and *Pima Indians Diabetes* data sets were selected.

Figure 2 displays the Mean Square Quantization Error of the M-estimators computed for all the data sets. Similarly to the previous section, the LS-estimator obtained better performance in this metric with respect to the robust estimators. Nevertheless, we can not assure the topology preservation of the LS-estimator as the robust estimators, if the real data are contaminated.

Figure 3 shows the Classification Accuracy of the NG modelling all the real data sets. In the Cancer data set (upper-left) the best performance were obtained with a low number of neurons, while in the other experiments better performance were obtained with increasing number of prototypes. When the number of the prototypes are less than

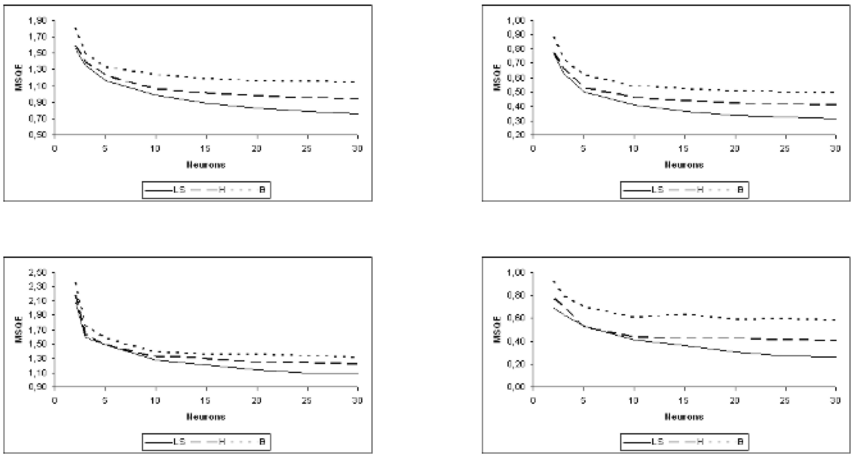

Fig. 2. MSQE performance evaluation for the Real data sets. Neural Gas modelling the Cancer (Upper-Left), Pima Indians Diabetes (Upper-Right), Glass Identification (Bottom-Right) and the Wine Recognition (Bottom-Left) data sets respectively.

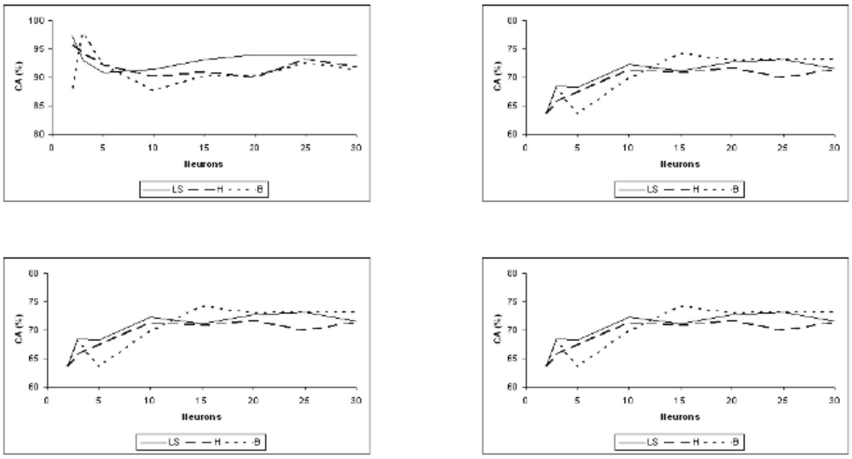

Fig. 3. Classification Accuray for the Real data sets. Neural Gas modelling the Cancer (Upper-Left), Pima Indians Diabetes (Upper-Right), Glass Identification (Bottom-Right) and the Wine Recognition (Bottom-Left) data sets respectively.

the number of classes the performance of the NG model for all the estimator was very poor (the Cancer, the Pima Indians Diabetes, the Glass Identification and the Wine Recognition data sets were composed of 2, 2, 8, 3 classes respectively). Only in the Wine Recognition data set one of the estimators (biweight) outperforms the others, while in the other cases we can not see the superiority of any of the estimators.

6 Concluding Remarks

In this paper we analyzed the robustness properties of the learning process of the Neural Gas model based on the M-estimator robust theory. We have demonstrated that the classical Neural Gas learning algorithm lacks of robustness and with small amount of contamination the codebooks are biased towards the outliers. We have introduced robust M-estimator to diminish the influence of outlying observations and making the learning process more stable.

In the synthetic experiment the robust estimators outperforms the LS-estimator in the topology preservation. However, we empirically showed that the $MSQE$ was not a good performance metric of the adaptation quality under contaminated data.

In the real data experiments, all the estimators showed similar performance in the classification accuracy. But we were not able to assure the topology preservation for the LS-estimator as in the robust estimator case. It is of great importance find algorithms to determine the learning parameters and the ordering properties. Finally, due to the similar behavior in real data sets of all algorithms is better to use a robust function learning to explore the data because it works with or without presence of outliers.

Further studies are needed to extend the results to other class of estimators (L, M and R estimators) as well as other self organizing models. Furthermore, index that measures the quality of topology preservation are needed instead of the MSQE.

References

1. H. Allende, S. Moreno, C. Rogel, and R. Salas, *Robust self organizing maps*, CIARP. LNCS **3287** (2004), 179–186.
2. H. Allende, C. Rogel, S. Moreno, and R. Salas, *Robust neural gas for the analysis of data with outliers*, IEEE-CS Press. SCCC 2004 (2004), 149–155.
3. C.L. Blake and C.J. Merz, *UCI repository of machine learning databases*, 1998.
4. F.R. Hampel, E.M. Ronchetti, P.J. Rousseeuw, and W.A. Stahel, *Robust statistics*, Wiley Series in Probability and Mathematical Statistics, 1986.
5. T. Kohonen, *Self-Organizing Maps*, Springer Series in Information Sciences, vol. 30, Springer Verlag, Berlin, Heidelberg, 2001, Third Extended Edition 2001.
6. T. Martinetz, S. Berkovich, and K. Schulten, *Neural-gas network for vector quantization and its application to time-series prediction*, IEEE Trans. on Neural Networks **4** (1993), no. 4, 558–568.
7. A. Qin and P. Suganthan, *Robust growing neural gas algorithm with application in cluster analysis*, Neural Networks **17** (2004), 1135–1148.
8. R. Salas, S. Moreno, and H. Allende C. Moraga, *A robust and flexible model of hierarchical self organizing maps for nonstationary environments*, To appear in Neurocomputing (2006).
9. U. Seiffert and L. Jain (eds.), *Self-organizing neural networks: Recent advances and applications*, Studies in Fuzziness and Soft Computing, vol. 78, Springer Verlag, 2002.
10. M. Su and H. Chang, *Fast self-organizing feature map algorithm*, IEEE Trans. on Neural Networks **11** (2000), no. 3, 721–733.