Proceedings of the 5th WSEAS International Conference on Signal Processing, Istanbul, Turkey, May 27-29, 2006 (pp127-134)

Clustering with Kernel-Based Self-Organized Maps trained with Supervised Bias

STERGIOS PAPADIMITRIOU KONSTANTINOS TERZIDIS

Dept of Information Management
Technological Educational Institute of Kavala
Kavala, 65404, GREECE

URL: http://infoman.teikav.edu.gr/~stpapad

Abstract

Self-Organized Maps (SOMs) are a popular approach for clustering data. However, most SOM based approaches ignore prior knowledge about potential categories. Also, Self Organized Map (SOM) based approaches usually develop topographic maps with disjoint and uniform activation regions that correspond to a hard clustering of the patterns at their nodes. We present a novel Self-Organizing map, the Kernel Supervised Dynamic Grid Self-Organized Map (KSDG-SOM). This model adapts its parameters in a kernel space. Gaussian kernels are used and their mean and variance components are adapted in order to optimize the fitness to the input density. The KSDG-SOM also grows dynamically up to a size defined with statistical criteria. It is capable of incorporating a priori information for the known categories. This information forms a supervised bias at the cluster formation and the model owns the potentiality of revising incorrect functional labels. The new method overcomes the main drawbacks of most of the existing clustering methods that lack a mechanism for dynamical extension on the basis of a balance between unsupervised and supervised drives.

Key-Words: Self-Organized Maps, Supervised Learning, Kernel Classifiers, Clustering

1 Introduction

The paper introduces the Kernel based Supervised Dynamic Grid Self-Organized Map model (KSDG-SOM) model. Clustering is one of the most widely used data analysis technique for many data analysis applications [2, 5, 6]. It aims to provide insight into the structure of the data and aids at the discovery of functional classes.

Bayesian clustering [7, 8], and the Self-Organizing Map (SOM) [6] usually ignore any existing class information. In addition many kernel-based developments of the SOM approach, do not have an explicit provision for incorporating supervised labeling [9, 10].

The standard SOM algorithm has a number of properties, which render it to a candidate of particular interest as a basis framework for building more flexible and advanced algorithms for analysis. SOMs can be implemented easily, are fast, robust and scale well to large data sets. They allow one to impose partial structure on the clusters and facilitate visualization and interpretation. In the case hierarchical information is required, it can be implemented on top of SOM, as in [11].

Recently, several dynamically extended schemes have been proposed that overcome the limitation of the fixed non-adaptable architecture of the SOM. Some examples are the Dynamic Topology Representing structures [12], the Growing Cell Structures [8, 13], Self-Organized Tree Algorithms [14, 5], the joint entropy maximization approach [15] and the Adaptive Resonance Theory [16].

The presented approach has some similarities to these dynamically extended schemes, from the point of view of its unsupervised component. However, one essential difference exists: all the forementioned schemes are purely unsupervised, lacking a design for the incorporation of problem domain knowledge. Instead, we focus on the design of such types of algorithms that aim to explore effectively existing *a priori supervised class labeling*, for *multi-class* and *multi-labeled* data. The multiple labeling, i.e. the possible assignment of more than one class label at each data record, perplexes the clustering and classification tasks. Also, in contrast to the complexity of some of these schemes, we design simple algorithms that through the restriction of growing on a rectangular grid, can be implemented easily and the training of the models is very efficient. In addition, most of the benefits of the more complex alternatives of dynamical extension are still retained.

We call the proposed model KSDG-SOM from Kernel Supervised Dynamic Grid SOM, since it is a model trained in kernel space and although it is SOM based it tightly integrates unsupervised and supervised learning components.

As a kernel the Gaussian one is used. The Gaussian kernel mapping implements more elaborate soft class separation boundaries than the hard separation onto Voronoi regions obtained by evaluating directly at the input space the inner products of the patterns and the prototype vectors. As with other kernel methods [17, 18], we aim to exploit a nonlinear transformation of the input space onto a high-dimensional feature space. Intuitively, the SOM based learning constructs Voronoi regions over this high-dimensional space in which the extra dimensions enhance the possibilities of defining more elaborately the cluster boundaries.

The KSDG-SOM has been designed in order to automatically detect the *appropriate level of expansion*, so that the number of clusters is controlled by a properly defined measure of the algorithm itself, with no need for any a priori specification. This is quite important for analysis tasks where very little (or nothing) can be claimed about an a priori estimate of the number of clusters.

Unfortunately, the functions of most data (e.g. gene expression recordings) are either unknown or partially and unreliably known. Motivated by this we have designed the KSDG-SOM in order to be simply another one clustering and classification device but instead and more important to be *a reclassification device*. It takes both input data and existing classification labels into account in order to evaluate a new set of classes. The cost function (minimized during learning) implements a trade-off between the unsupervised quantization error (arising from the input variance) and the supervised mislabeling error (quantified by an entropy-like term). The unsupervised error in traditional supervised approaches (e.g. Radial Basis Function, Support Vector Machines) is hidden behind the supervised error, i.e. the unsupervised techniques are used as hidden steps, not directly connected to the ouput labels. Hence, these approaches *cannot penalize explicitly the dispersion in the input space*. The presented approach accomplishes such a compromise between unsupervised and supervised drives that guides to a reclassification of the training set, that can expose new insights at the problem structure.

Comparing with other related approaches in the literature, that attempt to integrate both unsupervised and supervised learning, the approach of [19] preserves the topology of the space defined by the patterns themselves ("primary space" e.g. unsupervised part) and uses the class labelings ("auxiliary" data) to measure distances by similarity in the auxiliary space. The locality criterion in terms of the primary space implements the unsupervised contribution, while the computation of similarities in terms of the labeled "auxiliary" space constitutes the supervised part. However, such an approach is aimed for the cases where sufficient and reliable class information exist. To the contrary, we aim to exploit supervised information only when it exists and thus we confront the problems with both incomplete and unreliable class labelings of data.

Another related work, is the approach to reclassification with supervised clustering presented in [20]. The main differences and improvements of our approach compared with the former are:

1. We perform the determination of the proper number of clusters for each unsupervised/supervised relative strength via the dynamical extension process of the KSDG-SOM that is controlled with rigid statistical criteria.

2. We exploit the advantages of the kernel mapping. We prefer the use of Gaussian kernels due to their mathematical tractability.

3. We quantify the inconsensus in supervised labelings of the patterns represented by a node with an entropy-like term that is descriptive even for the symbolic unordered class labelings of the data.

4. We confront the case of multiple functional class labelings of the data.

The paper is outlined as follows: Section 2 deals with the definition of error measure minimization in kernel space. Section 3 deals with the learning algorithms that adapt both the structure and the parameters of the KSDG-SOM. The expansion phase of the KSDG-SOM learning is described in Section 4 separately since it is rather lengthy. Section Finally, Section 5 presents the conclusions along with some directions onto which further research can proceed for improvements.

2 Kernel Space Adaptation of Gaussian Means/Variances

This section derives learning rules for the adaptation of the Gaussian kernels of the KSDG-SOM. These rules are in turn used in

Section 3.4.1 that defines the update of weights and variances during the learning process in terms of them. Clearly, formulas for the adaptation of both the Gaussian means and of variances should be derived. The accommodated Gaussian nodes with overlapping activation regions, instead of the usual SOM nodes with disjoint and uniform activation regions (Voronoi tesselation), increase significantly the potentiality of the model for effective learning of the input density [21]. Furthermore, a proper higher-dimensional feature space (e.g. the product space) can reveal interesting structure of the data (e.g. higher order statistics) which may go unnoticed in the standard Euclidean space [22].

Denote by L a lattice of N nodes and by $I \subset \Re^d$ the input space. Each node $i \in L$ is characterized by its weight vector $\mathbf{w_i} = [\mathbf{w_{i1}} \dots \mathbf{w_{id}}] \in \mathbf{I}$ and by a lattice coordinate $\mathbf{r_i} \in \mathbf{L_A}$, where L_A is the lattice space.

Instead of directly computing the activation of a node with weight vector $\mathbf{w_i}$ from the input vector $\mathbf{x}$ by the inner product $\langle \mathbf{x}, \mathbf{w_i} \rangle$, we utilize a kernel function $K(\mathbf{x}, \mathbf{w_i}) = \langle \mathbf{\Phi}(\mathbf{x}), \mathbf{\Phi}(\mathbf{w_i}) \rangle$. The selected kernel is the Gaussian one:

$$K(\mathbf{x}, \mathbf{w_i}, \sigma_\mathbf{i}) = \exp(-\frac{\|\mathbf{x} - \mathbf{w_i}\|^\mathbf{2}}{\mathbf{2}\sigma_\mathbf{i}^\mathbf{2}})$$

Following the common practice of SOM-like algorithms, we want to minimize the distortion between the mapping of the input $\Phi(\mathbf{x})$, assigned to node i, and the mapping of the node $\Phi(\mathbf{w_i})$, i.e. the prototype of node i in kernel space. Therefore we perform gradient descent with respect to $\mathbf{w_i}$ in order to adapt it properly:

$$\begin{aligned} \frac{\partial}{\partial \mathbf{w}_i} \|\Phi(\mathbf{x}) - \mathbf{\Phi}(\mathbf{w_i})\|^\mathbf{2} &= \frac{\partial}{\partial \mathbf{w}_i} K(\mathbf{w_i}, \mathbf{w_i}, \sigma_\mathbf{i}) + \frac{\partial}{\partial \mathbf{w_i}} \mathbf{K}(\mathbf{x}, \mathbf{x}, \sigma_\mathbf{i}) \\ &- 2\frac{\partial}{\partial \mathbf{w}_i} K(\mathbf{x}, \mathbf{w_i}, \sigma_\mathbf{i}) \\ &= -2\frac{\partial}{\partial \mathbf{w}_i} \exp\left(-\frac{\|\mathbf{x} - \mathbf{w_i}\|^\mathbf{2}}{2\sigma_i^2}\right) \end{aligned} \quad (1)$$

Thus the update rule for the kernel centers $\mathbf{w_i}$ should be of the following form:

$$\Delta \mathbf{w_i} = \mu_\mathbf{w} \frac{(\mathbf{x} - \mathbf{w_i})}{\sigma_\mathbf{i}^\mathbf{2}} \mathbf{K}(\mathbf{x}, \mathbf{w_i}, \sigma_\mathbf{i}) \quad (2)$$

with μ_w the learning rate for the kernel centers, referred as *kernel centers learning rate parameter* μ_w.

The next step is to derive the learning rule for the kernel radii σ_i, $\forall i \in L$. The parameters σ_i correspond to the variance with a Gaussian mixture model interpretation. An effective policy for their initialization is to set them equal to the median of the Euclidean distances from each positive training set member to the nearest negative. By performing gradient descent on $\|\Phi(\mathbf{x}) - \mathbf{\Phi}(\mathbf{w_i})\|^\mathbf{2}$, with respect to σ_i, we obtain:

$$\Delta \sigma_i \propto \frac{\|\mathbf{x} - \mathbf{w_i}\|^\mathbf{2}}{\sigma_i^3} K(\mathbf{x}, \mathbf{w_i}, \sigma_\mathbf{i})$$

Therefore the update rule for the Gaussian centers will be of the form:

$$\Delta \sigma_i = \mu_\sigma \frac{\|\mathbf{x} - \mathbf{w_i}\|^\mathbf{2}}{\sigma_i^3} K(\mathbf{x}, \mathbf{w_i}, \sigma_\mathbf{i}) \quad (3)$$

with μ_σ the learning rate for the kernel variances (i.e. radii), referred as *kernel variance learning rate parameter* μ_σ.

Typical values that we have used for the kernel centers learning rate parameter μ_w and the kernel variance learning rate parameters μ_σ, are $\mu_w = 0.1$ and $\mu_\sigma = 0.1$. However, the convergence of the algorithm is not sensitive to the exact values of these parameters. We obtained convergence even for large values of μ_w, μ_σ, i.e. $\mu_w = 0.5$, $\mu_\sigma = 0.7$. Clearly, for small values (e.g. $\mu_w = 0.01, \mu_\sigma = 0.01$) convergence is obtained, as it was expected.

3 The KSDG-SOM algorithm

This section is organized in two parts: The first subsection elaborates further the general error criterion for optimization. The next subsection details on the algorithms for the minimization of this error.

3.1 Elaboration of the error criterion for optimization

The KSDG-SOM learning process should aim to minimize a heterogeneous [1] performance measure (i.e. error function). Denoting by K_n the number of KSDG-SOM nodes, which correspond (as noted previously) to *clusters* of patterns, clearly $K > K_n$ since each supercluster is composed of one or more clusters. We can reformulate the general form of the error function in terms of SDG-SOM nodes and with distinct unsupervised and supervised parts as:

$$E = \sum_{k=1}^{K_n} \text{UnsupervisedComponent}_\text{k} + \text{r}_\text{su} \cdot \sum_{\text{k=1}}^{\text{K}_\text{n}} \text{SupervisedComponent}_\text{k} + \text{ModelC} \quad (4)$$

The UnsupervisedComponent will be measured in terms of the AverageLocalError(abbreviatedALE) and the SupervisedComponent by means of an entropy-like quantity (abbreviated EntropyLike) that generally is not an entropy with the strict definition. Also, r_{su} is a parameter that controls the *relative significance of the supervised part*. Restating of these quantities we obtain a measure Θ_E to minimize:

$$\Theta_E = \min\left(\sum_{k=1}^{K_n} ALE_k + r_{su} \cdot \text{EntropyLike}_k + MOP\right) \quad (5)$$

where ALE denotes the Average Local Error, EntropyLike is the forementioned entropy-like parameter, MOP abbreviates the Model Order Penalty and K_n is the number of nodes. This minimization is achieved with the formulation of SOM-like learning rules and with a dynamic expansion process.

The parameter ALE of Equation 5 accounts for the unsupervised (quantization) error corresponding to the representation of each pattern i by the codebook of node k and can deal with the lack of class information. This measure tries to disperse patterns that are different according to some similarity metric, to different clusters, even if they are labeled with the same functional class label.

A commonly used measure for the Local Error (LE), and the one that we minimize with the formulation of equation 1, is the Euclidean distance between the kernel mapping $\Phi(\mathbf{x_i})$ of input vector

[1] Heterogeneous in the sense that integrates supervised, unsupervised and model complexity terms

$\mathbf{x_i}$ and the kernel mapping $\Phi(\mathbf{w_k})$ of the representative prototype $\mathbf{w_k}$ of its best matching node k, i.e.

$$\mathrm{LE}_i = \|\Phi(\mathbf{x_i}) - \mathbf{\Phi}(\mathbf{w_k})\|^{\mathbf{2}} \quad (6)$$

Then the Average Local Error (ALE) is obtained by averaging the Local Error (LE) over all the patterns mapped to the same node. The justification for this averaging is explained in Section 4.

The available a priori information for the functional class of the patterns is considered by the *entropy like measure*. This measure corresponds to an entropy like quantity that characterizes the uniformity of functional labeling of the node where the pattern is mapped. It is formally defined in Section 4. The entropy measure is effective for evaluating category discordances over the discrete nominal category space of functional labelings. Minimization of this measure is performed by *gathering patterns with the same (or at least similar) labels onto the same clusters*.

The *MOP* (ModelOrderPenalty) term punishes any increase in the model complexity. In this framework, the model complexity relates to the number K_s of superclusters of KSDG-SOM nodes.

The *superclusters* on the KSDG-SOM lattice (viewed as undirected graph) are defined as the *connected components* of the graph that *represent the same class labels*. It is clear that the superclusters can partition the state space in regions of *arbitrary shape* (i.e. not necessarily spherical as the case of the clusters corresponding to KSDG-SOM nodes).

Since superclusters define relatively homogenous regions of the state space, we found more appropriate to define the model complexity in terms of these. We define the *model order penalty* in terms of the squared difference between the number of the original classes and the number of developed superclusters that represent the newly formed (i.e. discovered) classes. We should note that very simple models consisted of less superclasses than the original classes are also punished.

Specifically, denoting (as usually) by K_0 the number of original classes and by K_s the number of superclusters, a term of the following form is exploited:

$$\mathrm{MOP} = \gamma \cdot (K_0 - K_s)^2 \quad (7)$$

In order to set properly the parameter γ we use a well working heuristic: we require that by doubling the number of original classes K_0 (i.e. $K_s = 2 \cdot K_0$) we get a penalty equal to the unsupervised + supervised part of the error. This criterion was found appropriate to automatically set a value for the γ parameter at the KSDG-SOM implementation.

With $r_{su} = 0$ we have pure unsupervised learning with model complexity penalization. As r_{su} increases, the cost Θ_E is minimized for configurations that fit better to the a priori classification. Finally, for sufficiently large values of r_{su}, the a priori component dominates completely. Clearly, since in this case the information provided by the data is demolished, care should be taken to avoid such r_{su} values.

After the former discussion on the error components that the KSDG-SOM aims to jointly minimize, we can now proceed to describe the corresponding learning algorithms.

3.2 The KSDG-SOM dynamic growing and adaptation

The KSDG-SOM is initialized with four nodes arranged in a 2X2 rectangular grid and grows nodes to represent the input data. This type of initialization is somehow arbitrary and different starting configurations can be used, e.g. 9 nodes arranged as a 3X3 grid. Weight values of the nodes are self-organized according to a new method inspired by the SOM algorithm. The self-organization process maps properties of the original high-dimensional data space onto the lattice consisted of KSDG-SOM nodes. The map is expanded to represent the input space by creating new nodes, either from the boundary nodes performing *boundary extension*, or by inserting whole columns (or rows) of new units with a *column extension* (or row extension).

A *training epoch* consists of the presentation of all the training patterns to the KSDG-SOM. A *training run* is defined as the training of the KSDG-SOM with a fixed number of neurons at its lattice i.e. the training between successive node insertions/deletions.

The top-level KSDG-SOM learning in algorithmic notation can be described as:

<Top-level KSDG-SOM learning algorithm>

1. Initialization (sets $r_{su} = 0$, i.e. pure unsupervised learning). (Subsection 3
Repeat // develop a series of models corresponding to increasing
// consideration of the supervised parameter, r_{su}
Repeat
2. Training Run Adaptation phase. (Subsection 3.4)
3. Expansion phase (Section 4)
until criteria for stopping map expansion are satisfied
4. Fine Tuning Adaptation phase (Subsection 3.6)
5. Save configuration of the map for the current supervised/unsupervised rati
6. Compute classification performance for the current r_{su}
7. Increment the significance of the supervised part, i.e. increase ratio r_{su}
until *Classification Performance* $\approx$ *1*
8. Model Selection Step (Subsection 3.8)

The details of the algorithm, i.e. the initialization, adaptation, fine tuning phases and the corresponding convergence criteria are described in detail below. The technical subleties involved in the expansion process are scrutinized detail in section 4.

3.3 Initialization phase.

The weight vectors of the four starting nodes that are arranged in a 2X2 grid are initialized with random numbers within the domain of feature values. Other initialization schemes are possible i.e. as it has been noted we can initialize to a 3X3 grid. The supervision parameter r_{su} controls the tradeof between unsupervised and supervised training and is discussed in detail in Section 4. It is initialized to 0, i.e. pure unsupervised learning is performed for the first KSDG-SOM model being generated.

3.4 Training Run Adaptation phase.

The purpose of this phase is to stabilize the current map configuration in order to be able to evaluate its effectiveness and the requirements for further expansion. During this phase, the input patterns are repeatedly presented and the corresponding self-organization actions are performed until the map converges sufficiently. The training run adaptation phase takes the following algorithmic form.

<Training Run Adaptation Phase>:
MapConverged := **false**;
while *MapConverged* = **false do**
for all input patterns **do**
present and adapt the map by applying the *map adaptation rules* (
endfor

Evaluate map *training run convergence condition* (Subsection 4.1) and set MapConverged accordingly

endwhile

The map adaptation rules are described below while the training run convergence condition is described in Section 4.1, after the appropriate terms have been defined.

3.4.1 Map adaptation rules

The map adaptation rules that govern the processing of each input pattern $\mathbf{x}_k$ are as follows:

1. Determination of the weight vector $\mathbf{w}_i$ for which its kernel mapping $\Phi(\mathbf{w_i})$ is closest to the kernel mapping $\Phi(\mathbf{x}_k)$, of the input vector $\mathbf{x_k}$, according to the utilized distance measure (i.e. determination of the winner node).

2. Adaptation of the weight vectors (i.e. Gaussian centers) only for the four nodes in the direct neighborhood of the winner and for the winner itself according to the following formula:

$$\mathbf{w}_j(k+1) = \begin{cases} \mathbf{w}_j(k), & j \notin N_k \\ \mathbf{w}_j(k) + \mu_w \cdot \Lambda_k(d(i,j)) \cdot \Delta\mathbf{w}_j(k), & j \in N_k \end{cases} \quad (8)$$

3. Adaptation of the Gaussian spreads also only for the four nodes in the direct neighborhood of the winner and for the winner itself according to the following formula:

$$\sigma_j(k+1) = \begin{cases} \sigma_j(k), & j \notin N_k \\ \sigma_j(k) + \mu_\sigma \cdot \Lambda_k(d(i,j)) \cdot \Delta\sigma_j(k), & j \in N_k \end{cases} \quad (9)$$

where the learning rates $\mu_w(k),\ \mu_\sigma(k),\ k \in N$, are monotonically decreasing sequence of positive parameters, N_k is the neighborhood of the winner node at the kth learning step and $\Lambda_k(d(j,i))$ is the neighborhood function implementing different adaptation rates even within the same neighborhood. Also, the gradient information $\Delta\mathbf{w}_j(k),\ \Delta\sigma_j(k)$ for the weight and variance adaptation, was derived in Section 2 (i.e. equations 2, 3). Clearly, both parameters are defined in terms of the kernel distance metric.

The learning rates $\mu_w(k),\ \mu_\sigma(k),\ k \in N$ typically start from a value of 0.1 and decrease down to 0.02. These values are specified with the empirical criterion of having relatively fast convergence, without however sacrificing the stability of the map.

The KSDG-SOM starts with a much smaller size than a usual SOM. Therefore a large neighborhood is not required to train the whole map at the first learning steps (e.g. with 4 nodes initially at the map, a neighborhood of 1 only is required). As training proceeds, during subsequent training epochs, the area defined by the neighborhood becomes localized near the winning neuron, not by shrinking the vicinity radius (as in the standard SOM) but by enlarging the SOM with the dynamic growing.

The neighborhood function $\Lambda_k(d(j,i))$, can thus be defined with the following simple formula (the row and column of a node i are denoted by i_r, i_c respectively):

$$\Lambda_k(d(j,i)) = \begin{cases} 1 & \text{if } \mathrm{j = i} \\ \alpha,\ 0 < \alpha < 1, & \text{if } |\mathrm{i_r - j_r}| + |\mathrm{i_c - j_c}| = 1 \\ 0, & \text{otherwise} \end{cases}$$

3.5 Expansion Phase

This phase constitutes the main core of the learning algorithms. It is described in detail in section 4.

3.6 Fine Tuning Adaptation Phase

The fine tuning phase aims to optimize the final KSDG-SOM configuration. This phase is similar to the training run adaptation phase described previously (subsection 3.4) with two differences:

1. The final criterion for map convergence is more elaborated. We require much smaller change of the Total Growth Parameter for accepting the condition for map convergence.

2. The learning rate decreases to a smaller value in order to allow fine adjustments to the final structure of the map.

Typically, the *ConvergenceErrorThreshold* for the fine tuning phase is about 0.00001 and the learning rate is set to 0.01 (or to an even smaller value).

3.7 Evaluation of classification performances

To each KSDG-SOM node is assigned a *classification vector* **CL** with elements $CL_l = pr_l$, where pr_l is the ratio of the patterns with functional label l, among all the patterns mapped to the node. This vector is considered as the predicted classification of new patterns that are evaluated on a trained KSDG-SOM device.

This classification is a soft one: each CL_l expresses the probability that to a node (and consequently the mapped patterns) is assigned a label l. We therefore compute the performance based on a metric that we describe below. Specifically, for each class label l of each pattern i, a score $sc_{l,i}$ is assigned. This score equals to the classification vector element for class label l, CL_l, formely defined as $CL_l = pr_l$, if the corresponding label is included in the original class assignment (i.e. $c_{l,i} = 1$) and equals $qr_l = 1 - CL_l$ in the other case (i.e. $c_{l,i} = 0$). In essence the score $sc_{l,i}$ measures the *consensus of the functional labeling* for class l between the pattern i and the node. Clearly, by summing over all labels we characterize with the $\mathrm{TotalScore_i}$ parameter, the consensus of the functional labeling between the pattern i and the node it is mapped onto. This parameter is calculated as:

$$\mathrm{TotalScore_i} = \sum_{\mathrm{l=1}}^{\mathrm{N_c}} \mathrm{sc_{l,i}}, \text{ where } \mathrm{sc_{l,i}} = \begin{cases} CL_l & \text{if } \mathrm{c_{l,i} = 1} \\ 1 - CL_l & \text{if } \mathrm{c_{l,i} = 0} \end{cases}$$

Intuitively, a small $CL_l \approx 0$ for a class that does not appear as a functional label (i.e. $c_{l,i} = 0$) for an input pattern is much more a success than a failure, therefore being considered by a score $1 - CL_i \approx 1$.

The performance for each pattern i, $\mathrm{perf_i}$ is then obtained by dividing this score with the total number of functional class labelings, N_c, i.e.

$$\mathrm{Perf_i} = \frac{\mathrm{TotalScore_i}}{\mathrm{N_c}}$$

The global measure of the performance *ClassificationPerformance*(r_{su}) for a given ratio r_{su} (i.e. the supervision weighting parameter of equation 5), is obtained by averaging the $\mathrm{Perf_i}$ values for all the patterns of the testing set, TS i.e.

$$\mathrm{ClassificationPerformance(r_{su})} = \frac{\sum_{\mathrm{i \in TS}} \mathrm{Perf_i}}{|\mathrm{TS}|}$$

where the notation $|TS|$ denotes the number of elements of the testing set. The testing set was obtained by splitting the original data sets into ten approximately equal parts and using the *tenfold cross validation* methodology to evaluate the performance [23].

3.8 Model Selection Step

During this step a well performing ratio r_{su} is selected by using the following criteria:

- The classification performance has obtained significant increase for the selected r_{su} parameter value at the corresponding classification performance curve and this increase is followed by a plateau. The increase at the classification performance with increasing r_{su}, means that a priori information for the application was taken into account by the second term of Equation 5 that enforces class labeling uniformity at the formation of the cluster boundaries. The plateau that we require to follow the steep increase implies that increasing further the strength of the a priori information, although it can bias the model heavily towards an imperfect domain theory, does not offer significant improvements to its generalization potential.

- The number of the KSDG-SOM nodes that grow for the "optimal" r_{su} value should be relatively small (small model complexity). This criterion prefers the models with the smallest complexity that offer adequate generalization performance.

We should note that these selection criteria are somewhat heuristic. However, there seem to perform an adequate model selection.

3.9 Node Deletion

Nodes that are selected as winners for very few (usually one or two) training patterns, termed *uncolonized* nodes, are not deleted by our scheme although they probably correspond to noisy outliers. The patterns that consistently (three times or more) are mapped to uncolonized nodes are very unique and can either be artifacts or if not they have the potential to provide knowledge. Therefore they are amenable to further consideration. These patterns therefore are marked and isolated for further study. Nodes that are not selected as winners for any pattern are removed from the map in order to keep it compact.

4 The Expansion Process

The expansion is based on the detection of the neurons with large *Growth Parameter* (GP), referred to as the *unresolved neurons*. The node with the largest GP becomes the current focus of map expansion.

The Growth Parameter for node i, denoted GP_i, is based on an heterogeneous type of error, computed as:

$$GP_i = ALE_i + r_{su} \cdot \text{EntropyLike}_\text{i} \tag{10}$$

where we recall that the ALE denotes the Average Local Error.

We describe in turn the two components of 10, i.e. the *average local error* and the *entropy*.

A *local error* term is commonly used for implementing dynamically growing schemes [5]. A general assessment of the local error, le_i, is given by

$$le_i = \sum_{\mathbf{x} \in S_i} \text{Dist}(\Phi(\mathbf{x}), \Phi(\mathbf{w}_\text{i})) \tag{11}$$

where we denote by S_i the set of patterns $\mathbf{x}$ mapped to node i, $\mathbf{w}_i$ the weight vector of node i that corresponds to the average expression profile of S_i and the Dist operator denotes the corresponding distance metric.

However, the peculiarities of the data analysis application domain motivated two significant modifications to the classic local error measure. Specifically:

1. Instead of the simple local error measure of Equation 11 we use the *average local error* ALE_i per pattern, defined as:

$$ALE_i = \frac{le_i}{|S_i|} \tag{12}$$

   where $|S_i|$ denotes the number of elements of the set $|S_i|$.

   This measure does not increase when many similar patterns are mapped to the same node.

2. The second provision applies when we have class information available (either complete or partial) and we want to exploit it in order to improve the expansion. The local error that accumulates to a winner node is amplified by a factor that is inversely proportional to the square root of the frequency ratio r_c of its corresponding class c. Specifically, let $r_c = \frac{\#\text{patterns of class c}}{\#\text{total patterns}}$ be the frequency ratio of class c. Then the amplification factor is $r_c^{-\frac{1}{2}}$.

The supervised contribution to the heterogeneous error of Equation 5 is based on the computation of a parameter HN_i characterizing the *entropy of the class assignment content* of each node i. An advantage of the entropy is that it is relatively insensitive to the over-representation of classes. This means that independently of how many patterns of a class are mapped to the same node, if the node does not represent significantly other classes, its entropy is very small.

We first consider the simple case of each pattern belonging only to one functional class. The assignment of a class label to each neuron of the KSDG-SOM is in this case performed according to a majority-voting scheme [24]. The *entropy* parameter, that quantifies the uncertainty of the class label of neuron m, can be directly evaluated by counting the votes at each SOM neuron for every class as [25]:

$$HN(m) = -\sum_{k=1}^{N_c} p_k \cdot \log p_k \tag{13}$$

where N_c denotes the number of classes and $p_k = \frac{V_k}{V_{pattern}}$, is the ratio of votes V_k for class k to the total number of patterns $V_{pattern}$ that vote to neuron m. The parameter p_k has a probability interpretation, i.e. it is computed as the relative frequency of votes for each class k. For the single label case, the number of labeled patterns $V_{pattern}$ is also equal to the number of votes.

Clearly, the entropy $HN(m)$ is zero for unambiguous neurons and increases as the uncertainty about the class label of the neuron

m increases. The upper bound of $HN(m)$ is $\log(N_c)$, and corresponds to the situation where all the classes are equiprobable (i.e. the voting mechanism does not favor a particular class).

For the multi-label case, the voting scheme remains the same, but each pattern in this case can vote for more than one class. The frequent case of having patterns not assigned to any functional class is handled as a vote to the *Unassigned* class.

A quantity $HR(m)$ is defined similarly:

$$HR(m) = -\sum_{k=1}^{N_c} r_k \cdot \log r_k \tag{14}$$

The r_k do not correspond to probabilities but they are class voting ratios defined as the p_k (i.e. $r_k = \frac{V_k}{V_{pattern}}$). However, in this case $\sum_k V_k > V_{pattern}$ and therefore $\sum_{k=1}^{N_c} r_k > 1$. Thus, $HR(m)$ is not mathematically an entropy of a probability distribution. However, this quantity retains properties similar to the entropy.

We consider an example in order to explain the handling of multiple labeling by equation 14. Let $N_c = 3$ and suppose that 30 patterns are assigned to node m_1, all of them having as a label all the three classes and that 90 patterns are assigned to neuron m_2 one third of them having as a label class 1, another third class 2 and the last third class 3. Although in each case there are 30 patterns voting for each class, the quantity HR will be high in the case of the 90 patterns (i.e. log(3)) and zero at the other case. Thus, the HR measure has quantified effectively the similarity of multiple class labeling between patterns of some cluster.

4.1 Evaluation of the map training run convergence condition

The Total Growth Parameter "(TGP) is defined by summing the Growth Parameter (defined with equation 10) of all nodes, i.e.

$$TGP = \sum_{i \in \text{KSDGSOMnodes}} GP_i$$

.

The reduction of the *Total Growth Parameter* (TGP, defined in section 4) controls the training run convergence condition. The corresponding convergence test is:

$$\text{MapConverged} := \left(\frac{|\text{TGP}_\text{b} - \text{TGP}_\text{a}|}{\text{TGP}_\text{a}} < \text{ConvergenceErrorThreshold} \right)$$

where $TGP_b = \sum_i (GP_i)_b$ and $TGP_a = \sum_i (GP_i)_a$ denote respectively the sum of the Growth Parameters for all nodes before and after the presentation of patterns (i.e. one training epoch) and the *ConvergenceErrorThreshold* is a given value.

The above formula states that the map converges when the relative change of the TGP parameter between successive epochs drops below the threshold value. The setting of the *ConvergenceErrorThreshold* is somewhat empirical but a value in the range 0.01 - 0.02 performs well in assuming sufficient convergence without excessive computation.

5 Conclusions

This work has presented a new self-growing adaptive neural network model fitted to the requirements for clustering and classification of multi-labeled data. This model, called KSDG-SOM overcomes elegantly the main drawbacks of most of the existing clustering methods that impose an a priori specification at the number of clusters. The KSDG-SOM determines adaptively the number of clusters with a dynamic extension process which is able to exploit class information whenever available. It grows within a rectangular grid that provides the potential for the implementation of efficient training algorithms. The expansion of the KSDG-SOM is based on an adaptive process. This process grows nodes at the boundary nodes, ripples weights from the internal nodes towards the outer nodes of the grid, and inserts whole columns within the map. The growing algorithm is simple and computationally effective. It prefers to grow from the boundary nodes in order to minimize the map readjustment operations. However, a mechanism for whole column (row) insertion is implemented in order to deal with the case that a large map should be expanded around a point that is deep within its interior. The growing process determines automatically the appropriate level of expansion in order the similarity between the potential patterns of the same cluster to fulfill a designer definable statistical confidence level of not being a random event.

Multiple KSDG-SOM models are constructed dynamically each for a different unsupervised / supervised balance. Model selection criteria are used to select an KSDG-SOM model that optimizes the contribution of the unsupervised part of the patterns with the a priori knowledge (supervised part). The model selected on the basis of these criteria can revise better unreliable and incomplete functional labeling on the basis of unsupervised drives.

Acknowledgment

This work was partially supported from a European Union funded EPEAK II project "Arximidis", code 04-3-001/5, performed at the Technological Educational Institute of Kavalas, Dept. of Information Management, Greece.

References

[1] H. Liu, L. Wong, "Data Mining Tools for Biological Sequences", Journal of Bioinformatics and Computational Biology, Vol. 1, No. 1, p. 139-168, April 2003

[2] Eisen Michael B., Spellman Paul T., Patrick O. Brown, and David Botstein, "Cluster analysis and display of genome-wide expression patterns", Proc. Natl. Acad. Sci. USA, Vol. 95, pp. 14863-14868, December 1998

[3] Mavroudi Seferina, Papadimitriou Stergios, Bezerianos Anastasios, "Gene Expression Analysis with a Dynamically Extended Self-Organized Map that Exploits Class Information", Bioinformatics, Vol. 18, no 11, 2002, p 1446-1453

[4] Papadimitriou S., Mavroudi S., Vladutu L., Bezerianos A., "Ischemia Detection with a Self Organizing Map Supplemented by Supervised Learning", IEEE Trans. On Neural Networks, Vol. 12, No. 3, May 2001, p. 503-515

[5] Herrero Javier, Valencia Alfonso, and Dopazo Joaquin, "A hierarchical unsupervised growing neural network for clustering gene expression patterns", Bioinformatics, (2001) Vol. 17, no. 2, pp. 126-136

[6] Tamayo, P., Slonim, D., Mesirov, J., Zhu, Q., Kitareewan, S., Dmitrovsky, E., Lander, E.S. and Golub, T.R. (1999) "Interpreting patterns of gene expression with self-organizing maps: methods and application to hematopoietic differentiation", Proc. Natl. Acad. Sci., USA, 92, pp. 2907-2912

[7] Friedman, N., M. Linial, I. Nachman, and D'Peier, "Using Bayesian networks to analyze expression data", J. Comp. Bio. 7, 2000, 601-620,

[8] Fritzke Bernd, "Growing Grid - a self organizing network with constant neighborhood range and adaptation strength", Neural Processing Letters, Vol. 2, No. 5, pp. 9-13, 1995

[9] Van Hulle, N.M., "Kernel-Based Topographic Map Formation", Neural Computation, Vol. 14, No 7, p. 1560-1573, 2002

[10] Van Hulle, N.M., "Kernel-based equiprobabilistic topographic map formation, Neural Computation, Vol. 10, No. 7, p. 1847-1871, 2002

[11] Vesanto Juha Alhoniemi, Esa, "Clustering of the Self-Organized Map", IEEE Transactions on Neural Networks, Vol. 11, No. 3, May 2000, p. 586-600

[12] Si J., Lin S., Vuong M. A., "Dynamic topology representing networks", Neural Networks, 13, pp. 617-627, 2000

[13] Cheng Guojian and Zell Andreas, "Externally Growing Cell Structures for Data Evaluation of Chemical Gas Sensors", Neural Computing & Applications, 10, pp. 89-97, Springer-Verlag, 2001

[14] Campos Marcos M., Carpenter Gail A., "S-TREE: self-organizing trees for data clustering and online vector quantization", Neural Networks 14 (2001), pp. 505-525

[15] Van Hulle, N.M., "Joint Entropy Maximization in Kernel-Based Topographic Maps", Neural Computation, Vol. 14, No 8, p. 1887-1906, 2002

[16] James R. Williamson, "Self-Organization of Topographic Mixture Networks Using Attentional Feedback", Neural Computation 13:563-593, 2001

[17] Bernhard Scholkopf, Alexander J. Smola, "Learning with Kernels: Support Vector Machines, Regularization and Beyond", MIT Press 2002

[18] B. Scholkopf, A. J. Smola, R. C. Williamson, P. L. Bartlett, "New support vector algorithms", Neural Computation:1207-1245, 2000

[19] Janne Sinkkonen, Samuel Kaski, "Clustering Based on Conditional Distributions in an Auxiliary Space", Neural Computation, 14:217-239, 2001

[20] A. Sierra, F. Corbacho, "Reclassification as Supervised Clustering", Neural Computation 12:2537-2546, 2000

[21] Bishop, C. M., Svensen, M., Williams, C. K., "GTM: The generative topographic mapping", Neural Computation, 10:215-234, 1998

[22] Andras, P., "Kernel-Kohonen networks", International Journal of Neural Systems, Vol. 12, No. 2 (2002) 117-135

[23] Ian H. Witten, Eibe Frank, *Data Mining*, Morgan Kaufmann Publishers, 2000

[24] Kohonen T., Self-Organized Maps, Springer-Verlag, Second Edition,1997.

[25] Haykin S, Neural Networks, Prentice Hall International, Second Edition, 1999.

[26] Troyanskaya Olga, Cantor Michael, Shelock Gavin, Brown Pat, Hastie Trevor, Tibshirani Robert, Botstein David, Altman Russ B., "Missing value estimation methods for DNA microarrays", Bioinformatics, Vol. 17, no 6, 2001

[27] Lawrence Hunter, Ronald C. Taylor, Sonia M. Leach and Richard Simon, "GEST: a gene expression search tool based on a novel Bayesian similarity metric", Bioinformatics, Vol. 17, Suppl. 1, p. S115-S122