Proceedings of International Joint Conference on Neural Networks, Montreal, Canada, July 31 - August 4, 2005

An Incremental Growing Neural Gas Learns Topologies

Yann Prudent
University of Rouen
Mont Saint Aignan, CP 76821
E-mail: yann.prudent@univ-rouen.fr

Abdellatif Ennaji
University of Rouen
Mont Saint Aignan, CP 76821
E-mail: abdel.ennaji@univ-rouen.fr

Abstract— **An incremental and Growing network model is introduced which is able to learn the topological relations in a given set of input vectors by means of a simple Hebb-like learning rule. We propose a new algorithm for a SOM which can learn new input data (plasticity) without degrading the previously trained network and forgetting the old input data (stability). We report the validation of this model on experiments using a synthetic problem, the IRIS database and the handwriting digit recognition problem over a portion of the NIST database. Finally we show how to use this network for clustering and semi-supervised clustering.**

Key Words : incremental learning, SOM, semi-supervised clustering, clustering

I. Introduction

Data clustering is one of the most traditional and important issues in computer sciences. Cluster analysis aims at discovering groups and identifying meaningful distributions and patterns in data sets. Numerous clustering algorithms have been proposed in the literature, where four major categories of clustering algorithms are identified, namely hierarchical, partitional, density based, and artificial Neural Network techniques [JF99], [Jan02]. A hierarchical clustering algorithm yields a dendogram representing the nested grouping of patterns and similarity levels at which groupings change. The dendogramm can be broken at different levels to yield different clusterings of the data. Most hierarchical clustering algorithms are variants of the single-link [SS73], complete-link [Kin67], and minimum-variance [War63], [Mur84] algorithms.

A partionnal clustering algorithm obtains a single partition of the data instead of a clustering structure, such as the dendogram produced by a hierarchical technique. Partional methods have an advantages in applications involving lage data sets for wich the construction of a dendogram is computationally prohibitive. The most known partitional algorithm is the K-means [McQ67] which employ a squared error criterion.

A major drawback of such algorithms is the *a priori* choice of the number of desired output clusters.

Artificial neural networks (ANNs) have been used extensively over the past three decades for both classification and clustering. Competitive neural networks are often used to cluster input data. In competitive learning, similar patterns are grouped by the network and represented by a single unit (neuron). This grouping is done automatically based on data correlations. Well-known exemples of ANNs used for clustering include Kohonen's self-organizing map [Koh82] and adaptative resonance theory models (ART) [CG90]. The architectures of these ANNs are simple : they are single-layered. The weigth between the nodes are iteratively modified during learning, until a terminantion criterion is satisfied. Further, self-organizing maps and ART models are suitable for detecting only spherical clusters. And if the number of clusters is not given, the partition obtained is often a subpartition of the optimum one.

In recent years, due to emerging complex applications such as data and text mining, data clustering has attracted a new round of attention [DHS01]. One of the main challenges in the design of efficient and robust clustering algorithms is that, in many applications, new data sets are continuously added into an already huge database. One way to tackle this challenge is to introduce a clustering algorithm that operates incrementally. However, few works in the development of incremental clustering algorithms are related in the literature [Fri96], [REL99].

In this paper we will introduce a new Incremental ANN which give a structure for a semi-supervised clustering or a clustering technique that is suitable for detecting non spherical clusters. This ANN is a Topology based neural network such as the self-organizing map. Thus, an overview of the existing self-organizing maps is giving in the next section. Section 3 introduces the proposed new incremental ANN algorithm. Section 4 show how to use this ANN for semi-supervised or unsupervised learning. Then we will validate our ANN with several experiments in the section 5 and finally we will conclude.

II. An overview of Self-organizing Maps

We first would like to state the properties shared by all the models described below. The network structure is a graph consisting of a set of nodes (units or neurons) A and a set of edges N connecting the nodes. Each unit c has an associated position (or reference vector) w_c in the input space. Adaptation, during learning, of the reference vectors is done by moving the position of the nearest (or the "winning") unit (neuron) c_1 and its topological neighbors in the graph toward the input signal. For an input signal ξ the nearest unit c_1 is :

$$c_1 = \min_{c \in A} dist(\xi, w_c) \quad (1)$$

0-7803-9048-2/05/$20.00 ©2005 IEEE

and the position is updated as follows :

$$\Delta w_c = \epsilon(t) h_{c,c_1} \|\xi - w_c\| \quad (2)$$

where $\epsilon(t)$ is the adaptation step and h_{c,c_1} is a neighborhood function.

We can differentiate two kinds of SOM, Static Self-organizing Maps and Growing Self-organizing Maps. The Static SOMs have a pre-defined structure which is chosen *a priori* and does not change during the parameter adaptation. Growing SOMs, however, have no pre-defined structure; this is generated by successive additions of nodes and connections.

- **Static Self-organizing Maps**

The *Kohonen's Self-Organizing Feature Map* (SOFM) [Koh82] method considers a hyper-rectangular structure on the graph. Adaptation steps as described above are performed with:

$$\epsilon(t) h_{c,c_i} = \begin{cases} f(t) & \text{if } c \in N_{c_1}(t) \\ 0 & \text{otherwise} \end{cases}$$

where $0 \leq f(t) \leq 1$ is a decreasing monotonous function and:

$$N_{c_1}(t) = \{c \in A / \exists \text{ path between } c_1 \text{ and } c < R_{c_1}(t)\}$$

$R_{c_1}(t)$ is a decreasing integer function.

The decreasing behavior of both function f and neighborhood area of the winning unit during learning, makes it possible to roughly explore the input space at the beginning of the training process, and to carry out a refinement of the unit position in the final phase.

Neural Gas (NG) [MS91] is a pure vector quantization method which does not define any topology among the units. Rather, adaptations are done based on the distance computation in the input space.

The main principle of NG is to adapt the k nearest units for each input signal ξ. During the learning, k decreases from a large initial to a small final value.

This approach supposes a constant number of units, and it is necessary to predefine the total number of adaptation steps due to decreasing parameters during learning. Indeed, a large initial value of k causes adaptation of a large number of units. Then k is decreased up to a final value, namely value one, and so only the nearest center for each input signal is adapted.

However, the NG model may be combined with Competitive Hebbian Learning [Mar93] to build up a topology during self-organization. The principle of the Competitive Hebbian Learning (CHL) method is simply to create an edge between the winning and the second winning unit at each adaptation step. The graph generated is a subgraph of the Delaunay triangulation corresponding to the reference vectors. The NG/CHL combination has been called "topology-representing networks" [MS94].

- **Self-organizing Map generated by a constructive process**

The *Growing Cell Structures* (GCS) model [Fri94] has a structure consisting of *Hypertetrahedrons* with a dimensionality chosen in advance. A k-dimensional hypertetrahedron is a k-polyhedron having only $k+1$ vertices. For example, k is respectively 1, 2 and 3 for lines, triangles and tetrahedrons. The model is initialized with one hypertetrahedron. Adaptation steps, as described above, are performed with:

$$\epsilon(t) h_{c,c_i} = \begin{cases} \epsilon_b & \text{if } c = c_i \\ \epsilon_n & \text{if there is an edge between } c \text{ and } c_i \\ 0 & \text{otherwise} \end{cases}$$

At each adaptation step local error information is accumulated in the winning unit c_1.

$$\Delta E_{c_1} = \|w_{c_1} - \xi\|^2$$

After a given number λ of these adaptation steps, unit q with the maximum accumulated error is determined and a new unit is inserted by splitting the longest edge emanating from q. Moreover, additional edges are inserted in order to have a structure with hypertetrahedrons. The GCS model has a fixed dimensionality, so it carries out a dimensionality-reducing mapping from the input space into a k-dimensional space. This can be used to visualize data.

The *Growing Neural Gas* (GNG) model [Fri95b] does not impose any explicit constraints on the graph topology. The graph is generated and continuously updated by competitive Hebbian Learning [Mar93]. The adaptation of reference vectors is identical in GNG and GCS. After a fixed number λ of adaptation steps, unit q with the maximum error is determined and a new unit is inserted between q and its neighbor p in the graph that has the maximum error. Error variables of q and p are re-distributed with the new unit. The topology of a GNG network reflects the topology of the input data distribution and can have different dimensionalities in different parts of the input space. For this reason it is only possible to visualize low-dimensional input data.

The *Growing Grid* (GG) method [Fri95a] supposes a hyper-rectangular structure on the graph. Stated otherwise, the graph is a rectangular grid of a certain dimensionality k which is chosen *a priori*. The starting configuration is a k dimensional hypercube.

For example, a 2×2-grid for $k = 2$ and a $2 \times 2 \times 2$-grid for $k = 3$. To keep the integrity of this structure, it is necessary to always insert complete rows or columns. Except for its structure, the Growing Grid has the same algorithm as the Growing Cell Structures network. A new row or column is inserted at each λ adaptation step and the accumulated error determination is used to determine where to insert this row or column.

The *Growing Self-Organizing Map* (GSOM) [BV97] combines the reference vector adaptation of the SOFM with a constructive procedure for hypercubal output space. It starts from an initial 2-neuron configuration, learns according to the SOFM-algorithm, adds neurons to the output space until a specified maximum number of units is reached. GSOM can grow by adding nodes in one of the directions already spanned

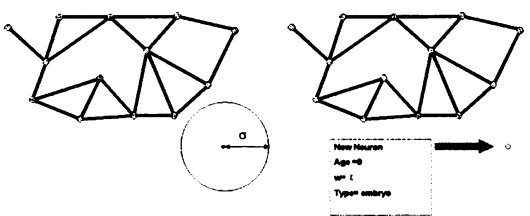
in the output space, or by adding a new dimension. It uses the back-propagation of the winning unit error along the different directions in order to determine how to to add nodes and in which direction.
Some of the neural network models presented in this section make it possible to learn new data dynamically, but do not guarantee the preservation (stability) of old knowledge as shown in the result section. It is the most important drawback when dealing with real and complex problems of data mining and knowledge discovery.

III. Incremental Growing Neural Gas

Our model is like the Growing Neural Gas model in that it does not impose any explicit constraint on the graph, which is generated and continuously updated by competitive Hebbian Learning; however unlike the networks described here, our approch distinguishes two kinds of neurons : *mature* neurons, and *embryo* neurons. In addition, connections or edges between neurons are created dynamically during learning following the principles of the CHL algorithm. And both neurons and edges are associated to an age which is set to zero when created and updated during learning. When a new neuron is inserted, it is an *embryo* neuron and its age is set to zero. Initially, the graph is empty. At each iteration, we search for the winning unit following (eq. 1). Then, as in the Adaptive Resonance Theory (ART), we perform the vigilance test of eq. 3 in order to decide if the winning unit is close enough to the input signal.

$$dist(\xi, w_c) \leq \sigma \tag{3}$$

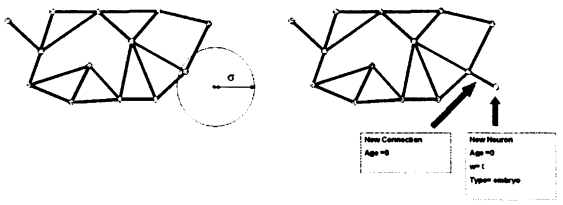
If the graph is empty, or the winning unit does not satisfy the vigilance test, we add a new *embryo* neuron with $w_{new} = \xi$ and the iteration is finished. This case is shown in figure 1. Figure 1(a) illustrates an IGNG network with new data to be learned (in dimension 2). This data is too far from the winning unit, so a new *embryo* neuron (fig 1(b)) is created. The reference vector of this new unit is the position of the data in the input space.



(a) A new training sample (b) Creation of a new neuron

Fig. 1. insertion of a new neuron when the new data is too far from its winning unit

If the vigilance test is satisfied for the winning unit, we apply it to the second-nearest unit. If there is only one unit in the graph, or if the test is not satisfied for the second nearest unit, a new *embryo* neuron is added with $w_{new} = \xi$. Figure 2 shows this case. In this figure the new input satisfies the vigilance test, but it is too far from the second-nearest neuron, so we create a new *embryo* neuron which is connected to the winning unit (fig 2(b)). The reference vector of the new unit is the position of the data in the input space.



(a) A new training sample (b) Creation of a new neuron

Fig. 2. insertion of a new neuron when the new data is too far from its second-winning unit

Finally, when the two nearest units satisfy the vigilance test, the reference vectors of the units are adapted (without unit creation) as in equation 2 with:

$$\epsilon(t)h_{c,c_i} = \begin{cases} \epsilon_b & \text{if } c = c_i \\ \epsilon_n & \text{if there is an edge between } c \text{ and } c_i \\ 0 & \text{otherwise} \end{cases}$$

The learning algorithm is then continued throughout the adaptation of both neurons and edges. If the connection between the two nearest units does not exist, an edge is created. Otherwise, the age of this edge is set to zero. On the other hand, the age of all the other edges connected to the winning unit is incremented. Afterward, we remove edges with an age higher than a_{max}, and if this leads to isolated *mature* neurons (without connections), we remove them as well. Then, we increment the age of all direct neighbor neurons of the winning unit. Consequently, if a given *embryo* neuron has an age higher than a_{mature}, this unit becomes a mature neuron. The final graph is only made up of mature neurons, as *embryo* neurons are useful only for the training.
This process is detailed in algorithm 1.

In order to avoid to give the parameter σ, it is initialized at the standard deviation of the database and is deacreasing during the learning process while the Calinski Harabasz's index of quality [CH74] ,which is given in eq.4, is increasing.

$$CHI = [B/(c-1)]/[W/(n-c)] \tag{4}$$

Where B denotes the sum of the squares of the distances between the neurons reference vector and the mean of all data, W stand for the sum of the squares of the distance between all data and the centroid of its wining unit, n is the total number of data and c is the total number of neurons.

IV. IGNG for semi-supervised

Let us consider that we have an huge base of unlabelled data, how to use our IGNG network to label this data ? First the

Algorithm 1: The incremental growing neural gas algorithm IGNG(age,sigma)

Data : **int** a_{mature}, **float** σ,**Training Database** S
Result : An IGNG network
begin
  while *a stopping criterion is not fulfilled* **do**
    Choose an input signal $\xi \in S$;
    Find the winning unit c_1;
    if *the graph is empty or* $dist(\xi, w_{c_1}) \geq \sigma$ **then**
      insert a new *embryo* neurons with $w_{new} = \xi$;
    else
      Find the second-nearest unit c_2;
      if *there is only one unit or* $dist(\xi, w_{c_2}) \geq \sigma$ **then**
        insert a new *embryo* neurons with $w_{new} = \xi$;
        create a connection between c_1 and ξ;
      else
        Increment the age of all edges emanating from c_1;
        $w_{c_1} += \epsilon_b(\xi - w_{c_1})$;
        $w_n += \epsilon_n(\xi - w_n)$; //($n$ are the directs neighbors of c_1)
        if c_1 *and* c_2 *are connected by an edge* **then**
          $age_{c_1 \to c_2} = 0$;
        else
          create a connection between c_1 and c_2;
        Increment the age of all direct neighbors of c_1;
        foreach *embryo neuron* c **do**
          if $age(c) \geq a_{mature}$ **then**
            c becomes a mature neuron
end

Algorithm 2: TrainIGNG()

Data : **Training Database** S
Result : An IGNG network
begin
  $\sigma = \sigma_S, x, y \in S$
  $IGNG(1, \sigma)$
  $old = 0$
  $calin = CHI$
  while $old - calin \leq 0$ **do**
    $\sigma = \sigma - \frac{\sigma}{10}$
    $IGNG(1, \sigma)$
    $old = calin$
    $calin = CHI$
end

Algorithm 3: semi-supervised labelling system algorithm

Data : **float** ρ **Training Database** S
Result : An IGNG network
begin
  Select $S_1 \in S$
  Set $S = S - S_1$
  TrainIGNG(S_1)
  Show the network to the user
  The user corrects the network
  The user labels the neurons of the nework
  while $S \neq \emptyset$ **do**
    Select $S_i \in S$
    Set $S = S - S_i$
    TrainIGNG(S_i)
    foreach $\xi \in S_i$ **do**
      if $dist(\xi, w_{c_1}) \leq \rho$ **then**
        Label ξ with the label of c_1;
    Show the network to the user with unlabelled data
    The user corrects the network
    The user labels the neurons of the nework
end

database have to be split . Then a database part is learned by an IGNG network. After, the result of this training is shown to the user (as we can see on the Fig.3). The user corrects the network by adding or deleting edges of the network. Afterward, the user label the data by giving a label to each disjoint subgraph. Then the learning process continue with another part of the database. After the learning, if the distance between an instance of the database and its winning unit is lower than a threshold (ρ), this instance is labelled with the label of its winning unit. The IGNG network is shown to the user without the labelled data. Then if it is needed he corrects the network. The process continue until all the data are labelled.

When the network is shown to the user (as we can see on the Fig.3), he can correct the IGNG topology by using three functions :

- removeEdge : The user selects an edge between two nodes that are not of the same class and delete it.
- addEdge : The user create an edge in order to link two disjoint subgraphs that are of the same class.
- changeNode : The user change the node of an instance when the class of the node is not the class of the instance.

When the clustering seems to be correct, the user gives a label to a node, and then the connexions spread this label to the other nodes of the same connected subgraphs. When all the nodes are labelled, all the instances are labelled.

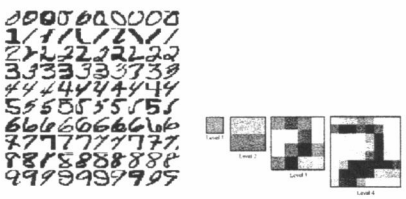
V. Experimental results

In this section, we will report some experimental results to demonstrate the general behavior and performances of the IGNG network. To visualize data, we use two synthetic

(a) data to be labelled (b) Visualization of the IGNG

Fig. 3. Visualization of an Incremental Growing Neural Gas

problems with a low-dimensional input data space. The first one has been proposed by Martinetz and Schulten [MS91] to demonstrate the non-incremental "Neural Gas" model and was also used by Fritzke [Fri95b] to validate the "Growing Neural Gas" model. We propose a second synthetic problem which allows us to show the limits of the GNG model in incremental learning. In order to validate our model for high-dimensional problems, we report some results obtained for the handwritten digit recognition over a subset of the NIST database 3. The feature vector considered is constitued of the 85 (1+4+16+64) gray levels of a 4-level-resolution pyramid [BB82]. Fig.4a gives an example of digits from the NIST database while Fig.4b shows an example (digit 2) of the representation retained.



(a) Figures of NIST database (b) Pyramid resolution of digit "2"

Fig. 4. Examples of digits in the Nist database and the retained vector representation

In this experiment, each neuron is labeled for a classification task using a simple majority vote rule of the labelled learning data that have activated each neuron during the training process. Two subsets from the NIST database of respectively 2626 digits for training and 2619 other digits for testing are used in this experiment.

1) IGNG vs GNG in offline learning:

Martinetz Distribution: The data distribution given in fig. 5 has been proposed by Martinetz and Schulten and used by Fritzke [Fri95b] to show that his model quickly learns the important and complex topological relations in this distribution by forming structures of different dimensionalities.

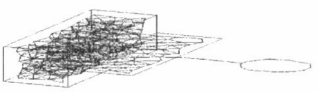


Fig. 5. Topological map structure obtained over Martinez distribution

As we can see, our approach performs well in this example. The topological structure of the data is well reproduced in the map topology. Neurons and connections retained in the network represent the real distribution of the learning data set.

Handwritten Digit Recognition: This experiment is done to compare our approach to the GNG and LVQ approaches for a classification problem. The three networks are compared using part of the NIST database described below. A learning cycle corresponds to a presentation to the network of all the training database samples. Table 1 shows that our approach

TABLE I

RECOGNITION RATE OF THE THREE NETWORKS

	Cycles	λ	ϵ_b	ϵ_n	Units	Recognition
LVQ	50	-	-	-	330	89.65%
GNG	50	400	0.1	0.006	330+0	91.44%
IGNG	**10**	-	0.01	0.002	313+9	**91.71**%

leads to faster learning (5 times faster) than the GNG network for a comparable accuracy, and with a fewer number of neurons. Note that in this experiment, 9 *embryo* neurons are generated during the learning process.

2) IGNG vs GNG in incremental learning:

Synthetic Problem: The synthetic problem we consider here is composed of two classes. The first class has a spherical distribution, while the second one has a cubic shape. The network was first trained with the spherical class. Then, to produce incremental learning, a data subset with only samples

from the second class was presented to the trained network. Fig.6 shows the result obtained with the GNG and our IGNG

(a) GNG (b) IGNG

Fig. 6. GNG (a) and IGNG (b) behavior in an incremental learning

models. On this figure, we can observe the degradation of the topology learned with the GNG model, reflecting a deterioration of the previously learned class. Such a phenomenon is avoided with our approach which allows a good behavior concerning the stability/plasticity dilemma in an incremental learning context.

Handwritten Digit Recognition: For the digit recognition problem, the portion of the NIST database used was split into four parts. Each model was then trained on one part after another. Results reported on table 2 confirm the previously observed behavior.

TABLE II

RECOGNITION RATE ON TEST DATASETS IN INCREMENTAL LEARNING

	Cycles	λ	ϵ_b	ϵ_n	Units	Recognition
GNG	50	400	0.1	0.006	328+0	81.29%
IGNG	**10**	-	0.01	0.002	244+16	**90.18%**

3) IGNG for semi-supervised clustering: As shown on the fig.3 user has only three egdes to delete in order to get a satisfting clustering. Then he can label the four obtained clusters. For this experiment the feature space used was the one obtained by a 4-level-resolution pyramid [BB82].

In the Iris database(UCI repository), with only one intervention of the user, the IGNG has found 3 clusters with 3.37% of misclassification and a standard deviation of 0.75%. We can compare this model with the result of well known algorithms such as EM(9.3333% of incorrectly clustered instances) or Kmeans (11,33% of incorrectly clustered instances).

VI. CONCLUSION

The "Incremental Growing Neural Gas" network presented in this paper is able to make explicit the important topological relations in a given distribution $P(\xi)$ of input signals. An advantage over other classical models is the stability of our network in incremental learning. Experiments carried out for low and high-dimensional problems demonstrate this stability. These results support the conclusion that this contribution can be considered as a response of the well known plasticity/stability dilemma in machine learning field.

Possible applications of our model are data visualization (only in low-dimensional input space), clustering ,semi-supervised and supervised learning (in the same way that LVQ).

Another promising way and future developments of this approach concern its combination with several supervised classifiers allowing the design of an incremental supervised decision system. Good and promising results have been already obtained in [PE05] with a multi-classifiers system which is managed by an IGNG network.

REFERENCES

[BB82] D.H. Ballard and C. M. Brown. *Computer Vision*. Prentice Hall, 1982.

[BV97] H. U. Bauer and T. Villmann. Growing an hypercubal output space in a self-organizing feature map. *IEE TNN*, (8):218–226, 1997.

[CG90] G. Carpenter and S. Grossberg. Art3: Hierarchical search using chemical transmitters in self-organizing pattern recognition architectures. *Neural Networks*, (3):129–152, 1990.

[CH74] T. Calinski and J. Harabasz. A dendrite method for cluster analysis. *Communications in Statistic*, 3:1–27, 1974.

[DHS01] P. O. Duda, P. E. Hart, and D.G. Storck. Pattern classification. New York, 2001.

[Fri94] B. Fritzke. Growing cell structures — A self-organizing network for unsupervised and supervised learning. *Neural Networks*, 7(9):1441–1460, 1994.

[Fri95a] B. Fritzke. Growing grid - a self organizing network with constant neighborhood range and adaptation strength. *Neural Processing Letter*, (2):9–13, 1995.

[Fri95b] B. Fritzke. A growing neural gas network learns topologies. In *Advances in Neural Information Processing Systems 7*, pages 625–632. 1995.

[Fri96] B. Fritzke. Growing self-organizing networks - why ? In *ESANN*, pages 61–72, 1996.

[Jan02] M.F. Janowitz. A combinatorial introduction to cluster analysis. Technical report, Classification Society of North America, 2002.

[JF99] N. Jain, A. K.and Murty and P. J. Flyn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[Kin67] B King. Step-wise clustering procedures. *J. Am. Stat. Assoc.*, 1967.

[Koh82] T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43:59–69, 1982.

[Mar93] T. Martinetz. Competitive Hebbian learning rule forms perfectly topology preserving maps. In *Proc. ICANN'93*, pages 427–434. Springer, 1993.

[McQ67] J. McQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and probability*, 1967.

[MS91] T. Martinetz and K. Schulten. A "Neural-Gas" network learns topologies. In *Proc. ICANN*, volume I, pages 397–402, Amsterdam, Netherlands, 1991.

[MS94] T. Martinetz and K. Schulten. Topology representing networks. *Neural Networks*, 7(2), 1994.

[Mur84] F. Murtagh. A survey of recent advances in hierachical clustering algorithms which use cluster centers. *Comp. J.*, 1984.

[PE05] Y. Prudent and A. Ennaji. A k nearest classifier design. *Electronic Letter on Computer Vision and Image Analysis*, 2005.

[REL99] A. Ribert, A. Ennaji, and Y. Lecourtier. An incremental hierarchichal clustering. In *Proc. of Vision Interface Conf*, pages 586–591, 1999.

[SS73] P. H. A. Sneat and R. R. Sokal. Numerical taxonomy. *freeman*, 1973.

[War63] J. H. Jr Ward. Hierarchical grouping to optimize an objective function. *J. Am .Stat. Assoc*, 1963.