Review of Hierarchical Models for Data Clustering and Visualization

Lola Vicente & Alfredo Vellido

Grup de Soft Computing
Secció d'Intel·ligència Artificial
Departament de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya (UPC)
Jordi Girona, 1-3. E-08034 Barcelona
{mvicente, avellido}@lsi.upc.es

Abstract. Real data often show some level of hierarchical structure and its complexity is likely to be underrepresented by a single low-dimensional visualization plot. Hierarchical models organize the data visualization at different levels, and their ultimate goal is displaying a representation of the entire data set at the top-level, perhaps revealing the presence of clusters, while allowing the lower levels of the hierarchy to display representations of the internal structure within each of the clusters found, providing the definition of lower level sets of sub-clusters which might not be apparent in the higher-level representation. Several unsupervised hierarchical models are reviewed, divided into two main categories: *Heuristic Hierarchical Models*, with a focus on Self-Organizing Maps, and *Probabilistic Hierarchical Models*, mainly based on Gaussian Mixture Models.

1 Introduction

The structural complexity of high-dimensional datasets can hardly be captured by means of single, low-dimensional representations, and information which is structured at different representation levels is likely to escape them. Often, real-world datasets involve some level of hierarchical structure.

Hierarchical models organize the data visualization at different levels, and their ultimate goal is displaying a representation of the entire data set at the top-level, perhaps revealing the presence of clusters, while allowing the lower levels of the hierarchy to display representations of the internal structure within each of the clusters found, providing the definition of lower level sets of sub-clusters which might not be apparent in the higher-level representation. The definition of a hierarchy will allow the analyst to drill down the data in order to discover patterns that might be hidden to other, more simple models (See example in figure 1).

The construction of a hierarchy, in all the models hereupon reviewed, is carried out in a top-down fashion, in a procedure known as *divisive* hierarchical clustering. This review separates the unsupervised models under consideration into two categories: *Heuristic Hierarchical Models* and *Probabilistic Hierarchical Models*. Heuristic Hierarchical Models focus on variations on the well known *Self-Organizing Map* (SOM, [1, 2]). This model has been widely used over the last twenty-odd years due to its powerful visualization properties, and despite some of its intrinsic limitations. Probabilistic Hierarchical Models are based on density estimations: *Hierarchical Mixture of Latent Variable Models* and *Hierarchical Generative Topographic Model*, will be reviewed.

Approaches to hierarchical data visualization incorporating SOM entail a "hard" data partition, while probabilistic models allow a "soft" partitioning in which, at any level of hierarchy, data points can effectively belong to more than one model.

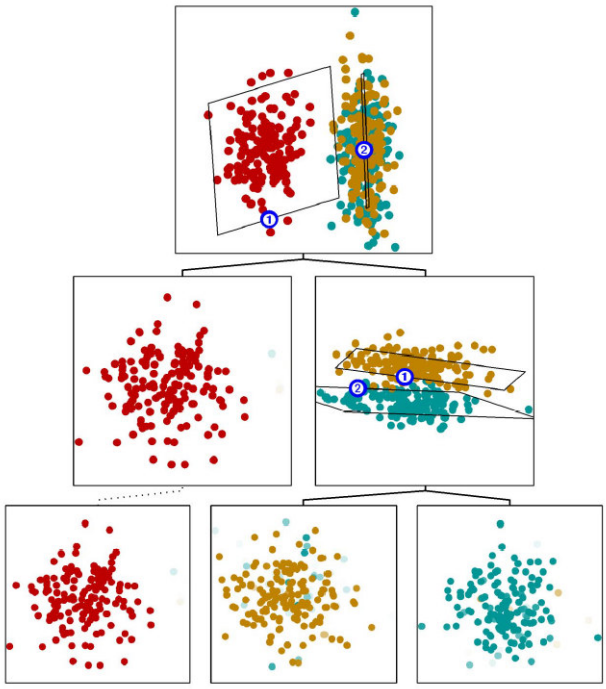

Fig. 1. An example of a hierarchical model, where more details on the structure of the data are revealed in each level, from Bishop & Tipping [3]

2 Heuristic Hierarchical models based on the SOM

SOM is an unsupervised, neural network-inspired model for clustering and data visualization, in which the prototypes[1] are encouraged to reside in a one- or two-dimensional manifold in the feature space. The resulting manifold is also referred to as a *constrained topological map*, since the original high-dimensional observations are mapped down onto a fixed, ordered, discrete grid on a coordinate system.

Kohonen's [1, 2] SOM is an unsupervised neural network providing a mapping from a high-dimensional input space to a one- or two-dimensional output space while preserving topological relations as faithfully as possible.

The SOM consists of a set of units arranged usually arranged in a 1- or 2-dimensional grid with a weight vector $m_i \in \Re^n$ attached to each unit. The basic training algorithm can be summarized as follows:

- Observations from the high-dimensional input space, referred to as input vectors $x \in \Re^n$, are presented to the SOM and the activation of each unit for the presented input vector is calculated, usually resorting to an activation function based on the distance between the weight vector of the unit and the input vector.

[1] Prototype methods represent the training data by a set of points in the feature space. These prototypes are typically not examples from the training sample.

- The weight vector of the unit showing the highest activation (smallest distance) is selected as the "winner" and is modified as to more closely resemble the presented input vector by means of a reorientation of the weight vector towards the direction of the input vector, weighted by a time-decreasing learning rate α .
- Furthermore, the weight vectors of units in the neighborhood of the winner are modified accordingly (to a lesser extent than the winner) as described by a time-decreasing neighborhood function.
- This learning procedure finally leads to a topologically ordered mapping of the presented input signals. Similar input data are mapped onto neighboring regions on the map.

Several developments on the basic algorithm have addressed the issue of adaptive SOM structures; amongst them: *Dynamic Self-Organizing Maps* [4], *Incremental Grid Growing* [5], or *Growing Grid* [6], where new units are added to map areas where the data are not represented at a satisfying degree of granularity.

As mentioned in the introduction, hierarchical models can provide more information from a data set. SOM has been developed in several ways in order to set it within hierarchical frameworks, which are commonplace as part of more standard statistical clustering procedures.

2.1 Hierarchical Feature Map

The key idea of hierarchical feature maps as proposed in [7] is to use a hierarchical setup of multiple layers where each layer consists of a number of independent SOMs. One SOM is used at the root or first layer of the hierarchy. For every unit in this map a SOM is created in the next layer of the hierarchy. This procedure is repeated in further layers of the hierarchical feature map. A 3-layered example is provided in figure 2. The first layer map consists of 2x2 units, thus we find four independent self-organizing maps on the second layer. Since each map on the second layer consists again of 2x2 units, there are 16 maps on the third layer.

The training process of hierarchical feature maps starts with the root SOM on the first layer. This map undergoes standard training. When this first SOM becomes stable, i.e. only minor further adaptation of the weight vectors occurs, training proceeds with the maps in the second layer. Here, each map is trained with only that portion of the input data that is mapped on the respective unit in the higher layer map. This way, the amount of training data for a particular SOM is reduced on the way down the hierarchy. Additionally, the vectors representing the input patterns may be shortened on the transition from one layer to the next, due to the fact that some input vector components can be expected to be equal among those input data that are mapped onto the same unit. These equal components may be omitted for training the next layer maps without loss of information.

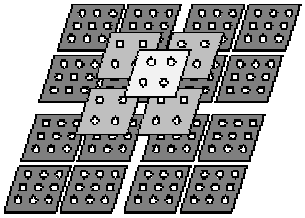

Fig. 2. Architecture of a three-layer hierarchical feature map, from Merkl [18]

Hierarchical feature maps have two benefits over SOM which make this model particularly attractive:

- First, hierarchical feature maps entail substantially shorter training times than the standard SOMs. The reason for that is twofold: On the one hand, there is the input vector dimension reduction on the transition of one layer to the next. Shorter input vectors lead directly to reduced training times. On the other hand, the SOM training is performed faster because the spatial relation of different areas of the input space is maintained by means of the network architecture rather than by means of the training process.
- Second, hierarchical feature maps may be used to produce fairly isolated, i.e. disjoint, clusters of the input data than can be gradually refined when moving down along the hierarchy. In its basic form, the SOM struggles to produce isolated clusters. The separation of data items is a rather tricky task that requires some insight into the structure of the input data. Metaphorically speaking, the standard SOM can be used to produce general *maps* of the input data, whereas hierarchical feature maps produce an *atlas* of the input data. The standard SOM provides the user with a snapshot of the data; as long as the map is not too large, this may be sufficient. As the maps grow larger, however, they have the tendency of providing too little orientation for the user. In such a case, hierarchical feature maps are advisable as models for data representation.

2.2 Hierarchical SOM

The *Hierarchical SOM* (HSOM) model usually refers to a tree of maps, the root of which acts as a preprocessor for subsequent layers. As the hierarchy is traversed upwards, the information becomes more and more abstract. Hierarchical self-organizing networks were first proposed by Luttrell [9]. He pointed out that although the addition of extra layers might yield a higher distortion in data reconstruction, it might also effectively reduce the complexity of the task. A further advantage is that different kinds of representations would be available from different levels of the hierarchy. A multilayer HSOM for clustering was introduced by Lampinen and Oja [10]. In the HSOM, the best matching unit (BMU) of an input vector **x** is sought from the first-layer map and its index is given as input to the second-layer map. If more than one data vector concurs within the same unit of the first layer map, the whole data histogram can be given to the second layer instead of a single index. This approach has been applied to document database management [11].

HSOM consists of a number of maps organized in a pyramidal structure, such as that displayed in figure 3. Note that there is a strict hierarchy and neighborhood relation implied with this architecture. The size of the pyramid, i.e. the number of levels as well as the size of the maps at each level, has to be decided upon in advance, meaning there is no dynamic growth of new maps based on the training process itself. However, since the training of the pyramid is performed one level at a time, it is theoretically possible to add a further level if required. Furthermore, note that, usually, the number of nodes at the higher levels is small as compared to other SOM models using multiple maps.

During the training process, the input vectors that are passed down in the hierarchy are compressed: if certain vector entries of all input signals that are mapped onto the same node show no or little variance, they are deemed not to contain any additional information for the subordinate map and thus are not required for training the corresponding sub-tree of the hierarchy. This leads to the definition of different weight vectors for each map, created dynamically as the training proceeds.

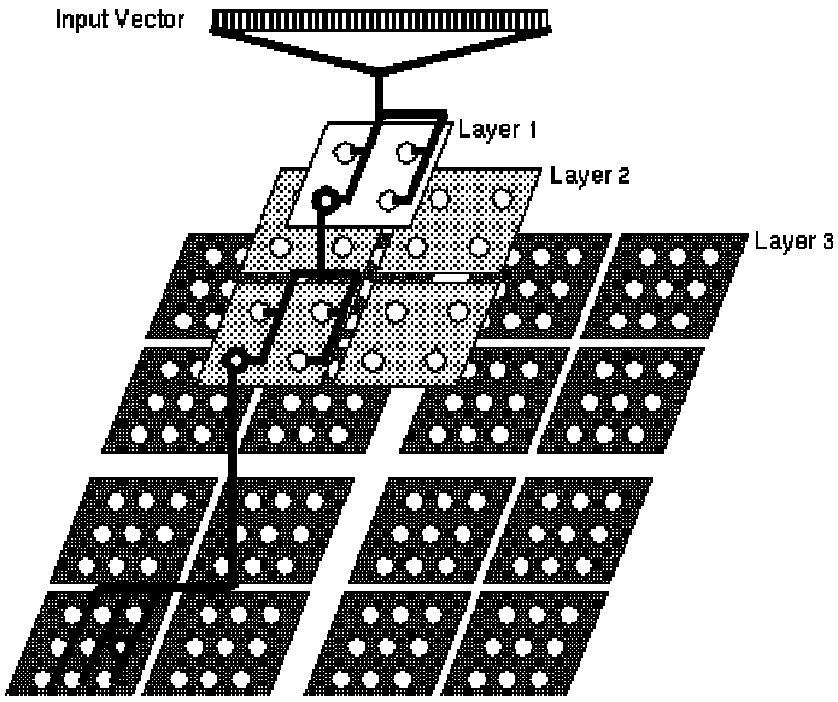

Fig. 3. Hierarchical SOM Architecture: 3 layers with 2x2 (layers 1 and 2) and 3x3 (layer 3) from Koikkalainen & Oja [12]

2.3 Growing Hierarchical SOM

The *Growing Hierarchical Self-organizing Map* [13, 14] (GHSOM) is proposed as an extension to the SOM [1, 2] and HSOM [9] with these two issues in mind:

1. SOM has a fixed network architecture i.e. the number of units to use as well as the layout of the units has to be determined before training.
2. Input data that are hierarchical in nature should be represented in a hierarchical manner for clarity of representation.

GHSOM uses a hierarchical structure of multiple layers where each layer consists of a number of independent SOMs. Only one SOM is used at the first layer of the hierarchy. For every unit in this map a SOM might be added to the next layer of the hierarchy. This principle is repeated with the third and any further layers of the GHSOM.

In order to avoid SOM fixed size in terms of the number of units an incrementally growing version of SOM is used, similar to the *Growing Grid*.

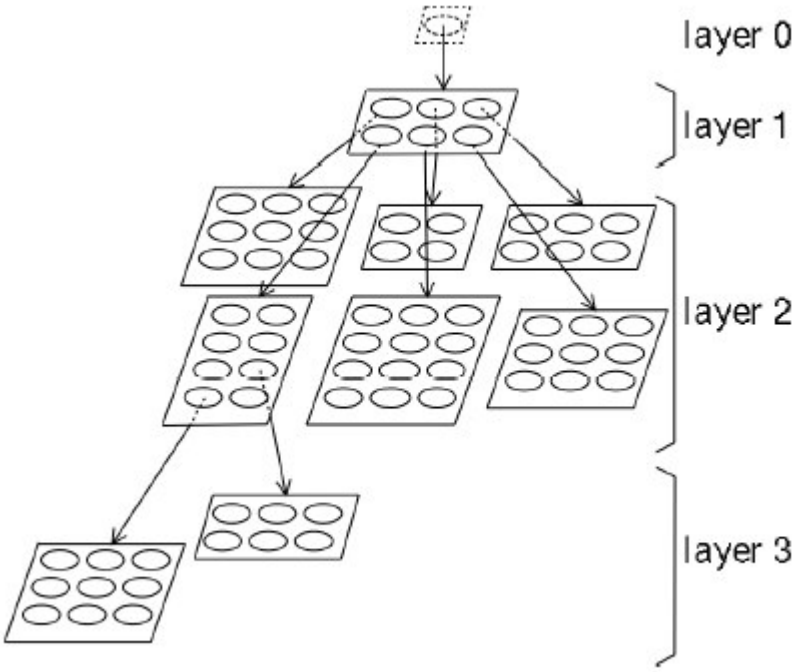

Fig. 4. GHSOM reflecting the hierarchical structure of the input data, from Dittenbach, Merkl & Rauber [13]

The GHSOM will grow in two dimensions: in width (by increasing the size of each SOM) and in depth (by increasing the levels of the hierarchy).

For growing in width, each SOM will attempt to modify its layout and increase its total number of units systematically so that each unit is not covering too large an input space. The training proceeds as follows:

1. The weights of each unit are initialized with random values.
2. The standard SOM training algorithm is applied.
3. The unit with the largest deviation between its weight vector and the input vectors that represents is chosen as the *error unit.*
4. A row or a column is inserted between the *error unit* and the most dissimilar neighbour unit in terms of input space
5. Steps 2-4 are repeated until the mean quantization error (MQE) reaches a given threshold, a fraction of the average quantification error of unit *i*, in the proceeding layer of the hierarachy.

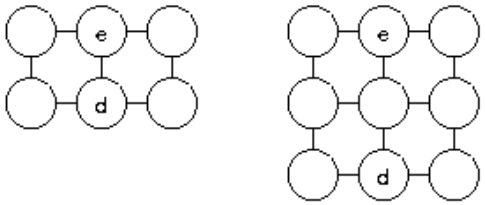

Fig. 5. Inserting a row in SOM, from Dittenbach, Merkl & Rauber [13]

The picture on the left of figure 5 is the SOM layout before insertion. “e” is the error unit and “d” is the most dissimilar neighbor. The picture on the right shows the SOM layout after inserting a row between “e” and “d”.

As for deepening the hierarchy of the GHSOM, the general idea is to keep checking whether the lowest level SOMs have achieved enough coverage for the underlying input data.

The details are as follows:

1. Check the average quantification error of each unit to ensure it is above certain given threshold: it indicates the desired granularity level of a data representation as a fraction of the initial quantization error at layer 0
2. Assign a SOM layer to each unit with an average quantification error greater than the given threshold, and train SOM with input vectors mapped to this unit.

GHSOM provides a convenient way to self organize inherently hierarchical data into layers and it gives users the ability to choose the granularity of the representation at the different levels of the hierarchy. Moreover, the GHSOM algorithm will automatically determine the architecture of the SOMs at different levels. This is an improvement over the Growing Grid as well as HSOM.

The drawbacks of this model include the strong dependency of the results on a number of parameters that are not automatically tuned. High thresholds usually result in a flat GHSOM with large individual SOMs, whereas low thresholds result in a deep hierarchy with small maps.

3 Probabilistic Hierarchical models

Probabilistic models offer a consistent framework to deal with problems that entail uncertainty. When probability theory lays at the foundation of a learning algorithm, the risk that the reasoning performed in it be inconsistent in some cases is lessened ([15, 16]) Next, we present several hierarchical models developed within a probabilistic framework. The presentation of each model is preceded by a brief summary of the theory laying its foundations.

3.1 Gaussian Mixture Modeling

The Gaussian Mixture Model (GMM) is based on density estimation. It is a *semi-parametric* estimation method since it defines a very general class of functional forms for the density model where the number of adaptive parameters can be increased in a systematic way (by adding more components to the model) so that the model can be made arbitrarily flexible.

In a mixture model, a probability density function is expressed as a linear combination of basis functions. A model with *M* components is written in the form

$$p(x) = \sum_{j=1}^{M} P(j)\, p(x \mid j) \ , \qquad (\mathbf{1})$$

where the parameters $P(j)$ are called the mixing coefficients and the parameters of the component density functions $p(x \mid j)$ typically vary with *j*. To be a valid probability density, a function must be non-negative throughout its domain and integrate to 1 over the whole space. Constraining the mixing coefficients

$$\sum_{j=1}^{M} P(j) = 1 \tag{2}$$

$$0 \le P(j) \le 1 \ , \tag{3}$$

and choosing normalized density functions

$$\int p(x \mid j) dx = 1 \tag{4}$$

guarantees that the model does represent a density function.

The mixture model is a *generative* model and it is useful to consider the process of generating samples from the density it represents, as they can be considered as *representatives* of the observed data. First, one of the components *j* is chosen at random with probability $P(j)$; thus we can view $P(j)$ as the *prior* probability of the *j*th component. Then a data point is generated from the corresponding density $p(x \mid j)$. The corresponding posterior probabilities can be written, using Bayes' theorem, in the form

$$P(j \mid x) = \frac{p(x \mid j) P(j)}{p(x)} \tag{5}$$

where p(x) is given by (1). These posterior probabilities satisfy the constraints

$$\sum_{j=1}^{M} P(j \mid x) = 1, \ 0 \le P(j \mid x) \le 1 \tag{6}$$

It only remains to decide on the form of the component densities. These could be Gaussian distributions with a different form of covariance matrix.

- **Spherical** The covariance matrix is a scalar multiple of the identity matrix,

$$\sum\nolimits_j = \sigma_j^2 I \ \text{ so that } \ p(x \mid j) = \frac{1}{(2\pi\sigma_j^2)^{d/2}} \exp\left\{ -\frac{\left\| x - \mu_j \right\|^2}{2\sigma_j^2} \right\} \tag{7}$$

- **Diagonal** The covariance matrix is diagonal $\sum\nolimits_j = diag(\sigma_{j,1}^2, ..., \sigma_{j,d}^2)$ and the density function is

$$p(x \mid j) = \frac{1}{(2\pi \prod_{i=1}^{d} \sigma_{j,i}^2)^{d/2}} \exp\left\{ -\sum_{i=1}^{d} \frac{(x_i - mu_{j,i)}{}^2}{2\sigma_{j,i}^2} \right\} \tag{8}$$

- **Full** The covariance matrix is allowed to be any positive definite $d \times d$ matrix $\sum\nolimits_j$ and the density function is

$$p(x \mid j) = \frac{1}{2\pi^{d/2} \left| \sum\nolimits_j \right|^{1/2}} \exp\left\{ -\frac{1}{2} (x - \mu_j)^T \Sigma^{-1} (x - \mu_j) \right\} \tag{9}$$

Each of these models is a universal approximator, in that they can model any density function arbitrarily closely, provided that they contain enough components. Usually a mixture model with full covariance matrices will need fewer components to model a given density, but each component will have more adjustable parameters.

The method for determining the parameters of a Gaussian mixture model from a data set is based on the maximization a data likelihood function. It is usually convenient to recast the problem in the equivalent form of minimizing the negative log likelihood of the data set

$$E = -L = -\sum_{n=1}^{N} \log p(x^n), \tag{10}$$

which is treated as an error function. Because the likelihood is a differentiable function of the parameters, it is possible to use a general purpose non-linear optimizer such as the expectation-maximization (E-M) algorithm [17]. It is usually faster to converge than general purpose algorithms, and it is particularly suitable to deal with incomplete data.

3.2 Hierarchical Mixture Models

Bishop and Tipping [3] introduced the concept of hierarchical visualization for probabilistic PCA. By considering a probabilistic mixture of latent variable models we obtain a "soft" partition of the data set at the top level of the hierarchy into "clusters", corresponding to the second level of the hierarchy. Subsequent levels, obtained using nested mixture representations, provide increasingly refined models of the data set. The construction of the hierarchical tree proceeds top-down, and can be driven interactively by the user. At each stage of the algorithm the relevant model parameters are determined using the expectation-maximization (E-M) algorithm [17].

The density model for a mixture of latent variable models takes the form:

$$p(t) = \sum_{i=1}^{M_0} \pi_i p(t \mid i) \tag{11}$$

where M_0 is the number of components of the mixture, and the parameters π_i are the mixing coefficients, or prior probabilities, corresponding to the mixture components $p(t \mid i)$. Each component is an independent latent variable model with parameters μ_i, W_i and σ_i^2.

The hierarchical mixture model is a two-level structure consisting of a single latent variable model at the top level and a mixture of M_0 such models at the second level. The hierarchy can be extended to a third level by associating a group G_i of latent variable models with each model *i* in the second level. The corresponding probability density can be written in the form

$$p(t) = \sum_{i=1}^{M_0} \pi_i \sum_{j \in G_i} \pi_{i|i} p(t \mid i, j) \tag{12}$$

where the $p(t \mid i, j)$ again represent independent latent variable models, and the π_{jli} correspond to sets of mixing coefficients, one for each *i*, satisfying $\sum_j \pi_{jli} = 1$. Thus, each level of the hierarchy corresponds to a generative model, with lower levels giving more refined and detailed representations.

Determination of the parameters of the models at the third level can again be viewed as a missing data problem in which the missing information corresponds to labels specifying which model generated each data point.

3.3 Generative Topographic Mapping

The aim of the *Generative Topographic Mapping* (GTM, [18]), a probabilistic alternative to the SOM, that also resorts to Bayesian statistics, is to allow a non-linear transformation from latent space to data space but keeping the model computationally tractable. In this approach, the data is modeled by a mixture of Gaussians (although alternative distributions can be used), in which centers of the Gaussians are constrained to lie on a lower dimensional manifold. The *topographic* nature of the mapping comes about because the kernel centers in the data space preserve the structure of the latent space. By adequate selection of the form of the non-linear mapping, it is possible to train the model using a generalization of the EM algorithm.

The GTM provides a well-defined objective function (something that the SOM lacks) and its optimisation, using either non-linear standard techniques or the EM-algorithm, has been proved to converge. As part of this process, the calculation of the GTM learning parameters is grounded in a sound theoretical basis. Bayesian theory can be used in the GTM to calculate a posterior probability of each point in latent space being responsible for each point in data space, instead of the SOM sharp map unit membership attribution for each data point.

The GTM belongs to a family of latent space models that model a probability distribution in the (observable) data space by means of latent, or hidden variables. The latent space is used to visualize the data, and is usually a discrete square grid on the two-dimensional Euclidean space. GTM creates a generative probabilistic model in the data space by placing a radially symmetric Gaussian with zero mean and inverse variance.

3.4 Hierarchical GTM

The probabilistic definition of the GTM allows its extension to a hierarchal setting in a straightforward and principled way [19]. The Hierarchical GTM (HGTM) models the whole data set at the top level, and then breaks it down into clusters at deeper levels of the hierarchy. The hierarchy is defined as follows:

- HGTM arranges a set of GTMs and their corresponding plots in a tree structure **T**.
- The *Root* is considered to be at level 1, i.e. *Level*(Root) = 1. *Children* of a model N with *Level*(N) = i are at level i + 1, i.e. Level(M) = i + 1, for all M ∈ *Children*(N).
- Each model M in the hierarchy, except for the *Root*, has an associated parent-conditional mixture coefficient: a prior distribution defined as: p (M|*Parent*(M)).

  The priors are non-negative and satisfy the consistency condition:

$$\sum_{M \in Children(N)} p(M \mid N) = 1 \qquad (13)$$

- Unconditional priors for the models are recursively calculated as follows:

p (Root) = 1, and for other models

$$p(\mathrm{M}) = \prod_{i=2}^{Level(M)} p(Path(M)_i \mid Path(M)_{i-1}) \quad (14)$$

where Path(M)=(*Root*,…,M) is the N-tuple (N=*Level*(M)) of nodes defining the path in T from *Root* to M.

- The distribution associated to the hierarchical model is a mixture of leaf models,

$$p(t \mid T) = \sum_{M \in Leaves(T)} p(M)\, p(t \mid M) \quad (15)$$

The training of the HGTM is straightforward and proceeds in a recursive fashion (top-down):

1. A root GTM is trained and used to generate an overall visualization of the data set.
2. The user identifies regions of interest on the visualization map.
3. These regions of interest are transformed into the data space and form the basis for building a collection of new, child GTMs.
4. The EM algorithm works with responsibilities (posterior probabilities of unit membership given data observations) moderated by the parent-conditional prior previously described.
5. After assessing the lower level visualization maps, the user may decide to proceed further and model in greater detail some specific portions of these.

An automated initialization, resorting to Minimum Description Length (MDL) principles, can be implemented to choose the number and location of sub-models.

4 Conclusions

In this brief paper, we have reviewed a number of recent advances on the development of unsupervised hierarchical models for data visualization and clustering. The setting of data exploration elements, such as clustering and visualization, into a hierarchical framework augments the amount of information about a data set that models manage to convey. Most real-world problems entail complex data sets that seldom provide enough information in a single snapshot, and interactive hierarchical methods are more likely to provide an adequate insight into the fine details of the structure of data patterns.

Two sub-groups of models have been considered: *Heuristic Hierarchical Models* and *Probabilistic Hierarchical Models.* Many advantages can be expected from the definition of data analysis models according to principled probabilistic theory, amongst them the possibility of developing these models in a coherent way.

References

1. Kohonen, T.: *Self-Organizing Maps*. Berlin: Springer Verlag (1995)
2. Kohonen, T.: Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43(1), (1982) 59-69
3. Bishop, C.M. & Tipping, M.E.: A hierarchical latent variable model for data visualization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3), (1998) 281-293
4. Alahakoon, D., Halgamuge, S.K. & Srinivasan, B.: Dynamic self-organizing maps with controlled growth for knowledge discovery. *IEEE Transactions Neural Networks*, 11(3), (2000) 601-614

5. Blackmore, J. & Miikkulainen, R.: Incremental grid growing: Encoding high-dimensional structure into a two-dimensional feature map. In *Proceedings of the International Conference on Neural Networks (ICANN'93)*. San Francisco, CA, (1993) 450-455
6. Fritzke, B.: Growing grid – a self-organizing network with constant neighborhood range and adaptation strength. *Neural Processing Letters*, 2(5), (1995) 1-5
7. Miikkulainen, R.: Script recognition with hierarchical feature maps. In *Connection Science*, 2, (1990) 83-101
8. Merkl, D.: Exploration of text collections with hierarchical feature maps. *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'97)*, Philadelphia, USA (1997)
9. Luttrell, S. P.: Hierarchical self-organizing networks. In *Proceedings of the International Conference on Neural Networks (ICANN'89)*. London, U.K (1989) 2-6
10. Lampinen, J. & Oja, E.: Clustering properties of hierarchical self-organizing maps. *Journal of Mathematical Imaging and Vision*, 2, (1992) 261-272
11. Kohonen, T., Kaski, S., Lagus, K. & Honkela, T.: Very large two-level SOM for the browsing of newsgroups. In *Proceedings of the International Conference on Neural Networks (ICANN'96)*, Bochum, Germany (1996) 269-274
12. Koikkalainen, P. & Oja, E.: Self-organizing hierarchical feature maps. In *Proceedings of the International Joint Conference on Neural Networks*. San Diego, California, U.S.A. 2, (1990) 279-284
13. Dittenbach, M, Merkl, D. & Rauber, A.: The growing hierarchical self-organizing map. In *Proceedingsof the International Joint Conference on Neural Networks. (IJCNN 2000)*, Como, Italy 6, (2000) 15-19
14. Dittenbach, M, Rauber, A. & Merkl, D.: Uncovering the hierarchical structure in data using the growing hierarchical self-organizing map. *Neurocomputing* 48(1–4), (2002) 199–216.
15. Jaynes, E.: *Probability Theory: The Logic of Science*. Cambridge University Press (2003)
16. Cerquides, J.: Improving Bayesian Network Classifiers. PhD Thesis. U.P.C. Barcelona. Spain (2003)
17. Dempster, A.P., Laird, N.M., & Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, B* ,39, (1977) 1-38
18. Bishop, C.M., Svensén, M., & Williams, C.K.I.: GTM: the generative topographic mapping. *Neural Computing*, 10, (1998) 215-234
19. Tiňo, P. & Nabney, I.: Hierarchical GTM: constructing localized non-linear projection manifolds in a principled way. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5), (2002) 639-656.