Visualizing High-Dimensional Structure with the Incremental Grid Growing Neural Network

Justine Blackmore and Risto Miikkulainen
Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712
justine,risto@cs.utexas.edu

Abstract

Understanding high-dimensional real world data usually requires learning the structure of the data space. The structure may contain high-dimensional clusters that are related in complex ways. Methods such as merge clustering and self-organizing maps are designed to aid the visualization and interpretation of such data. However, these methods often fail to capture critical structural properties of the input. Although self-organizing maps capture high-dimensional topology, they do not represent cluster boundaries or discontinuities. Merge clustering extracts clusters, but it does not capture local or global topology. This paper proposes an algorithm that combines the topology-preserving characteristics of self-organizing maps with a flexible, adaptive structure that learns the cluster boundaries in the data.

1 INTRODUCTION

Real world data is often very high-dimensional, and often has a structure that is difficult both to recognize and describe. For instance, human blood can be tested for the presence or absence of hundreds of inherited traits such as blood types, HLA factors, proteins, and DNA markers (Cavalli-Sforza et al. 1994). Similarity between blood samples is an indication of genetic similarity between individuals. On a larger scale, the structure of a set of samples from populations around the world reflects the global organization of human genetics. Learning the structure of such a data set would yield knowledge of how human populations are related.

Because the complicated relationships in real world data are difficult to perceive, tools for visualizing high-dimensional data are crucial in discovering patterns in the data. Visualization involves mapping an unknown high-dimensional space (the data set) onto a drawable structure. Current approaches to visualization of arbitrary data focus either on extracting and representing the global topology of the space, or on extracting information about clusters in the data.

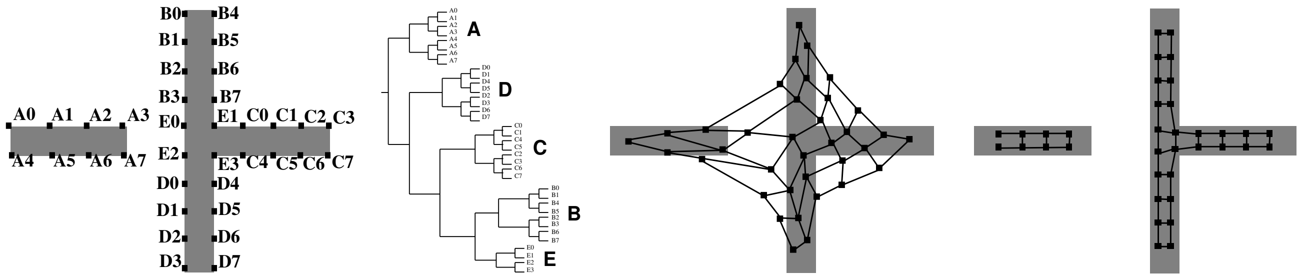
The merge clustering algorithm, for example, represents cluster properties of the data in a 2-dimensional merge tree, such that leaves and branches under each merge node have variances that are closer to each other than to the variances of any other cluster in the tree. The high-dimensional topology, however, is lost or unrecognizable (figure 1b). There may be important relationships between branches that are not represented because of the fixed connectivity of the tree.

On the other hand, methods that focus on dimension reduction, such as principal component analysis, multidimensional scaling, and the self-organizing map (Kohonen 1989; Kohonen 1990), do not represent cluster boundaries explicitly in a 2-D structure. For example, the self-organizing map maps high-dimensional data onto a fixed network of nodes. Vectors nearby in the high-dimensional input space are mapped onto nearby nodes of the network. The network structure preserves the topology of the data set as much as possible. However, the network cannot learn or represent discontinuities in the data due to its full connectivity (figure 1c).

Self-organizing map algorithms have recently been developed that incrementally grow and prune a network (Fritzke 1991a, 1991b, 1992, Jockusch 1990; Kangas et al. 1990; Martinetz and Schulten 1991; Ritter 1991; Rodriques and Almeida 1990; Xu and Oja 1990). These methods employ heuristics to ensure that nodes are added only where the network needs them to represent the input space, and that nodes are deleted only when they do not represent any part of the input space. Most of the resulting networks, however, have arbitrary dimensionality and connectivity. There is no guarantee the final network structure can be easily visualized in two dimensions.

This paper presents an incremental self-organizing algorithm called Incremental Grid Growing (IGG) that captures both complicated topology and high-

In: A. Prieditis and S. Russell (eds.) *Machine Learning: Proceedings of the 12th International Conference*, 1995.



(a) (b) (c) (d)

Figure 1: **Capturing topology and cluster boundaries in a 2-D input space.** (a) The input space consists of 36 two-dimensional vectors uniformly distributed in the shaded regions. The vectors were labeled A through E according to the region and numbered. A represents the separated cluster, B, C, and D the connected arms, and E the area connecting B, C, and D. (b) The merge clustering representation. The tree structure has captured the clusters in the data, but the overall topology of the data set is not apparent. (c) The self-organizing map representation. The black dots indicate locations of the network's weight vectors after the map has been organized. Lines indicate network connectivity. Although the map has captured the overall 2-D topology, its connections span the cluster boundaries. Note also that nodes have been allocated to areas where there is no input, representing the structure of the data set inaccurately. (d) A more desirable 2-D network visualization that can be obtained with incremental grid growing. This network has 4 clusters: 3 are connected through a small number of nodes, and the fourth is separate from the others.

dimensional cluster boundaries. The network is strictly 2-dimensional, but incrementally adapts its shape and connectivity to the structure of the data set. The local and global structure of the input is automatically embedded in the 2-D drawing (figure 1d).

The organization of the paper is as follows. After a brief discussion of the self-organizing map, a detailed description of the incremental grid growing algorithm is presented. IGG is compared to other visualization methods in a minimum spanning tree task. IGG is then demonstrated in a large, real world semantic data set. Finally, future possibilities for speeding up the algorithm, tuning the parameters automatically, and applying IGG to visualizing population genetics are discussed.

2 INCREMENTAL GRID GROWING

Following a brief review of the standard self-organizing map algorithm, the incremental grid growing approach is presented in this section. IGG incorporates the standard self-organizing process, but new techniques have been added that allow the network structure to be dynamic and adaptive.

2.1 STANDARD SELF-ORGANIZING MAPS

The self-organizing map (Kohonen 1989; Kohonen 1990) represents the topology of an N-dimensional input space in a network of nodes, which is usually a regular 2-D grid. Each node in the network has an N-dimensional input weight vector. Vectors from the input space are presented to the network and mapped onto the node whose weight vector most closely resembles the input. The weight vector is then modified so that its distance to the current input is reduced by a fixed fraction. Similarly, the weight vectors of the neighboring nodes in the network are modified, although by a smaller fraction. The size of the modification neighborhood is reduced as organization progresses. In this way, the network learns to organize its nodes to resemble closely the topology of the input space. Note, however, that there is no opportunity for the network to represent any cluster boundaries that exist in the data. The neighborhood relations are set a priori and remain fixed throughout the learning.

2.2 THE IGG ALGORITHM

The incremental grid growing algorithm is designed to overcome the limitation of the fixed grid in self-organizing maps. IGG embeds the cluster boundaries directly in its 2-D network structure. IGG builds the network incrementally, dynamically adapting its structure and connectivity according to the input data.

Initially, the grid consists of four connected nodes with weight vectors chosen at random from the input space (figure 2a). The following three steps are then iterated:

1. Adapting the current grid to the input distribution through the self-organizing map process;
2. Adding new nodes to the perimeter of the grid where the network is exhibiting a large errors in representation (figures 2a-d); and
3. Examining the vectors of neighboring nodes to de-

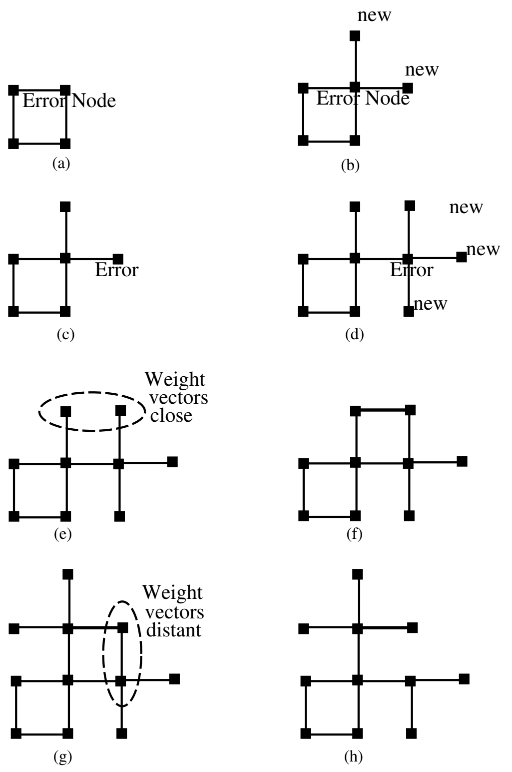


Figure 2: **Growing the map grid.** Figure (a) shows the initial structure after the first organization stage; the boundary node with the highest error value is marked. (b) New nodes are "grown" into any open grid location that is an immediate neighbor of the error node. (c) After organizing the new structure with the standard self-organization process, a new error node is found. (d) Again, new nodes are grown into any open grid location that is an immediate neighbor of the error node. (e) During self-organization of this new structure, the algorithm detects that the circled nodes have developed weight vectors very close in Euclidean distance. (f) These "close" nodes are connected. (g) After further organization, the algorithm discovers connected neighboring nodes whose weight vectors occupy distant areas of the input (i.e. the nodes have a large Euclidean distance). (h) These "distant" nodes are then disconnected in the grid.

termine whether a connection between the nodes should be deleted from the map, or a new connection added (figures 2e-h).

The core of the learning of IGG, therefore, consists of the self-organizing map algorithm. The additional techniques of growing new nodes and adding and deleting connections allow the network to evolve a 2-D structure that reflects the relationships in the data. These techniques are described in detail below.

2.2.1 Growing New Nodes

A boundary node is defined as any node in the grid that has at least one directly neighboring position in the 2-D grid space not yet occupied by a node. Each boundary node in the current network maintains an error value E over each organizational pass. Whenever an input vector is mapped onto a boundary node, the square of the distance between the input vector and the node's weight vector is added to the error value:

$$E(t) = E(t-1) + \sum_k (x_k - w_k)^2, \qquad (1)$$

where E is the cumulative error, $\mathbf{w}$ the weight vector of the winning unit, and $\mathbf{x}$ is the input vector.

Large cumulative error values occur at nodes that have too many input vectors mapped onto them. Their weight vectors fail to adequately represent all of the input vectors in that area. Therefore, new nodes are added to the grid in the areas that have high cumulative error. New nodes are only added on the boundary, so that the structure remains 2-D and drawable at all times. During self-organization, these new nodes on the perimeter develop weight vectors for those areas of the input space that were previously inadequately represented.

The new nodes are directly connected to the error node. If any other directly neighboring grid spots are occupied (as in figure 2d), the new node's weight vector is initialized to be the average value of all the neighboring weight vectors:

$$w_{\mathrm{NEW},k} = 1/n \sum_{i \in N} w_{i,k}, \qquad (2)$$

where $w_{\mathrm{NEW},k}$ is the kth component of the new unit's weight vector and N is the set of the n neighboring nodes of the new unit. Otherwise (as in figure 2b), the new node's weight vector is initialized so that the weight vector of the error node is the average of the new node's vector and the vectors of any already existing neighbors of the error node:

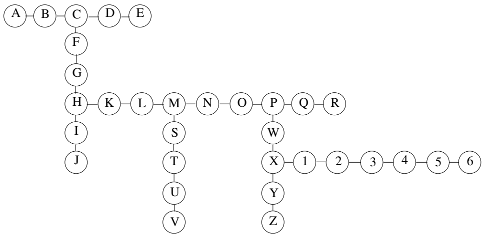
$$w_{\mathrm{ERR},k} = 1/(m+1)\left(w_{\mathrm{NEW},k} + \sum_{i \in M} w_{i,k}\right), \qquad (3)$$

where $w_{\mathrm{ERR},k}$ is the kth component of the error node's weight vector and M is the set of the m already existing neighbor units of the error node.

Because new nodes are added only to areas that need them, each node in the structure always represents some region of the input space that lies inside a cluster. Therefore, node deletion is not necessary in incremental grid growing. Also, because the algorithm is incremental and locally adaptive, it is unlikely to settle into a local minimum that would distort global organization, such as a twist in the map.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	1	2	3	4	5	6
1	2	3	4	5	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
0	0	0	0	0	1	2	3	4	5	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
0	0	0	0	0	0	0	0	0	0	1	2	3	4	5	6	7	8	3	3	3	3	6	6	6	6	6	6	6	6	6	6
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2	3	4	1	2	3	4	2	2	2	2	2	2
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2	3	4	5	6

(a)



(b)

Figure 3: **(a) The input vectors of Kohonen (1990). (b) The minimal spanning tree of the data.** The example was designed to illustrate the self-organizing map's capacity to represent the general topology of hierarchical data. In this case, the minimal spanning tree is one relational description that happens to capture the structure well.

2.2.2 Adding and Deleting Connections

Initially, the new nodes are connected to the structure only through the high error node. During organization, the new weight vectors may become similar to the weight vectors of neighbors to which they are not connected. In this case, a new connection is added joining these nodes. An adjustable threshold parameter is used to decide if such a new connection should be grown. After each organizational pass, the similarity metric (e.g. Euclidean distance) between unconnected neighboring nodes is examined. If the value is below the "connect" threshold, a connection is added (figures 2e-f). Similarly, a "disconnect" threshold is used to determine if there are two nodes in the map that are connected although they have evolved into separated areas of the input space. Exceeding such a threshold may indicate that a connection crosses a cluster boundary, and should be deleted from the grid (figures 2g-h).

Adding nodes only at the perimeter ensures that the map remains drawable at all times. The dynamic addition and deletion of connections allows the grid to learn high-dimensional cluster boundaries in the input, avoiding the limitation of standard self-organizing maps. As a result, the grid-growing algorithm is a flexible, dynamic tool for discovering the structure of complicated high-dimensional data.

3 COMPARISON USING MINIMUM SPANNING TREE DATA

To illustrate how the grid-growing algorithm differs from the standard visualization methods such as merge clustering and self-organizing maps, consider the minimum spanning tree example of Kohonen (1990). In this example, the input consists of the 5-dimensional vectors listed in figure 3a. A superficial look at the input indicates that there are clusters of similar vectors in the data, as well as topological relationships between them. The high-dimensional topology of this data set is difficult to describe, but a minimum spanning tree is one 2-D structure that in this particular case captures the hierarchical nature of the data quite well.

Conventional 2-D visualizations fail to represent the essential properties of the spanning tree. When merge clustering is applied to this data set (figure 4a), clustering is apparent in the resulting tree. However, the merge tree does not make the global and local relationships between elements clear. On the other hand, the hexagonally connected self-organizing map learned both the global and local topology of the data; however, it has no way of representing the cluster boundaries (figure 4b). Both representations are therefore incomplete.

The incremental grid growing algorithm applied to the same data set results in a structure that captures both the cluster boundaries and the topology of the data (figure 4c). The arms of the spanning tree are clustered in delineated regions of the map. Also, the relationships between the clusters are narrowly specified by the limited connectivity between clusters. In other words, the algorithm has evolved a network that visualizes the underlying structure in the data.

4 DEMONSTRATION USING REAL WORLD SEMANTIC DATA

The spanning tree example is a good illustration of the properties of visualization techniques, but it is still a toy problem. Real world data is usually more challenging to visualize in 2-D. Each vector may contain hundreds of features, and the data set may contain thousands of such vectors. Very complex relationships between data elements are possible. In addition, real world data sets often contain a significant amount of noise. Minimum spanning trees (and other such simple structures) are usually insufficient to represent the complicated relationships in unknown, high-dimensional data.

Consider for example word semantics. The variety

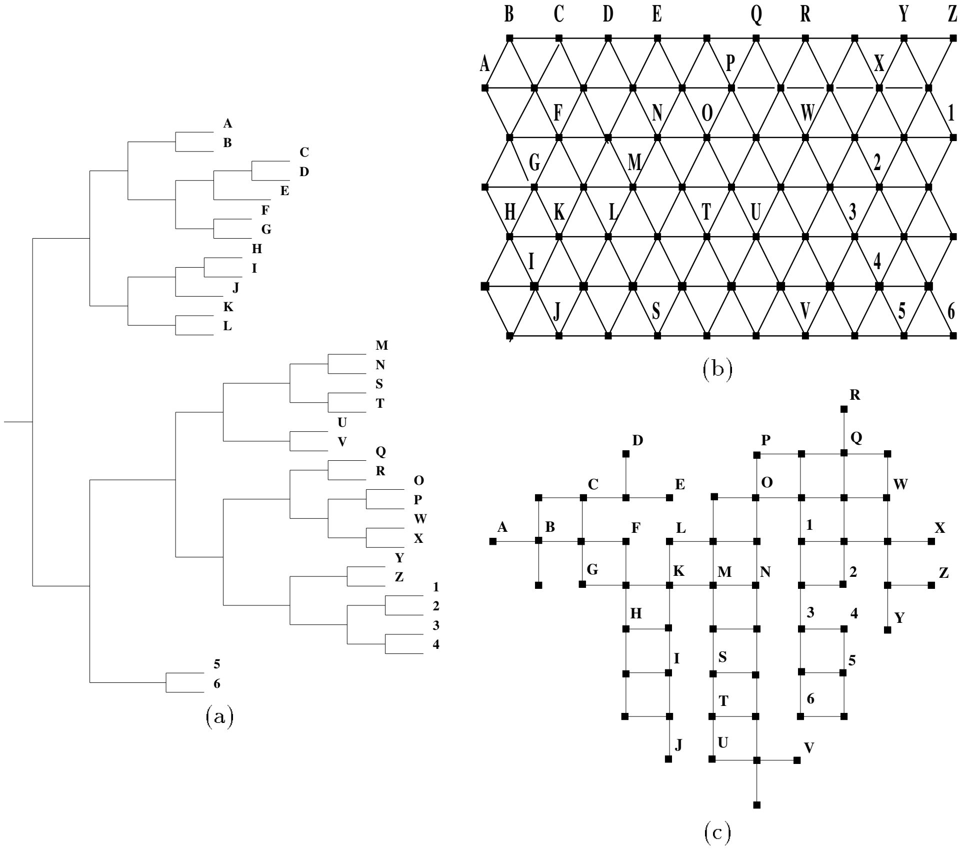


Figure 4: **2-D representations of the spanning tree data.** (a) The merge tree derived by the merge clustering algorithm. Cluster boundaries are apparent in the structure, but the global and local topology is not. (b) Map derived by the standard self-organizing algorithm (Kohonen 1990). The map is hexagonally connected. The spanning tree structure is clearly present in the map; however, the full connectivity makes it difficult to extract exact neighborhood relations between units. (c) Map derived by the incremental grid growing algorithm. The limited connectivity between clusters in the map closely resembles the structure of the spanning tree. This map was produced using 4 organizational phases for each incremental structure. In phase 1, the best matching unit was moved toward the input with an epsilon of 0.18 (epsbmu); the neighbors were moved by 0.13 (epsngb); the structure was trained for 25 epochs (nepochs). In phase 2, epsbmu = 0.14, epsngb = 0.1, nepochs = 15. In phase 3, epsbmu = 0.09, epsngb = 0.06, nepochs = 15. In phase 4, epsbmu = 0.05, epsngb = 0.02, nepochs = 15. The neighborhood size was a function of the number of nodes in current structure. When there were as many or fewer than 36 nodes in the map, the neighborhood size began at 1 at phase 1 and decreased to 0 by phase 4. If there were greater than 36 nodes, the neighborhood began at 2 and decreased to 0. Connections were added or deleted after phase 4: a connection was added between 2 nodes if their distance was less than 4.9 times the average distance in the structure (tconnect). A connection was deleted if the distance between the two nodes was more than 3.9 times the average distance in the map (tdelete). The parameter settings were determined experimentally, and have thus far served as good starting points for all other IGG experiments.

and complexity of relationships between word meanings overwhelms simple representation schemes. New learning and representation methods are often demonstrated using such semantic data (Miikkulainen 1993; Ritter and Kohonen 1989; Schütze 1993; Scholtes 1993).

The demonstrate visualization with IGG, a high-dimensional semantic data set was compiled from the "Webster" online thesaurus. In this thesaurus, relationships among words are represented explicitly as lists, where each list indicates a different type of relationship. However, only the most direct relations between words are immediately apparent to the user. Less obvious relationships require a search through all the intervening lists. In an effort to visualize the global structure of such semantic data, IGG was applied to a subset of the thesaurus words.

In the online thesaurus, words appear in alphabetical order. Each word is accompanied by its part of speech, definition and any idiomatic expressions it is used in, along with its cross reference lists. The lists include synonyms, related words, contrasted words, and antonyms. In order to apply IGG, feature vector representations for the words were created based on the cross-reference lists. The features are words, and each main entry in the thesaurus is represented by a feature vector indicating the presence or absence of each of the words in its lists. A value of 1.0 for a feature indicates that that word (feature) is present in the related list; a value of -1.0 indicates that the feature appears in the constrasted or antonym lists. A value of 0.0 indicates that the two words are not directly related by their lists.

For brevity, the example described here uses only verb entries from the thesaurus. Synonyms have been collapsed into single entries and single features. Also, only those verbs that appear most often are represented in the input set, and only those verbs that are used most often in the cross-reference lists are used as features. The resulting data set contains 197 verbs, each represented by 240 features. Because the data source is obtained from the real world, and because a very simple method for reducing its size was employed, the data set is very noisy, and therefore rather difficult to describe and visualize.

The final grid derived by IGG is shown in figure 5. Semantic similarities have been captured in the 2-D topology of the grid to the extent possible with this data. Semantic dissimilarities are implicit in the map boundaries. Note that although the vector representation scheme is very simple, the data is not synthetic or contrived. The vectors were generated from the straight text of the thesaurus, and the full data set reduced only for brevity. The grid has captured the relationships to the extent they exist in the reduced set, and has placed the noise words in areas where there are similar vectors.

In the top left arm of the map, verbs that connote a retarding function are clustered (e.g. hamper, ward, suppress, stultify). This arm merges into into the top central arm, which contains verbs suggesting deterioration (deteriorate, weaken, depreciate). This arm joins an area that contains words implying lessening (decrease, wither). This area in turn blends into the top right arm, which contains words connoting a decreasing or removal function (fall, droop, slip, retire). The bottom two arms contain words that have more positive connotations. The bottom right contains words suggesting increase (skyrocket, raise, increase). This area merges into an area connoting improvement and promotion (improve, speed, help). To the left, this area blends with an arm connoting accumulation or gathering (absorb, group, heap, accumulate). In the central area connecting the more positive arms with the more negative arms, there is a gradation of meaning from bottom to top: urge, incite, offend, attack, overwhelm, shatter, maim, revolt, demur. The gathering region also merges into the negative region: handle, keep, monopolize, dictate, demand, overwhelm.

Two other arms are present in the grid, containing words that do not support the general positive-negative gradation. On the left, words that imply giving (distribute, grant) blend with words that suggest giving in (lose, fawn). These in turn merge into the more negative areas of the grid (spain, yield, decay). On the right, words that connote success and encouragement (coax, succeed, reach, push) blend with words connoting support (base, correct, stabilize). This area in turn gradually blends with the positive region (incite, urge, assert).

The thesaurus data demonstrates the ability of IGG to represent complicated high-dimensional data, which is often both incomplete and noisy. To the extent possible, the network has learned the structure present in the input. The results support the idea that IGG can be used as a visualization tool for very complicated, unknown data.

5 DISCUSSION AND FUTURE WORK

Because of its adaptive and flexible nature, the incremental grid-growing algorithm is a potentially powerful tool for visualizing the structure of unknown high-dimensional data. An important question to consider is whether it will scale up to larger tasks. Because the central organizing step is based on the standard self-organizing algorithm, which spends most of its time computing the Euclidean distance between every network node and every input vector for tens of epochs, the algorithm's time complexity can be combinatorial in the number of inputs, number of dimensions, and network size.

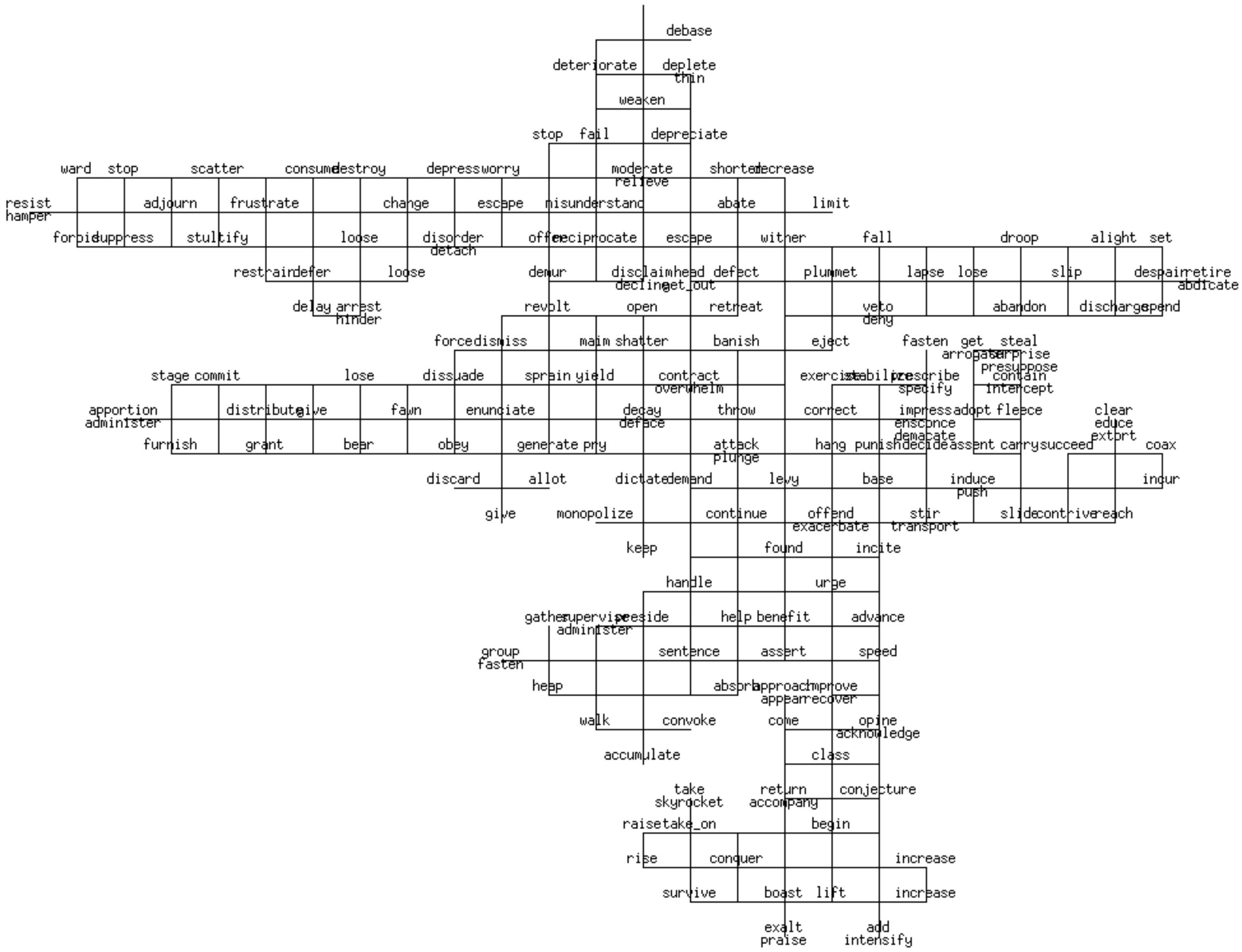


Figure 5: **The reduced Webster's thesaurus verb set.** The full verb set contains 1477 1477-dimensional vectors; here it has been reduced to 197 240-dimensional vectors for clarity. In the online thesaurus text, only direct relations between words are explicitly represented, and looser relationships must be deduced by chasing word entries. The grid encodes semantic similarities as well as semantic boundaries. The simulation parameters were otherwise the same is in the minimum spanning tree example (figure 4c), except nepochs for the 4 phases were 10, 6, 6, and 4, and tconnect was 2.8 and tdelete = 3.3.

Little can be done about the properties of the input, but the running time can be reduced to linear in the network size by making the search for the closest weight vector for each more local. By searching only a small neighborhood around the node that was closest to the current input in the previous epoch, the search time becomes constant for all epochs, regardless of the current network size. This optimization is possible because the network fully learns the input space during each organizational phase, making only a local search necessary. As an example, in our implementation, we saw a speedup factor of 7 when the search was localized to a 1×1 neighborhood of the map. (The computation time on the reduced verbs set went from approximately 2 hours on an IBM RS6000 to 20 minutes). This is a promising result, and indicates that building very large maps should be tractable.

Future work on the algorithm will focus on fine-tuning the implementation for efficiency. Like any neural net algorithm, incremental grid growing would benefit from a method for setting all organization and growth parameters automatically. Because the algorithm is incremental, it may accomodate dynamic parameter tuning based on statistical properties of the intermediate structures. Such a technique would add another dimension of flexibility to the algorithm, and is thus a promising research direction. Another challenge facing all SOM algorithms is the lack of an objective criterion for determining how good the map is. It might be possible to incorporate the stress metric used in multidimensional scaling analysis into a SOM evaluation measure. Additionally, future work will focus on ways to represent the distances between clusters and data items explicitly in the network, as in merge clustering and multidimensional scaling. Also, IGG may prove useful in combination with other dimensionality reducing techniques, such as principal component analysis or multidimensional scaling. PCA and MDS can be first employed to find a good subspace representation of the data set. If this sapce has more than 2 dimensions, incremental grid growing could then be used to visualize it in 2-D.

Future application work will concentrate on discovering the unknown structure of large high-dimensional data sets. Specifically, the visualization of high-dimensional genetics data acquired from human populations around the world is being investigated (Cavalli-Sforza et al. 1994). IGG will also be applied to high-dimensional linguistics data for the same populations (Nichols 1992) for comparison. Clearly, the relationships and interactions between human populations are complex. Visualizing the relationships within and between genetic and linguistic clusters may aid in understanding the evolution and migration of populations.

6 CONCLUSION

The incremental grid-growing algorithm constructs 2-D drawable representations of arbitrarily complex high-dimensional input distributions. The algorithm addresses the shortcomings of popular visualization tools. Self-organizing maps (like principal component analysis and multidimensional scaling) capture continuous high-dimensional topology, but cannot represent discontinuities in the data. Merge clustering breaks the input set into clusters, but does not illustrate the internal cluster structure. The IGG algorithm can extract and represent high-dimensional relationships within and between clusters, and is capable of capturing general tree and graph structures. Consequently, it is a promising new tool for discovering relationships in unknown, complex real-world data sets.

References

Cavalli-Sforza, L. L., Menozzi, P., and Piazza, A. (1994). *The History and Geography of Human Genes*. Princeton, NJ: Princeton University Press.

Fritzke, B. (1991a). Let it grow—Self-organizing feature maps with problem dependent cell structure. In *Proceedings of the International Conference on Artificial Neural Networks* (Espoo, Finland), 403–408. Amsterdam; New York: North-Holland.

Fritzke, B. (1991b). Unsupervised clustering with growing cell structures. In *Proceedings of the International Joint Conference on Neural Networks* (Seattle, WA), vol. II, 531–536. Piscataway, NJ: IEEE.

Fritzke, B. (1992). *Wachsende Zellstrukturen—ein selbstorganisierendes neuronales Netzwerkmodell*. PhD thesis, Technischen Fakultät, Universität Erlangen–Nürnberg, Erlangen, Germany.

Jockusch, S. (1990). A neural network which adapts its structure to a given set of patterns. In Eckmiller, R., Hartmann, G., and Hauske, G., editors, *Parallel Processing in Neural Systems and Computers*, 169–172. Amsterdam; New York: North-Holland.

Kangas, J., Kohonen, T., and Laaksonen, J. (1990). Variants of self-organizing maps. *IEEE Transactions on Neural Networks*, 1:93–99.

Kohonen, T. (1989). *Self-Organization and Associative Memory*. Berlin; Heidelberg; New York: Springer. Third edition.

Kohonen, T. (1990). The self-organizing map. *Proceedings of the IEEE*, 78:1464–1480.

Martinetz, T. M., and Schulten, K. J. (1991). A "neural gas" network learns topologies. In *Proceedings of the International Conference on Artificial Neural Networks* (Espoo, Finland), 397–402. Amsterdam; New York: North-Holland.

Miikkulainen, R. (1993). *Subsymbolic Natural Language Processing: An Integrated Model of Scripts, Lexicon, and Memory.* Cambridge, MA: MIT Press.

Nichols, J. (1992). *Linguistic Diversity in Space and Time.* Chicago, IL: University of Chicago Press.

Ritter, H. J. (1991). Learning with the self-organizing map. In *Proceedings of the International Conference on Artificial Neural Networks* (Espoo, Finland), 379–384. Amsterdam; New York: North-Holland.

Ritter, H. J., and Kohonen, T. (1989). Self-organizing semantic maps. *Biological Cybernetics*, 61:241–254.

Rodriques, J. S., and Almeida, L. B. (1990). Improving the learning speed in topological maps of patterns. In *Proceedings of the International Neural Networks Conference* (Paris, France), 813–816. Dordrecht; Boston: Kluwer.

Scholtes, J. C. (1993). *Neural Networks in Natural Language Processing and Information Retrieval.* PhD thesis, Universiteit van Amsterdam, Amsterdam, the Netherlands.

Schütze, H. (1993). Word space. In Giles, C. L., Hanson, S. J., and Cowan, J. D., editors, *Advances in Neural Information Processing Systems 5.* San Mateo, CA: Morgan Kaufmann.

Xu, L., and Oja, E. (1990). Adding top-down expectation into the learning procedure of self-organizing maps. In *Proceedings of the International Joint Conference on Neural Networks* (Washington, DC), vol. II, 531–534. Hillsdale, NJ: Erlbaum.