Indices to Evaluate Self-Organizing Maps for Structures

J. J. Steil[1], A. Sperduti[2]
[1] Neuroinformatics Group, Faculty of Technology,
Bielefeld University, Bielefeld, Germany
[2] Dipartimento di Matematica Pura ed Applicata,
University of Padua, Padua, Italy
email: jsteil@techfak.uni-bielefeld.de, sperduti@math.unipd.it

Keywords: SOM-SD, structured data, classification, performance measure

***Abstract*—** Self-Organizing Maps for Structures (SOM-SD) are neural networks models capable of processing structured data, such as sequences and trees. The evaluation of the encoding quality achieved by these maps can neither be measured exclusively by the quantization error as in the standard SOM, which fails to capture the structural aspects, nor by indices measuring topology preservation, because often there are no measures available for discrete structures. We propose new indices for the evaluation of encoding quality which are customized to the structural nature of input data. These indices are used to evaluate the quality of SOM-SDs trained on a benchmark dataset introduced earlier in [2]. We show that the proposed indices capture relevant structural features of the tree encoding additional to the statistical features of the training data vectors associated with the tree vertices.

1 Introduction

Self-Organizing Maps (SOMs) [6] have been developed with the aim to visualize high-dimensional data in few dimensions, usually a regular two-dimensional grid of nodes, while preserving the most important topological and/or metric relationships of the high-dimensional data elements on the grid. The parameters which define the function implemented by a SOM are obtained by an unsupervised learning procedure. The evaluation of the "quality" of a map is not an easy task, since the standard quantization error does not take into consideration how well the topology of the input data is preserved. For this reason, some measures of topology preservation have been proposed in the past, e.g. [1, 5, 8, 7].

Recently, a SOM based model, i.e. SOM-SD [2], capable of processing structured data, such as sequences and trees, has been proposed. Such varying size data structures occur in natural language processing, HTML document and structured text analysis, robotics, or cheminformatics and bioinformatics. The evaluation of the "quality" of SOM-SDs is even more difficult, since input data is structured. Of course, just using the quantization error is not enough and the topology preservation measures mentioned above are useless since they are defined on vectorial metric spaces and it is not so obvious how to generalize them to structured domains. Up to now, the only way to evaluate the "quality" of a SOM-SD was through visual inspection of the mapping realized for selected input structures.

In this paper, we address this problem and propose new indices for measuring, in a quantitative way, how well structural features of the input data are preserved.

2 The SOM-SD framework

SOM-SD is a SOM capable of processing structured input in the form of a directed acyclic graph $\boldsymbol{T}$ taken from a data set of input graphs $\mathcal{T}$. Each vertex v of the graph $\boldsymbol{T}$ has attached a data vector, which can be real or discrete valued. Because the graph is assumed to be directed, for each vertex v the set of child vertices is well defined and denoted by $ch[v]$. SOM-SD can be understood as the recursive application of a standard SOM where the input is properly coded to take in consideration the data vectors together with the structural information summarized by $ch[v]$. For SOM-SD, a network input vector $\boldsymbol{x}_v$ represents the information of a vertex $v \in V(\boldsymbol{T})$ concatenating the data vector $\boldsymbol{v}$ and the coordinates of the mapping, on the same network, of child nodes $\boldsymbol{y}_{ch[v]}$ so that $\boldsymbol{x}_v = [\boldsymbol{v}, \boldsymbol{y}_{ch[v]}]$. We consider only two-dimensional maps such that $\boldsymbol{y}_{ch[v]}$ contains for each child of v the two-dimensional map-coordinates of the winner node for that child. The vectors $\boldsymbol{y}_{ch[v]}$ can be made constant in size if the maximum out-degree of any vertex in the dataset is known. Assuming that the maximum out-degree is o, then for vertices with less than o children, padding with a default coordinate, typically the "impossible" coordinate (-1,-1), is applied. As a result, the $\mathbf{x}$ are $k = p + 2o$ dimensional vectors, where p is the dimension of the data vector. The codebook vectors $\mathbf{m}$ are of the same dimension.

In order to account for the coding of both vertex data vector and structural information in the same input item, the winning neuron is selected by using a modified Euclidean distance, i.e. $r = \arg\min_i \|(\mathbf{x}_v - \mathbf{m}_i)\mathbf{\Lambda}\|$ where $\mathbf{\Lambda}$ is a $k \times k$ dimensional diagonal matrix. Its diagonal elements $\lambda_{11} \cdots \lambda_{pp}$ are set to μ_1, all remaining diagonal ele-

ments are set to $1-\mu_1$. The constant μ_1 weights the contribution of the data vector component against the influence of the children's coordinates representing the structural information Intuitively, setting large values for μ_1 focusses on the data vectors but reduces the SOM-SD's ability to represent structural information and reduces the accuracy with which information is passed on to parent nodes. Small μ_1 emphasise mapping precision according to structural information but neglect the importance of the information provided by the data vectors. We will derive a new measure which shows exactly this tradeoff quantitatively. See [2] for more details on SOM-SD.

3 Evaluation measures

It is very difficult to evaluate the ability of a map to encode relevant spatio-temporal relationships contained in the input trajectories. There are different issues that must be considered. First of all we would like to have a small quantization error, i.e. we would like the weight vectors to retain all the statistical information about the data vector distribution. Moreover, we would like the map to preserve also the main temporal (or more in general, structural) features of the trajectories, i.e., the "history" or "contextual" information contained in the trajectories (e.g. that a point in space is reached by a trajectory only passing across a specific subspace and following a specific direction). This means that we expect that the same region of the input space is "covered" by several neurons, each one specialized to recognize from which origin and direction the trajectory reaches that region. Of course, this ability should be obtained with no reduction of the quantization capability of the network.

In this paper, we address this issue on the representation side, i.e. we ask how a trajectory of a structured object can be represented such that both contextual information is preserved and quantization error is minimized.

Quantization Error. If with $r(\boldsymbol{x}_v)$ we denote the index of the winning neuron for input $\boldsymbol{x}_v,\ v \in V(\boldsymbol{T})$, the (squared) quantization error for the standard SOM is defined as

$$E_{som} = \sum_{\boldsymbol{T}\in\mathcal{T}} \sum_{v\in V(\boldsymbol{T})} (\boldsymbol{x}_v - \boldsymbol{m}_{r(\boldsymbol{x}_v)})^2.$$

This way of computing the quantization error, however, includes also the structural component introduced by the input encoding. We can avoid this contribution, when considering SOM-SD, by computing the quantization error as:

$$E_{som-sd} = \sum_{\boldsymbol{T}\in\mathcal{T}} \sum_{v\in V(\boldsymbol{T})} (\boldsymbol{v} - \hat{\boldsymbol{m}}_{r(\boldsymbol{x}_v)})^2,$$

where $\hat{\boldsymbol{m}}$ is the sub-vector obtained by selecting the first p components of the code vector $\boldsymbol{m}$, i.e., the portion which refers to the input label.

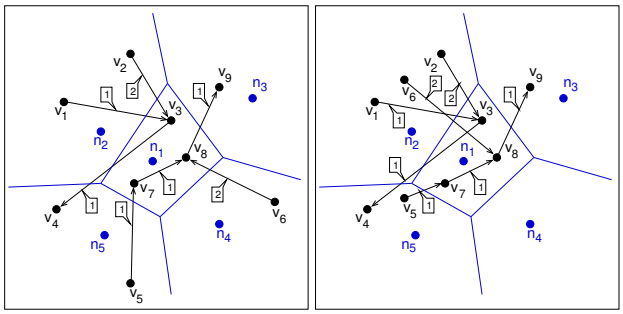

Figure 1: Representations of different mappings for trees with 2D labels. ν_i represent the 2D weight vectors associated to neurons, while labels attached to tree's vertices are represented by v_i. and are connected by directed arcs representing the trees' topology. The direction of the arcs is inverted with respect to a standard representation for trees in order to represent the fact that vertices are presented to the map according to an inverted topological order. Arcs are annotated by numbers (either 1 or 2) representing the position assigned to each child. The immediate receptive field for neuron ν_1 is $R_1(0) = \{v_1, v_2, v_5, v_6, v_7\}$, i.e. the children of vertices for which ν_1 is the winner node. The spread of data (2-D positions) attached to vertices belonging to $R_1(0)$ is larger for the data shown in the left side box than for data shown in the right side box, consequently the map encoding is better for the right hand side data.

Simple Contextual Error. In order to understand how structural information is codified, let us define the *immediate receptive field* $R_i(0)$ of a neuron with index i, i.e. all the child vertices of parents for which neuron n_i is the winner:

$$R_i(0) = \{u \in \boldsymbol{T}, \boldsymbol{T} \in \mathcal{T} | u \in ch[v],\ r(\boldsymbol{x}_v) = i\}.$$

We can then define the immediate context as the set of "predecessors" of inputs in $R_i(0)$ and, recursively the set of "predecessors" of "predecessors" of inputs in $R_i(0)$:

$$R_i(j) = \{u \in \boldsymbol{T}, \boldsymbol{T} \in \mathcal{T} | u \in ch[v],\ v \in R_i(j-1)\}.$$

The definition of these sets is particularly useful, since, given a neuron n_i, we can look at all the inputs which are j steps back with respect to inputs in the *immediate receptive field* and check if they are spread or concentrated. In the first case, it means that the neuron is not able to codify the history of the input, while in the last case it means that the neuron is actually able to do it. An example of immediate receptive field is given in Figure 1, assuming that the training data is constituted by just two trees with root nodes v_4 and v_9. Then the immediate receptive field for neuron n_1 is defined by the set $R_1(0) = \{v_1, v_2, v_5, v_6, v_7\}$. The left part of the figure shows an immediate receptive field for neuron n_1 which exhibits a larger spread than the one represented in the right side box. For a given j, we can

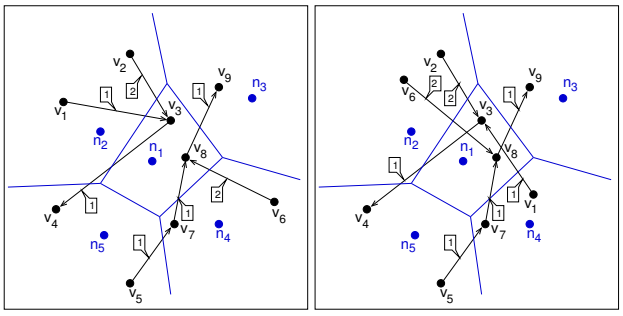

Figure 2: The spread of 2-D position data attached to vertices belonging to $R_1(0) = \{v_1, v_2, v_6, v_7\}$ is much smaller for the data shown in the right side box than for data shown in the left side box when considering children in the same position (either 1 or 2.)

quantify the degree of encoded context of neuron n_i, i.e. history, by computing the sample variance of the label data over the dataset $\mathcal{T}$:

$$\sigma^2(i,j) = \frac{1}{|R_i(j)|} \sum_{u \in R_i(j)} (\boldsymbol{u} - \overline{\boldsymbol{u}})^2$$

where $\overline{\boldsymbol{u}}$ is the sample mean (computed over $R_i(j)$) of the labels. Finally, we can average over all the neurons in a map to get a "global" evaluation of the map:

$$C(j) = \frac{1}{N} \sum_{i}^{N} \sigma^2(i,j).$$

The above measure can be understood as a direct generalization of the temporal context measure defined for sequences in [4, 9] to structures but in this simplicity it has a major drawback when considering structural data. It is impossible to discriminate contexts where children occurring in the same position are concentrated while looking across different positions they are quite different (Fig. 2, right), from contexts where children occurring in the same position are spread around as well as when looking across different positions (Fig. 2, left).

Structure Respecting Contextual Error. In order to discriminate the different contexts over structures, we need to take into consideration the co-occurence of children of the same vertex. This should be done considering both the topology of the input structures and the label attached at the vertices. First consider the special case that each input structure is constituted by a fully complete structure, e.g. a complete binary tree. Denote by $p \in path(\boldsymbol{T})$ a path p from the set of possible pathes $path(\boldsymbol{T})$ of the tree $\boldsymbol{T}$. Then, given an input structure, it is easy to keep track of co-occurences by focusing on the set of vertices which can be reached by a path p of length $len(p) = j$ starting from a specific vertex, and then building up a single vector obtained by the concatenation of the labels attached at each vertex of the above defined set. In order to exemplify, let consider the above case of complete binary trees. Then, each (relative) path of length l can be described by a string of length j from the alphabet $\{l, r\}$, where the symbol l indicates that the left child has been followed, and the symbol r indicates that the right child has been followed. Thus, if $j = 2$, we have the 4 possible paths ll, lr, rl, rr. For each input tree $\boldsymbol{T}$, given a vertex $u \in \boldsymbol{T}$, we can now build the following context vector $\gamma(u, 2) \equiv [\boldsymbol{v}_{ll}, \boldsymbol{v}_{lr}, \boldsymbol{v}_{rl}, \boldsymbol{v}_{rr}]$, where $\boldsymbol{v}_{ll}$ is the label attached at the vertex v reached by starting from u and following the path ll, and so on for the other paths. Using this definition, we can define the following sample variance:

$$\sigma^2_\gamma(i,j) = \frac{1}{|win_i|} \sum_{u \in win_i} \left(\gamma(u,j) - \overline{\gamma(u,j)}\right)^2,$$

where $\overline{\gamma(u,j)}$ is the sample mean of $\gamma(u,j)$ and win_i the set of vertices in the training set for which neuron i is the winner node.

The problem with this definition is that it is not clear what to do when the input trees are not all complete. In fact, in this case we are faced with the problem that going backward from a vertex towards the frontier of the input tree, different paths may end earlier than others, thus rendering impossible to produce a well formed vector $\gamma(u,j)$.

Here we suggest to proceed as follows. First of all, we compute the label variance on each single (relative) path, then we take the expected value of this variance over paths of the same length. More formally, given $win_i(p)$ be the subset of vertices u in win_i for which there exist a (relative) path $p(u)$, the sample mean for the labels reachable by p (starting from any vertex in $win_i(p)$) is defined as:

$$\mu_\pi(i,p) = \frac{1}{|win_i(p)|} \sum_{u \in win_i(p)} \boldsymbol{v}_{p(u)}.$$

We can now define the following sample variance:

$$\sigma^2_\pi(i,p) = \frac{1}{|win_i(p)|} \sum_{u \in win_i(p)} \left(\boldsymbol{v}_{p(u)} - \mu_\pi(i,p)\right)^2$$

and compute its expected value over paths of the same length j:

$$E^2_\pi(i,j) = \sum_{p \mid len(p)=j} f_{i,j,p}\, \sigma^2_\pi(i,p),$$

where

$$f_{i,j,p} = \frac{|win_i(p)|}{\sum_{q \mid len(q)=j} |win_i(q)|}.$$

Again, we can average over all the neurons in a map to get a "global" evaluation:

$$C_\pi(j) = \frac{1}{N} \sum_{i}^{N} \sigma^2_\pi(i,j).$$

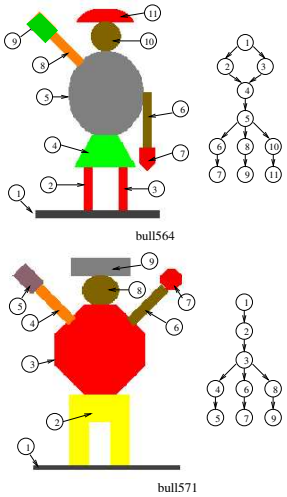

Figure 3: Graph structure of two policemen (data labels are not shown). Nodes are numbered to indicate which element they represent.

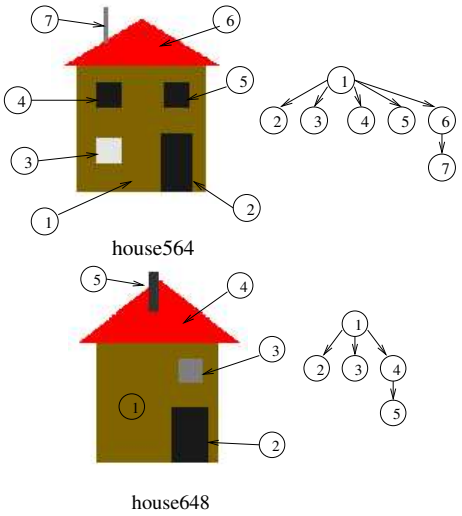

Figure 4: Artificially generated houses and associated graph structure. Data labels are not shown. Nodes are numbered to indicate which element is represented.

Computation. An efficient computation of the variance measure is not trivial, because the number of possible paths within a tree grows exponentially with their maximal length and the measures also require to compute variances for each node in the map separately. But in practice, only few of these paths actually occur in real applications and it is possible to exploit that each vertex of a tree contributes to the variance with respect to one path of a given length for a single winner. Given that all means and variances can be computed incrementally and recursively, we use the following strategy to determine all relevant quantities in two steps. First we determine for each vertex the corresponding winner node in the (trained) SOM-SD map. Then, for each vertex and up to a given depth, we determine all paths the vertex is part of and update the means and variances for these paths and the respective winner incrementally by moving to the parent and using the information which is the winning node for this parent. Only if a new type of path or a new winner node is encountered, memory is allocated and the incremental computations are initialized. Thereby all computations can be performed by two sweeps over all vertices, one for determining the winners, and one for computing the variances.

4 Data and experimental setup

For the experiments, we used the dataset given in [2]. In essence, the dataset is composed of labeled directed ordered acylic graphs (DOAGs) extracted from images produced by means of a context free attributed plex grammar. The dataset consists of visual patterns and associated graph structures from three different domains. A number of classes are defined for each domain. Structural representations are obtained by a scan line algorithm where images are scanned from bottom to top, and from left to right. The first colored object found constitutes the root node. Objects directly connected form the offsprings. Applied recursively, all objects are considered and a graph representation is obtained. Each node in the graph receives a two dimensional label stating the $\{x, y\}$ coordinate of the center of gravity of the corresponding object. Examples are illustrated in figure 3, figure 4, and figure 5.

The result is a dataset that provides directed acyclic graphs with the following properties:

Data set	Outdegr. Max.	Depth Min	Depth Max	Num. nodes Min	Num. nodes Max	Num. classes
Policemen	3	4	5	9	11	2
Houses	5	2	3	4	7	8
Ships	6	1	2	3	13	2

Hence, policemen patterns produce deep narrow graphs, ships and houses have a flat and wide data structure. Some graph structures produced by the ships and houses are identical in structure such as `house648` in figure 4 and `ship1034` in figure 5. There is no graph structure derived from policemen images that is contained in the domain houses or ships.

Some patterns can be distinguished only through features encoded in the labels. For example, when considering policemen, the location of the arm is not encoded in the graph structure, but in the data label, while the graph structure is not affected.

The maximum outdegree of all nodes in the data set is 6. The training (test) set contained $3,750$ $(3,750)$ graphs with a total of $29,864$ $(29,887)$ nodes. For the purpose of testing the ability of the network to autonomously organize the input structures according to their structural

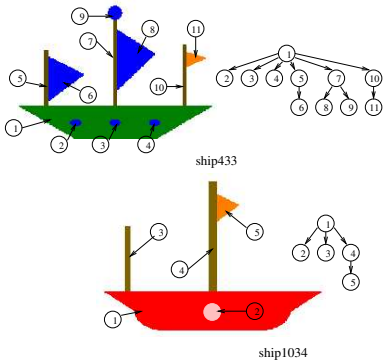

Figure 5: Artificially generated images of ships. Data labels are not shown.

and content (i.e., data attached to nodes) features, 12 classes are defined. The property of these classes is such that some classes can only be distinguished by considering information provided through data labels. Other classes require structural information in order to be distinguished. A detailed description of the classes can be found in [2].

SOM-SD preparation. To evaluate the capacity of the SOM-SD we train maps of different sizes from 20×10, 24×12, ..., 60×30, and 90×45. Map initialization differs for the weights representing node label, where values are uniformly randomized between the maximum and the minimum value of the respective component in the training data and weights referring to context, which are initialized to uniformly in $[-1, 0]$. We also compare different weightings of the labels against the context, i.e. settings of μ_1.

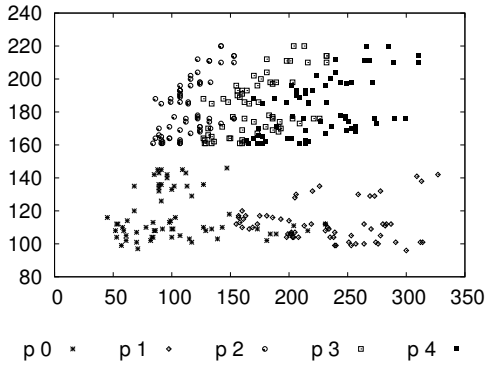

Figure 6: Clustering of the 2-D positions data vectors (pixel positions) of child vertices (immediate receptive field) represented by a single winner node. Symbols encode different positions in the fan-out tree, i.e. different paths p of length $len(p) = 1$.

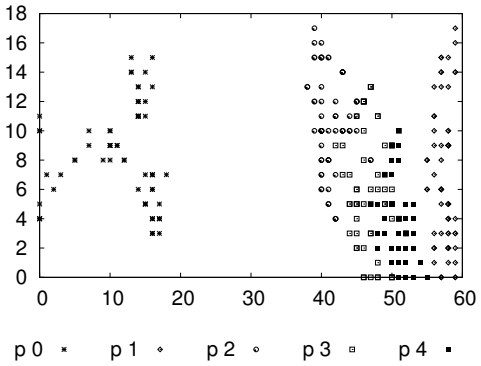

Figure 7: Clustering of winners for points shown in fig. 6 on the corresponding 60×30 map.

5 Results on the variance measures

Figures 6 and 7 illustrate the the importance of the path-respecting computation of the variance measure for a single exemplary node from a 60×30 trained map. It is clearly visible that the SOM-SD encodes the context by clustering winners for the predecessors respecting the path in the tree (here the position in the fan-out tree) in label space. The variance-based measures quantify this effect.

Figure 8 shows the development of the quantization error E_{som-sd}, the simple contextual variance C (as varnode), the quality of the context encoding measured by $C_\pi(1)$ (as var0) for the immediate predecessors and by $C_\pi(2)$ (as var1) for paths p of length $len(p) = 1, 2$. We see that the SOM-SD decreases the variance measures and the quantization error over training. All further results use 20 training iterations, i.e. 20 presentations of the full set of 3750 graphs in newly randomized order for each presentation.

In figure 9, the tradeoff from weighting the structural component higher than the label is considered. We get the best classification performance for most of the weight on the structure, however, the quantization error is larger than for a more balanced weighting of label and structure. On the other hand, the variance measure decreases consistently with the increase in classification performance and is therefore better suited to evaluate the map. This result motivates to use for the last experiment a fairly large weight on the structure ($1 - \mu_1 = 0.95$) and to evaluate performance with respect to the size of the map.

Figure 10 shows the respective results for maps of size $20 \times 10, \ldots, 60 \times 30$ for otherwise constant parameters and $1 - \mu_1 = 0.95$. While the quantization error is insignificant to show superiority of larger maps, the variance measures once more decrease consistently with the increase in classification performance.

6 Conclusion

We introduce several measures to quantify the success of encoding the structural properties of input graphs clustered

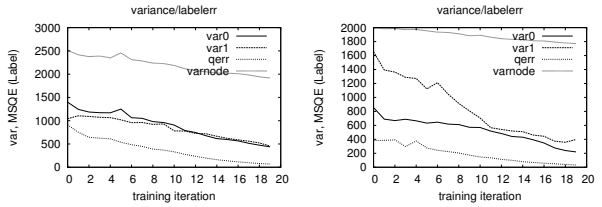

Figure 8: Development of error and variance measures over training iterations for different 90×45 maps with $\mu_1 = 0.01(left)$ and $\mu_1 = 0.1$. Clearly the structure respecting variance measures var0 for immediate and var1 for 2-nd order children are decreasing, while the simple receptive field variance (varnode) stays large. One training iteration corresponds to a single presentation of all graphs in randomized order.

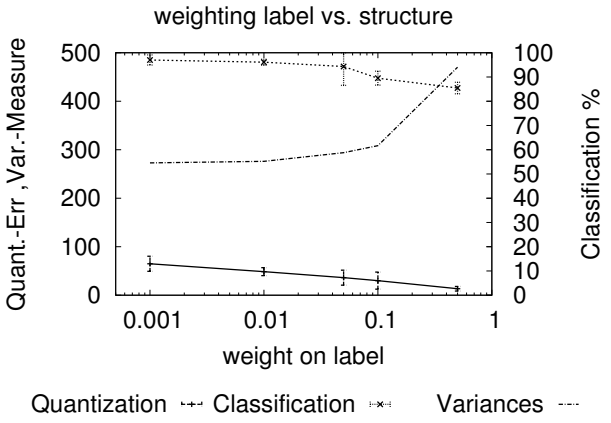

Figure 9: Quantization error, variance measure for immediate children, and classification with respect to the weighting of labels (μ_1). All measures averaged over 10 runs on 90×45 maps.

with SOM-SD. We show that the quantization error alone is an insufficient measure for structural encoding and a naive approach directly using the previously introduced receptive field variance like in [4, 9] for recursive SOMs also fails to differentiate with respect to children positions in the fan-out tree. Therefore we propose to generalize this approach to structure respecting variance measures and demonstrate on a SOM-SD benchmark dataset that these measures are consistent in the sense that they decrease over training and for larger maps. Further, if the comparison metric imposes large weight on the structural encoding components, the quantization error on the labels may increase while the variance measure decreases. It is interesting that this goes along with better classification performance on the given dataset, which motivates to investigate in future work whether an online-monitoring of the variances could be useful to modify the SOM-SD training scheme, for instance to introduce a schedule for changing μ_1 adaptively.

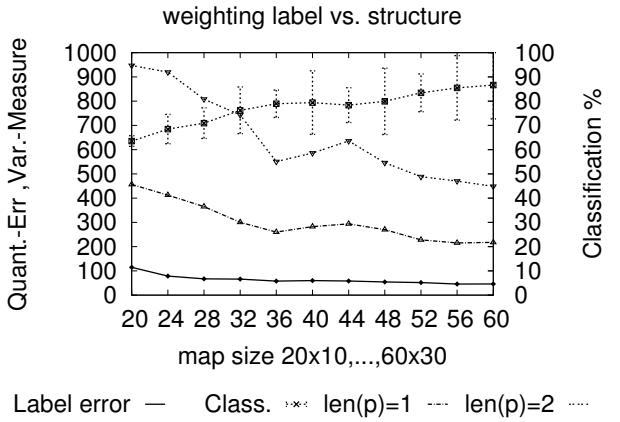

Figure 10: Evaluation of map size: The quantization error is insignificant while the variance measures evaluated with respect to paths of length 1 and 2 consistently decrease with improving the classification performance.

Acknowledgements. We would like to thank M. Hagenbuchner for providing us with an implementation of the SOM-SD as described in more detail in [3]. This work was partially supported by the "Vigoni Program", sponsored by DAAD and Conferenza Italiana dei Rettori.

References

[1] H. U. Bauer and K. R. Pawelzik. Quantifying the neighborhood preservation of self-organizing feature maps. *IEEE TNN*, 3(4):570–579, 1992.

[2] M. Hagenbuchner, A. Sperduti, and A.C. Tsoi. A self-organizing map for adaptive processing of structured data. *IEEE TNN*, 14(3):491–505, 2003.

[3] M. Hagenbuchner and A.C. Tsoi. A supervised self-organizing map for structures. In *IJCNN*, 2004.

[4] B. Hammer, A. Micheli, A. Sperduti, and M. Strickert. Recursive self-organizing network models. *Neural Networks*, 17(8–9):1061–1085, 2004.

[5] S. Kaski and K. Lagus. Comparing self-organizing maps. In *ICANN 96*, pages 809–814, 1996.

[6] T. Kohonen. *Self-Organizing Maps*, volume 30 of *Springer Series in Information Sciences*. Springer, Berlin, Heidelberg, 1995.

[7] D. Polani. Measures for the organization of self-organizing maps. In *Self-Organizing Neural Networks. Recent Advances and Applications*. Springer, 2001.

[8] M-C. Su, H-T. Chang, and C-H. Chou. A novel measure for quantifying the topology preservation of self-organizing feature maps. *Neural Processing Letters*, 15(2):137–145, 2002.

[9] T. Voegtlin. Recursive self-organizing maps. *Neural Networks*, 15:979–992, 2002.