Links Between Self–Organizing Feature Maps and Weighted Vector Quantization

Gregory R. De Haan[1] Ömer Eğecioğlu

Department of Computer Science
University of California, Santa Barbara
Santa Barbara, CA 93106

Abstract

A new learning algorithm for Self-Organizing Feature Maps (SOFM) is presented. The learning algorithm is based on an extension of vector quantization called weighted vector quantization (WVQ). WVQ distortion is a weighted sum of the distortion between an input vector and each of the codevectors in the codebook. A formulation of WVQ is given, as well as two optimality conditions which are analogous to the nearest neighbor and centroid conditions of vector quantization. We then incorporate the SOFM neighborhood mechanism into WVQ, and use the WVQ optimality conditions to derive the new algorithm.

1 Introduction

From the late 1970's to the present, Teuvo Kohonen has developed the Self-Organizing Feature Map (SOFM) neural networks. SOFM were found to have many remarkable capabilities [8]: they can preserve the topological relationships in a multi-dimensional space, while performing data reduction (*topology preservation*); they can represent those dimensions of the input space with high variance (*automatic selection of feature dimensions*); they can represent hierarchically related data on a single layer of processing units, instead of using separate layers to represent different levels of abstraction; they can be trained quickly and were successfully applied to a practical problem– automatic speech recognition[7].

Kohonen's update rule for SOFM is a pattern learning [11] version of the generalized Lloyd algorithm for vector quantizer design [3], where an SOFM unit's weight vector corresponds to a vector quantization codeword. Kohonen training adds a neighborhood mechanism which gives SOFM their additional capabilities. Like vector quantizers, SOFM can produce good codebooks. Unlike vector quantizers, SOFM can order the codevectors such that their position on the SOFM encodes information about the topological relationships between the codevectors. The other properties described above are by-products of this topology preserving ordering.

SOFM are trained to achieve topology preservation by first using large neighborhoods to obtain a reasonable ordering of vectors; and then reducing the neighborhoods to obtain good codebooks in a vector quantization sense: i.e. by minimizing the expected distortion introduced by mapping an input vector to a codevector. However, poor ordering of these vectors can occur [4][5]. While it is easy to detect poor ordering in simple "toy" problems, with complex multidimensional inputs, e.g. speech spectra, there has been no objective measure of how well the vectors are ordered.

Herein neighborhood distortion functions (NDF) are used to quantify the degree of topology preservation in an SOFM [2]. With NDF, all codevectors in the *neighborhood* of the best matching codevector contribute to the distortion. NDF are effective for determining the degree of topology preservation because lowering the distortion encourages the preservation of both adjacency and non-adjacency: distortion is lower when a unit and its neighbors have similar weight vectors, and distortion is higher when a unit and its neighbors have dissimilar weight vectors.

We are studying the special capabilities of SOFM partly for use in speech recognition systems [1]. Instead of assigning labels to the units, as in [7], our recognizers use the position *on the SOFM* of the

[1] Also with Speech Technology Laboratory, Div. of Panasonic Technologies, Inc., Santa Barbara, California.

CH 3065-0/91/0000-0887 $1.00 © IEEE

units which respond. In contrast, the approach taken by [7] does not utilize the topology preserving ordering produced by SOFM (or its by-products). Torkkola and Kokkonen have independently studied the direct use of topology preservation for speech recognition [10].

2 SOFM Learning

An SOFM consists of neuron-like units in a d-dimensional space called a *map*: typically $d = 2$, in analogy to the (essentially) 2-dimensional structure of human cortex. Usually the units are placed uniformly on the map, with either a rectangular or hexagonal tessellation. The map is trained iteratively, with each unit u_i updating its weight vector w_i. For each iteration, Kohonen's SOFM training algorithm performs the following:

1. *Find the unit u whose weight vector is closest to the input vector.*
2. *Update the weight vectors of unit u and all units in the neighborhood of u as a convex combination of the input vector and the corresponding weight vector, i.e.*

$$w_j(t+1) = \begin{cases} (1-\alpha_t)w_j(t) + \alpha_t x & \text{if } u_j \in \text{neighborhood } (u), \\ w_j(t) & \text{otherwise.} \end{cases}$$

Note that the best matching unit is chosen by assuming that the nearest neighbor rule [3] holds. Also, when an input is presented, a given unit u is updated if and only if the closest unit is in u's neighborhood; thus the expected value of a unit's weight vector is the *centroid* of the union of the Voronoi regions corresponding to the unit and its neighbors [6]. Therefore Kohonen learning appears to be an extension of the generalized Lloyd algorithm which accounts for the neighborhood mechanism. Based on these implicit assumptions of Kohonen learning, we have developed a batch training algorithm for SOFM which makes these goals explicit. For each pass through the training data, this batch algorithm does the following:

1. *Partition: Assign each input vector to the unit with the closest weight vector.*
2. *Update: using the partition from step 1, update the weight vector for each unit u to be the centroid of the set of input vectors assigned to either u or any of the neighbors of u.*

While this batch algorithm appears to be a simple extension of the generalized Lloyd algorithm, it has an important deficiency when the neighborhood radius $r > 0$. Upon running experiments with the batch algorithm and $r > 0$, we found that the distortion[2] was *not* monotonically decreased with each pass through the training data. We therefore performed an analysis of SOFM which minimize neighborhood distortion functions [2]. For the simple case of a 1-D SOFM with 1-D inputs and $r \geq 1$, neither the partitioning nor the centroid assumptions of the batch algorithm (and Kohonen's algorithm) held for optimal codebooks. Even for Voronoi SOFM, where we enforce the nearest neighbor rule, optimal codebooks violated the centroid assumption.

3 Weighted Vector Quantization

Based on the findings in [2], we formulated and found optimality conditions for Weighted Vector Quantization (WVQ). The objective in WVQ is to construct codebooks which minimize the expected value of a "multiple-vector" distortion function. We consider distortion functions which are a weighted sum of the distortion between the input vector and each of the codevectors. Since we did not want to make any restrictions on the neighborhood distortion functions for SOFM, we did not place any constraints on the weights used in the distortion function.

[2] The distortion was defined as the sum of the mean-square distortion between the input and each unit in the neighborhood of the best-matching unit.

Associated with a weighted vector quantizer is a finite set of N codevectors $Y = \{y_1, y_2, \ldots, y_N\} \subset \Re^n$; an $N \times N$ weight matrix of real numbers $\mathbf{W}$, an N-region partition $R_1, R_2, \ldots, R_N$ of $\Re^n$, where R_i is the region associated with codevector y_i; an encoder function $\mathcal{C} : \Re^n \rightarrow [\mathrm{N}]$, where $[\mathrm{N}] = \{1, 2, \ldots, N\}$ which takes the input $x \in \Re^n$ and yields the index i of the region R_i, i.e. $\mathcal{C}(x) = i \longleftrightarrow x \in R_i$; and a decoder function $\mathcal{W} : [\mathrm{N}] \rightarrow \Re^N$, defined below.

A weighted vector quantizer maps an input $x \subset \Re^n$ to a weight vector, as follows:

$$\mathcal{W}(\mathcal{C}(x)) = (\mathbf{W}_{\mathcal{C}(x),1}, \mathbf{W}_{\mathcal{C}(x),2}, \ldots, \mathbf{W}_{\mathcal{C}(x),N}), \tag{1}$$

where the first component of the resultant vector corresponds to codevector y_1, and the second component corresponding to y_2, and so on. Thus a weighted vector quantizer maps an input x to row $\mathcal{C}(x)$ of of the matrix $\mathbf{W}$.

Given N and the weight matrix $\mathbf{W}$, the performance of a weighted–vector quantizer is measured by the average distortion

$$\mathcal{D}(\mathcal{W}) = E\left[\Delta(x, \mathcal{C}(x))\right],$$

where $\Delta : \Re^n \times [\mathrm{N}] \rightarrow \Re$, the weighted vector quantization distortion function, is defined as

$$\Delta(x, \mathcal{C}(x)) = \sum_{j=1}^{N} \delta(x, y_j) \mathbf{W}_{\mathcal{C}(x),j}. \tag{2}$$

where $\delta : \Re^n \times \Re^n \rightarrow \Re$ is a distance function.

Given N, $\mathbf{W}$, and the input distribution, a weighted vector quantizer $\mathcal{W}'$ is optimal if, for any weighted vector quantizer $\mathcal{W}$,

$$\mathcal{D}(\mathcal{W}') \leq \mathcal{D}(\mathcal{W})$$

Example 1: Standard (Noiseless Channel) vector quantization.

Standard vector quantization, and Kohonen learning with neighborhood radius $r = 0$, are special cases of WVQ, where the weight matrix $\mathbf{W}$ is an $N \times N$ identity matrix.

Example 2: Noisy Channel VQ

Another special case of WVQ is vector quantization in the presence of discrete, memoryless channel noise [12], where $\mathbf{W}_{ij} = \Pr[\mathcal{Q}(x) = y_j | \mathcal{C}(x) = i]$.

3.1 WVQ Optimality Conditions

In this section we present two necessary conditions for optimal WVQ, which are extensions of the Nearest Neighbor Partitioning and Centroid Conditions of standard vector quantization [3].

3.1.1 Optimal WVQ Partitioning

Here we are given N, $\mathbf{W}$, the input distribution and the set of codevectors Y. We are interested in the optimal partitioning of the input space to minimize WVQ distortion.

The WVQ Partitioning Condition: For any optimal weighted vector quantizer, the partition is defined as follows:

$$R_i \supset \{x \in \Re^n : \Delta(x, i) < \Delta(x, j), \text{ for all } i, j \in [\mathrm{N}], i \neq j\} \tag{3}$$

PROOF

$$\begin{aligned} E\left[\Delta(x, \mathcal{C}(x))\right] = \sum_{i=1}^{N} E\left[\Delta(x, i) | x \in R_i\right] Pr\left[x \in R_i\right] &= \sum_{i=1}^{N} E\left[\sum_{j=1}^{N} \delta(y_j, x) \mathbf{W}_{ij} | x \in R_i\right] Pr\left[x \in R_i\right] \\ &= \sum_{i=1}^{N} \left[\left(\sum_{j=1}^{N} \int_{x \in R_i} \delta(y_j, x) \mathbf{W}_{ij} dx\right) Pr\left[x \in R_i\right]\right] \end{aligned}$$

Note that

$$\sum_{i=1}^{N}\left[\left(\sum_{j=1}^{N}\int_{x\in R_i}\delta(y_j,x)\mathbf{W}_{ij}dx\right)Pr[x\in R_i]\right]\geq\sum_{i=1}^{N}\left[\min_{1\leq k\leq N}\left(\sum_{j=1}^{N}\int_{x\in R_i}[\delta(y_k,x)\mathbf{W}_{kj}]\,dx\right)Pr[x\in R_i]\right]$$

and thus distortion is minimized if and only if the partitioning ensures that equality holds. Equality occurs only if (3) holds. □

3.1.2 Optimal WVQ Codevector Placement

Here we are given N, $\mathbf{W}$, the input distribution and the partition. We are interested in the optimal values of the codevectors Y to minimize WVQ distortion.

The WVQ Codevector Placement Condition: For any optimal weighted vector quantizer where the distance function δ is squared Euclidean distance, i.e. $\delta(x,y)=\|x-y\|^2$, the codevectors are given by

$$y_i=\sum_{j=1}^{N}\frac{\mathbf{W}_{ji}Pr[x\in R_j]}{\sum_{k=1}^{N}\mathbf{W}_{ki}Pr[x\in R_k]}E[x\mid x\in R_j]. \tag{4}$$

Since $E[x\mid x\in R_j]$ is the centroid of the region R_j, optimal codevectors are placed at a linear combination of the centriods of the N regions in the partition. If $\mathbf{W}_{ij}\geq 0$ for all $i,j\in$ [N], then (4) implies that optimal codevectors are placed at a convex combination of the centroids.

PROOF

$$\begin{aligned}E[\Delta(x,C(x))]=E[\sum_{i=1}^{N}\delta(x,y_i)\mathbf{W}_{C(x),i}] &= \sum_{i=1}^{N}\sum_{j=1}^{N}(E[\delta(x,y_i)\mathbf{W}_{ji}\mid x\in R_j]\;Pr[x\in R_j])\\ &= \sum_{i=1}^{N}\sum_{j=1}^{N}(\mathbf{W}_{ji}E[\delta(x,y_i)\mid x\in R_j]\;Pr[x\in R_j])\\ &= \sum_{i=1}^{N}\sum_{j=1}^{N}(\mathbf{W}_{ji}Pr[x\in R_j])\,E[\delta(x,y_i)\mid x\in R_j]\end{aligned}$$

Now consider the inner sum with the squared Euclidean distance distortion, and let $F_j=\mathbf{W}_{ji}Pr[x\in R_j]$.

$$\begin{aligned}\sum_{j=1}^{N}F_jE\left[\|y_i-x\|^2\mid x\in R_j\right] &= \sum_{j=1}^{N}F_jE\left[\|x\|^2|x\in R_j\right]+\sum_{j=1}^{N}F_jE\left[\|y_i\|^2\right]-2\sum_{j=1}^{N}F_jE\left[x\cdot y_i\mid x\in R_j\right]\\ &= \sum_{j=1}^{N}F_jE\left[\|x\|^2|x\in R_j\right]+\sum_{j=1}^{N}F_j\|y_i\|^2-2\sum_{j=1}^{N}F_j\left(y_i\cdot E\left[x\mid x\in R_j\right]\right)\\ &= \sum_{j=1}^{N}F_jE\left[\|x\|^2|x\in R_j\right]+\sum_{j=1}^{N}F_j\left(\|y_i\|^2-2(y_i\cdot E\left[x\mid x\in R_j\right])\right)\end{aligned}$$

Since

$$\|y_i-E[x\mid x\in R_j]\|^2=\|y_i\|^2+\|E[x\mid x\in R_j]\|^2-2(y_i\cdot E[x\mid x\in R_j])$$

we have

$$\begin{aligned}\sum_{j=1}^{N}F_j\left(\|y_i\|^2-2(y_i\cdot E\left[x\mid x\in R_j\right])\right) &= \sum_{j=1}^{N}F_j\left(\|\,y_i-E[x\mid x\in R_j]\,\|^2-E[x\mid x\in R_j]\right)\\ &= \sum_{j=1}^{N}F_j\|\,y_i-E[x\mid x\in R_j]\,\|^2-\sum_{j=1}^{N}E[x\mid x\in R_j]\end{aligned}$$

Therefore the codevector y_i must be placed so as to minimize

$$\sum_{j=1}^{N} F_j \| y_i - E[x \mid x \in R_j] \|^2 \tag{5}$$

and thus

$$y_i = \frac{\sum_{j=1}^{N} F_j E[x \mid x \in R_j]}{\sum_{j=1}^{N} F_j} = \sum_{j=1}^{N} \frac{F_j}{\sum_{k=1}^{N} F_k} E[x \mid x \in R_j] = \sum_{j=1}^{N} \frac{\mathbf{W}_{ji} Pr[x \in R_j]}{\sum_{k=1}^{N} \mathbf{W}_{ki} Pr[x \in R_k]} E[x \mid x \in R_j],$$

which is (4). □

4 WVQ with Neighborhood Distortion

In this section we incorporate neighborhood functions into the WVQ distortion measure, which will lead to a learning algorithm for SOFM.

We define a neighborhood function $\mathcal{N} : Y \times Y \rightarrow \Re$, which is used to determine the response of each unit on the SOFM, given a unit's weight vector and the weight vector of that of the best matching unit. Typically, $\mathcal{N}(y_i, y_j)$ will yield a value determined by the distance (on the SOFM) between the units corresponding to y_i and y_j. For instance, for the 1-dimensional SOFM analyzed in [6], with a neighborhood radius $r = 1$, $\mathcal{N}(y_i, y_j) = 1$ when $|i - j| \leq r$, and 0 otherwise.

When the neighborhood function is used to weight a unit's contribution to distortion, we find that the total distortion for an input x can be given as

$$\Delta(x, \mathcal{C}(x)) = \sum_{y \in Y} \mathcal{N}(y_{\mathcal{C}(x)}, y) \times \delta(x, y). \tag{6}$$

The average distortion per codevector is then

$$\Delta(x, \mathcal{C}(x)) = \frac{\sum_{y \in Y} [\mathcal{N}(y_{\mathcal{C}(x)}, y) \delta(x, y)]}{\sum_{y \in Y} \mathcal{N}(y_{\mathcal{C}(x)}, y)} = \sum_{y \in Y} \left[\frac{\mathcal{N}(y_{\mathcal{C}(x)}, y)}{\sum_{y \in Y} \mathcal{N}(y_{\mathcal{C}(x)}, y)} \times \delta(x, y) \right] \tag{7}$$

Thus we incorporate the neighborhood function into WVQ by defining $\mathbf{W}_{ij}$ in terms of the SOFM distortion function. For example, WVQ distortion is defined by using the neighborhood distortion function to determine the values for the matrix $\mathbf{W}$. For total distortion, we find that

$$\mathbf{W}_{ij} = \mathcal{N}(y_i, y_j). \tag{8}$$

For the average distortion per node, $\mathbf{W}_{ij}$ is the relative (weighted) contribution of y_i in y_j's neighborhood when $\mathcal{Q}(x) = y_j$, i.e.

$$\mathbf{W}_{ij} = \frac{\mathcal{N}(y_i, y_j)}{\sum_k \mathcal{N}(y_k, y_j)}. \tag{9}$$

Note that $\sum_{i=1}^{N} \mathbf{W}_{ij} = 1$.

The neighborhood distortion functions (6) and (7) can now be expressed using the notation of WVQ, as in (2). We use $\delta(x, y) = |x - y|^2$ so that the WVQ codevector placement condition holds.

Intuitively, the average meansquare distortion appears to be the more useful measure, because it compensates for the fact that at the boundary of the SOFM fewer nodes contribute to the distortion.

5 A Learning Algorithm for SOFM

By incorporating SOFM neighborhoods into WVQ, we obtain a the following new algorithm for training SOFM. Significantly, distortion is monotonically reduced (and thus topology preservation is monotonically increased) with this algorithm. Like the generalized Lloyd algorithm, this algorithm produces locally optimal codebooks. We plan to study the use of simulated annealing and related techniques to produce better codebooks.

1. *Initialize codevectors, set threshold for termination.*
2. *Loop until decrease in distortion is below threshold.*
   (a) *Partition the training set using the WVQ partitioning rule.*
   (b) *Compute new codebook using the WVQ codevector placement condition and the partitioning from step (a)*
   (c) *Compute WVQ distortion summed over the training set, and compute the decrease in WVQ distortion compared to the last iteration.*

Our current implementation of this algorithm stores the SOFM weight vectors as a linear array, and the W matrix defines the arrangement of the SOFM units and the neighborhood function. By specifying the weight matrix W, the number of units n, and the number of components in input vector, any SOFM can be trained. This includes SOFM with units in more than two dimensions.

The use of neighborhood distortion measures to train SOFM also yields a criterion for stopping training, unlike Kohonen learning where training is stopped in an ad hoc manner.

6 Conclusions

Our formulation for weighted vector quantization enables us to train SOFM to preserve topology by minimizing neighborhood distortion measures. The algorithm is simple, easy to vectorize, and has a uniform representation for any SOFM. Furthermore, neighborhood distortion is monotonically reduced with each pass through the training set.

Currently we are studying how to decrease the size of the neighborhood to minimize the distortion when the neighborhood is small. Also, we are developing a pattern learning version of the algorithm, which will be more amenable to neural implementation. We are also interested in improving the efficiency of the partitioning, which is currently the main time bottleneck of the algorithm.

This paper has focused on the theoretical portion of our work. On the practical side, we are also running speech recognition experiments to compare the performance of SOFM trained with this algorithm to those trained with Kohonen's algorithm.

References

[1] De Haan, G. R., "Kohonen Nets for ASR", *STL Symposium*, Osaka, Japan, 1990. Speech Technology Laboratory, 3888 State Street Suite 202, Santa Barbara, CA 93117 USA.

[2] De Haan, G. R., and Ö. Eğecioğlu, "Neighborhood distortion functions and self–organizing feature maps", *International Joint Conference on Neural Networks*, July 1991.

[3] Gersho, A., "On the structure of vector quantizers," *IEEE Transactions on Information Theory*, IT-28, 157–166, 1982.

[4] Hecht-Nielsen, R., *Neurocomputing*, Addison-Wesley, Reading, MA, 1990.

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

[6] Kohonen, T., *Self-Organization and Associative Memory*, (Second Edition), Springer-Verlag, Berlin, 1988.

[7] Kohonen, T., "The "neural" phonetic typewriter," *Computer*, 21, 11–22, 1988.

[8] Kohonen, T., "The self-organizing map," *Proceedings of the IEEE*, 78, 1464–1480, 1990.

[9] Luttrel, S. P., "Derivation of a class of training algorithms," *IEEE Transactions on Neural Networks*, 1, 229–232, 1990.

[10] Torkkola, K., and M. Kokkonen,, "Using the topology-preserving properties of SOFMS in speech recognition," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Volume I, 261–264, 1991.

[11] Werbos, P. J., "Generalization of backpropagation with application to a recurrent gas market model," *Neural Networks*, 1, 339–356, 1989.

[12] Zeger, K., *Source and Channel Coding with Vector Quantization*, Ph.D. Dissertation, UC Santa Barbara, 1990.