In Proceedings of ICANN'99, Ninth International Conference on Artificial Neural Networks, Edinburgh, UK, 7-10 September, to appear

Fast Winner Search for SOM-Based Monitoring and Retrieval of High-Dimensional Data

Samuel Kaski
Helsinki University of Technology
Neural Networks Research Centre
P.O. Box 5400, FIN-02015 HUT, Finland
samuel.kaski@hut.fi

Abstract

Self-Organizing Maps (SOMs) are widely used in engineering and data-analysis tasks, but so far rarely in very large-scale problems. The reason is the amount of computation: while small SOMs can be computed starting from the basic principles, rapid computation of large maps of high-dimensional data requires special methods. Winner search, finding the position of a data sample on the map, is the computational bottleneck: comparison between the data vector and all of the model vectors of the map is required. In this paper a method is proposed for reducing the amount of computation by restricting the search to certain small-dimensional subspaces of the original space. The method is suitable for applications in which the map can be computed off-line, for instance in data monitoring, classification, and information retrieval. In a case study with the WEBSOM system that organizes text document collections on a SOM, the amount of computation was reduced to about 14% of the original, and even to 6.6% when approximations were utilized.

1 Introduction

The self-organizing map (SOM) algorithm (Kohonen, 1982; Kohonen, 1995) forms a kind of a nonlinear regression of an ordered set of model vectors $\mathbf{m}_i$, $i = 1, \ldots, M$, into the data space $\mathbb{R}^n$. Each model vector has been attached to a map unit on a regular map lattice. In SOM-based data analysis the aim is to extract and illustrate the essential structures within a statistical data set by a map that, as a result of the unsupervised learning process, follows the distribution of the data in the input space. Each data sample is mapped to the unit having the most similar model vector, whereby the relations of the data samples become reflected in geometrical relations (order) of the samples on the map.

The mapping of a data vector $\mathbf{x}$ onto the SOM is done by a winner search, i.e. by finding the vector $\mathbf{m}_c$ that is closest to $\mathbf{x}$. In this paper the closeness is judged based on the inner products $\mathbf{x}^T\mathbf{m}_i$ between the normalized vectors $\mathbf{x}$ and $\mathbf{m}$.

If the dimensionality n of the input space is high and the number of map units M is large, the number of computations required for mapping an input vector is very large, of the order of Mn. The aim of this paper is to reduce this amount of computation. The presented methods are aimed especially at reducing the amount of on-line computation needed for mapping a new data sample onto a fixed SOM that has been computed off-line.

2 Tree search in low-dimensional subspaces

In this section the principle of the method will be presented in a general form. An efficient practical implementation of the principle will be presented later in Section 3.

The general task is to find, from a set of model vectors $\{\mathbf{m}_i\}_{i=1}^M$, the model vector for which the inner product with the input vector $\mathbf{x}$ is the largest. This general task is a part of many clustering and information retrieval problems.

2.1 Search in a low-dimensional subspace

If the model vectors would lie on a small-dimensional subspace of the original space, then the rest of the dimensions of the original space could be neglected in the winner search. Even if the subspace spanned by the vectors was not low-dimensional it is, of course, possible to *approximate* it with a lower-dimensional one. The intention here is to build such a low-dimensional subspace, and to compute the inner products first in that subspace to find a *tentative winner*, indexed with c^*. The tentative winner is the model vector that is closest to the input vector, *measured within the subspace.* The projections of all model vectors into the subspace can be computed off-line, and therefore in the on-line phase the dimensionality of the distance computations is effectively reduced to that of the subspace.

Once the tentative winner has been selected and the full inner product $\mathbf{x}^T\mathbf{m}_{c^*}$ with it has been computed, an upper bound is computed for all the other inner products. Thereafter only those model vectors for which the upper bound is larger than $\mathbf{x}^T\mathbf{m}_{c^*}$ need be checked. These computation steps are explained in more detail below.

It should perhaps be emphasized that this winner search method always finds the real winner. The approximations are used only as intermediate steps to speed up the computation. The amount of speed-up depends on the chosen subspace.

2.1.1 On-line computation

Denote the projection of $\mathbf{x}$ onto the low-dimensional subspace by $\hat{\mathbf{x}}$ and the residual by $\tilde{\mathbf{x}}$, and similarly $\mathbf{m}_i = \hat{\mathbf{m}}_i + \tilde{\mathbf{m}}_i$.

Step 1: Find the tentative winner by computing inner products within the subspace. The inner product

$$\hat{\mathbf{x}}^T\hat{\mathbf{m}}_i \tag{1}$$

is computed and tabulated for each i, and the model vector for which the inner product is largest is chosen to be the tentative winner indexed with c^*.

Step 2: Computation of full inner product with the tentative winner,

$$\mathbf{x}^T\mathbf{m}_{c^*} \ . \tag{2}$$

Step 3: Find upper bounds for the other full inner products. Each inner product can be decomposed into

$$\begin{aligned} \mathbf{x}^T\mathbf{m}_i &= (\hat{\mathbf{x}}+\tilde{\mathbf{x}})^T(\hat{\mathbf{m}}_i+\tilde{\mathbf{m}}_i) \\ &= \hat{\mathbf{x}}^T\hat{\mathbf{m}}_i + \tilde{\mathbf{x}}^T\tilde{\mathbf{m}}_i \ . \end{aligned}$$

The first term in the sum has already been tabulated in Step 1, and the second is bounded from above by $\|\tilde{\mathbf{x}}\|\|\tilde{\mathbf{m}}_i\|$. The whole inner product is bounded from above by

$$\mathbf{x}^T\mathbf{m}_i \le \hat{\mathbf{x}}^T\hat{\mathbf{m}}_i + \|\tilde{\mathbf{x}}\|\|\tilde{\mathbf{m}}_i\| \equiv B_i(\mathbf{x}) \ , \tag{3}$$

where $B_i(\mathbf{x})$ has been used to denote the upper bound.

Step 4: Check the "rivals". Go through all units i, and if the upper bound for the inner product is larger than the inner product with the tentative winner,

$$B_i(\mathbf{x}) > \mathbf{x}^T\mathbf{m}_{c^*} \ , \tag{4}$$

compute the full inner product $\mathbf{x}^T\mathbf{m}_i$. If it is larger than the full inner product of the present tentative winner then choose the unit i the new tentative winner.

After Step 4 has been completed the tentative winner is the final winner.

2.1.2 Off-line computation

Before mapping new samples, the projection $\hat{\mathbf{m}}_i$ and the norm of the residual, $\|\tilde{\mathbf{m}}_i\|$, must be computed for each i.

2.2 Tree search

If the set of model vectors is very inhomogeneous it may be not be possible to approximate the whole set well with a low-dimensional subspace. However, if the model vectors are divided into relatively more homogeneous subsets then a better approximation can be made in each subset.

If there is a different subspace in each subset then input needs to be projected onto all subspaces. It is, however, possible to divide the subsets of model vectors recursively into smaller subsets, whereby a tree of subsets and the corresponding subspaces is formed.

Let us first consider a two-level tree consisting of one set and its subsets.

2.2.1 On-line computation

Step 1: Find a tentative winner. The subsets can be utilized here: we can first search for the *tentative subset*, the centroid of which is closest to the input, and then search for the closest model vector within the subspace associated with the tentative subset.

It may be useful to note that goodness of the method of choosing the tentative winner may affect the speed of computation but it does not affect the result itself.

Step 2: Computation of full inner product (2) with the tentative winner.

Step 3: Find upper bounds for the full inner products *for whole subsets.*

Denote by $I^{(j)}$ the set of indexes of model vectors that belong to the jth subset. If $i \in I^{(j)}$ the inner product can be bounded from above by

$$\begin{aligned} \mathbf{x}^T\mathbf{m}_i &= \hat{\mathbf{x}}^T\hat{\mathbf{m}}_i + \tilde{\mathbf{x}}^T\tilde{\mathbf{m}}_i \\ &\leq \|\hat{\mathbf{x}}\| \max_{i\in I^{(j)}} \|\hat{\mathbf{m}}_i\| + \|\tilde{\mathbf{x}}\| \max_{i\in I^{(j)}} \|\tilde{\mathbf{m}}_i\| \\ &\equiv B^{(j)}(\mathbf{x}) \ , \end{aligned} \tag{5}$$

The limit $B^{(j)}(\mathbf{x})$ depends only on the subset j and not on the individual model vectors within the subset.

Step 4: Check the "rival subsets". Go through all subsets j, and if the upper bound for the inner products is larger than the inner product with the tentative winner,

$$B^{(j)}(\mathbf{x}) > \mathbf{x}^T\mathbf{m}_{c^*} \ , \tag{6}$$

then search the jth subset with the methods described in Section 2.1, starting from Step 3.

After Step 4 has been completed the current tentative winner is the final winner.

2.2.2 Off-line computation.

In addition to the off-line computations for each of the subsets, described in Section 2.1, the maximal norms and residuals, $\max_{i\in I^{(j)}} \|\hat{\mathbf{m}}_i\|$ and $\max_{i\in I^{(j)}} \|\tilde{\mathbf{m}}_i\|$, must be computed for each subset j. Additionally, the centroids needed when searching for the tentative subset in Step 1 must be computed.

Above, only a two-level tree consisting of a set and its subsets was considered for simplicity. The search can, however, be easily generalized to a tree search in which each subset is divided recursively into subsets. At each level the maximums in (5) must be computed over all subsets of the current set.

3 Fast search in SOMs

In the general description above we did not discuss methods for choosing the subsets or subspaces. In principle any method can be applied, but the better the methods are the larger the savings in the computational time.

In this section a method is presented for choosing both the subsets and the subspaces in the special case that the model vectors have been computed with the SOM algorithm. It will be described below how special properties of the SOM can be utilized in making reasonable choices that require relatively little computational time and memory capacity. The usefulness of the approach will be demonstrated in Section 5.

3.1 Selection of subsets and subspaces

The model vectors of the SOM that are close-by on the map lattice are generally close-by in the input space as well. Moreover, the map can be thought of as an "elastic network" in the input space, formed by connecting the model vectors of neighboring units with lines (for demonstrations, cf. Kohonen, 1995). If the map lattice is two-dimensional the elastic network is two-dimensional as well, and the more "stiff" the network is the better it can be approximated locally with even a two-dimensional subspace. The stiffness can be controlled by changing the width of the neighborhood function of the SOM.

These properties of the SOM suggest a computationally efficient strategy for selecting the subsets and subspaces: Each set of model vectors corresponds to a local area on the map lattice, and the corresponding subspace represents the "elastic network" locally at the area. In this paper the following computationally simple methods are used in choosing the areas and the subspaces.

Choosing the subsets. On a hexagonal map lattice choose every Nth unit to form a sparser lattice (Fig. 1). The triangles of the

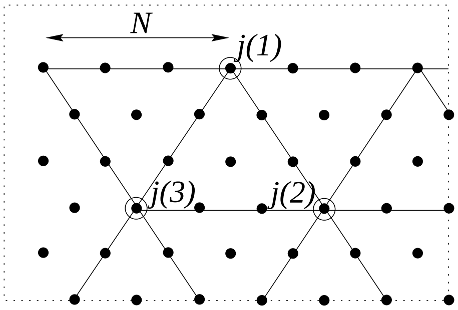

Figure 1: The method used for forming the subsets on a hexagonal SOM. A sparser lattice of triangles is formed by joining every Nth (here N=3) unit on the map lattice, with appropriate shifts on every second row. Each subgroup contains the model vectors that are attached to the units within one triangle. The units located exactly at the borders in between the triangles may be assigned to either triangle or both, and the "extra" units on the edges may be assigned to their closest triangle. The indexes of the units in the corners of the jth triangle have been marked with $j(1)$, $j(2)$, and $j(3)$.

new lattice will be used as the local areas, and each subset consists of the units within one triangle.

Choosing the subspaces. For subset ("triangle") j, choose the subspace spanned by the three model vectors at the corners of the triangle. A basis for such a three-dimensional subspace can be formed rapidly and stored into a small amount of memory as will be discussed below.

3.2 Characteristics of the chosen subspaces

Denote the model vectors at the corners of the jth triangle by $\mathbf{m}_{j(1)}$, $\mathbf{m}_{j(2)}$, and $\mathbf{m}_{j(3)}$ (cf. Fig. 1). The vectors are generally not orthonormal but they can be orthonormalized with the so-called Gram-Schmidt orthogonalization procedure (see, for example, Kohonen, 1995). Denote by $U^{(j)}$ the matrix in which the rows are the orthonormalized vectors. After some manipulation of the Gram-Schmidt-equations it can be shown that

$$U^{(j)} = S^{(j)} \begin{bmatrix} \mathbf{m}_{j(1)}^T \\ \mathbf{m}_{j(2)}^T \\ \mathbf{m}_{j(3)}^T \end{bmatrix} , \qquad (7)$$

where $S^{(j)}$ is a lower triangular matrix having three rows and three columns.

Since the model vectors have already been stored in memory, equation (7) implies that only the 6 lower-diagonal values of $S^{(j)}$ need be stored instead of the whole matrix $U^{(j)}$, irrespective of the dimensionality of the input space.

When computing the projection $U^{(j)}\mathbf{x}$ we get as a side product the full inner products with three model vectors; cf. (7).

3.3 The search

Step 1: Find the tentative winner. First find the tentative subset in which the inner product between the input and the centroid of the set is the largest. Memory and computational time can be saved by approximating the centroid with the centroid of the endpoints of the triangle. Then the inner product between the input and the centroid, denoted by $\bar{\mathbf{x}}^{(j)}$, is simply

$$\mathbf{x}^T \bar{\mathbf{x}}^{(j)} = \sum_{k=1}^{3} \mathbf{x}^T \mathbf{m}_{j(k)} \ . \qquad (8)$$

Continue this search to the bottom of the tree. At the bottom, find the tentative winner from the most promising subset using the method described in Section 2.1.

Step 2: Computation of the full inner product (2) with the tentative winner.

Step 3: Start from the top of the tree, and make a depth-first search to find out which subsets need to be searched. At each level of the tree use (6) as the criterion, and if the criterion is met descend to the subsets. When at the bottom of the tree use the methods described in Section 2.1 to find a potential new tentative winner.

After step 3 has been completed the tentative winner is the final winner.

4 Approximate search in SOMs

If some proportion of errors can be allowed, then additional speed-up can be achieved by skipping some of the checks for rivals.

In the approximate search used in Section 5 below the rivals will be checked only within the subset containing the tentative winner selected in Step 1. Some "expert knowledge" on SOMs will, however, be used for enhancing the accuracy of the search.

It is easy to guarantee that the winner is a *local winner* on the map grid by comparing the inner product of the tentative winner with the inner products of neighboring units on the map lattice. If the inner product is larger for some of the neighbors, the neighbor will be chosen to be the new tentative winner. The search will be continued until the middle-most vector remains the winner. This approach has been used by Kohonen (1997) to speed up the computation of SOMs.

5 Experiments: mapping of textual documents in the WEBSOM system

5.1 The WEBSOM system

WEBSOM (Honkela et al., 1996; Kaski et al., 1998; Kohonen et al., 1999) is a method for organizing textual documents onto a two-dimensional map display. Nearby locations on the display contain similar documents which aids in browsing the document collection. The map can also be used for content-addressable search, and for filtering interesting documents from an incoming document stream.

Mapping new documents onto a document map formed by the SOM algorithm will be used here as a case study of the new winner search methods. For the sake of clarity the case study does not, however, include all of the possible ingredients of the WEBSOM system.

5.2 The setting

In a simple but yet very effective document encoding method, called the vector space model (Salton and McGill, 1983), the documents are represented by vectors in a space where each dimension corresponds to one word. The value of each component is equal to the relative frequency of occurrence of the corresponding word in the document. Alternatively, some function of the frequency of occurrence and the importance of the word may be used. The resulting vectors can be thought of as representing the *word histograms* of the documents. When the length of the document vector is normalized the direction of the vector will reflect the contents of the document.

In the present case study the document collection contained 118767 patent abstracts that had been divided into 21 groups according to their topic. The documents were encoded with the vector space model, in which the dimensions were weighted with an entropy-based weight (cf. Kaski et al., 1998). After the rarest and some common words were removed there remained 5532 words, and the input vectors were therefore *5532-dimensional.*

The document map contained 12096 units.

The new winner search methods were tested by mapping a total of 114573 new patent abstracts onto the document map. A three-level search tree was constructed by selecting, at each level, every third unit to the coarser map lattice that defines the subsets, as was described in Sec. 3. The approximate search described in Section 4 was tested with a two-level tree, in which every fourth unit was selected. These values were chosen based on preliminary experiments with a small cross-validation set.

5.3 Results

When the three-level search tree was used, the number of full distance computations reduced to **14 %** of the original. With the approximate search the number was reduced to **6.6 %** of the original. These figures include the inner product computations needed for finding the tentative winner and for projecting the input into the low-dimensional subspaces. The amount of other computation is negligible compared to the full distance computations in a very high-dimensional space. In the present case study, for instance, even if inner products between the input and *all* model vectors were computed in a three-dimensional subspace, the total number of multiply-accumulate operations would still be less than 1% of the number resulting from the full distance computations in the fast approximate search mode.

To evaluate the quality of the approximate search we labeled each map unit with the patent group that formed the majority within that unit. We then counted how large a proportion of the new samples ended up in units where the majority belonged to the same group. In effect, then, we classified the new samples to patent classes using the document map, and measured the classification error. It may be useful to note that the classes here are very overlapping and therefore the "classification error" is expected to be large. The results are, however, useful for comparing different methods, like the different winner search methods here.

Accurate winner searches placed 57% of the abstracts in map units in which the same class dominated, whereas for the approximate search the proportion was 56%.

The result suggests that even the approximate search is probably accurate enough for many applications.

6 Discussion

The methods presented in this paper have been aimed at speeding up the mapping of new high-dimensional data vectors onto an already computed Self-Organizing Map. Winners can be searched accurately from any SOMs. This, in addition to the new method of using low-dimensional subspaces, are the main differences compared with the tree-structured SOM (Koikkalainen, 1995), a variant of SOM that uses tree search.

Applications in which fast mapping methods would be useful include data monitoring, in which a large map has been computed off-line and new data samples are monitored on-line using the map display. In clustering and classification tasks the clusters or class prototypes can often be computed off-line as well. Such clusterings are useful in, for example, information retrieval applications.

The search methods may even be beneficial for *computation of SOMs*, if the Batch Map algorithm (Kohonen, 1995) is used and if the map has been initialized well enough so that sets of model vectors can be approximated reasonably well with low-dimensional subspaces. In the Batch Map algorithm the winning unit is first searched for for each data vector, and the map is modified only after that. Therefore, the subspaces need to be recomputed only once for each batch.

References

Honkela, T., Kaski, S., Lagus, K., and Kohonen, T. (1996). Newsgroup exploration with WEBSOM method and browsing interface. Technical Report A32, Helsinki University of Technology, Laboratory of Computer and Information Science, Espoo, Finland.

Kaski, S., Honkela, T., Lagus, K., and Kohonen, T. (1998). WEBSOM—self-organizing maps of document collections. *Neurocomputing*, 21:101–117.

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

Kohonen, T. (1995). *Self-Organizing Maps*. Springer, Berlin. (Second, extended edition 1997).

Kohonen, T., Kaski, S., Lagus, K., Salojärvi, J., Honkela, J., Paatero, V., and Saarela, A. (1999). Self organization of a massive text document collection. In Oja, E. and Kaski, S., editors, *Kohonen Maps*, pages 171–182. Elsevier, Amsterdam. In press.

Koikkalainen, P. (1995). Fast deterministic self-organizing maps. In Fogelman-Soulié, F. and Gallinari, P., editors, *Proceedings of ICANN'95, International Conference on Artificial Neural Networks*, volume II, pages 63–68, EC2, Nanterre, France.

Salton, G. and McGill, M. J. (1983). *Introduction to modern information retrieval.* McGraw-Hill, New York.