On the Basis Updating Rule of Adaptive-Subspace Self-Organizing Map (ASSOM)

Huicheng Zheng, Christophe Laurent, Grégoire Lefebvre

On the Basis Updating Rule of Adaptive-Subspace Self-Organizing Map (ASSOM)*

Huicheng Zheng, Christophe Laurent, and Grégoire Lefebvre

France Telecom R&D – DIH/HDM
4, Rue du Clos Courtel
35512 Cesson Sévigné Cedex, France

Abstract. This paper gives other views on the basis updating rule of the ASSOM proposed by Kohonen. We first show that the traditional basis vector rotation rule can be expressed as a correction to the basis vector which is proportional to component vectors in the episode. With the latter form, some intermediate computations can be reused, leading to a computational load only linear to the input dimension and the subspace dimension, whereas naive implementation of the traditional rotation rule has a computational load quadratic to the input dimension. We then proceed to propose a batch-mode updating of the basis vectors. We show that the correction made to each basis vector is a linear combination of component vectors in the input episode. Computations can be further saved. Experiments show that the proposed methods preserve the ability to generate topologically ordered invariant-feature filters and that the learning procedure is largely boosted.

1 Introduction

Adaptive-subspace self-organizing map (ASSOM) [1] is basically a combination of the competitive selection and cooperative learning as in the traditional SOM [2] and a subspace method. The single weight vectors at map units in SOM are replaced by modules of basis vectors in ASSOM that span some linear subspaces. ASSOM is an alternative to the standard principal component analysis (PCA) method of feature extraction. An earlier neural approach for PCA can be found in [3]. The ASSOM can generate spatially ordered feature filters thanks to spatial interactions among processing units [4]. Each module in ASSOM can be realized as a neural network which receives input vectors and outputs their orthogonal projections on the represented subspaces.

The input to an ASSOM array is typically an episode, i.e. a sequence of pattern vectors supposed to approximately span some linear subspace. These vectors shall also be referred to as component vectors of the episode in this paper. By learning the episode as a whole, ASSOM is able to capture the transformation coded in the episode. The simulation results in [1] and [4] have demonstrated that ASSOM can induce ordered filter banks to account for translation, rotation and scaling. The relationship between the neurons in the ASSOM architecture and their biological counterparts are reported [4]. ASSOM has been applied to speech processing [5], texture segmentation [6], image

* This work was carried out during the tenure of a MUSCLE Internal fellowship (http://www.muscle-noe.org).

retrieval [7] and image classification [7], [8], etc. in the literature. A supervised variant of ASSOM, called supervised adaptive-subspace self-organizing map (SASSOM), was used by Ruiz del Solar in [6].

The basis vector rotation rule in the traditional ASSOM implementation takes a form of matrix multiplication. This rule is hard to understand and more seriously, naive implemenation of the basis vector rotation rule leads to a computational load which is quadratic to the input dimension, not to mention large amount of memory required by the usually high-dimensional matrix operations. This deficiency renders naive implementation of the basic ASSOM learning very costly for practical applications.

There were efforts in the literature to reduce the computational load associated with the basic ASSOM learning. De Ridder et al. [7] dropped topological ordering to reduce the computations involved in the cooperative learning. Furthermore, they performed a batch-mode updating of subspaces with PCA to avoid the time-consuming iterative updating. Similarly, López-Rubio et al. [9] used PCA with ASSOM to perform PCA while retaining self-organization of generated features. The resulting algorithm is named PCASOM. According to their report, under similar classification performance, their algorithm runs about twice faster than the basic ASSOM. McGlinchey et al. [10] replaced the traditional basis vector updating formula with one proposed by Oja [11]. According to their paper, the computational load is only linear to the input dimension, but quadratic to the subspace dimension.

In this paper, we first show that with the traditional basis rotation rule, the correction made to each basis vector is in fact a vector proportional to the component vector of the input episode. With this modified form, some intermediate computations can be reused, leading to a computational load only linear to both the input dimension and the subspace dimension. We then proceed to propose a batch-mode updating of the basis vectors, where the correction made to each basis vector is a linear combination of component vectors in the episode. This modified rule further accelerates the learning procedure by saving large amounts of computations.

This paper will be organized as follows: In Sect. 2, we review briefly the basic ASSOM learning procedure. The proposed alternative updating rules will be presented in Sect. 3. Section 4 is dedicated to experiments which demonstrate the performance of the proposed methods. This paper will be concluded by Sect. 5.

2 The Basic ASSOM Learning

An ASSOM is composed of an array of modules. Each module in the ASSOM can be realized by a two-layered neural network [4], as shown in Fig. 1. It calculates the projection of an input vector $\mathbf{x}$ on the subspace $\mathcal{L}$ of the module. Supposing $\mathcal{L}$ is spanned by a set of basis vectors $\{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_H\}$, where H is the dimension of $\mathcal{L}$, the neurons in the first layer take the orthogonal projections $\mathbf{x}^{\mathrm{T}}\mathbf{b}_h$ of the input vector $\mathbf{x}$ on the individual basis vectors $\mathbf{b}_h$. The basis vectors are supposed to be orthonormalized. The only quadratic neuron of the second layer sums up the squared outputs of the first-layer neurons. The output of the module is then $\|\hat{\mathbf{x}}_{\mathcal{L}}\|^2$, the squared norm of the projection of $\mathbf{x}$ on the subspace $\mathcal{L}$. It can be regarded as a measure of the matching between the input vector $\mathbf{x}$ and the subspace $\mathcal{L}$. For an input episode $\mathbf{x}(s)$, $s \in S$, where S is the index set

of vectors in the episode, Kohonen proposed to use the *energy* $\sum_{s\in S} \|\hat{\mathbf{x}}_{\mathcal{L}}(s)\|^2$, i.e. the sum of squared norms of the projections of the individual vectors, as the measure [4].

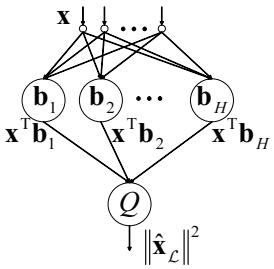


Fig. 1. A module of ASSOM realized as a neural network. $\mathbf{x}$ is an input vector. $\{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_H\}$ is an orthonormal basis of the linear subspace $\mathcal{L}$ of the module. Q is a quadratic neuron that sums up squares of its inputs

The learning process of ASSOM approximately minimizes an error function in an iterative way [4]. The iteration at step t of the basic ASSOM learning procedure proceeds as follows:

1. For the episode $\mathbf{x}(s)$, $s \in S$, locate the winning module indexed by $c = \arg\max_{i\in I} \sum_{s\in S} \|\hat{\mathbf{x}}_{\mathcal{L}_i}(s)\|^2$, where I is the index set of modules in the ASSOM.
2. For each module i in the neighborhood of c, including c itself, update the subspace $\mathcal{L}_i$ for each component vector $\mathbf{x}(s)$, $s \in S$, that is, update the basis vectors $\mathbf{b}_h^{(i)}$, according to the following rules:
   (a) Rotate each basis vector according to:

$$\mathbf{b}_h^{(i)} = \mathbf{P}_c^{(i)}(\mathbf{x}, t)\mathbf{b}_h^{'(i)} \quad , \tag{1}$$

   where $\mathbf{b}_h^{(i)}$ is the new basis vector and $\mathbf{b}_h^{'(i)}$ the old one. The matrix $\mathbf{P}_c^{(i)}(\mathbf{x}, t)$ is a rotation operator defined by:

$$\mathbf{P}_c^{(i)}(\mathbf{x}, t) = \mathbf{I} + \lambda(t)h_c^{(i)}(t)\frac{\mathbf{x}(s)\mathbf{x}^{\mathrm{T}}(s)}{\|\hat{\mathbf{x}}_{\mathcal{L}_i}(s)\|\|\mathbf{x}(s)\|} \quad , \tag{2}$$

   where $\mathbf{I}$ is the identity matrix, $\lambda(t)$ a learning-rate factor that diminishes with t. $h_c^{(i)}(t)$ is a neighborhood function defined on the ASSOM lattice.
   (b) Dissipate the basis vectors $\mathbf{b}_h^{(i)}$ to improve stability of the results [4] and then orthonormalize these basis vectors.

Through this competitive and cooperative learning procedure, the ASSOM will finally arrive at a topologically organized status, where nearby modules represent similar feature subspaces. Naive implementation of (1) requires a matrix multiplication which needs not only a large amount of memory, but also a computational load quadratic to the input dimension. It would be costly for practical applications of ASSOM.

3 On the Basis Updating Rule of ASSOM

3.1 Insight on the Basis Vector Rotation

In the first place we propose to replace the formulae (1) and (2) through a little mathematical deduction. The term $\mathbf{b}_h^{'(i)}$ in (1) can be distributed to the right side of (2), leading to the current basis vector and a correction to it:

$$\mathbf{b}_h^{(i)} = \mathbf{b}_h^{'(i)} + \Delta\mathbf{b}_h^{(i)} \ , \tag{3}$$

where

$$\Delta\mathbf{b}_h^{(i)} = \lambda(t)h_c^{(i)}(t)\frac{\mathbf{x}(s)\mathbf{x}^{\mathrm{T}}(s)\mathbf{b}_h^{'(i)}}{\|\hat{\mathbf{x}}_{\mathcal{L}_i}(s)\|\|\mathbf{x}(s)\|} \ . \tag{4}$$

$\mathbf{x}^{\mathrm{T}}(s)\mathbf{b}_h^{'(i)}$ is in fact a scalar value. The equation can be rewritten as:

$$\Delta\mathbf{b}_h^{(i)} = \alpha_{c,h}^{(i)}(s,t)\mathbf{x}(s) \ . \tag{5}$$

Here $\alpha_{c,h}^{(i)}(s,t)$ is a scalar value defined by:

$$\alpha_{c,h}^{(i)}(s,t) = \lambda(t)h_c^{(i)}(t)\frac{\mathbf{x}^{\mathrm{T}}(s)\mathbf{b}_h^{'(i)}}{\|\hat{\mathbf{x}}_{\mathcal{L}_i}(s)\|\|\mathbf{x}(s)\|} \ . \tag{6}$$

This shows that the correction $\Delta\mathbf{b}_h^{(i)}$ is in fact proportional to the component vector $\mathbf{x}(s)$, as illustrated in Fig. 2, which seems to have been ignored by many practitioners. Equation (5) gives a clearer way to understand the basis vector rotation in ASSOM learning than the traditional rotation matrix (2). Note that in (6), $\mathbf{x}^{\mathrm{T}}(s)\mathbf{b}_h^{'(i)}$ is the projection of the component vector on the basis vectors represented by the neurons of the first layer, which we have already when computing the projection $\|\hat{\mathbf{x}}_{\mathcal{L}_i}(s)\|$ (cf. Fig. 1). If we calculate the scaling factor $\alpha_{c,h}^{(i)}(s,t)$ first, and then scale the component vector $\mathbf{x}(s)$ with this factor, the computations associated with the basis vector updating will be dramatically reduced. This implementation will be referred to as FL-ASSOM for fast-learning ASSOM. It is completely equivalent to the basic ASSOM in terms of generating topologically ordered invariant-feature filters.

Let us compare the computational loads of the basis vector updating in the basic ASSOM and FL-ASSOM by analyzing the respective updating formulae. We assume the input dimension to be N, and the subspace dimension to be M. We first evaluate the computations required by naive implementation of the traditional basis vector updating rule (1). For each component vector $\mathbf{x}(s)$, $\mathbf{x}(s)\mathbf{x}^{\mathrm{T}}(s)$ in (2) needs N^2 multiplications, the matrix multiplication in (1) amounts to MN^2 multiplications. There are totally about $MN^2 + N^2$ multiplications. Similarly, the number of additions required by (1) can be shown to be around $MN^2 + N^2$. Finally, the computational load of naive implementation of the traditional updating rule is approximately $O(MN^2)$, i.e. quadratic to the input dimension and linear to the subspace dimension. The replacement proposed by McGlinchey et al. [10] leads to a computational load of $O(M^2N)$, as shown in their paper, i.e. linear to the input dimension but quadratic to the subspace dimension. Now

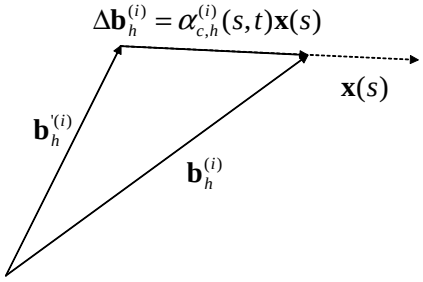


Fig. 2. An alternative view of the basis vector updating rule of ASSOM. The correction $\Delta\mathbf{b}_h^{(i)}$ is proportional to the component vector $\mathbf{x}(s)$. After updating, $\mathbf{b}_h^{(i)}$ represents better the component vector $\mathbf{x}(s)$

with the proposed updating rule (5), for each component vector $\mathbf{x}(s)$, the computations of $\|\hat{\mathbf{x}}_{\mathcal{L}_i}(s)\|$ and $\|\mathbf{x}(s)\|$ in (6) need about $MN+2N$ multiplications, and $\alpha_{c,h}^{(i)}(s,t)\mathbf{x}(s)$ in (5) about MN multiplications. In all (5) needs about $2MN + 2N$ multiplications. Similarly, the number of additions can be shown to be about $2MN + 2N$. So with (5), the computational load is approximately $O(MN)$, i.e. linear to both the input dimension and the subspace dimension. So there is an obvious benefit in using (5) other than naive implementation of (1).

3.2 Further Boosting: Batch-mode Basis Vector Updating

Basis vector updating can be further boosted by working in a batch mode. We can avoid computing the value of $\|\hat{\mathbf{x}}_{\mathcal{L}_i}(s)\|$ in (6) by using the value computed previously during module competition. However this could not be done inside the framework of FL-ASSOM since the subspaces are continuously changing in receiving each component vector of the episode. To save computation of $\|\hat{\mathbf{x}}_{\mathcal{L}_i}(s)\|$, the following batch-mode rotation operator [4] will be useful:

$$\mathbf{B}_c^{(i)}(t) = \mathbf{I} + \lambda(t)h_c^{(i)}(t)\sum_{s\in S}\frac{\mathbf{x}(s)\mathbf{x}^{\mathrm{T}}(s)}{\|\mathbf{x}(s)\|^2} \quad . \tag{7}$$

With this rotation operator, each basis vector in the subspace will be rotated only once for the whole input episode.

For stability of the solution, we expect the magnitude of the correction made to the basis vectors to be monotonically decreasing with respect to $\|\hat{\mathbf{x}}_{\mathcal{L}_i}(s)\|$. We borrow the idea from the basic rotation operator $\mathbf{P}_c^{(i)}(\mathbf{x},t)$ [4] to divide the learning-rate factor $\lambda(t)$ by the scalar value $\|\hat{\mathbf{x}}_{\mathcal{L}_i}(s)\|/\|\mathbf{x}(s)\|$, which only changes the effective learning rate. The batch-mode rotation operator then becomes:

$$\mathbf{B}_c^{(i)}(t) = \mathbf{I} + \lambda(t)h_c^{(i)}(t)\sum_{s\in S}\frac{\mathbf{x}(s)\mathbf{x}^{\mathrm{T}}(s)}{\|\hat{\mathbf{x}}_{\mathcal{L}_i}(s)\|\|\mathbf{x}(s)\|} \quad . \tag{8}$$

As for FL-ASSOM, we distribute $\mathbf{b}_h^{'(i)}$ to terms in this operator. With similar deduction as in FL-ASSOM, the basis vector updating rule becomes:

$$\mathbf{b}_h^{(i)} = \mathbf{b}_h^{'(i)} + \Delta\mathbf{b}_h^{(i)} \quad , \tag{9}$$

where

$$\Delta\mathbf{b}_h^{(i)} = \sum_{s\in S} \left(\alpha_{c,h}^{(i)}(s,t)\mathbf{x}(s) \right) \quad . \tag{10}$$

The correction made to each basis vector is thus a linear combination of the component vectors in the episode. The difference between the updating rule (9) here and (3) is that the former updates the basis vectors in a batch mode for the whole episode while the latter updates the basis vectors for each component vector one by one.

The scalar parameter $\alpha_{c,h}^{(i)}(s,t)$ has the same form as (6) in FL-ASSOM:

$$\alpha_{c,h}^{(i)}(s,t) = \lambda(t)h_c^{(i)}(t)\frac{\mathbf{x}^{\mathrm{T}}(s)\mathbf{b}_h^{'(i)}}{\|\hat{\mathbf{x}}_{\mathcal{L}_i}(s)\|\|\mathbf{x}(s)\|} \quad . \tag{11}$$

The meaning of this equation is a little different from that of (6), where the subspace $\mathcal{L}_i(s)$ of the module i should be updated for each component vector $\mathbf{x}(s)$ in the episode and thus we could not reuse the computational results of module competition. Here in (11) the basis vector updating works in a batch mode, i.e. updating is performed only after the whole episode has been received. Therefore, $\|\hat{\mathbf{x}}_{\mathcal{L}_i}(s)\|$ and $\mathbf{x}^{\mathrm{T}}(s)\mathbf{b}_h^{'(i)}$ can reuse the results previously calculated during module competition. What we need to do is only store the calculated values in registers and fetch them when needed. The computational load of (11) is thus trivial. Furthermore, the dissipation as well as orthonormalization of basis vectors can be performed only once for each episode without loosing accuracy since the basis vectors are not updated during the episode. The computational load can thus be further reduced. This method will be referred to as BFL-ASSOM for batch-mode fast-learning ASSOM.

Let us estimate the computational load of BFL-ASSOM. For basis vector updating with (10), we estimate the computational load averaged on each component vector of the episode as we did for the basic ASSOM and FL-ASSOM. As has been mentioned, the calculation of $\alpha_{c,h}^{(i)}(s,t)$ according to (11) needs only trivial computation. The majority of computation is in (10). Averaged on each vector in the episode, the computational load required by basis vector updating with BFL-ASSOM is about MN multiplications and MN additions. Furthermore, since the dissipation and orthonormalization of basis vectors can be performed only once for each episode, the whole learning time can be further reduced.

4 Experiments

We first show that BFL-ASSOM can also generate the topologically ordered invariant-feature filters as the basic ASSOM. The results of FL-ASSOM will be shown as the ground truth since FL-ASSOM is mathematically equivalent to the basic ASSOM. One of the most common transformations occurred to images is translation. We will show

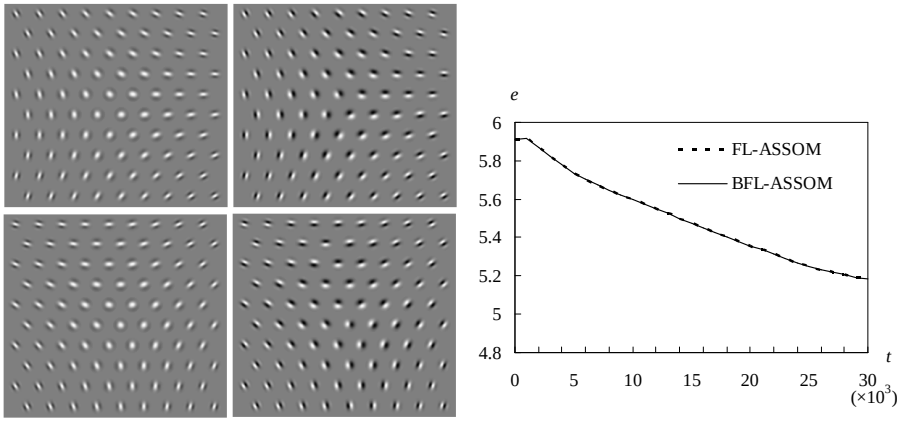
that BFL-ASSOM permits to generate Gabor type filters from episodes subject to translation.

The input episodes are constructed from a colored noise image, which is generated by filtering a white noise image with a second-order Butterworth filter. The cut-off frequency is set to 0.6 times of the Nyquist frequency of the sampling lattice. Each episode is composed of 6 vectors, each of which is formed on a circular receptive field on the sampling lattice composed of 349 pixels. The vectors in the same episode have only random translation of no more than 5 pixels in both the horizontal and the vertical directions. The episodes are generated on random locations of the colored noise image. The mean value of components of each input vector is subtracted from each component of the vector. In order to symmetrize the filters with respect to the center of the receptive field, the input samples are weighted by a Gaussian function symmetrically placed at the center of the receptive field with a full width at half maximum (FWHM) that varies linearly with respect to the learning step t from 1 to 16 sampling lattice spacings. Each vector is normalized before entering into the ASSOM array. The ASSOM array is composed of 9×10 modules aligned in a hexagonal lattice with two basis vectors at each module. The basis vectors of all the modules are initialized randomly and orthonormalized at the beginning of the learning process. The radius of the circular neighborhood function $h_c^{(i)}(t)$ decreases linearly from 6.73 ($= 0.5 \times (9^2 + 10^2)^{1/2}$) to 0.9 ASSOM array spacings with t. The learning-rate factor has the form $\lambda(t) = 0.1 \cdot T/(T + 99t)$, where T is the total number of learning steps and set to $30,000$ for the current experiment.

The translation-invariant filters generated by BFL-ASSOM compared to those by FL-ASSOM are shown in Fig. 3(a) with a gray scale. We can see that both methods generated topologically ordered Gabor-like filters. For either method, the two basis vectors at the same array locations have the same frequencies but 90 degrees of phase difference. Figure 3(b) shows how the average projection error e changes with the learning step t for FL-ASSOM and BFL-ASSOM. For each input episode $\mathbf{X} = \{\mathbf{x}(s), s \in S\}$, the projection error is calculated according to $e(\mathbf{X}) = \sum_{s \in S} \frac{\|\mathbf{x}(s) - \hat{\mathbf{x}}(s)\|^2}{\|\mathbf{x}(s)\|^2}$, where $\hat{\mathbf{x}}(s)$ is the orthogonal projection of $\mathbf{x}(s)$ on the subspace of the winning module. e is the average of $e(\mathbf{X})$ over all the training episodes. We can see that the curves of FL-ASSOM and BFL-ASSOM match very well in Fig. 3(b), which reveal that their difference in terms of generating the filters is indeed very little.

In the second experiment, we compare the computational loads of the basic ASSOM, FL-ASSOM and BFL-ASSOM. We designed the experiment by using C++ implementations of all the methods. In this experiment, the input dimension as well as the subspace dimension vary. We count the elapsed CPU seconds for different methods. The number of iterations are fixed to $1,000$. Each episode is composed of 6 vectors. These vectors are generated randomly according to a uniform probability distribution. The rectangular ASSOM array contains 10×10 modules.

The timing results are summarized in Table 1. As was anticipated, the time of updating basis vectors with the basic ASSOM increased sharply with the input dimension and moderately with the subspace dimension. Basis vector updating is the bottleneck of the basic learning procedure, especially when the input dimension is high. With FL-ASSOM, it is clear that the time of updating basis vectors increases much more mod-



(a) (b)

Fig. 3. (a) The Gabor type filters generated by BFL-ASSOM compared to those by FL-ASSOM on episodes subject to translation. *Top*: Filters generated by FL-ASSOM; *Bottom*: Filters generated by BFL-ASSOM. *Left*: First basis vectors. *Right*: Second basis vectors. (b) Change of the projection error e according to the learning step t for FL-ASSOM and BFL-ASSOM

erately with the input dimension. The response to the subspace dimension is also quite mild. Basis vector updating is no longer a bottleneck for the learning procedure. As a result, the learning time drops dramatically compared to the basic ASSOM. However learning time outside basis vector updating is not reduced. Now with BFL-ASSOM, we can observe that the basis vector updating time is further reduced. Moreover, learning time outside the basis vector updating is also reduced considerably compared to the basic ASSOM and FL-ASSOM.

The relationship between the basis vector updating time and the input dimension or the subspace dimension for the three implementations of ASSOM is visualized in Fig. 4. The basis vector updating time increases approximately linearly with respect to the input dimension for FL-ASSOM and BFL-ASSOM, but apparently nonlinearly for the basic ASSOM. In all the cases, the updating time increases approximately linearly with respect to the subspace dimension.

5 Conclusions

The focus of this paper is on the basis updating rule of the ASSOM learning. We first showed that the traditional basis rotation rule amounts to a correction made to the basis vectors which is proportional to the component vectors of the input episode. This gives us a better understanding of the basis updating in ASSOM learning. With this modified form of updating rule, some computations can be saved by reusing some intermediate computations. The resulting method is referred to as FL-ASSOM. Naive implementation of the traditional basis updating rule leads to a computational load linear to the

Table 1. The timing results for the basic ASSOM, FL-ASSOM and BFL-ASSOM. VU (vector updating time) denotes the time for the basis vector updating. WL (whole learning time) denotes the time for the whole learning procedure, including those for module competition, basis vector dissipation and orthonormalization. All the times are given in seconds

The basic ASSOM

	M=2		M=3		M=4	
	VU	WL	VU	WL	VU	WL
N=50	29.36	41.27	30.44	47.53	32.45	54.72
N=100	134.92	154.95	145.95	172.85	149.91	188.06
N=200	742.34	786.63	769.56	828.47	814.80	895.08
N=400	4529.69	4626.43	4956.64	5090.56	5200.78	5367.35

FL-ASSOM

	M=2		M=3		M=4	
	VU	WL	VU	WL	VU	WL
N=50	1.53	13.31	2.31	18.92	3.18	25.03
N=100	2.39	21.09	3.15	30.41	3.86	40.80
N=200	3.30	37.86	4.81	55.44	5.93	73.98
N=400	5.68	70.88	7.51	105.01	9.92	139.25

BFL-ASSOM

	M=2		M=3		M=4	
	VU	WL	VU	WL	VU	WL
N=50	0.58	4.83	1.28	6.72	1.33	8.75
N=100	0.87	6.88	1.50	10.01	1.58	13.37
N=200	1.03	11.61	1.67	17.37	2.10	22.86
N=400	1.46	21.02	2.06	31.01	2.99	41.51

subspace dimension but quadratic to the input dimension. This computational load is reduced by FL-ASSOM to be linear to both the subspace dimension and the input dimension. The ability of FL-ASSOM in generating topologically ordered invariant-feature filters is altogether preserved since FL-ASSOM is mathematically equivalent to the basic ASSOM. We then proceeded to present BFL-ASSOM, where the basis vectors are updated in a batch mode. We showed that the correction made to each basis vector is a linear combination of the component vectors in the input episode. What's more, large amount of computations can be further saved by reusing more of previous computations and performing only one dissipation and orthonormalization for each episode.

Our experiments showed that BFL-ASSOM can also generate topologically ordered Gabor-like translation-invariant filters and that the bottleneck of the basis vector updating in the learning procedure is totally removed by FL-ASSOM and BFL-ASSOM. The proposed methods can be easily adapted to the supervised ASSOM used in [6]. There is an obvious benefit in using the proposed rules instead of naive implementation of the basic learning rule.

References

1. Kohonen, T.: The adaptive-subspace som (assom) and its use for the implementation of invariant feature detection. In Fogelman-Soulié, F., Gallinari, P., eds.: Proc. ICANN'95, Int.

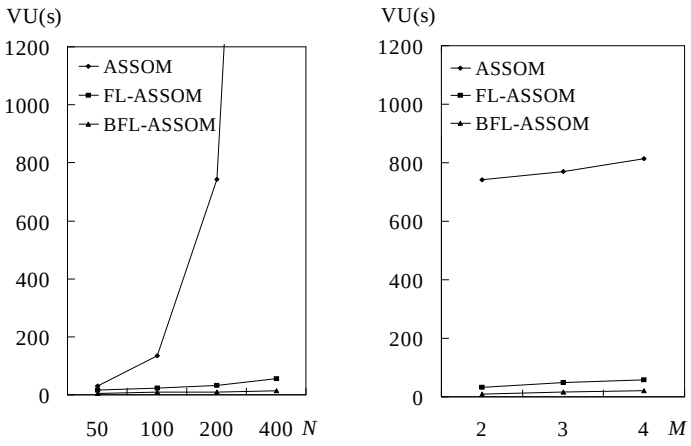


Fig. 4. *Left*: Relationship between the basis vector updating time (VU) and the input dimension N at the subspace dimension $M = 2$. *Right*: Relationship between VU and the subspace dimension M at the input dimension $N = 200$. For sake of clarity, the updating times of FL-ASSOM and BFL-ASSOM are magnified by a factor of 10

Conf. on Artificial Neural Networks. Volume 1., Paris (1995) 3–10
2. Kohonen, T.: Self-Organizing Maps. 3rd edn. Springer-Verlag, Berlin Heidelberg New York (2001)
3. Oja, E.: Principal components, minor components, and linear neural networks. Neural Networks **5** (1992) 927–935
4. Kohonen, T., Kaski, S., Lappalainen, H.: Self-organized formation of various invariant-feature filters in the adaptive-subspace som. Neural Computation **9**(6) (1997) 1321–1344
5. Hase, H., Matsuyama, H., Tokutaka, H., Kishida, S.: Speech signal processing using adaptive subspace som (assom). Technical Report NC95-140, The Inst. of Electronics, Information and Communication Engineers, Tottori University, Koyama, Japan (1996)
6. Ruiz del Solar, J.: Texsom: texture segmentation using self-organizing maps. Neurocomputing **21**(1–3) (1998) 7–18
7. De Ridder, D., Lemmers, O., Duin, R.P., Kittler, J.: The adaptive subspace map for image description and image database retrieval. In Ferri, F., et al., eds.: SSPR&SPR 2000. Volume 1876 of LNCS., Berlin Heidelberg, Springer-Verlag (2000) 94–103
8. Zhang, B., Fu, M., Yan, H., Jabri, M.: Handwritten digit recognition by adaptive-subspace self-organizing map (assom). IEEE Transactions on Neural Networks **10**(4) (1999) 939–945
9. López-Rubio, E., Muñoz Pérez, J., Gómez-Ruiz, J.: A principal components analysis self-organizing map. Neural Networks **17** (2004) 261–270
10. McGlinchey, S., Fyfe, C.: Fast formation of invariant feature maps. In: European Signal Processing Conference (EUSIPCO'98), Island of Rhodes, Greece (1998)
11. Oja, E.: Neural networks, principal components and subspaces. International Journal of Neural Systems **1** (1989) 61–68