Adaptive Offset Subspace Self-Organizing Map: An Application to Handwritten Digit Recognition*

Huicheng Zheng, Pádraig Cunningham, and Alexey Tsymbal

Dept of Computer Science
Trinity College Dublin, Ireland
{zhengh, cnnnghmp, tsymbalo}@tcd.ie

Abstract. An Adaptive-Subspace Self-Organizing Map (ASSOM) can learn a set of ordered linear subspaces which correspond to invariant classes. However the basic ASSOM cannot properly learn linear manifolds that are shifted away from the origin of the input space. In this paper, we propose an improvement on ASSOM to amend this deficiency. The new network, named AOSSOM for Adaptive Offset Subspace Self-Organizing Map, minimizes a projection error function in a gradient-descent fashion. In each learning step, the winning module and its neighbors update their offset vectors and basis vectors of the target manifolds towards the negative gradient of the error function. We show by experiments that the AOSSOM can learn clusters aligned on linear manifolds shifted away from the origin and separate them accordingly. The proposed AOSSOM is applied to handwritten digit recognition and shows promising results.

1 Introduction

The Adaptive-Subspace Self-Organizing Map (ASSOM) [1] is basically a combination of the traditional SOM and the subspace method. The single weight vectors at map units in the SOM are replaced by sets of basis vectors that span some linear subspaces in the ASSOM. By setting filters to correspond to pattern subspaces, some transformation groups, such as translation, rotation and scaling, can be taken into account. The simulation results in [1] and [2] have demonstrated that the ASSOM can induce ordered filter banks to account for translation, rotation and scaling. The ASSOM is an alternative to the standard Principal Component Analysis (PCA) method of feature extraction. An earlier neural approach for the PCA problem can be found in [3]. The ASSOM can learn topologically ordered filters corresponding to feature subspaces thanks to spatial interactions between processing units of the network. It has been successfully applied to speech processing [4], texture segmentation [5], image retrieval [6] and image classification [6] [7], etc. in the literature. A supervised variant of the ASSOM (SASSOM) was proposed by Ruiz del Solar in [5].

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

Although each module (a processing unit at the lattice locations in an ASSOM) performs an incremental PCA-like operation, in the traditional realization of the ASSOM, the module only learns a subspace of pattern vectors whose corresponding linear manifold must pass through the origin of the input space. Supposing we have a cluster of pattern vectors spreading on a linear manifold of the input space which does not pass through the origin, the ASSOM cannot learn this cluster properly. The learned network will thus lack discriminability.

In order to amend the above-mentioned deficiency, López-Rubio *et al.* [8] proposed a Principal Components Analysis Self-Organizing Map (PCASOM), where the manifold learning is realized by using an incremental PCA. However computations related to the updating of covariance matrices and the corresponding eigenproblem make PCASOM computationally expensive. Liu [9] devised an Adaptive Manifold Self-Organizing Map (AMSOM) as an extension of the basic algorithm of the ASSOM, which attempts to learn linear manifolds. The AMSOM was applied to face recognition and demonstrated superior performance to the standard PCA method as shown in [9]. An extension of AMSOM that uses a kernel method to account for nonlinear manifolds was proposed in [10].

In this paper, we propose to learn organized linear manifolds based on gradient descent. A previous gradient-descent method for the ASSOM has been introduced in [11], where a new basis updating rule of the ordinary subspaces has been proposed. The experiments in [11] showed that a gradient-descent method achieved faster convergence and less average projection error than the conventional ASSOM learning method. The method that we propose in this paper updates the offset vector and the basis vectors of a linear manifold simultaneously at each learning step. The resulting algorithm will be referred to as AOSSOM for Adaptive Offset Subspace Self-Organizing Map. The AOSSOM is a theoretically plausible method in the sense that there exists a predefined error function. It is more robust to local minima than the AMSOM, as will be justified by experiments in the paper.

The AOSSOM is applied to handwritten digit recognition in this paper. A major difficulty of handwritten digit recognition is the large variety of writing styles depending on national and regional origins of people, their individual habits and the circumstances in which they write [12]. Transformations to the handwritten digit images such as translation, rotation and scaling are common in real writing. It is hard to devise a hand-crafted feature extractor to cope with all the varieties. On the other hand, hand-crafted feature extraction can be advantageously replaced by carefully designed learning machines that operate directly on pixel images [13]. Since ASSOM-type networks can learn transformation groups with subspace competition, they can be used for handwritten digit recognition and learn filters invariant to a certain extent of variations.

This paper is organized as follows. Section 2 reviews the basic ASSOM algorithm. The proposed AOSSOM is derived in Section 3. Section 4 demonstrate the ability of AOSSOM to separate clusters by learning linear manifolds. Section 5 is devoted to the application of AOSSOM to handwritten digit recognition. Finally, Section 6 concludes this paper and suggests some perspectives.

2 ASSOM Learning

2.1 Orthogonal Subspace Projection

Each module in the ASSOM can be realized by a two-layer neural network [2], as shown in Fig. 1. Supposing a subspace $\mathcal{L}$ 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 h-th neuron in the first layer take the orthogonal projection $\mathbf{x}^{\mathrm{T}}\mathbf{b}_h$ of $\mathbf{x}$ on $\mathbf{b}_h$, where $1 \leq h \leq H$. 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 $\|\hat{\mathbf{x}}_{\mathcal{L}}\|^2$, with $\hat{\mathbf{x}}_{\mathcal{L}}$ being the orthogonal projection of $\mathbf{x}$ on $\mathcal{L}$. It can be regarded as a measure of the matching between $\mathbf{x}$ and $\mathcal{L}$.

The input to an ASSOM network is typically an episode, i.e. a sequence of pattern vectors supposed to approximately span some linear subspace. Typical examples of episodes used in the literature include sequences of temporally consecutive speech signals, transformations of image patchs. These vectors shall also be referred to as component vectors of the episode in this paper. By learning the episode as a whole, the ASSOM is able to capture the transformation coded in the episode. For an input episode $\mathbf{X} = \{\mathbf{x}(s),\ s \in S\}$, where S is the index set of vectors in the episode, Kohonen proposed to use the *energy* $m(\mathbf{X}, \mathcal{L}) = \sum_{s \in S} \|\hat{\mathbf{x}}_{\mathcal{L}}(s)\|^2$ as the measure of matching between $\mathbf{X}$ and $\mathcal{L}$ [2].

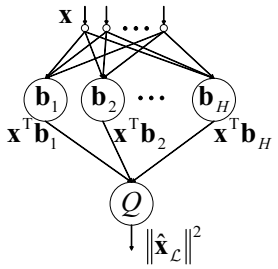

Fig. 1. A module of the ASSOM realized as a neural network

2.2 ASSOM Learning Process

The learning process of ASSOM approximately minimizes the average projection error of input vectors in an iterative way. In each learning iteration, there are two basic stages: 1) Competition of the modules for an input episode; 2) Updating of the winner and its neighbors towards the input in a weighted fashion. A detailed account of the learning process at each iteration step t is as follows:

1. For the current input episode $\mathbf{X} = \{\mathbf{x}(s),\ s \in S\}$, locate the winning module $c = \arg\max_{i \in I} m(\mathbf{X}, \mathcal{L}_i)$, 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 $\mathbf{b}_h^{(i)}$ according to:

$$\mathbf{b}_h^{(i)} = \left[\mathbf{I} + \lambda(t) h_c^{(i)}(t) \frac{\mathbf{x}(s)\mathbf{x}^{\mathrm{T}}(s)}{\|\hat{\mathbf{x}}_{\mathcal{L}_i}(s)\|\|\mathbf{x}(s)\|}\right] \mathbf{b}_h^{'(i)} \quad , \tag{1}$$

   where $\mathbf{b}_h^{(i)}$ is the new basis vector and $\mathbf{b}_h^{'(i)}$ the old one. $\mathbf{I}$ 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 $\mathbf{b}_h^{(i)}$, i.e. discard very small "noisy" components of the basis vectors, to force the basis vectors to learn the stronger and more fundamental features [2]. Orthonormalize these basis vectors afterwards.

A 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. In fact the formula (1) can be further simplified. It is not hard to get the following alternative formula:

$$\mathbf{b}_h^{(i)} = \mathbf{b}_h^{'(i)} + \Delta\mathbf{b}_h^{(i)} \quad , \tag{2}$$

where

$$\Delta\mathbf{b}_h^{(i)} = \alpha_{c,h}^{(i)}(s,t)\mathbf{x}(s) \tag{3}$$

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

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

This shows that the correction $\Delta\mathbf{b}_h^{(i)}$ is in fact a scaling of the component vector $\mathbf{x}(s)$, as illustrated in Fig. 2. After updating, $\mathbf{b}_h^{(i)}$ represents better $\mathbf{x}(s)$. Careful examination of (3) would reveal similarity of this formula with a recursive PCA suggested in [14]. The main difference is that here the gain of stochastic approximation is modulated by a neighborhood function dependent on the module competition. Note that the computation of $\mathbf{x}^{\mathrm{T}}(s)\mathbf{b}_h^{'(i)}$ in (4) can be saved since it was computed when calculating 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 use it to scale the component vector $\mathbf{x}(s)$, the basis vector updating speed can be dramatically improved.

2.3 A Deficiency of the ASSOM

As mentioned in Introduction, an intrinsic deficiency of the basic ASSOM is that it cannot learn linear manifolds which have a shift from the origin. An illustration

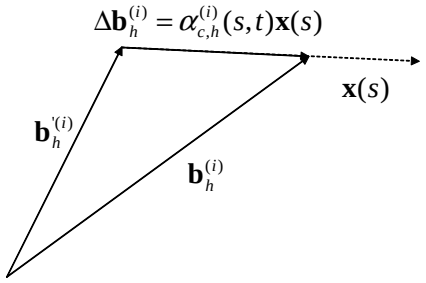

Fig. 2. An insightful view of the basis vector updating rule of ASSOM

of this problem is the two clusters shown in Fig. 3. The basic ASSOM is unable to separate these two clusters. For a network of two modules with 1-D subspaces, each module learned one of the two basis vectors marked by the two dotted lines in the graph. This is also a difficult case for PCA since the first principal component will point to the vertical direction. However if we can also learn the offsets of the clusters, then theoretically the learning would be more accurate and consequently the two clusters can be separated properly. There has been some work in this direction in the literature. In the next section, we shall review some of this work.

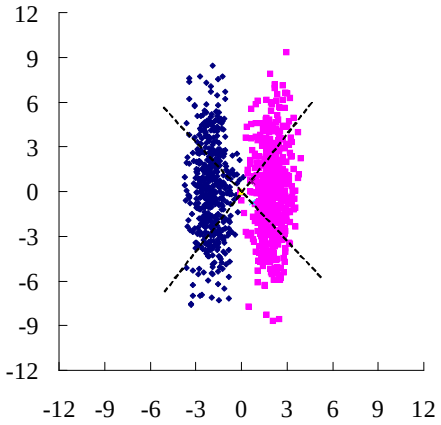

Fig. 3. Two 1-D subspaces (marked by dotted lines) learned by the basic ASSOM

3 Learning Linear Manifolds with the AOSSOM

The PCASOM and the AMSOM were proposed to improve the ASSOM so that the map can learn linear manifolds which do not necessarily pass through the origin. However the PCASOM is computationally expensive due to the covariance matrices to update and the eigenproblem to solve at each learning step. The AMSOM [9] is a direct extension of the ASSOM by appending a mean vector $\mathbf{m}$ to each learned subspace. At each learning step, $\mathbf{m}$ is first updated according to the input vector by using an SOM training rule. Then the AMSOM shifts the manifold to the origin and updates the basis vectors in the same way as the ASSOM does. Liu did not provide an objective error function in [9]. Minimization of the projection error by AMSOM is still to be theoretically justified. Contrary to the AMSOM, the AOSSOM proposed in this paper starts from an objective function, which is the average projection error. The algorithm is derived by minimizing this error function in a gradient-descent manner. Convergence of the algorithm will at least guarantee a local minimum of the error function.

3.1 Projection Error Minimization by Gradient Descent

Each module i of an AOSSOM represents a linear manifold $\mathcal{A}_i$ described by an offset vector $\mathbf{r}^{(i)}$ and a set of basis vectors $\mathbf{b}_h^{(i)}$, $h = 1, \ldots, H$. It is not necessary for the offset vector to be the mean vector of the samples lying in $\mathcal{A}_i$. An error function is defined for the learning procedure:

$$E = \int \sum_{i \in I} h_c^{(i)} d(\mathbf{X}, \mathcal{A}_i) P(\mathbf{X}) \mathrm{d}\mathbf{X} \ , \tag{5}$$

where $P(\mathbf{X})$ is the distribution of the random episode $\mathbf{X} = \{\mathbf{x}(s); s \in S\}$. c is the index of the module which wins $\mathbf{X}$. $h_c^{(i)}$ is a neighborhood function as in the basic ASSOM. $d(\mathbf{X}, \mathcal{A}_i)$ is a measure of the distance from $\mathbf{X}$ to $\mathcal{A}_i$ and defined as the projection error of $\mathbf{X}$ on $\mathcal{A}_i$:

$$d(\mathbf{X}, \mathcal{A}_i) = \sum_{s \in S} \left\| \mathbf{x}(s) - \hat{\mathbf{x}}_{\mathcal{A}_i}(s) \right\|^2 \ . \tag{6}$$

In the basic ASSOM, $\mathcal{A}_i$ is a linear subspace and the orthogonal projection of $\mathbf{x}(s)$ on $\mathcal{A}_i$ is simply

$$\hat{\mathbf{x}}_{\mathcal{A}_i}(s) = \sum_{h=1}^{H} (\mathbf{x}^{\mathrm{T}}(s) \mathbf{b}_h^{(i)}) \mathbf{b}_h^{(i)} \ . \tag{7}$$

While in the more general case where $\mathcal{A}_i$ is a linear manifold defined by an offset vector $\mathbf{r}^{(i)}$ and the basis $\{\mathbf{b}_1^{(i)}, \ldots, \mathbf{b}_H^{(i)}\}$, the projection is defined by:

$$\hat{\mathbf{x}}_{\mathcal{A}_i}(s) = \mathbf{r}^{(i)} + \sum_{h=1}^{H} \left(\mathbf{x}'^{\mathrm{T}}_{\mathcal{A}_i}(s) \mathbf{b}_h^{(i)} \right) \mathbf{b}_h^{(i)} \ , \tag{8}$$

where

$$\mathbf{x}'_{\mathcal{A}_i}(s) = \mathbf{x}(s) - \mathbf{r}^{(i)} \ . \tag{9}$$

The projection residual of $\mathbf{x}(s)$ on the linear manifold $\mathcal{A}_i$ is therefore defined by

$$\tilde{\mathbf{x}}_{\mathcal{A}_i}(s) = \mathbf{x}(s) - \hat{\mathbf{x}}_{\mathcal{A}_i}(s) \ . \tag{10}$$

The relationship between parameters of the linear manifold $\mathcal{A}_i$, $\mathbf{x}'_{\mathcal{A}_i}(s)$, $\hat{\mathbf{x}}_{\mathcal{A}_i}(s)$ and $\tilde{\mathbf{x}}_{\mathcal{A}_i}(s)$ is shown in Fig. 4 for the case where $\mathcal{A}_i$ is 1-dimensional.

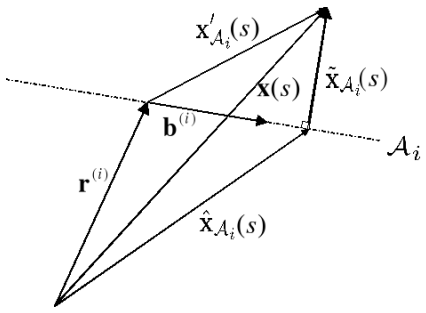

Fig. 4. The relationship between parameters of the 1-D linear manifold $\mathcal{A}_i$, $\mathbf{x}'_{\mathcal{A}_i}(s)$, $\hat{\mathbf{x}}_{\mathcal{A}_i}(s)$ and $\tilde{\mathbf{x}}_{\mathcal{A}_i}(s)$

In (5), the index c of the winning module depends on the input episode $\mathbf{X}$ as well as on the status of all the modules in the network. The exact minimization of an objective function like (5) may be very complicated [2]. Instead, by using a stochastic approximation, we target the "sample" objective function at a certain learning step t based on the last input episode $\mathbf{X}$ and the last status of the network:

$$E_{\mathrm{s}}(t) = \sum_{i \in I} h_c^{(i)}(t) d(\mathbf{X}, \mathcal{A}_i) \ . \tag{11}$$

The same strategy has been used to derive the basic ASSOM algorithm [2].

Taking gradients of $E_{\mathrm{s}}(t)$ with respect to the basis vectors, we have

$$\frac{\partial E_{\mathrm{s}}}{\partial \mathbf{b}_h^{(i)}}(t) = h_c^{(i)}(t) \frac{\partial d(\mathbf{X}, \mathcal{A}_i)}{\partial \mathbf{b}_h^{(i)}} \ . \tag{12}$$

This is all that we need for the basic ASSOM. Here for AOSSOM however we have the offset vector $\mathbf{r}^{(i)}$ to update for each linear manifold $\mathcal{A}_i$. Thus we need an additional partial derivative for $\mathbf{r}^{(i)}$:

$$\frac{\partial E_{\mathrm{s}}}{\partial \mathbf{r}^{(i)}}(t) = h_c^{(i)}(t) \frac{\partial d(\mathbf{X}, \mathcal{A}_i)}{\partial \mathbf{r}^{(i)}} \ . \tag{13}$$

Details of the proof for solving out the partial derivatives are omitted in order not to distract the reader from the main focus. However with some linear

algebraic and calculus skills they are not hard to get. We suppose that the basis vectors are kept orthonormal in the gradient-descent process. Here are the results, for the offset vector $\mathbf{r}^{(i)}$,

$$\frac{\partial E_{\mathrm{s}}}{\partial \mathbf{r}^{(i)}}(t) = -2h_c^{(i)}(t) \sum_{s \in S} \tilde{\mathbf{x}}_{\mathcal{A}_i}(s) \ , \tag{14}$$

and for the basis vectors,

$$\frac{\partial E_{\mathrm{s}}}{\partial \mathbf{b}_h^{(i)}}(t) = -2h_c^{(i)}(t) \sum_{s \in S} \left(\mathbf{x}'^{\mathrm{T}}_{\mathcal{A}_i}(s) \mathbf{b}_h^{(i)} \right) \tilde{\mathbf{x}}_{\mathcal{A}_i}(s) \ . \tag{15}$$

Taking a step length $\frac{1}{2}\lambda_{\mathbf{b}}(t)$ in the opposite direction to the gradients for the basis vectors, where $\lambda_{\mathbf{b}}(t)$ is a learning-rate factor for the basis vectors, the correction made to the basis vectors should be:

$$\Delta \mathbf{b}_h^{(i)} = h_c^{(i)}(t) \lambda_{\mathbf{b}}(t) \sum_{s \in S} \left(\mathbf{x}'^{\mathrm{T}}_{\mathcal{A}_i}(s) \mathbf{b}_h^{(i)} \right) \tilde{\mathbf{x}}_{\mathcal{A}_i}(s) \ , \tag{16}$$

and updating of the offset vector $\mathbf{r}^{(i)}$ would be:

$$\Delta \mathbf{r}^{(i)} = h_c^{(i)}(t) \lambda_{\mathbf{r}}(t) \sum_{s \in S} \tilde{\mathbf{x}}_{\mathcal{A}_i}(s) \ , \tag{17}$$

where $\lambda_{\mathbf{r}}(t)$ is the learning-rate factor for the offset vector $\mathbf{r}^{(i)}$.

Since $\tilde{\mathbf{x}}_{\mathcal{A}_i}(s) \perp \mathbf{b}_h^{(i)}, \forall s, h$, it is easy to show that

$$\Delta \mathbf{b}_{h_1}^{(i)} \perp \mathbf{b}_{h_2}^{(i)}, \ \forall h_1, h_2 \tag{18}$$

$$\text{and} \quad \Delta \mathbf{r}^{(i)} \perp \mathbf{b}_h^{(i)}, \ \forall h \ . \tag{19}$$

That is, updating of the linear manifold $\mathcal{A}_i$ is perpendicular to any current basis vector of $\mathcal{A}_i$. This is also the steepest direction to update $\mathcal{A}_i$ towards the input. We remark that in the traditional ASSOM and in the AMSOM, the updating of the basis vectors has a redundant direction corresponding to the projection of the input vector on the (affine) subspace (cf. (3)). This might not be desirable since it would introduce instability to the basis vectors. Let us suppose that an input vector $\mathbf{x}(s)$ lies perfectly in the (affine) subspace of the module i. With the ASSOM and the AMSOM, the basis vectors are still updated as long as $\mathbf{x}(s)$ has a non-zero projection on the (affine) subspace. While with the AOSSOM, since the projection residual $\tilde{\mathbf{x}}_{\mathcal{A}_i}(s)$ is zero, the basis vectors remained unchanged, which is a more desirable behavior.

3.2 AOSSOM Learning Process

The AOSSOM follows a similar learning procedure as the ASSOM learning. Each learning iteration consists of the module competition and the updating of the winner and its neighbors. The learning procedure of AOSSOM at each iteration step t can be summarized as follows:

1. For an input episode $\mathbf{x}(s)$, $s \in S$, locate the winning module indexed by

$$c = \arg\min_{i \in I} \sum_{s \in S} \|\tilde{\mathbf{x}}_{\mathcal{A}_i}(s)\|^2 \ . \tag{20}$$

2. For each module i in the neighborhood of c, including c itself, update the linear manifold $\mathcal{A}_i$, i.e. update the offset vector $\mathbf{r}^{(i)}$ and basis vectors $\mathbf{b}_h^{(i)}$ for each component vector $\mathbf{x}(s)$, $s \in S$. To update $\mathbf{r}^{(i)}$, we use:

$$\mathbf{r}^{(i)} = \mathbf{r}'^{(i)} + \Delta\mathbf{r}^{(i)} \ , \tag{21}$$

where $\Delta\mathbf{r}^{(i)}$ is defined by (17). To update $\mathbf{b}_h^{(i)}$, we use

$$\mathbf{b}_h^{(i)} = \mathbf{b}_h'^{(i)} + \Delta\mathbf{b}_h^{(i)} \ , \tag{22}$$

where $\Delta\mathbf{b}_h^{(i)}$ is defined by (16). The basis vectors $\mathbf{b}_h^{(i)}$ are orthonormalized at the end of this step.

4 Cluster Separating Experiments

In this section, we demonstrate the capacity of the AOSSOM to learn and separate clusters which lie in linear manifolds away from the origin. For easier reading of this section, we first present some common parameters of the ASSOM, the AMSOM and the AOSSOM used in the following experiments. For each of the networks, the learning takes $T = 10,000$ steps. The neighborhood function is fixed to be Gaussian:

$$h_c^{(i)}(t) = \exp\left(-\frac{\|\mathbf{u}_c - \mathbf{u}_i\|^2}{2\sigma^2(t)}\right) \ , \tag{23}$$

where $\mathbf{u}_c$ and $\mathbf{u}_i$ are respectively the coordinates of the winning module c and the module i in the network lattice. $\sigma(t)$ is related to the Full Width at Half Maximum (FWHM) $w(t)$ by $\sigma(t) = w(t)/2.3548$. For all the networks throughout this section except the "AMSOM 1" in the third experiment, $w(t)$ has the common form:

$$w(t) = w_0 \frac{T}{T + 99t} \ , \tag{24}$$

where w_0 should be set as such that the whole lattice can be properly covered at the beginning of the learning procedure. The learning-rate parameters are $\lambda(t) = \lambda_0 \frac{T}{T+99t}$ for the ASSOM, $\lambda_{\mathbf{m}}(t) = \lambda_{\mathbf{b}}(t) = \lambda_0 \frac{T}{T+99t}$ for the AMSOM, and $\lambda_{\mathbf{r}}(t) = \lambda_{\mathbf{b}}(t) = \lambda_0 \frac{T}{T+99t}$ for the AOSSOM. $\lambda_{\mathbf{m}}(t)$ is the learning-rate function for the mean vectors of the AMSOM. The initial learning rate λ_0 shall be properly chosen in the respective experiments.

4.1 First Experiment

In the first experiment, the goal is to learn the two Gaussian clusters in Fig. 3. The basic ASSOM is unable to learn the two clusters properly. Even with PCA, the first principal component would be directed to a vertical orientation, which does not give the best description of the two clusters and cannot separate them properly.

The settings of the first experiment are as follows. Each cluster in Fig. 3 is generated with a 2-D Gaussian distribution which shows an evident orientation, so that the cluster can be approximated by a 1-D linear manifold, which is a line, in the 2-D input space. 500 samples are generated for each cluster. We first show that the AOSSOM is able to learn the corresponding linear manifolds. Two modules are implemented in the AOSSOM network with each containing an offset vector and a basis vector. The initial learning rate $\lambda_0 = 1$. The learning results are shown in Fig. 5. In the figure, the solid lines with arrows show the offset vectors learned by the AOSSOM. The dotted lines mark the 1-D linear manifolds learned by the AOSSOM. We can appreciate how well AOSSOM learned the two clusters and separated them by different offset vectors. Both modules found the linear manifolds which describe best the clusters in the sense of MSE (mean squared error).

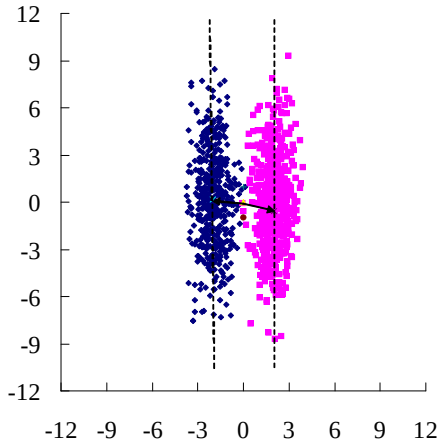

Fig. 5. Result of learning two 1-D linear manifolds with the AOSSOM

We have run the basic ASSOM, the AMSOM and the AOSSOM to compare their performance in separating such a distribution. Two modules are implemented in each network. The ASSOM tries to learn 1-D linear subspaces from the data whereas the AMSOM and the AOSSOM try to learn 1-D linear manifolds. Efforts were made for each network so that all of them work optimally.

After learning, each module was labeled with the cluster that it won for the most of the times. In testing, each input sample was assigned the label of the winning module. The experiments were repeated ten times for each network. In each repeat, 500 training samples and 500 test samples were randomly generated for each cluster and the networks were initialized randomly with different random seeds. The resulting classification accuracies are summarized in Table. 1. In the table, TR represents the training sets and TT the test sets. We can see that ASSOM works very poorly on such a distribution, it is practically a random predictor. Both AOSSOM and AMSOM can separate the two clusters with very high accuracies across different runs with various initializations of the networks.

Table 1. Classification accuracies for ten runs of the ASSOM, the AMSOM and the AOSSOM to separate the two Gaussian clusters in Fig. 5

n	ASSOM		AMSOM		AOSSOM	
	TR	TT	TR	TT	TR	TT
1	50.3%	52.3%	99.6%	99.9%	99.7%	99.9%
2	51.5%	51.3%	99.7%	99.8%	99.8%	99.9%
3	51.8%	50.9%	99.7%	99.9%	99.9%	99.8%
4	52.4%	50.3%	99.8%	99.9%	99.9%	99.7%
5	51.7%	50.9%	99.8%	99.8%	100%	99.7%
6	51.5%	51.4%	99.7%	99.9%	99.8%	99.8%
7	51.5%	51.4%	99.8%	99.8%	99.8%	99.7%
8	50.4%	50.1%	99.8%	100%	99.9%	100%
9	53%	49.8%	99.9%	99.8%	99.7%	100%
10	50.7%	52.2%	99.9%	99.8%	99.8%	99.9%

4.2 Second Experiment

In the second experiment, the goal is to learn the three Gaussian clusters in Fig. 6. Each cluster is generated with a 2-D Gaussian distribution with one of the shown orientations, so that it can be approximated by a 1-D linear manifold. We intentionally introduced overlapping between different clusters to add to the difficulty of learning. 500 samples were generated for each cluster. We first show the capacity of the AOSSOM to learn the three linear manifolds. Three modules are implemented in the AOSSOM network with each containing an offset vector and a basis vector. As shown in Fig. 6, the AOSSOM learned the three clusters correctly despite the overlapping between clusters.

The basic ASSOM, the AMSOM and the AOSSOM are compared in terms of their ability to separate the three clusters. The experimental settings are the same as the previous experiment with two clusters except that here we use three modules in each network. $\lambda_0 = 1$ was used for the ASSOM and the AOSSOM, $\lambda_0 = 1.5$ for the AMSOM. In the experiment we observed unstable

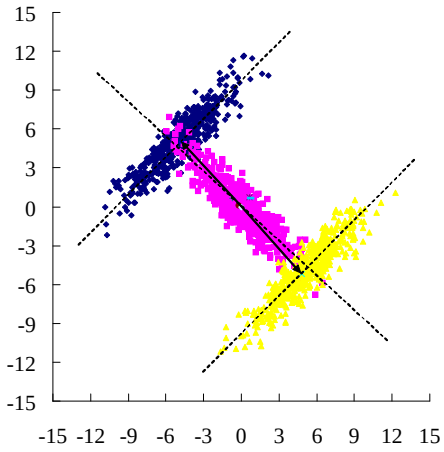

Fig. 6. Result of learning three 1-D linear manifolds with the AOSSOM. The offset vector of the middle cluster is not plotted since it is too close to a zero vector

phenomena of the AMSOM when $\lambda_0 = 1$. Even though the learning rate should conventionally be in $[0, 1]$, as suggested by Liu [9], a larger starting learning rate seemed to have worked better for the AMSOM in this experiment. We will discuss this problem in the third experiment. For now we just choose the good parameters. The experiment was repeated ten times for each of the networks. 500 training samples and 500 test samples were randomly generated for each cluster at each run. The resulting classification accuracies are summarized in Table. 2. Due to overlapping between clusters, no classifiers can give perfect separation. The performance of the ASSOM is very poor for such a distribution. Both the AOSSOM and the AMSOM can separate these clusters with reasonably high accuracies across different runs with various initializations of the networks.

We may conclude that the AOSSOM is able to properly learn and separate clusters of these kinds, which are best described by distributions along linear manifolds shifted away from the origin. Such distributions cannot be adequately identified by linear subspaces used in the basic ASSOM.

4.3 Third Experiment

In the previous experiment we observed unstable phenomena of the AMSOM when $\lambda_0 = 1$. A larger starting learning rate $\lambda_0 = 1.5$ seemed to have worked better. The explanation for this could be that a smaller learning rate has a better convergent property, however it cannot pull the network out of "bad" local minima. However in a general case, a large initial learning rate does not always lead to a good performance, as we observed in other experiments on the AMSOM. The sensitivity of the AMSOM to local minima seems to be evident in some cases. To demonstrate this, we devised the following experiment.

Table 2. Classification accuracies for ten runs of the ASSOM, the AMSOM and the AOSSOM to separate the three Gaussian clusters in Fig. 6

n	ASSOM		AMSOM		AOSSOM	
	TR	TT	TR	TT	TR	TT
1	37.7%	34.9%	90.5%	91.2%	89.9%	91.7%
2	37.1%	36.4%	90.2%	90.5%	89.9%	91.2%
3	37.9%	36.8%	89.6%	91.3%	90.5%	89.9%
4	36.3%	37.7%	90.5%	91.3%	89.9%	90.9%
5	37.5%	36.5%	90.9%	90%	90.5%	91.2%
6	36.5%	37.5%	89.6%	90.5%	89.9%	90.6%
7	36.6%	37.5%	91%	89.7%	91.7%	90.6%
8	37%	37.3%	91.1%	91%	89.9%	90.9%
9	36.8%	35.5%	90.4%	91.1%	90.2%	91.3%
10	39%	38.3%	90.5%	91.7%	89.9%	91.1%

We randomly generated 10 different data sets. Each data set was composed of 4 clusters with 2-D Gaussian distributions. 500 training samples and 500 test samples were generated for each cluster. The mean vector of each cluster is random, with components in $[-4, 4]$ and a Euclidean norm not less than 2. Except the above settings, the clusters are allowed to have any amount of overlapping. So in general, they cannot be perfectly separated. Each AMSOM network as well as each AOSSOM network contains 4 modules. There is one basis vector in each module to simulate a 1-D linear manifold.

For the AMSOM, we experimented two configurations. In the first configuration, we used the exponential neighborhood-decreasing scheme suggested in [9]:

$$w(t) = w_0 \exp(-\frac{t}{F}) \ , \tag{25}$$

where F is the learning step when $w(F) = 0.368 w_0$. In our experiments, we set $F = 5000$. Changes to F did not show improved performance in our experiments. w_0 was set to 4 so that the full lattice could be covered at the beginning. The initial learning rate $\lambda_0 = 1$. We did not observe improvement of the performance with different λ_0 in this experiment. The results we got on the first configuration of the AMSOM are shown in Table 3 in the column "AMSOM 1". In the table, D is the serial number of the data sets generated. Different data sets have different four-cluster distributions. For a fair comparison, in the second configuration of the AMSOM, we chose the same neighborhood-decreasing function as that of the AOSSOM, which is shown in (24). The initial learning-rate was set to $\lambda_0 = 1$. Other values of λ_0 did not show improved performance in our experiments. The results for the AMSOM of the second configuration are summarized in Table 3 in the column "AMSOM 2". The configuration of the AOSSOM was the same as the second configuration of the AMSOM. The results of the AOSSOM are also summarized in Table 3. According to the table, the AOSSOM worked the best in 8 runs out of 10. In the second run even though the AOSSOM did not

Table 3. Classification accuracies of the AMSOM and the AOSSOM for separating four Gaussian clusters. The boldface font emphasizes the best results at each run

D	AMSOM 1		AMSOM 2		AOSSOM	
	TR	TT	TR	TT	TR	TT
1	54.7%	53.6%	41.2%	42%	**78.9%**	**79%**
2	53.2%	51.8%	**67.6%**	**68.2%**	65.5%	65.7%
3	48.4%	48.6%	63%	63.1%	**74.5%**	**74.1%**
4	42.9%	42.3%	45.2%	45.2%	**78.1%**	**75.4%**
5	45.9%	45.7%	50.1%	49.2%	**57.4%**	**57.9%**
6	42.9%	42.7%	**52.4%**	**53.8%**	46.5%	46.4%
7	43.9%	45.4%	44.9%	44.9%	**57.7%**	**59.1%**
8	60.3%	60%	44.9%	47.8%	**61.9%**	**62.3%**
9	47%	46.2%	59.6%	61.3%	**75.3%**	**73.3%**
10	53.4%	54.4%	46.4%	47.1%	**72.3%**	**70.9%**

work the best, it still achieved a fairly good performance which is only at a little distance from the best.

To see what happened when the AMSOM was trapped in a local minimum, we recorded the average projection errors on the test data set 1 in Table 3 with the AMSOM and the AOSSOM, which are shown in Fig. 7. As we can see from the figure, the "AMSOM 1" and the "AMSOM 2" converged to higher error levels than the AOSSOM. Higher error levels correspond to worse local minima. The "AMSOM 1" and the "AMSOM 2" seem to have stablized on the "bad" local minima and could not get out of them. In the process of convergence the AOSSOM also encountered a "bad" local minimum at around $t = 4,000$, where the error was 0.47. However it was able to get out of that "bad" local minimum and finally stablized at a better minimum, which was 0.40. In fact, even the "bad" local minimum 0.47 of the AOSSOM is already smaller than the converged local minima of the "AMSOM 1" and the "AMSOM 2". The reason that the AOSSOM worked better than the AMSOM in most of the cases could be that the AOSSOM learning algorithm was derived from the gradient descent of the projection error function. Updating of the linear manifolds in a direction other than the negative gradient direction is more likely to be stuck in undesirable local minima.

5 Application of the AOSSOM to Handwritten Digit Recognition

5.1 Related Work and Database

Zhang *et al.* [7] have applied a variant of the ASSOM for handwritten digit recognition. They designed an ASSOM network in which the modules are realized by three-layer neural networks trained as autoencoders. To improve the stability of the autoencoders and overcome restrictions of linear subspaces in the

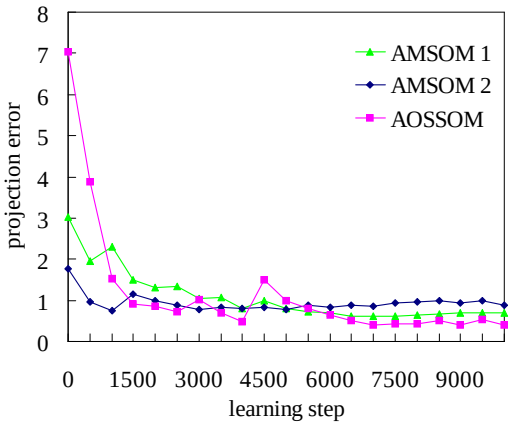

Fig. 7. Average projection errors of the first data set in Table 3 with the "AMSOM 1", the "AMSOM 2" and the AOSSOM

basic ASSOM, they introduced nonlinearity to the hidden-layer neurons of the autoencoders. Their experiments showed impressive performance on a database of the U.S. National Institute of Standards and Technology (NIST), which consists of 20,000 numerals. In this section, we apply the AOSSOM, an alternative improvement of the basic ASSOM, to handwritten digit recognition.

The handwritten digit image database used in this paper is a modified NIST (MNIST) database by mixing NIST's datasets. The database is made publicly available by LeCun *et al.* [13]. Images in this database were size normalized and centered in a 28×28 pixel field. The resulting images contain gray levels as a result of the interpolation technique used by the normalization algorithm. The foreground is coded with high gray levels and the background with low gray levels. The database contains a training set of 60,000 digits and a test set of 10,000 digits. Some examples from this database are shown in Fig. 8. A large variety of writing styles are covered by the database as shown by these examples.

5.2 Learning Handwritten Digits with the AOSSOM

As an application of the AOSSOM, our handwritten digit recognition system is designed as follows. An AOSSOM network is trained for each class of examples. An input digit image is classified by examining which network gives the best reconstruction. Similar ideas were proposed in [15] and [7]. In [15], PCA and factor analysis (FA) were proposed as local models of handwritten digit image manifolds. In [7], ASSOM networks realized by non-linear autoencoders were implemented for the local modeling.

Before a digit image is input into the AOSSOM networks, its mean value was subtracted. Then the pattern vector representing the image was normalized

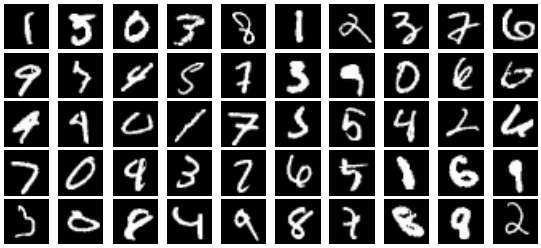

Fig. 8. Some examples from the MNIST handwritten digit image database

before entering the networks. The learning steps of each AOSSOM network is set to $T = 30,000$. The learning-rate factor has the form $\lambda_{\mathbf{r}}(t) = \lambda_{\mathbf{b}}(t) = \lambda_0 \frac{T}{T+99t}$ where λ_0 was set to 1. The neighborhood function is Gaussian with the FWHM $w(t) = w_0 \frac{T}{T+99t}$ where the initial value w_0 depends on the network size. The trained AOSSOM networks of 5×5 modules with two basis vectors are shown in Fig. 9. Each module in the k-th network $\mathcal{M}_k$, $k = 0, \ldots, 9$, learned a linear manifold for the images of the digit k. Each manifold contains an offset vector $\mathbf{r}$ and two basis vectors $\mathbf{b}_1$ and $\mathbf{b}_2$. These vectors are visualized by normalizing the grayscales into $[0, 255]$. For a mean-subtracted and normalized test input digit image $\mathbf{x}$, each network $\mathcal{M}_k$ gives a reconstruction vector $\hat{\mathbf{x}}_{\mathcal{M}_k}$. The class label of $\mathbf{x}$ is determined by the network which gives the minimum reconstruction error:

$$k^* = \arg\min_k \|\mathbf{x} - \hat{\mathbf{x}}_{\mathcal{M}_k}\| \ . \tag{26}$$

Now it comes to the question of how to build the reconstruction $\hat{\mathbf{x}}_{\mathcal{M}_k}$ for each network. The idea is to combine the set of reconstruction vectors $\hat{\mathbf{x}}_{\mathcal{A}_{ki}}$ from the $|I_k|$ modules in the network $\mathcal{M}_k$, where I_k is the set of modules in $\mathcal{M}_k$ and $\mathcal{A}_{ki}$ the linear manifold learned by the i-th module in $\mathcal{M}_k$. The combination is a weighted average:

$$\hat{\mathbf{x}}_{\mathcal{M}_k} = \frac{\sum_{i \in I_k} a_{ki} \hat{\mathbf{x}}_{\mathcal{A}_{ki}}}{\sum_{i \in I_k} a_{ki}} \ . \tag{27}$$

This weighted average was also used in [7]. There is no clear evidence that the choice of the weighting function is critical [16]. One choice is the Gaussian function which has infinite extent [7]:

$$a_{ki} = \exp\left(-\frac{\|\mathbf{x} - \hat{\mathbf{x}}_{\mathcal{A}_{ki}}\|^2}{2\sigma_{ki}^2}\right) \ , \tag{28}$$

where σ_{ki} controls the response range of the corresponding module. In general, σ_{ki} can be chosen in a reasonable range without significant difference [7]. We have chosen $\sigma_{ki} = 0.1$ in our experiments.

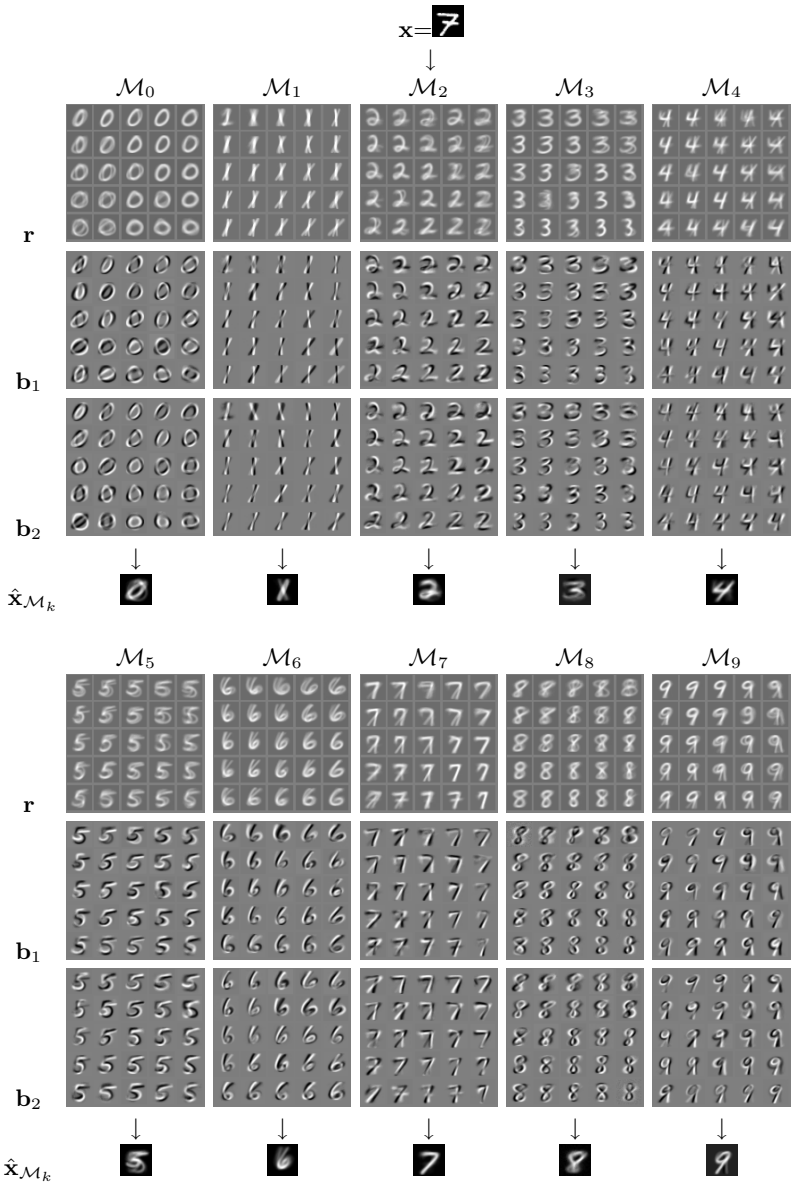

Fig. 9. AOSSOM networks trained for the MNIST database. For a mean-subtracted and normalized test input digit image $\mathbf{x}$, each network builds a reconstruction vector $\hat{\mathbf{x}}_{\mathcal{M}_k}$

5.3 Experimental Results

We have experimented on different sizes of AOSSOM networks with different numbers of basis vectors in each module. The results are summarized in Table 4. The network lattices are squares with the dimension W varying from 3 to 8. Thus the number of modules in each network is W^2. As we can see, in general, a larger network size leads to a better recognition accuracy. But this also increases the training and testing time. A higher manifold dimension can achieve a better performance, this is more evident when the network size is smaller. In general, this classification system exhibits a promising recognition performance on the handwritten digit images.

Table 4. Handwritten digit recognition accuracies of the AOSSOM network with different lattice sizes W and linear manifold dimensions H

W	$H=1$		$H=2$		$H=3$	
	TR	TT	TR	TT	TR	TT
3	94.6%	94.7%	95.6%	95.6%	96.3%	96.2%
4	95.7%	95.7%	96.4%	96.3%	96.9%	96.5%
5	96.3%	96.2%	96.9%	96.7%	97.2%	96.8%
6	96.7%	96.3%	97.1%	96.8%	97.7%	97.1%
7	97.1%	96.6%	97.4%	96.9%	97.6%	96.9%
8	97.5%	97%	97.8%	97.1%	98%	97.3%

Some of the correctly recognized digits are shown in Fig. 10. In the figure, the first column shows the input digit images. The other columns show the images reconstructed by the individual networks. The images are equalized to 255 gray levels. In the first column, numbers to the left of the arrows are the ture labels and numbers to the right are the labels assigned by the system. Numbers in the other columns are reconstruction errors of the individual networks. For each shown input digit image, the network corresponding to the correct digit label gives the best reconstruction, as confirmed by the output reconstruction errors. The other networks often output ambiguous reconstruction vectors, e.g. the output of the 4-th network for the input digit 2 and that of the 8-th network for the input digit 6. The written style of the input digit 9 is similar to 4, and this is confirmed by the output of the 4-th network, whose reconstruction error is close to that of the 9-th network.

Figure 11 shows some incorrectly classfied digits. Some of the input examples are really ambiguous, e.g. $4 \rightarrow 9$ and $9 \rightarrow 1$. We have observed in the experiments that the most probable errors come from $2 \rightarrow 7$, $4 \rightarrow 9$ and $6 \rightarrow 0$. By examining the output reconstruction errors, we can find that even though the recognition failed, the error of the true-class network could be very close to the minimum error. For example, in the false classification $9 \rightarrow 1$, the shown error of the network $\mathcal{M}_1$ is practically the same as that of $\mathcal{M}_9$. In fact, the errors are discriminable only by using more numerical precision. The error from $\mathcal{M}_1$

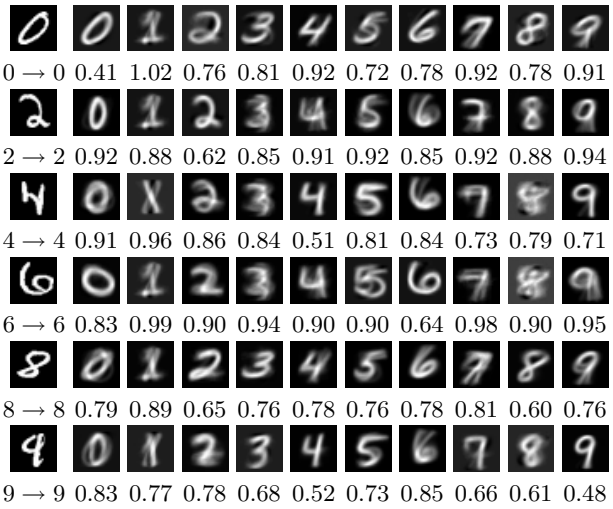

Fig. 10. Some digits correctly recognized by the AOSSOM networks

is effectively 0.4637 and that from $\mathcal{M}_9$ is 0.4639. The difference is practically negligible here. Better accuracies could be achieved by rejecting some input instances when the smallest reconstruction error and the second smallest are very close.

6 Conclusions and Perspectives

In this paper, the AOSSOM, an improvement of the basic ASSOM, was proposed. It is able to learn a set of ordered linear manifolds. The AOSSOM works by minimizing a projection error function in a gradient-descent manner. We demonstrated by experiments that the AOSSOM is able to learn linear manifolds of clusters shifted away from the origin and separate the clusters accordingly. The basic ASSOM cannot identify clusters of such kinds adequately. The AOSSOM is more robust to local minima than the AMSOM as revealed by the cluster separating experiments. The reason could be that the updating of the manifolds follows the negative gradient direction. Other directions, such as the one used in the AMSOM, could be more prone to local minima. The proposed network was then applied to handwritten digit recognition and showed promising results.

The number of modules in our AOSSOM network (which correponds to the number of linear manifolds) and the number of basis vectors in each module (which corresponds to the minifold dimension) were determined heuristically. In a real-world data mining problem, the potential number of clusters and their dimensions are often not known. So networks which can learn the optimal numbers of modules and basis vectors from the empirical data would be desirable. This issue will be considered in our further research.

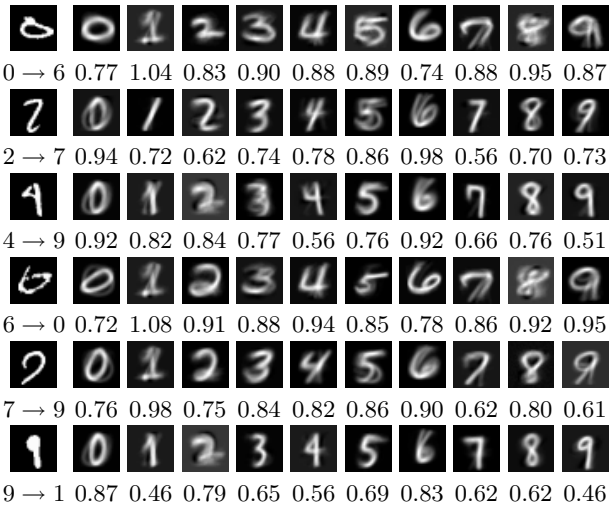

Fig. 11. Some digits incorrectly recognized by the AOSSOM networks

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. Conf. on Artificial Neural Networks. Volume 1., Paris (1995) 3–10
2. 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
3. Oja, E.: Principal components, minor components, and linear neural networks. Neural Networks **5**(6) (1992) 927–935
4. 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)
5. Ruiz-del-Solar, J.: TEXSOM: Texture segmentation using Self-Organizing Maps. Neurocomputing **21**(1-3) (1998) 7–18
6. de Ridder, D., Lemmers, O., Duin, R.P.W., Kittler, J.: The Adaptive Subspace Map for image description and image database retrieval. In: SSPR/SPR. (2000) 94–103
7. Zhang, B., Fu, M., Yan, H., Jabri, M.: Handwritten digit recognition by Adaptive-Subspace Self-Organizing Map (ASSOM). IEEE Trans. Neural Networks **10**(4) (1999) 939–945
8. López-Rubio, E., Muñoz-Pérez, J., Gómez-Ruiz, J.A.: A Principal Components Analysis Self-Organizing Map. Neural Networks **17**(2) (2004) 261–270
9. Liu, Z.Q.: Retrieving faces using Adaptive Subspace Self-Organising Map. In: Proc. International Symposium on Intelligent Multimedia, Video and Speech Processing, Hong Kong (2001) 377–380

10. Kawano, H., Horio, K., Yamakawa, T.: Adaptive Affine Subspace Self-organizing Map with kernel method. In Pal, N.R., Kasabov, N., Mudi, R.K., Pal, S., Parui, S.K., eds.: Proc. 11th Int. Conf. on Neural Information Processing. Volume 3316 of Lecture Notes in Computer Science., Calcutta, India, Springer (2004) 387–392
11. López-Rubio, E., Muñoz-Pérez, J., Gómez-Ruiz, J.A., Domínguez-Merino, E.: New learning rules for the ASSOM network. Neural Comput & Applic **12**(2) (2003) 109–118
12. Suen, C.Y., Nadal, C., Legault, R., Mai, T.A., Lam, L.: Computer recognition of unconstrained handwritten numerals. Proc. IEEE **80**(7) (1992) 1162–1180
13. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11) (1998) 2278–2324
14. Oja, E.: Subspace Methods of Pattern Recognition. Research Studies Press, Letchworth, UK (1983)
15. Hinton, G.E., Dayan, P., Revow, M.: Modeling the manifolds of images of handwritten digits. IEEE Trans. Neural Networks **8**(1) (1997) 65–74
16. Atkeson, C.G., Moore, A.W., Schaal, S.: Locally weighted learning. Artificial Intelligence Review **11**(1-5) (1997) 11–73