ESANN 2010 proceedings, European Symposium on Artificial Neural Networks - Computational Intelligence and Machine Learning. Bruges (Belgium), 28-30 April 2010, d-side publi., ISBN 2-930307-10-2.

Efficient online learning of a non-negative sparse autoencoder

Andre Lemme, R. Felix Reinhart and Jochen J. Steil

Research Institute for Cognition and Robotics (CoR-Lab), Bielefeld University
{alemme, freinhar, jsteil}@CoR-Lab.Uni-Bielefeld.de

Abstract. We introduce an efficient online learning mechanism for non-negative sparse coding in autoencoder neural networks. In this paper we compare the novel method to the batch algorithm non-negative matrix factorization with and without sparseness constraint. We show that the efficient autoencoder yields to better sparseness and lower reconstruction errors than the batch algorithms on the MNIST benchmark dataset.

1 Introduction

Unsupervised learning techniques that can find filters for relevant parts in images are particularly important for visual object classification [1]. Lee proposed a non-negative matrix factorization (NMF) in [2] to produce a non-negative encoding of input images by combining a linear matrix factorization approach with non-negativity constraints. Hoyer introduced an additional sparseness constraint to the factorization process in order to increase the sparseness of the produced encoding [3, 4]. Sparse codes enhance the probability of linear separability, a crucial feature for object classification [5]. The main drawbacks of matrix factorization approaches are high computational costs and the fact that the costly matrix factorization has to be redone each time new images are perceived. This is also true for the sort-of-online NMF introduced in [6]. Therefore, NMF is not suited for problems that require processing in real-time and life-long learning. A non-negative and online version of the PCA was introduced recently [7]. This approach addresses the problem of non-negativity and computational efficiency. However, PCA is intrinsically a non-sparse method.

We propose a modified autoencoder model that encodes input images in a non-negative and sparse network state. We use logistic activation functions in the hidden layer in order to map neural activities to strictly positive outputs. Sparseness of the representation is achieved by an unsupervised self-adaptation rule, which adapts the non-linear activation functions based on the principle of intrinsic plasticity [8]. Non-negativity of connection weights is enforced by a novel, asymmetric regularization approach that punishes negative weights more than positive ones. Both online learning rules are combined, use only local information, and are very efficient. We compare the reconstruction performance as well as the generated codes of the introduced model to the non-negative offline algorithms NMF and non-negative matrix factorization with sparseness constraints (NMFSC) on the MNIST benchmark data set [9]. The efficient online implementation outperforms the offline algorithms with respect to reconstruction errors and sparseness of filters as well as code vectors.

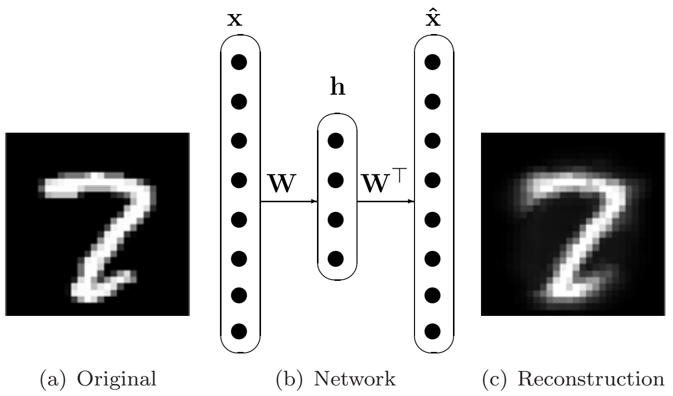

Fig. 1: Autoencoder network with input image $\mathbf{x}$ (left) and reconstructed image $\hat{\mathbf{x}}$ (right). $\mathbf{W} \equiv \mathbf{W}^T$ is the tied weight matrix and $\mathbf{h}$ the hidden network state.

2 Non-negative sparse autoencoder network (NNSAE)

We modify an autoencoder network in order to obtain non-negative and sparse encodings with only positive network weights from non-negative input data. The network architecture is shown in Fig. 1. We denote the input image by $\mathbf{x} \in \mathbb{R}^n$ and the network reconstruction by $\hat{\mathbf{x}} = \mathbf{W}^T\mathbf{f}(\mathbf{W}\mathbf{x})$, where $\mathbf{W} \in \mathbb{R}^{m \times n}$ is the weight matrix and $\mathbf{f}(\cdot)$ are parameterized activation functions $f_i(g_i) = (1 + exp(-a_i g_i - b_i))^{-1} \in [0, 1]$ with slopes a_i and biases b_i that are applied component-wise to the neural activities $\mathbf{g} = \mathbf{W}\mathbf{x}$ ($\mathbf{g} \in \mathbb{R}^m$). Non-negative code vectors are already assured by the logistic activation functions $f_i(g_i)$.

Learning to reconstruct: Learning of the autoencoder is based on minimizing the reconstruction error $E = \frac{1}{N}\sum_{i=1}^{N} ||x_i - \hat{x}_i||^2$ on the training set. Note that $\mathbf{W} \equiv \mathbf{W}^T$ is the same matrix for encoding and decoding, i.e. the autoencoder has *tied* weights [10, 11], which reduces the number of parameters to be adapted. Then, learning boils down to a simple error correction rule for all training examples $\mathbf{x}$

$$\tilde{w}_{ij} = w_{ij} + \eta\,(x_i - \hat{x}_i)\,h_j, \tag{1}$$

$$\Delta w_{ij} = \eta\,(x_i - \hat{x}_i)\,h_j + d(\tilde{w}_{ij}), \tag{2}$$

where w_{ij} is the connection from hidden neuron j to the output neuron i and h_j is the activation of neuron j. The decay function $d(\tilde{w}_{ij})$ will be discussed in the next section. We use an adaptive learning rate $\eta = \tilde{\eta}\,(||\mathbf{h}||^2 + \epsilon)^{-1}$ that scales the gradient step size $\tilde{\eta}$ by the inverse network activity and is similar to backpropagation-decorrelation learning for reservoir networks [12].

Asymmetric decay enforces non-negative network parameters: In order to enforce non-negative weights, we introduce an asymmetric regularization approach. Typically, a Gaussian prior for the network weights is assumed, which is expressed by a quadratic cost term $\lambda||\mathbf{W}||^2$ added to the error function E. Using gradient descent, the Gaussian prior results in a so called decay term

$d(w_{ij}) = -\lambda w_{ij}$. We assume a virtually deformed Gaussian prior that is skewed with respect to the sign of the new computed weight $\tilde{w}_{ij}$ (eq. 1). Gradient descent then yields the following asymmetric, piecewise linear decay function:

$$d(\tilde{w}_{ij}) = \begin{cases} -\alpha\, \tilde{w}_{ij} & \text{if } \tilde{w}_{ij} < 0 \\ -\beta\, \tilde{w}_{ij} & \text{else} \end{cases} \tag{3}$$

The main advantage of (3) is the parameterized decay behavior depending on the sign of the weight: it is possible to allow a certain degree of negative weights ($0 \leq \alpha \ll 1$) or to prohibit negative weights completely ($\alpha = 1$). The decay function (3) falls back to the case of a symmetric prior for $\alpha = \beta$.

Adaptation of non-linearities achieves sparse codes: In the context of the non-negative autoencoder, it is crucial to adapt the activation functions to the strictly non-negative input distribution caused by the non-negative weight and data matrix. We therefore apply an efficient, unsupervised and local learning scheme to the parameterized activation functions $f_i(g_i)$ that is known as intrinsic plasticity (IP) [8]. In addition, IP enhances the sparseness of the input representation in the hidden layer [13]. For more details about IP see [8, 13].

We use a small learning rate $\eta_{IP} = 0.0001$ throughout the experiments and initialize the parameters of the activation functions $\mathbf{a} = \mathbf{1}$ and $\mathbf{b} = \mathbf{-3}$ in order to accelerate convergence. IP provides a parameter to adjust the desired mean network activation, which we set to $\mu = 0.2$ corresponding to a sparse network state. In addition, we use a constant bias decay $\Delta\mathbf{b} = \Delta\mathbf{b}_{IP} - 0.00001$ to further increase the sparseness of the encoding.

3 Encoding and decoding of handwritten digits

We analyze the behavior of the non-negative sparse autoencoder (NNSAE) and compare the new NNSAE approach to the batch algorithms NMF and NMFSC on a benchmark data set.

Dataset and network training: The MNIST data set is commonly used to benchmark image reconstruction and classification methods [14]. For the following experiments the "MNIST-basic" set [9] is used, which comprises 12000 trainings and 50000 test images of handwritten digits from 0 to 9 in a centered and normalized 28×28 pixel format. We subdivide the training set into 10000 samples for learning and use the remaining 2000 examples as validation set. Some example digits from the test set are shown in the top row of Fig. 2.

We use an autoencoder with 784 input neurons and 100 hidden neurons to encode and decode the input images. The weight matrix is initialized randomly according to a uniform distribution in $[0.95, 1.05]$. We set $\tilde{\eta} = 0.01$ and $\epsilon = 0.002$ in (1). Each autoencoder is trained for 100 epochs, where the whole training set is presented to the model in each epoch. To account for the random network initialization, we present all results averaged over 10 trials.

Impact of asymmetric decay: We first investigate the impact of the asymmetric decay function (3). The following variates are calculated to quantify the

	$\alpha=0$	$\alpha=0.1$	$\alpha=0.2$	$\alpha=0.3$	$\alpha=1.0$
MSE	0.0107	0.0149	0.0149	0.0150	0.0151
$STD \cdot 10^{-4}$	0.5966	0.4079	0.7153	0.7143	0.6893
$s(\mathbf{H})$	0.3954	0.3465	0.3474	0.3460	0.3512
$s(\mathbf{W})$	0.5334	0.9379	0.9363	0.9397	0.9322

Tab. 1: MSE with standard deviation on the test set and sparseness of the encodings $\mathbf{H}$ and network weights $\mathbf{W}$ depending on asymmetric decay rate α.

model properties in dependence on α: (a) Pixelwise mean square error

$$MSE = \frac{1}{MN} \sum_{i=1}^{N} \sum_{j=1}^{M} (x_j^i - \hat{x}_j^i)^2 \tag{4}$$

of the reconstructions, where M is the dimension of the data and N the number of samples. (b) Average sparseness of the code matrix $\mathbf{H}$ and filters $\mathbf{W}$, where we estimate the sparseness of a vector $\mathbf{v} \in \mathbb{R}^n$ by

$$s(\mathbf{v}) = (\sqrt{n} - (\Sigma_{i=1}^{n} |v_i|)/\sqrt{\Sigma_{i=1}^{n} v_i^2}))(\sqrt{n} - 1)^{-1}. \tag{5}$$

This function was introduced by Hoyer [4] and rates the energy of a vector $\mathbf{v}$ with values between zero and one, where sparse vectors $\mathbf{v}$ are mapped to $s(\mathbf{v}) \gg 0$.

Tab. 1 shows the MSE and the average sparseness of hidden states as well as connection weights depending on α while $\beta \equiv 0$. The results in Tab. 1 for the case without decay ($\alpha = 0$) show that a certain amount of negative weights seems to be useful in terms of the reconstruction error. However, if we commit to non-negativity, it does not matter whether there is just a small set of negative ($\alpha = 0.1$) or only positive entries ($\alpha = 1$) in the weight matrix. This indicates that a rather moderate non-negativity constraint causes the weight dynamics and the final solution to change completely. We therefore set $\alpha = 1$ in the following experiments and focus on strictly non-negative filters $\mathbf{w}_i \in \mathbb{R}^n$.

4 Online NNSAE versus offline NMF and NMFSC

The batch algorithms NMF and NMFSC solve the equation $\mathbf{X} = \mathbf{WH}$ starting from the data matrix $\mathbf{X}$ and initial filters $\mathbf{W}$ and code matrix $\mathbf{H}$. Then, iterative updates of the code and weight matrix are conducted in order to reduce the reconstruction error. We iterate the factorization algorithm 100 times on the training set to make the comparison with the NNSAE fair. We further set the sparseness parameters provided by NMFSC for the weight and code matrix to $s(\mathbf{W}) = 0.9$ and $s(\mathbf{H}) = 0.35$ according to the values obtained for the NNSAE. For more details about NMF and NMFSC see [2, 4]. To evaluate the generalization of NMF and NMFSC on a test set, we keep the trained filters $\mathbf{W}$ fixed and update only the code matrix $\mathbf{H}$ iteratively as in the training.

	MSE			Sparseness	
	Test	Validation	Train	$sp(\mathbf{H})$	$sp(\mathbf{W})$
NMF	$0.047 \pm 1 \cdot 10^{-4}$	$0.222 \pm 5 \cdot 10^{-3}$	$0.011 \pm 3 \cdot 10^{-4}$	0.26	0.89
NMFSC	$0.109 \pm 6 \cdot 10^{-6}$	$0.108 \pm 3 \cdot 10^{-5}$	$0.014 \pm 2 \cdot 10^{-4}$	0.29	0.90
NNSAE	$0.015 \pm 6 \cdot 10^{-5}$	$0.016 \pm 6 \cdot 10^{-5}$	$0.012 \pm 7 \cdot 10^{-5}$	0.35	0.93

Tab. 2: Mean square reconstruction errors for training, validation and test set as well as sparseness of code and weight matrices for NMF, NMFSC and NNSAE.

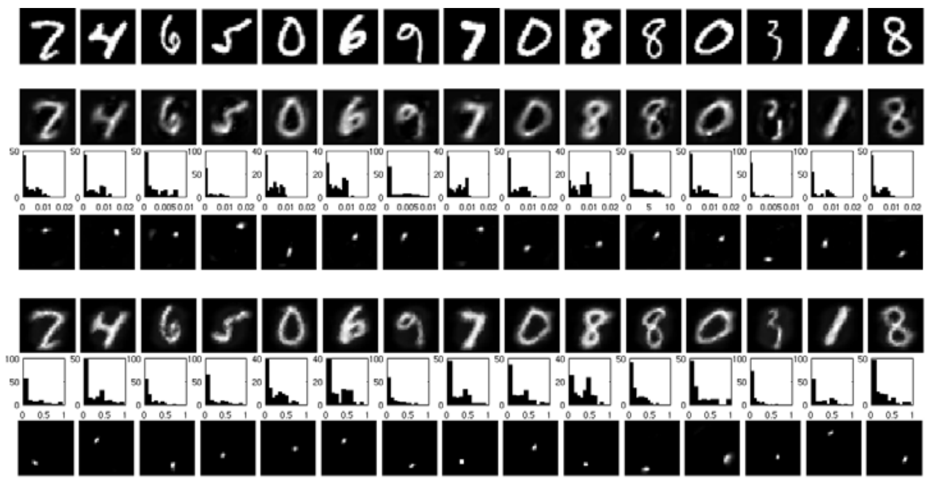

Fig. 2: 15 images from the test set (top row) with reconstructions (2nd row), code histogram (3rd row) and some filters (4th row) performed by the NMFSC. Reconstruction (5th row), code histogram (6th row) and some filters (7th row) performed by the NNSAE.

Comparison of reconstruction errors and sparseness: Tab. 2 gives a review of the performance of NMF/NMFSC and compares it to the NNSAE. The NNSAE outperforms NMF/NMFSC with respect to the MSE on the test set, sparseness of the code matrix $\mathbf{H}$ and sparseness of the weight matrix $\mathbf{W}$. It is surprising that even NMFSC does not reach the sparse codes that we achieve with the NNSAE, although the constraint is set to be equal ($s(\mathbf{H}) = 0.35$). We show for some examples the corresponding code histogram of NMFSC and NNSAE in Fig. 2. The sparseness of the filters $\mathbf{w}_i$ does not differ significantly for the different methods, which seems to be the effect of the non-negative constraint itself. In Fig. 2 we show in the 4th and the last row some filters of NMFSC and NNSAE. The filters of both methods cover only blob-like areas, i.e. are local feature detectors, which corresponds to a sparse representation.

5 Discussion and conclusion

We present an autoencoder for efficient online learning of sparse and non-negative encodings. Therefore, we combine a mechanism to enhance the sparseness of encodings and an asymmetric decay function to create non-negative weights. The online trained autoencoder outperforms the offline techniques NMF/NMFSC when reconstructing images from the test set, underlining its generalization capabilities. Additionally, the autoencoder produces sparser codes compared to the offline methods while generating similar filters at the same time. Besides the life-long learning capability, the main advantage of the new approach is the efficient encoding of novel inputs: the state of the network has simply to be updated with the new input, where the matrix factorization approaches in contrast require a full optimization step on the new data.

It is of particular interest to use the NNSAE in a stacked autoencoder network for pattern recognition in the future: Does the non-negative representation improve classification performance, and how is fine-tuning of the hierarchy, e.g. by backpropagation learning, affected by the asymmetric decay function? Also, the interplay of IP and synaptic plasticity in the context of the NNSAE has to be further investigated.

References

[1] M. Spratling. Learning Image Components for Object Recognition. *Mach. Learn.*, 2006.

[2] D. D. Lee, and H. S. Seung. Learning the parts of objects by nonnegative matrix factorization. *Nature*, pp. 788–791, 1999.

[3] P. O. Hoyer. Non-negative sparse coding. *Neural Networks for Signal Processing XII*, pp. 557-565, 2002.

[4] P. O. Hoyer. Non-negative Matrix Factorization with Sparseness Constraints. *Journal of Machine Learning Research*, pp. 1457–1469, 2004.

[5] Y-lan Boureau and Y. Lecun. Sparse Feature Learning for Deep Belief Networks. *NIPS*, pp. 1–8, 2007.

[6] B. Cao. Detect and Track Latent Factors with Online Nonnegative Matrix Factorization *Matrix*, pp. 2689–2694, 1999.

[7] M. D. Plumbley and O. Erkki. A "nonnegative PCA" algorithm for independent component analysis. *Neural Networks*, pp. 66–76, 2004.

[8] J. Triesch. A Gradient Rule for the Plasticity of a Neuron's Intrinsic Excitability. *Neural Computation*, pp. 65–70, 2005.

[9] Variations of the MNIST database. `http://www.iro.umontreal.ca/~lisa/ptwiki`.

[10] G. E. Hinton and S. Osindero. A fast learning algorithm for deep belief nets. *Neural Computation*, pp. 1527–54, 2006.

[11] P. Vincent and H. Larochelle and Y. Bengio. Extracting and composing robust features with denoising autoencoders *ICML '08*, pp. 1096–1103, 2008.

[12] J. J. Steil. Backpropagation-Decorrelation: online recurrent learning with O(N) complexity *Proc. IJCNN*, pp. 843-848, 2004.

[13] J. Triesch. Synergies Between Intrinsic and Synaptic Plasticity Mechanisms. *Neural Computation*, pp. 885–909, 2007.

[14] Y. Lecun and C. Cortes. The MNIST database of handwritten digits. `http://yann.lecun.com/exdb/mnist`.