Training of Sparsely Connected MLPs

Markus Thom[1], Roland Schweiger[1], and Günther Palm[2]

[1] Department Environment Perception (GR/PAP), Daimler AG, Ulm, Germany
[2] Institute of Neural Information Processing, University of Ulm, Germany

Abstract. Sparsely connected Multi-Layer Perceptrons (MLPs) differ from conventional MLPs in that only a small fraction of entries in their weight matrices are nonzero. Using sparse matrix-vector multiplication algorithms reduces the computational complexity of classification. Training of sparsely connected MLPs is achieved in two consecutive stages. In the first stage, initial values for the network's parameters are given by the solution to an unsupervised matrix factorization problem, minimizing the reconstruction error. In the second stage, a modified version of the supervised backpropagation algorithm optimizes the MLP's parameters with respect to the classification error. Experiments on the MNIST database of handwritten digits show that the proposed approach achieves equal classification performance compared to a densely connected MLP while speeding-up classification by a factor of seven.

1 Introduction

Multi-Layer Perceptrons [1] have been widely employed in a vast range of classification tasks, especially for handwritten digit recognition [11]. MLP parameter tuning is traditionally achieved using the backpropagation algorithm [18]. This procedure has recently tied the record [3] on the MNIST database of handwritten digits [12], achieving an error of 0.35%. For this, the MLP comprised six layers with 12 million synaptic connections. The learning set was extended by generating artificial training examples using elastic distortions [19].

This paper focuses on posing sparseness constraints on the weight matrices of MLPs, so that only a small fraction of entries are nonzero. Thus, sparsely connected MLPs or briefly sparse MLPs (SMLPs) are yielded. This is motivated in part by the fact that neurons in biological neuronal systems are not connected to every other neuron in the network [22]. Sparse connectivity helps to reduce both memory usage and computational complexity of the classification task. By exploiting the structure of sparsely populated weight matrices, the principal module of an MLP's feeding-forward mechanism can be sped up. This is especially advantageous in industrial applications, where a system may be rendered real-time capable or where hardware cost may be reduced [15,17].

The main focus of this paper is the training of sparsely connected MLPs. The amount of nonzero connection weights is set a-priori, and can hence be chosen to obtain a classifier with predictable time complexity. In the first of two consecutive stages, a matrix factorization algorithm that uses sparse filter matrices for

R. Mester and M. Felsberg (Eds.): DAGM 2011, LNCS 6835, pp. 356–365, 2011.

generating sparse and information-preserving representations is used to initialize the sparse MLP's parameters. In the second stage, a modified backpropagation learning algorithm finds an MLP that is optimal with respect to the classification error, simultaneously fulfilling sparseness constraints.

The remainder of this paper is organized as follows: Section 2 addresses the problem of computing sparse representations in an unsupervised manner suitable for sparse MLP initialization. In Sect. 3, an algorithm for training sparsely connected MLPs is proposed. The results of this technique applied to handwritten digits are demonstrated in Sect. 4, and compared with alternative approaches in Sect. 5. The final section contains a summary and conclusion.

2 Sparse Generative Models for SMLP Initialization

Training of sparsely connected MLPs from randomly initialized parameters is an ill-conditioned problem. Network parameter initialization constitutes the most crucial part in the whole procedure. One key idea that has recently found its way into the pattern classification community is the unsupervised pre-training of classifiers [8]. Here, the computation of sparse representations is of particular interest. Considering linear generative models, a sample $x \in \mathbb{R}^d$ is approximated by the linear combination of a matrix of bases $W \in \mathbb{R}^{d \times n}$ with a code word $h \in \mathbb{R}^n$, such that $x \approx Wh$. Traditional sparse coding [7] is achieved by requiring most entries of h to be zero. Another notion is the restriction that W be sparsely populated, which is the main focus of this section.

One of the most important mathematical models known to reproduce certain data sets using a sparse matrix of bases is Non-Negative Matrix Factorization (NMF) [14]. It aims to factorize a data matrix $X \in \mathbb{R}_{\geq 0}^{d \times M}$ of M samples with non-negative entries into the product of a matrix of bases $W \in \mathbb{R}_{\geq 0}^{d \times n}$ and a matrix of code words $H \in \mathbb{R}_{\geq 0}^{n \times M}$, both with non-negative entries. Since NMF is only allowed to make additive combinations in a linear generative model framework, sparse matrices W can be achieved [14]. However, there are data sets where NMF fails to produce sparse representations without further modifications to the algorithm itself [9].

2.1 Non-Negative Matrix Factorization with Sparseness Constraints

NMF is extended by Non-Negative Matrix Factorization with Sparseness Constraints (NMFSC) [9] so that the sparseness of the representation becomes easily controllable. In doing so, a formal sparseness measure σ based on a normalized quotient of the L^1 norm and the L^2 norm of a vector has been proposed:

$$\sigma \colon \mathbb{R}^d \setminus \{0\} \to [0,1], \qquad x \mapsto \frac{\sqrt{d} - \frac{\|x\|_1}{\|x\|_2}}{\sqrt{d} - 1} \ . \tag{1}$$

Using σ, NMFSC must minimize the reproduction error in Frobenius norm, $E_{\mathrm{NMF}}(W,H) := \|X - WH\|_F^2$, subject to $\sigma(We_i) = \sigma_W$ and $\sigma(H^t e_i) = \sigma_H$ for

all $i \in \{1, \ldots, n\}$ for constant sparseness degrees $\sigma_W, \sigma_H \in (0, 1)$. Here, e_i is the i-th canonical basis vector. σ_W controls the sparseness of the individual columns of W. σ_H controls the fraction of samples each column of W contributes to.

The biconvex objective function is minimized via alternating gradient descent on W and H, with each step followed by a sparseness-enforcing projection to meet the sparseness constraints. This sparseness-enforcing projection is computed by iterative projection on a hyperplane satisfying an L^1 norm constraint and on a hypersphere satisfying an L^2 norm constraint [9,20].

The major drawback of NMFSC is that it is only a generative architecture. However, computation of sparse code words h from arbitrary samples x is possible through a cost-intensive optimization. Unfortunately, no real-time capable algorithm to solve this code word inference problem is known.

2.2 Extension for Fast Inference of Sparse Code Words

To address this issue, an extension ensuring fast inference of sparse code words has recently been proposed [21]. There, inference is modeled as feeding forward the training samples through a one-layer perceptron, employing a non-linearity that approximates a soft-shrinkage operation. Similar to [16], the matrix of bases is used for code word inference as well.

As sparseness is enforced through NMFSC's projection operator, the non-negativity constraint is no longer needed. Hence let $X \in \mathbb{R}^{d \times M}$, $W \in \mathbb{R}^{d \times n}$ and $H \in \mathbb{R}^{n \times M}$, allowing for a more general range of applications. This enables the normalization of the training samples to zero mean and unit variance, which is advantageous in classification scenarios [13].

To extend NMFSC for fast inference, a vector of thresholds $\theta \in \mathbb{R}^n$ and a nonlinear transfer function $f\colon \mathbb{R}^n \to \mathbb{R}^n$ are introduced. Feed-forward code words are defined to be a one-layer perceptron's output, $x \mapsto f(W^t x + \theta)$. Here, f is chosen to be a hyperbolic tangent raised to an odd exponent greater or equal to three, that is $f\colon \mathbb{R} \to \mathbb{R}$, $x \mapsto (\tanh(\beta x))^q$ with $\beta > 0$ and $q \in 2\mathbb{N}_0 + 3 = \{3, 5, 7, \ldots\}$. The exponentiation has a similar effect as applying a soft-shrinkage function after a hyperbolic tangent transfer function, thus allowing for sparse inferred code words. However, this function has the advantage of being differentiable everywhere. In this paper, $\beta = 1$ and $q = 3$ were chosen. By computing the squared difference between the code word matrix and the matrix of feed-forward code words, the inference error is given by $E_{\text{Inf}}(W, \theta, H) := \|H - f(W^t X + \theta \cdot J_{1\times M})\|_F^2$. Here, $J_{1\times M} \in \mathbb{R}^{1\times M}$ denotes a matrix containing only ones and is employed for repeating the threshold over all M samples.

The objective function is defined to lie between reconstruction error and inference error, controlling the trade-off with a parameter $\alpha_{\text{Inf}} \in [0, 1]$:

$$E(W, \theta, H) := (1 - \alpha_{\text{Inf}}) \cdot E_{\text{NMF}}(W, H) + \alpha_{\text{Inf}} \cdot E_{\text{Inf}}(W, \theta, H) \ . \qquad (2)$$

Here, α_{Inf} was chosen to start from zero and reach $1/2$ asymptotically. E is minimized by alternating gradient descent. Each update of W and H is followed by sparseness-enforcing projections, ensuring that the sparseness constraints are met: $\sigma(We_i) = \sigma_W$ and $\sigma(H^t e_i) = \sigma_H$ for all $i \in \{1, \ldots, n\}$.

3 Sparsely Connected MLPs

In the preceding section, only generative models were investigated. It turns out that state-of-the-art performance in classification problems can not be achieved solely by using unsupervised learning algorithms. This is because the representations have only been optimized to retain a maximum of the information in the training samples, regardless of their class labels. However, the unsupervised algorithms provide adequate initializers for more discriminative architectures. Exploiting the emergence of sparse matrices of bases, an efficient way of training sparsely connected MLPs is proposed in this section.

3.1 Architecture

Consider an MLP with L layers, where $\mathcal{W} := (W_1, \ldots, W_L)$ and $\Theta := (\theta_1, \ldots, \theta_L)$ denote the weight matrices and the threshold vectors of the individual layers, respectively. Let $W_i \in \mathbb{R}^{d_{i-1} \times d_i}$ and $\theta_i \in \mathbb{R}^{d_i}$ for all $i \in \{1, \ldots, L\}$, where $d_0, d_1, \ldots, d_{L-1}, d_L \in \mathbb{N}$ denote the complexity of the individual layers. Given a dataset of samples and teacher signals, $\mathcal{W}$ and Θ are adjusted by minimizing the deviation between network output given the samples and the corresponding teacher signals. This deviation can be measured using the mean square error function or the cross-entropy error function [1]. In classification problems with $n > 2$ distinct classes, the teacher signals are represented using 1-of-n codes. The final layer's transfer function then is set to the softmax function [1].

In sparse MLPs, the weight matrices are enforced to meet specific sparseness constraints. Let $\sigma_{W_1}, \ldots, \sigma_{W_L} \in (0, 1)$ be sparseness degrees for every layer. The problem of training SMLPs then becomes minimizing the error function subject to $\sigma(W_i e_j) = \sigma_{W_i}$ for all $i \in \{1, \ldots, L\}$ and for all $j \in \{1, \ldots, d_i\}$. Here, σ is the sparseness measure from Sect. 2.1. In practice, posing sparseness constraints to the final layer is often not beneficial, as it constitutes the final weighting of the MLP's internal state. In the remainder of this paper, analysis is restricted to final layers without sparseness constraints. In the case of two-layer MLPs, let $\sigma_W := \sigma_{W_1}$ denote the sparseness degree of the only hidden layer.

3.2 Training Algorithm

Similar to Radial Basis Function (RBF) networks [1], a proper initialization of $\mathcal{W}$ and Θ is crucial for the success in training SMLPs. Using unsupervised pre-training, this is not limited to RBF networks [8]. By employing the learning algorithm from Sect. 2.2, layers 1 to $L-1$ are initialized in a layer-wise manner. This guarantees that the respective weight matrices are sparsely populated, and every layer is able to reproduce its individual input.

Supervised linear SVM training [6] is used to initialize the final layer. If the classification problem is not binary, each hidden unit in layer L represents one class using 1-of-n codes. Linear SVM training is then run in a one vs. all fashion, and the corresponding column of W_L and the corresponding entry of θ_L are set to contain the linear SVM's weight vector and threshold, respectively.

Supervised training of sparsely connected MLPs is achieved through sequential gradient descent, using the backpropagation algorithm [18]. With the original backpropagation algorithm, the connection matrices that were initialized to be sparsely populated lose this property during training. This indicates that sparse connection matrices do not minimize the unconstrained classification error. Minimization of the error function subject to sparseness constraints is achieved by projected gradient descent, that is the sparseness-enforcing projection from Sect. 2.1 is carried out after presentation of a few thousand samples to the network. It is important not to project too early, because then $\mathcal{W}$ will not be modified effectively during learning. By projecting too late, the changes to $\mathcal{W}$ between projections become too intense, resulting in divergence.

After training has converged, an MLP with connection matrices meeting sparseness constraints results. The connectivity rate in the weight matrices is set a-priori using sparseness degrees. The relationship between both values is non-linear, but can be calculated beforehand. Naive pruning of connections with small absolute values can increase sparseness further. The threshold for pruning has to be verified on the learning set so that the classification error is not increased.

3.3 Computational Complexity of Classification

In practice, the computational complexity of the classification routine is relevant for designing the classifier's architecture, adjusting it to specific hardware needs. Let $A \in \mathbb{R}^{d\times n}$ be a matrix and $x \in \mathbb{R}^d$ be a vector. Investigating the problem of computing the matrix-vector product $A^t x$, the kernel of this computation is a multiply-accumulate (MAC) operation, $D \leftarrow D + A(j,i) \cdot x(j)$. Here, D is a data register employed to avoid having to store the product in memory after every execution. The MAC operation is executed dn times, and $2dn$ numbers have to be fetched from memory. If A is a sparsely populated matrix with exactly k nonzero entries in each column, it can be stored as pair of a matrix $P \in \mathbb{N}_0^{k\times n}$ of positions of nonzero elements with a matrix of according values $V \in \mathbb{R}^{k\times n}$. The kernel of the sparse matrix-vector product computation then becomes the MAC operation $D \leftarrow D + V(j,i) \cdot x(P(j,i))$, and is carried out kn times. In total, $3kn$ numbers now have to be fetched from memory, also accounting for the entries of P. If $k < \frac{2}{3}d$, the sparse matrix-vector multiplication needs less read accesses than the dense counterpart does. Method effectiveness thus increases with d.

The computation of matrix-vector products is the dominating part of the overall computational complexity of MLP classification. The application of the thresholds is equivalent to initializing register D with the concrete threshold value and requires only n additional memory accesses. Transfer function evaluation can be sped up by employing a function that can be evaluated by means of algebraic operations on the argument [5]. Hence, no auxiliary data needs to be read from memory, as look-up table entries or coefficients of a Taylor expansion would be required otherwise. The final layer's transfer function can always be replaced with a linear one since the classification decision is equivalent to thresholding in binary classification problems and to finding the maximum entry in the network's output vector if more than two classes are involved.

In summary, the computational complexity of classification with MLPs depends strongly on the number of nonzero entries in the weight matrices. Sparsely connected MLPs require some overhead with respect to the read accesses, as the more complex data structure has to be stored as well. For comparison, a simple computational model that accounts for MAC operations and memory accesses is defined. Let $Z \in \mathbb{R}$ be the memory latency relative to the cost of a MAC operation. Considering a two layer $d - h - n$ MLP, the cost for classifying one sample is $(1 + 2Z)\, h\, (d + n)$ operations. A sparse MLP with a connectivity rate of $\rho \in (0, 1)$ in the first layer needs $(1 + 3Z)\, \rho dh + (1 + 2Z)\, hn$ operations. The speed-up factor $S\,(Z, \rho)$ is then the quotient of the two quantities and is independent of the number of hidden units h. $S(Z, \rho)$ converges decreasingly for very high memory latencies to

$$S_\infty\,(\rho) := \lim_{Z \to \infty} S(Z, \rho) = \lim_{Z \to \infty} \frac{\left(\frac{1}{Z} + 2\right)(d + n)}{\left(\frac{1}{Z} + 3\right)\rho d + \left(\frac{1}{Z} + 2\right) n} = \frac{2\,(d + n)}{3\rho d + 2n} \quad . \tag{3}$$

In this computational model, a lower bound of sparse MLP speed-up over dense MLPs is given by $S_\infty\,(\rho)$, which is independent of memory latencies.

4 Experiments on Handwritten Digits

The performance of sparse MLPs is evaluated on the MNIST database of handwritten digits [12]. It consists of 70 000 samples, divided into a learning set of 60 000 samples and an evaluation set of 10 000 samples. Each sample represents a digit of size 28×28 pixels and has a class label from $\{0, \ldots, 9\}$ associated with it. The samples were normalized to achieve zero mean and unit variance.

Three variants of the learning set have been employed for the experiments. The first one is the original learning set with 60 000 samples. To investigate the effect of small translations, the samples have been jittered by 1 pixel in each of 8 directions, yielding 540 000 samples. Finally, by applying additional elastic distortions [19], a learning set with 13.5 million samples was generated.

The architecture was fixed to two-layer MLPs with 1000 neurons in the hidden layer. Although employing more layers has improved classification performance [3], this paper focuses on creating classifiers with very low computational complexity. For reference, conventional densely connected MLPs have been trained on the three learning sets and tested on the 10 000 samples of the evaluation set, yielding errors of 1.9%, 0.89%, and 0.61%, respectively. Thus, similar to [2,4], adding jittered samples to the learning set significantly improves the classification error. The localization uncertainty can be explained by the original placement of the digits based on the center of mass of their pixel values [12]. This illustrates that MLPs are not able to learn invariants well, and hence rely heavily on artificial training samples added to the learning set. Adding even more samples using elastic distortions further decreases the classification error.

Two-thirds of the connections in the first layer of the MLP trained on the jittered learning set have been removed through naive pruning based on their

Table 1. Comparison of a conventional MLP with sparsely connected MLPs on the MNIST dataset. All MLPs had two layers with 1000 units in the hidden layer. Only the first connection matrix W_1 was required to be sparse, W_2 was densely populated.

Sparseness degree σ_W	none	0.75	0.80	0.85	0.90
Connectivity rate ρ in W_1 [%]	100	12.9	8.7	6.6	3.7
Number of synaptic connections	800 000	110 000	78 000	62 000	39 000
Speed-up $S_\infty(\rho)$ to dense MLP	1	4.9	7.1	9.1	15
Error on evaluation set [%]	0.89	0.83	0.86	0.91	0.94

absolute values. This increased the error from 0.89% to 1.2%. If only enough connections were removed so that the classification error did not increase, 55.7% of the connections still remained. Thus, only a mild speed-up factor of 1.2 could be achieved while retaining classification capabilities. If instead of naive pruning the sparseness-enforcing projection from Sect. 2.1 is used, the error increases drastically to 3.0%. This indicates that the solution found by unconstrained minimization is very distant from a minimum fulfilling sparseness constraints.

The results can be improved by training sparsely connected MLPs using the method from Sect. 3.2. As 96% of all samples possess a sparseness of less than 0.75, this value was chosen for the lower bound on the MLP sparseness. A summary of the results on the jittered learning set is given in Table 1. For $\sigma_W = 0.75$, an error of 0.83% was achieved, rendering it slightly better than the conventional MLP. Nevertheless, only 12.9% of the entries in the first connection matrix were nonzero, resulting in a lower bound to the speed-up factor of 4.9 compared to the dense MLP. For comparison, a dense MLP with a reduced number of hidden units has been trained. Though the dense MLP was adjusted for equal computational complexity as the SMLP, only an error of 1.1% could be achieved. For $\sigma_W = 0.80$, the error slightly increased, but the lower connectivity rate allowed for a speed-up of at least factor seven. The speed-ups increase more for higher sparseness degrees, while the classification capabilities degrade. However, even for very sparse connection matrices, the error remains below 1%.

Sparse MLPs for sparseness degrees 0.75, 0.80, 0.85, and 0.90 have also been trained on the learning set generated by elastic distortions, achieving significantly lower errors of 0.59%, 0.64%, 0.67%, and 0.72%, respectively. Though the SMLP with $\sigma_W = 0.75$ achieved an error statistically equal to the one of the conventional MLP, the connectivity rate in the hidden layer was 14.2%, which results in a minimum speed-up of factor 4.5.

To verify the initialization as described in Sect. 3.2, an MLP was initialized using random values and then trained on the jittered learning set with the backpropagation algorithm subject to sparseness constraints. After convergence, an error of 1.1% was achieved, which is significantly higher than the corresponding error of 0.89% using the unconstrained backpropagation algorithm. Thus, reasonable results could only be achieved using the combination of a sophisticated initialization scheme with a constrained backpropagation algorithm.

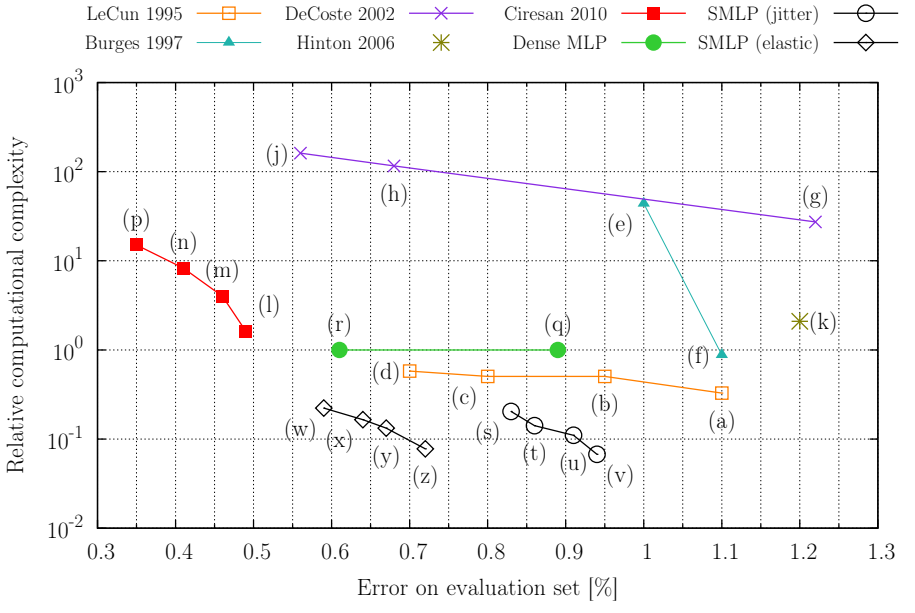

Fig. 1. Comparison of various approaches to MNIST considering classification error and computational complexity of classification. Computational complexity is given relative to the complexity of a dense two-layer MLP with 1000 hidden units. A detailed discussion is given in Sect. 5. This figure is best viewed in color.

5 Comparison with Alternative Approaches

The results described in the previous section are compared with alternative approaches from LeCun 1995 [10,11], Burges 1997 [2], DeCoste 2002 [4], Hinton 2006 [8], and Cireşan 2010 [3]. The two major points of this comparison are the error achieved on the evaluation set and the computational complexity of classification relative to a densely connected MLP with 1000 hidden units. The results of this discussion are illustrated in Fig. 1 and are referenced using signs (a)–(z). Though [19] achieved an error of 0.4% using convolutional neural networks, the computational complexity of their approach has not been published. Thus it is not included in this discussion.

A very efficient solution using a family of convolutional neural networks has been given by LeCun 1995 [10,11]: (a) LeNet-4 and (b) LeNet-5 using the original learning set, (c) LeNet-5 and (d) boosted LeNet-4 using an augmented learning set. The computational complexity has been determined based on the number of MAC operations from [11].

Burges 1997 [2] proposed a Virtual Support Vector Machine, adding samples jittered in four directions to the learning set. Their approach (e) achieves an error of 1.0%, but the computational complexity is very demanding. The computational cost can be reduced by approximating the SVM's hyperplane normal

by a reduced set of vectors, resulting in a speed-up of factor 22 [2], involving a slightly larger error (f). DeCoste 2002 [4] improve on the classification error by using samples jittered in eight directions. The error improves from (g) the original learning set to (h) jittering by 1 pixel and (j) jittering by 2 pixels. Computational complexity of [2] and [4] has been determined based on the number of support vectors. The complexity in the latter case is very high, but there no reduced set approach has been applied.

Hinton 2006 [8] used a deep network of Restricted Boltzmann Machines, pretrained using unsupervised algorithms. They (k) achieved an error of 1.2%, without having to augment the learning set using artificial training samples. Thus, their approach is completely invariant to permutations of the pixels of the input samples. Using a two-layer MLP, a significantly higher error of 1.9% was achieved using the very same learning set, see Sect. 4.

By employing very large MLPs, Cireşan 2010 [3] currently hold the record on MNIST classification. They employed elastic and affine distortions to obtain a huge learning set. MLP training was sped up by greatly exploiting parallelism on a graphics processing unit. They trained MLPs with (l) three layers, (m) four layers, (n) five layers, and (p) six layers. Unlike recent trends, they did not employ unsupervised pre-training before using the backpropagation algorithm to tune the MLP parameters.

The results obtained in this paper are quite competitive. The conventional two-layer MLP with 1000 hidden units trained on the jittered samples achieved (q) an error of 0.89%. By training on the elastically distorted samples, an error of 0.61% was achieved (r). By exploiting the sparse connection matrix of sparse MLPs, significant speed-ups can be gained while retaining similar classification performance. For the jittered learning set, SMLPs with sparseness degrees σ_W of (s) 0.75, (t) 0.80, (u) 0.85, and (v) 0.90 have been trained. On the elastic learning set, SMLPs with sparseness degrees of (w) 0.75, (x) 0.80, (y) 0.85, and (z) 0.90 have been trained, achieving significantly better classification performance but also higher computational complexity due to a higher connectivity rate.

6 Conclusions

Traditional Multi-Layer Perceptrons have been studied and applied intensely over the past decades. They have recently achieved state-of-the-art performance in handwritten digit recognition. In this paper, an algorithm for the training of sparsely connected MLPs has been proposed. In doing so, the MLP's parameters are initialized using the solution to an unsupervised matrix factorization problem. Then, the parameters are tuned to optimize classification capabilities using a projected gradient descent algorithm. A comparison with alternative approaches has shown that sparse MLPs achieve competitive classification performance, while computational complexity can be reduced using sparse matrix-vector multiplication algorithms. This enables using sparse MLPs in embedded systems, where real-time capable algorithms are mandatory, and each speed-up results in an effective reduction of hardware cost.

References

1. Bishop, C.M.: Neural Networks for Pattern Recognition. Clarendon Press, Oxford (1995)
2. Burges, C.J.C., Schölkopf, B.: Improving the Accuracy and Speed of Support Vector Machines. In: NIPS, vol. 9, pp. 375–381 (1997)
3. Cireşan, D.C., Meier, U., Gambardella, L.M., Schmidhuber, J.: Deep, Big, Simple Neural Nets for Handwritten Digit Recognition. Neural Computation 22(12), 3207–3220 (2010)
4. DeCoste, D., Schölkopf, B.: Training Invariant Support Vector Machines. Machine Learning 46, 161–190 (2002)
5. Elliott, D.: A Better Activation Function for Artificial Neural Networks. Tech. Rep. ISR TR 93-8, Institute for Systems Research, University of Maryland (1993)
6. Fan, R.E., Chang, K.W., Hsieh, C.J., Wang, X.R., Lin, C.J.: LIBLINEAR: A Library for Large Linear Classification. JMLR 9, 1871–1874 (2008)
7. Field, D.J.: What is the Goal of Sensory Coding? Neural Computation 6, 559–601 (1994)
8. Hinton, G.E., Salakhutdinov, R.R.: Reducing the Dimensionality of Data with Neural Networks. Science 313(5786), 1527–1554 (2006)
9. Hoyer, P.O.: Non-negative Matrix Factorization with Sparseness Constraints. JMLR 5, 1457–1469 (2004)
10. LeCun, Y., Jackel, L., Bottou, L., Brunot, A., Cortes, C., Denker, J., Drucker, H., Guyon, I., Müller, U., Säckinger, E., Simard, P., Vapnik, V.: Comparison Of Learning Algorithms For Handwritten Digit Recognition. In: Proceedings of ICANN, pp. 53–60 (1995)
11. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-Based Learning Applied to Document Recognition. Proceedings of the IEEE 86, 2278–2324 (1998)
12. LeCun, Y., Cortes, C.: The MNIST Database of Handwritten Digits, `http://yann.lecun.com/exdb/mnist`
13. LeCun, Y., Kanter, I., Solla, S.A.: Eigenvalues of Covariance Matrices: Application to Neural-Network Learning. Physical Review Letters 66(18), 2396–2399 (1991)
14. Lee, D.D., Seung, H.S.: Learning the parts of objects by nonnegative matrix factorization. Nature 401, 788–791 (1999)
15. Ortigosa, E.M., Cañas, A., Rodríguez, R., Díaz, J., Mota, S.: Towards an Optimal Implementation of MLP in FPGA. In: Bertels, K., Cardoso, J.M.P., Vassiliadis, S. (eds.) ARC 2006. LNCS, vol. 3985, pp. 46–51. Springer, Heidelberg (2006)
16. Ranzato, M., Boureau, Y., LeCun, Y.: Sparse Feature Learning for Deep Belief Networks. In: NIPS, vol. 20, pp. 1185–1192 (2008)
17. Rast, A.D., Welbourne, S., Jin, X., Furber, S.: Optimal Connectivity In Hardware-Targetted MLP Networks. In: Proceedings of IJCNN, pp. 2619–2626 (2009)
18. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Nature 323, 533–536 (1986)
19. Simard, P.Y., Steinkraus, D., Platt, J.C.: Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis. In: Proceedings of ICDAR, pp. 958–962 (2003)
20. Theis, F.J., Stadlthanner, K., Tanaka, T.: First results on uniqueness of sparse non-negative matrix factorization. In: Proceedings of EUSIPCO (2005)
21. Thom, M., Schweiger, R., Palm, G.: Supervised Matrix Factorization with Sparseness Constraints and Fast Inference. In: Proceedings of IJCNN (to appear, 2011)
22. Yoshimura, Y., Dantzker, J.L.M., Callaway, E.M.: Excitatory cortical neurons form fine-scale functional networks. Nature 433(7028), 868–873 (2005)