ESANN'2006 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 26-28 April 2006, d-side publi., ISBN 2-930307-06-4.

Recognition of handwritten digits using sparse codes generated by local feature extraction methods

Rebecca Steinert[1], Martin Rehn[2] and Anders Lansner[3] *

Royal Institute of Technology
CBN, School of Computer Science and Communication (CSC)
SE-100 44 Stockholm, SWEDEN
rebste@kth.se[1], rehn@nada.kth.se[2], ala@nada.kth.se[3]

Abstract. We investigate when sparse coding of sensory inputs can improve performance in a classification task. For this purpose, we use a standard data set, the MNIST database of handwritten digits. We systematically study combinations of sparse coding methods and neural classifiers in a two-layer network. We find that processing the image data into a sparse code can indeed improve the classification performance, compared to directly classifying the images. Further, increasing the level of sparseness leads to even better performance, up to a point where the reduction of redundancy in the codes is offset by loss of information.

1 Introduction

Real-world data, such as visual inputs, are often highly redundant [1]. It has been hypothesized that the aim of cortical information processing is to transform the input into a representation in terms of higher-order objects, reducing redundancy [2].

In the last four decades the interest in feature extraction methods generating codes with *sparse* neural activity and reduced redundancy has increased. In part, this interest is motivated by evidence that sensory coding in the early stages of cortical information processing produces sparse representations [1][3]. It has been shown that the use of sparse codes can improve performance when storing and retrieving information in attractor memories [4]. In addition, it has been suggested that such a coding scheme could simplify problems of pattern recognition [5]. Though a range of sparse-coding feature extraction algorithms have been developed over time, based on e.g. independent component analysis [6][3][7] or anti-Hebbian competitive learning [8], the benefit that such methods might offer when combined with standard pattern recognition methods has not been systematically investigated.

In this paper we study the performance of different neural classifiers, when operating on sparse codes produced by one of several feature extraction methods, applied to handwritten digits from the MNIST image database [9]. We address the question whether sparse coding can improve the number of correctly classified

*Acknowledgement: This work was supported in part by Swedish Science Council (grant no. 621-2004-3807).

patterns, in particular when the degree of sparseness is high. Our intent is specifically to study the performance of scalable and neurobiologically inspired methods, rather than aiming for optimal classification performance.

2 A two-layer network model

To study the classification properties of sparse codes, we employed a two-layer network that consists of a local feature extraction layer connected to a neural classifier. We investigated combinations of one feature extraction algorithm and one classification method at a time, and recorded the classification rate on the test set for each combination.

Database. As input, we used the MNIST image database of handwritten digits, which consists of training and test sets containing 60000 and 10000 images, respectively. Each image has 28×28 pixels and 256 grey levels ranging from 0-255. To reduce computational demands, we here used a randomly selected subset of 20000 unique training samples, containing 2000 images per digit 0-9. Before being fed into the encoding layer the images were whitened, as described in the methods section. The reduction of the training set should not affect the relative performance between different neural classifiers, since we used the same reduced training set in all of the experiments.

Feature extraction. In short, local feature extraction aims to learn local filters in the image domain. The outcome is a set of *basis functions* that reacts to localized and oriented features (such as edges and lines). These basis functions often resemble the receptive fields in primary visual cortex [3]. Neural network models of this process commonly use a 'weight-sharing' scheme, in which a single set of basis functions is applied to the whole visual field. In our case, however, a *hypercolumnar* structure of the encoding layer was used, in which each hypercolumn learns a set of basis functions specifically for its partition of the image domain. This kind of network model is both scalable and biologically motivated, in the sense that it resembles the hypercolumnar structures in primary visual cortex [10]. The local codes produced by each hypercolumn were concatenated and fed into the classification layer.

We used a total of four different feature extraction algorithms, of which the first three were based on independent component analysis (ICA) and the fourth on competitive learning. In general, ICA-based feature extraction methods represent the input as factorial codes [2]. The objective is to learn a set of n basis functions, written as a matrix $\mathbf{\Psi} \in \mathbb{R}^{m \times n}$, such that an image vector of pixels $\mathbf{x} = \{x_1, x_2, \ldots, x_m\}$ can be represented by statistically independent components $\mathbf{a} = \{a_1, a_2, \ldots, a_n\}$. The image can then be reconstructed as $\mathbf{x} \approx \hat{\mathbf{x}} = \mathbf{\Psi a}$. The resulting sparse code $\mathbf{a}$ is characterized by a probability distribution that is peaked near zero.

Our first method, Infomax ICA [6], utilizes a non-linear version of Linsker's information maximization principle [11]. The algorithm learns the matrix $\mathbf{\Psi}$ by maximizing the joint entropy of $\mathbf{u} = \{f(a_1), f(a_2), \cdots, f(a_n)\}$ where $f(a)$ is a smooth non-linear function.

The second method that we used was Sparsenet [3]. The independent components are extracted by maximizing the likelihood that a set of images presented to the method were produced from the basis functions. This is accomplished by minimizing an energy function incorporating reconstruction quality and sparseness:

$$E = \frac{1}{2}\sum_{i}^{m}(x_i - \hat{x}_i)^2 + \theta\sum_{j}^{n} f(a_j).$$

The importance of sparseness is controlled by the parameter θ and the form of sparseness by a smooth non-linear function $f(a)$. Energy minimization is accomplished in two steps. For each image, the energy is minimized by adjusting $\mathbf{a}$ holding $\mathbf{\Psi}$ fixed, followed by adjusting $\mathbf{\Psi}$ over the whole set of images.

In the third method, the sparse-set coding network (SSC) [7], images are encoded using the same energy-minimization as in Sparsenet, but with the non-smooth $f(a)$ that counts the number of non-zero a_j:s. The method uses a binary gating vector $\mathbf{y}$ to select a subset from the available basis functions to code for a particular image. By iteratively flipping the entries in $\mathbf{y}$, a near-optimal code is produced.

The fourth algorithm that we used is based on competitive selective learning [12]. The method finds n clusters represented by $\boldsymbol{\psi} \in \mathbb{R}^m$. By the use of Euclidean norm as a metric $\delta_i = d(\mathbf{x}, \boldsymbol{\psi}_i)^\theta$, the feature space is divided into clusters with equal variance. The degree of sparseness is controlled by θ. A selection mechanism keeps the number of units fixed, by pruning and replacing dead units. The output $\mathbf{a} = \{a_1, a_2, \ldots, a_n\}$ is non-negative and equivalent to $\boldsymbol{\delta} = \{\delta_1, \delta_2, \ldots, \delta_n\}$. The total activity in each hypercolumn is normalized.

Classification. In our two-layer network, the classification layer was implemented by one of four different algorithms, namely gradient descent (GD), the linear Hebb rule [13], nearest centroid and the Bayesian Confidence Propagation Neural Network (BCPNN) [14]. The GD rule aims to minimize the mean squared error between the output response of the network and the actual target values of an input pattern. The output response is squashed by a sigmoid activation function $(1 + e^{-x})^{-1}$. The class labels between 0-9 were mapped into binary target vectors in $\mathbb{R}^{10}$, such that the component corresponding to a specific class was set to 1. The linear Hebbian network uses a weight matrix that is calculated by the outer product between the set of target vectors and the corresponding pattern vectors. The class labels of the patterns were mapped into bipolar target vectors in [-1,1]. In the nearest centroid classifier, the centroids are trained by adding each training pattern once and then calculating the average pattern corresponding to each class. A test pattern is assigned the same class as the closest centroid, using Euclidean distance. The BCPNN calculates the conditional probability that a pattern belongs to a certain class using a modified version of the naïve Bayes rule. The input activities can here be interpreted as the confidence that corresponding features are present in the visual field.

3 Methods

The images were prepared by whitening [3] to equalize the variances in all directions and to remove second-order statistics. This was followed by partitioning the image domain into 16 parts (7×7 pixels in size) corresponding to 16 hypercolumns. The image variances were normalized to $\sigma^2 = \{0.1, 0.1, 4.0\}$ before being used as input to Sparsenet, SSC and ICA, respectively.

We performed two series of experiments; in the first series the total number of hidden units, i.e. the dimensionality of the sparse representation, was gradually increased. The number of features were allowed to vary between the hypercolumns, such that fewer features were extracted in image partitions with many inactive pixels. This was accomplished by measuring the ℓ_2-norm of each pixel over the training set, assigning each hypercolumn one output unit for each pixel in the corresponding partition that had an ℓ_2-norm above a threshold value l_c. By lowering l_c, units were successively added until there was a complete set of 49 units in each hypercolumn. Since Infomax ICA requires equal size of the input and output dimensions, pixels with an ℓ_2-norm below l_c were discarded. The total number of hidden units were varied between $n = \{132, \ldots, 784\}$. The base sparseness parameters were set to $\theta_{SSC} = 0.1$, $\theta_{Sparsenet} = 0.022$ and $\theta_{CSL} = 8$.

In the second series of experiments, the sparseness parameters in Sparsenet and SSC were varied for a fixed set of 414 hidden units, distributed among the hypercolumns as described. We computed the sparseness level of each hidden unit a_i over the whole test set by the measure $\tau = \langle \tanh(|a_i - \mathrm{med}(a_i)|^2) \rangle$ [15]. Sample kurtosis was not used due to occurrences of inactive units in the SSC when θ_{SSC} was increased, for which kurtosis is undefined. The sparseness levels were varied by the parameter settings $\theta_{SSC} = \{0.01, \ldots, 0.80\}$ and $\theta_{Sparsenet} = \{0.002, \ldots, 0.372\}$. Increased sparseness in CSL has not yet been tested.

The GD classifier was (after normalization of the data to zero mean and unit variance) trained using the learning rate $\eta = 0.005$ and a batchsize of 100 images. To prevent overfitting, a validation set of 6000 patterns was randomly split off from the training set, using the remaining 14000 as training patterns. The BCPNN was implemented with the same hypercolumnar structure as in the encoding layer. To match the requirement of inputs being confidence values, the attributes of the encoded representations produced by the ICA-based methods were squared and linearly rescaled to the interval [0,1]. The sum of the total activity of each hypercolumn was then normalized.

4 Results

As a reference measure for our baseline parameter settings, we classified both the whitened and unwhitened data directly without the encoding step. Compared to these identity mappings, the performance among all but the centroid classifier could be improved by the use of our baseline versions of the feature extraction methods (fig. 1a). For instance, the error rate compared to the unwhitened data decreased 13.65 percentage units (p.u.) when the Hebbian classifier was

ESANN'2006 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 26-28 April 2006, d-side publi., ISBN 2-930307-06-4.

combined with the SSC and 4.14 p.u. in the combination of GD and CSL. To further analyze the improvement of the latter combination compared to using the unwhitened data, we computed the ROC-curves and confusion matrices [16]. The ROC-curve was constructed by varying the decision threshold of one output unit corresponding to digit 3. Compared to using the unwhitened data, the area under the ROC-curve increased 0.02 area units (a.u.) to 0.99 a.u. when the GD classifier was combined with CSL. The confusion matrices are shown in fig. 1b.

We observe that the improvement in classification depends mainly on the match between the encoding and classification layers. The performance can then be fine-tuned by varying the number of hidden units. To obtain optimal recognition performance, the encoded representations should have properties favorable to the classifier in the next layer. For example, since CSL is designed to produce non-negative and bi-modal activities, the performance of the BCPNN is much better using this method than when provided codes with symmetric activity produced by the ICA-based methods.

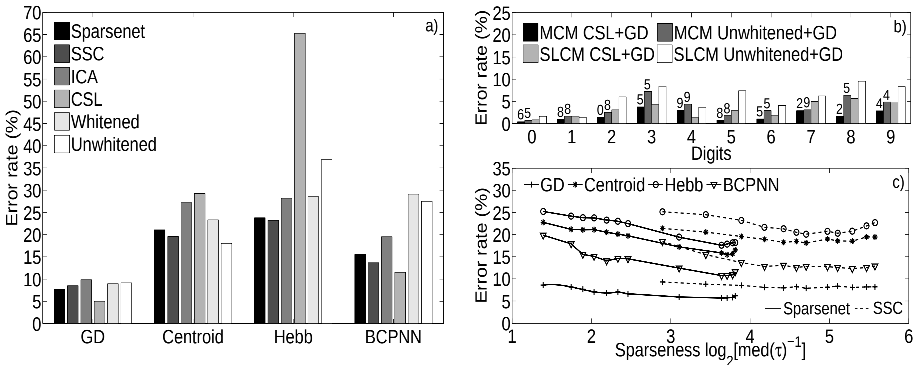

Fig. 1: a) the error rate on the test set using 414 hidden units and fixed sparseness parameter settings: b) the most common misclassification (MCM) indicated by a digit and the sum of less common misclassifications (SLCM): c) variation of the error rate with increased sparseness, using 414 hidden units. The plot shows various combinations of feature extraction methods (solid and dashed lines) and neural classifiers (markers).

Compared to the results in fig. 1a, we find that the performance is improved with increased sparseness up to a certain point, after which the error rates again increase (fig. 1c). For instance, the error rate of the nearest centroid classifier combined with Sparsenet decreased 5.63 p.u. compared to the baseline parameter settings. We observe that the improvement is 2.59 p.u. compared to the unwhitened data in fig. 1a. Thus, the classification performance can be fine-tuned also by an optimal setting of the sparseness parameter, such that all classification methods used can benefit from sparse coding.

5 Discussion

We have shown that sparse coding methods can be used as preprocessing to improve the classification performance of various simple neural classifiers. Our results indicate an additional reason that sparse representations may be preferred in the sensory cortex, in addition to other reasons such as reduced metabolic energy cost [7]. We show here that increased sparseness in fact improves recognition (while further lowering the metabolic energy cost), but only to a certain point. The reason is that recognition performance depends on a trade-off between effectiveness of the codes and preservation of relevant information.

We have observed that the performance of our two-layer network is satisfactory, provided that the feature extraction method matches the specific input requirements of the neural classifier. Further, we have observed that the performance can be fine-tuned by varying sparseness and the number of hidden units. Since we have used simple neurobiologically inspired models, using only local information processing, our two-layer network model could easily be adapted to large-scale applications, while scaling favorably in terms of computational demands.

References

[1] H. B. Barlow. Single units and sensation: A neuron doctrine for perceptual psychology? *Perception*, 1:371–394, 1972.

[2] H. B. Barlow. Unsupervised learning. *Neural Computation*, 1:295–311, 1989.

[3] B. A. Olshausen and D. J. Field. Sparse Coding with an Overcomplete Basis Set: A Strategy Employed by V1? *Vision Research*, 37:3311–3325, 1997.

[4] D. J. Willshaw, O. P. Buneman, and H. C. Longuet-Higgins. Non-holographic associative memory. *Nature*, 222:960–962, 1969.

[5] D. J. Field. What Is the Goal of Sensory Coding? *Neural Comp.*, 6(4):559–601, 1994.

[6] A. J. Bell and T. J. Sejnowski. Edges are the 'Independent Components' of Natural Scenes. *Adv. in Neural Information Processing Systems 9*, pages 831–837, 1996.

[7] M. Rehn and F. T. Sommer. A network that uses few active neurones to code visual input predicts the diverse shapes of cortical receptive fields. *Submitted*, 2005.

[8] P. Földiák. Forming sparse representations by local anti-Hebbian learning. *Biological Cybernetics*, 64:165–170, 1990.

[9] Y. Lecun. The MNIST database of handwritten digits. Available at `http://yann.lecun.com/exdb/mnist/`. 2004-11-04.

[10] D. H. Hubel and T. N. Wiesel. Functional Architecture of Macaque Monkey Visual Cortex. *Proc. R. Soc. Lond. B.*, 198(1130):1–59, May 1977.

[11] R. Linsker. An Application of the Principle of Maximum Information Preservation to Linear Systems. *Adv. in Neural Information Processing Systems 1*, pages 186–194, 1989.

[12] N. Ueda and R. Nakano. A New Competitive Learning Approach Based on an Equidistortion Principle for Designing Optimal Vector Quantizers. *Neural Networks*, 7(8):1211–1227, 1994.

[13] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice-Hall, 1999.

[14] A. Sandberg, A. Lansner, K. M. Petersson, and Ö. Ekeberg. A Bayesian attractor network with incremental learning. *Network: CNS*, 13:179–194, 2002.

[15] J. Karvanen and A. Cichocki. Measuring Sparseness of Noisy Signals. *4th Int. Symp. on Independent Component Analysis and Blind Signal Separation (ICA2003)*, 2003.

[16] A. P. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, 1997.