Unsupervised coding with Lococode

In Proc. ICANN'97, p. 655-660

Sepp Hochreiter and Jürgen Schmidhuber

Technische Universität München, 80290 München, Germany
and IDSIA, Corso Elvezia 36, CH-6900-Lugano, Switzerland

Abstract. Traditional approaches to sensory coding use code component-oriented objective functions (COCOFs) to evaluate code quality. Previous COCOFs do not take into account the information-theoretic complexity of the code-generating mapping itself. We do: "*Lo*w-*co*mplexity *co*ding and *de*coding" (LOCOCODE) generates so-called *lococodes* that (1) convey information about the input data, (2) can be computed from the data by a low-complexity mapping (LCM), and (3) can be decoded by a LCM. We implement LOCOCODE by training autoassociators with *Flat Minimum Search* (FMS), a general method for finding low-complexity neural nets. LOCOCODE extracts optimal codes for difficult versions of the "bars" benchmark problem. As a preprocessor for a vowel recognition benchmark problem it sets the stage for excellent classification performance.

1 Introduction

Several COCOFs have been proposed to evaluate the quality of sensory codes. Many COCOFs explicitly favor factorial codes [1] of input data. Other approaches favor local codes, e.g., [8]. Recently there also has been much work on COCOFs for biologically plausible sparse distributed codes, which share some advantages of both minimally redundant and local codes, e.g., [4,3,6].

But what about coding costs? COCOFs emphasize desirable properties of the code itself, while neglecting the costs of constructing the code from the data. For instance, coding input data in a redundancy-free fashion may be very expensive in terms of information bits required to describe many finely tuned free parameters in the code-generating network. In this paper we will shift the focus onto the information-theoretic costs of code-generation. See abstract. We will see that LOCOCODE encourages noise-tolerant "feature detectors" reminiscent of those observed in the mammalian visual cortex, because they are easily codable and decodable.

2 Flat minimum search: review

To implement LOCOCODE we apply Flat Minimum Search (FMS) [7] to a 3-layer autoassociator (AA) whose hidden unit (HU) activations represent the code.

FMS is a general, gradient-based method for finding low-complexity networks that can be described with few bits of information.

FMS Overview. FMS finds a large region in weight space such that each weight vector from that region has *similar* small error. Such regions are called "flat minima". A flat minimum corresponds to weights many of which can be given with low precision. In contrast, a "sharp" minimum corresponds to weights which have to be specified with high precision. In the terminology of the theory of minimum description length (MDL), fewer bits of information are required to pick a "flat" minimum (corresponding to a low complexity-network). As a natural by-product of net complexity reduction, FMS automatically prunes units, weights, and input lines, reduces output sensitivity with respect to remaining weights and units, and generalizes well. In a previous application to stock market prediction [7], FMS led to better results than "weight decay" and "optimal brain surgeon".

Architecture. We use a 3 layer feedforward net. Each layer is fully connected to the next layer. Let O, H, I denote index sets for output, hidden, input units, respectively. For $l \in O \cup H$, the activation y^l of unit l is $y^l = f(s_l)$, where $s_l = \sum_m w_{lm} y^m$ is the net input of unit l ($m \in H$ for $l \in O$ and $m \in I$ for $l \in H$), w_{lm} denotes the weight on the connection from unit m to unit l, f denotes the activation function, and for $m \in I$, y^m denotes the m-th component of an input vector. $W = |(O \times H) \cup (H \times I)|$ is the number of weights.

Algorithm. FMS' objective function E features an unconventional term:

$$B = \sum_{i,j \in O \times H \cup H \times I} \log \sum_{k \in O} (\frac{\partial y^k}{\partial w_{ij}})^2 + W \log \sum_{k \in O} \left(\sum_{i,j \in O \times H \cup H \times I} \frac{|\frac{\partial y^k}{\partial w_{ij}}|}{\sqrt{\sum_{k \in O} (\frac{\partial y^k}{\partial w_{ij}})^2}} \right)^2 .$$

$E = E_q + \lambda B$ is minimized by gradient descent, where E_q is the training set mean squared error (MSE), and $\lambda > 0$ scales B's influence. B measures the weight precision (number of bits needed to describe all weights in the net). Reducing B without increasing E_q means removing weight precision without increasing quadratic error. All of this can be done efficiently, namely, with standard backprop's order of computational complexity. For more details see [7].

3 Experiment 1: independent bars

The task is adapted from [2]. The input is a 5×5 pixel grid with horizontal and vertical 1×5 and 5×1 bars at random, independent positions. The goal is to extract the independent features corresponding to the bars. According to [2], even a simpler variant (no mixing of vertical and horizontal bars) is not trivial: *"Although it might seem like a toy problem, the* 5×5 *bar task with only 10 HUs turns out to be quite hard for all the algorithms we discuss. The coding cost of making an error in one bar goes up linearly with the size of the grid, so at least one aspect of the problem gets* easier *with large grids." [2].* We will see that even the difficult version of this task is not hard for LOCOCODE.

Training and testing. For each of the 25 pixels there is an input unit. Input units that see a pixel of a bar take on activation 0.5; others -0.5. Each of

the 10 possible bars appears with probability $\frac{1}{5}$. In contrast to [2] vertical and horizontal bars may be mixed in the same input. This makes the task harder (see [2], p. 570). To test LOCOCODE's ability to reduce redundancy, we use many more HUs (namely 25) than the required minimum of 10. [2] (p. 570) reports that an AA trained without FMS (and more than 10 HUs) "consistently failed". We have confirmed this result.

Following [2], the net is trained on 500 randomly generated patterns (there may be pattern repetitions). Learning is stopped after 5,000 epochs. Then the net is tested on 500 additional random patterns. We say that a pattern is processed correctly if the absolute error of all output units is below 0.3. The sigmoid HUs are active in [0,1], the sigmoid output units are active in [-1,1]. Noninput units have an additional bias input. The target is -0.7 for -0.5 and 0.7 for 0.5. Normal weights are initialized in $[-0.1, 0.1]$, bias weights with -1.0, λ with 0.5. *Parameters:* learning rate: 1.0, $E_{tol} = 0.16$, $\Delta\lambda = 0.001$. *Architecture:* (25-25-25).

Results: factorial codes based on feature detectors. Training MSE is 0.11; test MSE is 0.15 (averages over 10 trials). The net generalizes well: only one of the 500 test patterns is not processed correctly. 15 of the 25 HUs are indeed automatically pruned. Figures 1 and 2 depict typical weights to and from HUs. For each of the 25 HUs there is a 5×5 square depicting the 25 post-training weights on connections from 25 inputs. The corresponding bias weight sits on top of the upper left corner. White (black) circles on gray (white) background are positive (negative) weights. The circle radius is proportional to the weight's absolute value. All but 10 units are effectively pruned away. The surviving HUs become binary bar detectors. They exactly mirror the statistics of the pattern generation process: LOCOCODE finds an optimal factorial code by producing optimal feature detectors.

Backprop fails. For comparison we run this task with conventional backpropagation with 25 (BP25), 15 (BP15) and 10 (BP10) HUs. BP25: test MSE 0.20; BP15: test MSE 0.22; BP10: test MSE 0.27. Backprop does not prune any units; the resulting weight patterns are highly unstructured, and the underlying input statistics are not discovered.

Noisy bars. Similar lococodes were obtained even when we randomly varied bar intensities and added Gaussian noise to the input.

Conclusion. Unlike standard backprop, LOCOCODE easily solves hard variants of the standard "bars" problem. It discovers the underlying statistics and extracts the essential, statistically independent features, even in case of noisy inputs.

4 Experiment 2: vowel recognition

Next we will show that lococodes can help to achieve superior generalization performance on a supervised learning benchmark problem.

Task. We recognize vowels, using data [9] from Scott Fahlman's CMU benchmark collection. There are 11 vowels and 15 speakers. Each speaker spoke each vowel 6 times. Data from the first 8 speakers is used for training, other data for testing. This means 528 frames for training and 462 frames for testing. Each

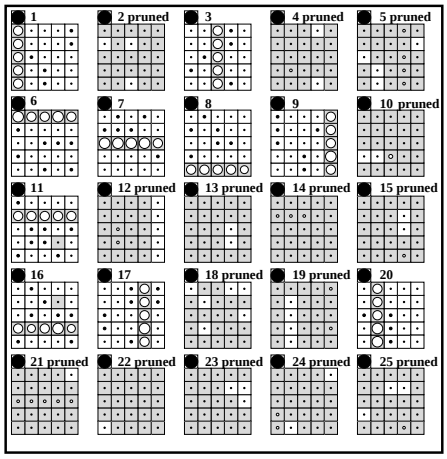

Fig. 1. *Independent bars: incoming weights to hidden units.*

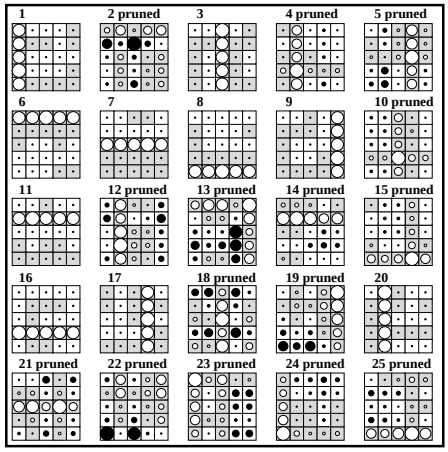

Fig. 2. *Independent bars: weights from hidden to output units.*

frame consists of 10 input components obtained by low pass filtering at 4.7kHz, digitized to 12 bits with a 10 kHz sampling rate. A twelfth order linear predictive analysis was carried out on six 512 sample Hamming-windowed segments from the steady part of the vowel. The reflection coefficients were used to calculate 10 log area parameters, providing the 10 dimensional input space.

Coding. The training data is coded using FMS for autoassociation. Architecture: (10-30-10). The sigmoid HUs are active in [0,1], the sigmoid output units are active in [-1,1]. Noninput units have an additional bias input. The input components are linearly scaled in [-1,1]. The AA is trained with 10^7 pattern presentations. Then its weights are frozen.

Classification. From now on, the vowel codes across all nonconstant HUs are used as inputs for a conventional supervised backprop classifier, which is trained to recognize the vowels from the code. The classifier's architecture is $((30-c)$-11-11), where c is the number of nonvarying (pruned) HUs in the AA. The hidden and output units are sigmoid and active in [-1,1], and receive an additional bias input. The classifier is trained with another 10^7 pattern presentations.

Parameters. AA net: learning rate: 0.02, $E_{tol} = 0.015$, $\Delta\lambda = 0.2$, $\gamma = 2.0$. Backprop net: learning rate: 0.002.

Overfitting. We confirm Robinson's results: the classifier tends to overfit when trained by simple backprop — during learning, the test error rate first decreases and then increases again.

Comparison. See Table 1. FMS generates 3 different lococodes. Each is fed into 10 conventional, overfitting backprop classifiers with different weight initializations: the table entry for "LOCOCODE/Backprop" represents the mean of 30 trials. The results for neural nets and nearest neighbor are taken from Robinson [9]. The other results (except for LOCOCODE's) are taken from Hastie

	Technique		error rates	
		nr. hidden units	training	test
(1.1)	Single-layer perceptron	–	–	0.67
(1.2.1)	Multi-layer perceptron	88	–	0.49
(1.2.2)	Multi-layer perceptron	22	–	0.55
(1.2.3)	Multi-layer perceptron	11	–	0.56
(1.3.1)	Modified Kanerva Model	528	–	0.50
(1.3.2)	Modified Kanerva Model	88	–	0.57
(1.4.1)	Radial Basis Function	528	–	0.47
(1.4.2)	Radial Basis Function	88	–	0.52
(1.5.1)	Gaussian node network	528	–	0.45
(1.5.2)	Gaussian node network	88	–	0.47
(1.5.3)	Gaussian node network	22	–	0.46
(1.5.4)	Gaussian node network	11	–	0.53
(1.6.1)	Square node network	88	–	0.45
(1.6.2)	Square node network	22	–	0.49
(1.6.3)	Square node network	11	–	0.50
(2)	Nearest neighbor	–	–	0.44
(3)	LDA	–	0.32	0.56
(4)	Softmax	–	0.48	0.67
(5)	QDA	–	0.01	0.53
(6.1)	CART	–	0.05	0.56
(6.2)	CART (linear comb. splits)	–	0.05	0.54
(7)	FDA / BRUTO	–	0.06	0.44
(8)	Softmax / BRUTO	–	0.11	0.50
(9.1)	FDA / MARS (degree 1)	–	0.09	0.45
(9.2)	FDA / MARS (degree 2)	–	0.02	0.42
(10.1)	Softmax / MARS (degree 1)	–	0.14	0.48
(10.2)	Softmax / MARS (degree 2)	–	0.10	0.50
(11)	LOCOCODE / Backprop	30/11	0.05	0.42

Table 1. *Vowel recognition task: generalization performance of different methods. Surprisingly, FMS-generated lococodes fed into a conventional, overfitting backprop classifier led to best results. See text for details.*

et al. [5]. *Our method led to best generalization results.* The error rates after backprop learning vary between 39 and 45 %.

Backprop fed with LOCOCODE code sometimes goes down to 38 % error rate, but due to overfitting error grows again. Given that backprop by itself is a very naive approach, the fact that its generalization performance can be dramatically enhanced by feeding it *nongoal-specific* lococodes appears quite surprising.

Hastie et al. also obtained additional, even slightly better results with an FDA/MARS variant: down to 39 % average error rate. It should be mentioned, however, that their data was subject to goal-directed pre-processing with splines, such that there were many clearly defined classes. Furthermore, to determine the input dimension, Hastie et al. used a special kind of generalized cross-validation error, where one constant was obtained by unspecified "simulation studies".

Typical feature detectors. The number of (pruned) HUs with constant activation varies between 5 and 10. 2 to 5 HUs become binary, and 4 to 7 trinary.

With all codes we observed: apparently, certain HUs become feature detectors for speaker identification. Another HU's activation is near 1.0 for the words "heed" and "hid" ("i" sounds). Another HU's activation has high values for the words "hod", "hoard", "hood" and "who'd" ("o"-words) and low but nonzero values for "hard" and "heard". LOCOCODE supports feature detection.

5 Conclusion

Unlike previous approaches, LOCOCODE does not define code optimality solely by properties of the code itself. Instead, LOCOCODE's notion of code optimality takes into account the information-theoretic complexity of the mappings used for coding and decoding. Lococodes typically compromise between conflicting goals. They tend to exhibit *low but not minimal* redundancy — if the complexity costs of generating minimal redundancy are too high.

LOCOCODE easily solves coding tasks that have been described as hard by other authors. Our experiments also demonstrate the usefulness of LOCOCODE-based data pre-processing for subsequent classification. Although we made no attempt to prevent classifier overfitting, we achieved excellent results. From this we conclude that the lococodes fed into the classifier already conveyed the "essential", almost noise-free information necessary for excellent classification. We are led to believe that LOCOCODE is a promising and general method for data pre-processing.

6 Acknowledgments

This work was supported by *DFG grant SCHM 942/3-1* from "Deutsche Forschungsgemeinschaft". Results for the bars problem stem from M. Baumgartner's diploma thesis (TUM 1996).

References

1. H. B. Barlow, T. P. Kaushal, and G. J. Mitchison. Finding minimum entropy codes. *Neural Computation*, 1(3):412–423, 1989.
2. P. Dayan and R. Zemel. Competition and multiple cause models. *Neural Computation*, 7:565–579, 1995.
3. B. A. Olshausen; D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, 1996.
4. D. J. Field. What is the goal of sensory coding? *Neural Computation*, 6:559–601, 1994.
5. T. J. Hastie, R. J. Tibshirani, and A. Buja. Flexible discriminant analysis by optimal scoring. Technical report, AT&T Bell Laboratories, 1993.
6. G. E. Hinton and Z. Ghahramani. Generative models for discovering sparse distributed representations. Technical report, University of Toronto, Department of Computer Science, Toronto, Ontario, M5S 1A4, Canada, 1997. A modified version to appear in *Philosophical Transactions of the Royal Society* **B**.
7. S. Hochreiter and J. Schmidhuber. Flat minima. *Neural Computation*, 9(1):1–42, 1997.
8. T. Kohonen. *Self-Organization and Associative Memory*. Springer, second ed., 1988.
9. A. J. Robinson. *Dynamic Error Propagation Networks*. PhD thesis, Trinity Hall and Cambridge University Engineering Department, 1989.