Low-Complexity Coding and Decoding

Sepp Hochreiter and Jürgen Schmidhuber

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

Abstract. We present a novel approach to sensory coding and unsupervised learning. It is called "*Lo*w-*co*mplexity *co*ding and *de*coding" (LOCOCODE). Unlike previous methods it explicitly takes into account the information-theoretic complexity of the code generator: *lococodes* (1) convey information about the input data and (2) can be computed and decoded by low-complexity mappings. To implement LOCOCODE we train autoassociators with *Flat Minimum Search*, a recent method for discovering neural nets that can be described with few bits of information. Experiments show: unlike codes obtained with standard autoencoders, lococodes are based on familiar feature detectors, never unstructured, usually sparse, sometimes factorial or local (depending on the data). Unlike, e.g., independent component analysis (ICA) LOCOCODE does not need to know the number of independent data sources.

1 INTRODUCTION

What is the goal of sensory coding [8]? There is no generally agreed-upon answer yet. Several information-theoretic objective functions (OFs) have been proposed to evaluate the quality of sensory codes. Most OFs focus on statistical properties of the code components (such as mutual information) — we refer to them as code component-oriented OFs, or COCOFs. Some COCOFs explicitly favor near-factorial, minimally redundant codes of the input data (see, e.g., [3,18,26,30,33]). Such codes can be advantageous for (1) data compression, (2) speeding up subsequent gradient descent learning (e.g., [4]), (3) simplifying subsequent Bayes classifiers (e.g., [29]). Other approaches favor local codes, e.g., [24,15]. They can help to achieve (1) minimal crosstalk, (2) subsequent gradient descent speed-ups, (3) facilitation of post training analysis, (4) simultaneous representation of different data items. Recently there also has been much work on COCOFs encouraging biologically plausible sparse distributed codes, e.g., [9,21,34,8,25,17,6,20,10]. Sparse codes share certain advantages of both local and dense codes.

Coding costs. COCOFs express desirable properties of the code itself, while neglecting the costs of constructing the code from the data. A previous argument for ignoring coding costs [11,34] based on the principle of minimum description length (MDL, e.g., [31,32,22]) focuses on hypothetical costs of transmitting the data from some sender to a receiver — how many bits are necessary to enable the receiver to reconstruct the data? It goes more or less like this: *"Total transmission cost is the number of bits required to describe (1) the data's code, (2)*

the reconstruction error and (3) the decoding procedure. Since all input exemplars are encoded/decoded by the same mapping, the coding/decoding costs are negligible because they occur only once."

We are not quite convinced by this argument. We doubt that sensory coding's sole objective should be to transform data into a compact code that is cheaply transmittable across some ideal, abstract channel. In fact, *generating* such a code may be very expensive in terms of information bits required to describe the code-generating network, which may need many finely tuned free parameters. Hence we believe that one of sensory coding's objectives should be to reduce the cost of code generation through data transformations in *existing* channels (e.g., synapses etc.). Without denying the usefulness of certain COCOFs, we postulate that an important scarce resource is the bits required to describe the mappings that generate and process the codes — after all, it is these mappings that need to be implemented, given some limited hardware.

Lococodes. For such reasons we shift the point of view and focus on the information-theoretic costs of code-generation. We will present a novel approach to unsupervised learning called "*lo*w-*co*mplexity *co*ding and *de*coding" (LOCOCODE). Without assuming particular goals such as data compression, simplifying subsequent classification, etc., but in the MDL spirit, LOCOCODE generates so-called *lococodes* that (1) convey information about the input data and (2) can be computed and decoded by low-complexity mappings.

Typical lococodes are sparse. We will see that LOCOCODE encourages noise-tolerant feature detectors reminiscent of those observed in the mammalian visual cortex, because such detectors typically reduce coding and decoding costs: to code a given input all we need to know is which features are present and which are not. Inputs that are mixtures of a few regular features, such as edges in images, can be described well in a sparse fashion (only code components corresponding to present features are non-zero).

Lococodes through FMS. To implement LOCOCODE we apply *Flat Minimum Search* (FMS [12]) to an autoassociator (AA) whose hidden layer activations represent the code. FMS is a general, gradient-based method for finding networks that can be described with few bits of information.

Previous AAs. Backprop-trained AAs *without* a narrow hidden bottleneck typically produce redundant, continuous-valued codes and unstructured weight patterns. Linear AAs *with* a hidden layer bottleneck, however, produce codes that are orthogonal projections onto the subspace spanned by the first principal eigenvectors of a covariance matrix associated with the training patterns [2]. The mean squared error (MSE) surface has an unique minimum [2]. Nonlinear codes have been obtained by nonlinear bottleneck AAs with more than 3 (e.g., 5) layers, e.g., [16,7]. None of these methods produces sparse, factorial or local codes — instead they produce first principal components or their nonlinear equivalents ("principal manifolds"). We will see that FMS-based AAs yield quite different results.

2 FLAT MINIMUM SEARCH: REVIEW

FMS Overview. FMS is a general method for finding low complexity-networks with high generalization capability. 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". In MDL terminology, few bits of information are required to pick a weight vector in a "flat" minimum (corresponding to a low complexity-network) — the weights may be given with low precision. In contrast, weights in a "sharp" minimum require a high-precision specification. As a natural by-product of net complexity reduction, FMS automatically prunes weights (by setting them to zero) and units (e.g., by giving them a strong bias), and reduces output sensitivity with respect to remaining weights and units. Previous FMS applications focused on supervised learning [12]: FMS led to better stock market prediction results than various alternative methods. In this paper, however, we will use it for unsupervised coding only.

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 includes an unconventional term:

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

$E_q + \lambda B$ is minimized by gradient descent, where E_q is the training set mean squared error (MSE), and λ is a positive "regularizer" scaling B's influence. Defining λ corresponds to choosing a tolerable error level (there is no *a priori* "optimal" way of doing so). 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 MSE. Given a constant number of output units, all of this can be done efficiently, namely, with standard BP's order of computational complexity. For details see [12]. For even more general attempts at reducing net complexity see [27].

3 EXPERIMENTS

In all our experiments we will associate input data with itself, using an FMS-trained 3-layer autoassociator (AA) with semilinear activation function in the hidden layer. Unless stated otherwise we use 700,000 training exemplars.

3.1 EXPERIMENT 1: local, sparse, factorial codes

Task. The data consists of 8 input vectors with 8 components each. The i-th component of the i-th vector is 0.8. The other components are 0.2. The sigmoid hidden units (HUs) are active in [0,1]. Code units and output units have an additional bias input. Code units are initialized with a negative bias of -2.0.

Experiment 1.1: uniformly distributed inputs, 500,000 training examples. Each input vector has probability $\frac{1}{8}$ of being chosen next. *Parameters:* learning rate: 0.1, the "tolerable error" $E_{tol} = 0.1$, $\Delta\lambda = 1.0$, $\gamma = 2.0$ ($\Delta\lambda$ is a parameter for the λ update, and γ is a parameter for 2-phase learning — see [12]). *Architecture:* (8-5-8) (8 input units, 5 HUs, 8 output units).

Results: factorial codes. In 7 out of 10 trials, FMS effectively pruned 2 HUs (huge negative bias $\rightarrow$ zero unit activation $\rightarrow$ incoming weights pruned due to extremely low required precision), and produced a *factorial binary code* with statistically independent code components. In 2 trials FMS pruned 2 HUs and produced an almost binary code — with one trinary unit taking on values of 0.0, 0.5, 1.0. In one trial FMS produced a binary code with only one HU being pruned away. Obviously, under certain constraints on the input data, FMS tends towards the compact, nonredundant codes advocated by numerous researchers.

Experiment 1.2: like Experiment 1.1, but with architecture (8-8-8), 200,000 training examples and input values in $\{0.0, 1.0\}$ (as opposed to input values in $\{0.2, 0.8\}$). We use more HUs than in Experiment 1.1, to make clear that in this case fewer units are pruned.

Results: local codes. 10 trials were conducted. FMS always produced a binary code. In 7 trials, only 1 HU was pruned, in the remaining trials 2 HUs. Unlike with standard BP, *almost all inputs almost always were coded in an entirely local manner*, i.e., only one HU was switched on, the others switched off. Recall that local codes were also advocated by many researchers – but they are precisely "the opposite" of the factorial codes from the previous experiment. How can LOCOCODE justify such different codes? How to explain this apparent discrepancy? The reason is: with the different input representation, the additional HUs do not necessarily result in much more additional complexity of the mappings for coding and decoding. The zero-valued inputs allow for low weight precision (low coding complexity) for connections leading to HUs (similarly for connections leading to output units). In contrast to Experiment 1.1 it is possible to describe the i-th possible input by the following feature: "the i-th input component does not equal zero".

Experiment 1.3: like Experiment 1.2, but architecture (1-8-1) and *one-dimensional* inputs: 0.05, 0.1, 0.15, 0.2, 0.8, 0.85, 0.9, 0.95. *Parameters:* learning rate: 0.1, $E_{tol} = 0.00004$, $\Delta\lambda = 1.0$, $\gamma = 2.0$.

Results: feature detectors. 10 trials were conducted. FMS always produced the following code: one binary HU making a distinction between input values less than 0.5 and input values greater than 0.5, 2 HUs with continuous values, one of which is zero (or one) whenever the binary unit is on, while the other is zero (one) otherwise. All remaining HUs adopt constant values of either

1.0 or 0.0, thus being essentially pruned away. The binary unit serves as a binary *feature detector*, grouping the inputs into 2 classes.

Experiment 1.4: nonuniformly distributed inputs. Input vectors like in Experiment 1.1, but occurring with probabilities $\frac{1}{4}, \frac{1}{4}, \frac{1}{8}, \frac{1}{8}, \frac{1}{16}, \frac{1}{16}, \frac{1}{16}, \frac{1}{16}$. *Parameters:* learning rate: 0.005, $E_{tol} = 0.01$, $\Delta\lambda = 1.0$, $\gamma = 2.0$. *Architecture:* (8-5-8).

Results: sparse codes. In 4 out of 10 trials, FMS found a binary code (no HUs pruned). In 3 trials: a binary code with one HU pruned. In one trial: a code with one HU removed, and a trinary unit adopting values of 0.0, 0.5, 1.0. In 2 trials: a code with one pruned HU and 2 trinary HUs. Obviously, with this set-up, FMS prefers codes known as *sparse distributed representations*.

Explanation. Why is the result different from Experiment 1.1's? To achieve equal error contributions for all inputs, the weights for coding/decoding highly probable inputs have to be given with higher precision than the weights for coding/decoding inputs with low probability: the input distribution from Experiment 1.1 will result in a more complex network. The next experiment will make this effect even more pronounced.

Experiment 1.5: like Experiment 1.4, but with architecture (8-8-8).

Results: sparse codes. In 10 trials, FMS always produced binary codes. In 2 trials only 1 HU was pruned. In 1 trial 3 units were pruned. In 7 trials 2 units were pruned. Unlike with standard BP, *almost all inputs almost always were coded in a sparse, distributed manner:* typically, 2 HUs were switched on, the others switched off, and most HUs responded to exactly 2 different input patterns. The mean probability of a unit being switched on was 0.28, and the probabilities of different HUs being switched on tended to be equal.

3.2 EXPERIMENT 2: Independent Bars (Overview)

In separate publications [13,14] we successfully use LOCOCODE to solve difficult variants of the "bars" benchmark problem [6,25,10]. LOCOCODE discovers the underlying statistics and extracts the essential, statistically independent features, even in presence of noise. Standard BP AAs accomplish none of these feats (Dayan and Zemel, 1995) — this has been confirmed by additional experiments conducted by ourselves. LOCOCODE achieves success solely by reducing information-theoretic (de)coding costs. Unlike previous approaches, it does not depend on explicit terms enforcing independence [26], zero mutual information among code components [18], or sparseness [34,8,20,10]. Like "independent component analysis" (ICA, e.g., [1,5]), LOCOCODE untangles mixtures of independent data sources. Unlike ICA, however, it does not need to know in advance the number of such sources — like "predictability minimization" [26] it simply prunes away superfluous code components.

3.3 EXPERIMENT 3: More Realistic Visual Data

Task 3.1 (overview). In a separate publication [13] we successfully use LOCOCODE to code image data (based on the aerial shot of a village). LOCOCODE

generates sparse codes based on well-known on-center-off-surround feature detectors. Superfluous HUs are pruned, few survive. Standard backprop, however, does not produce any recognizable weight structures or simple feature detectors, and does not prune any of the AA's HUs.

Task 3.2. Figure 1 shows a wood cell image with 150×150 pixels, each taking on one of 256 gray levels. 7×7 pixels subsections are randomly chosen as training inputs, where gray levels are scaled to input activations in $[-0.5, 0.5]$ and to targets in $[-0.7, 0.7]$. The sigmoid HUs (output units) are active in [0,1] ([-1,1]). Normal weights are initialized in $[-0.1, 0.1]$, bias weights with -1.0, λ with 0.5. Training stop: after 250,000 training examples. *Parameters:* learning rate: 1.0, $E_{tol} = 1.0$, $\Delta\lambda = 0.01$. *Architecture:* (49-25-49).

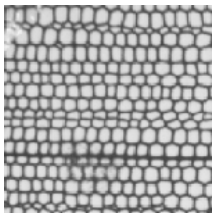

Fig. 1. *Task 3.2 — wood cells: image section used for training.*

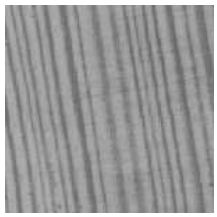

Fig. 2. *Task 3.3 — striped wood: image section used for training.*

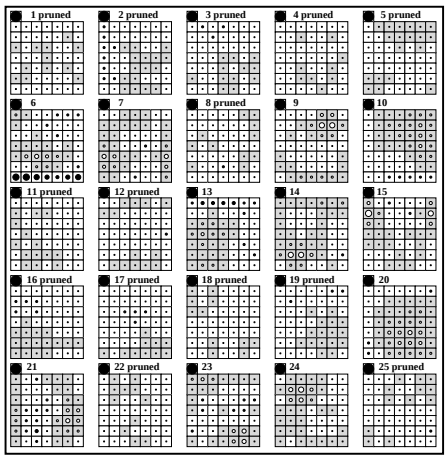

Fig. 3. *Task 3.2 (cells): weights from input to hidden units after 250,000 examples. 11 units survive.*

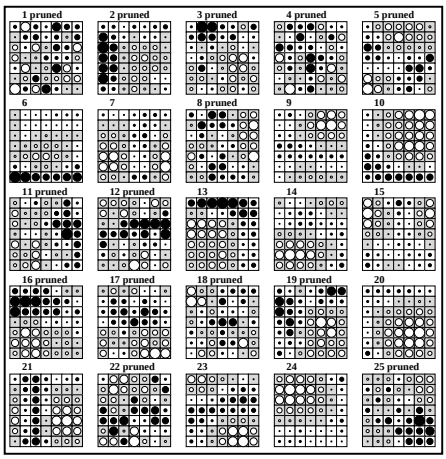

Fig. 4. *Task 3.2: weights from hidden to output units after 250,000 examples.*

Results. 4 trials led to similar results. Bias weights to HUs are negative. Figures 3 and 4 depict typical weights on connections to and from HUs after 250,000 training examples. For each of the 25 HUs, there is a 7×7 square depicting 49 typical post-training weights on connections from 49 inputs (and to 49 outputs). White (black) circles on gray (white) background are positive (neg-

ative) weights. The circle radius is proportional to the weight's absolute value. Bias weights of HUs are shown on top of the squares' upper left corners. The circle representing some HU's maximal absolute weight has maximal possible radius (circles representing other weights are scaled accordingly). Output bias weights are all negative. To activate some HU, its input must match the structure of the incoming weights to cancel the inhibitory bias. 9 to 11 units survive. They are obvious feature detectors and can be characterized by the positions of the centers of their on-center-off-surround structures relative to the input field: the entire input is covered by them.

Task 3.3. Like Task 3.2 — but now we use the image of a *striped* piece of wood. See Figure 2. $E_{tol} = 0.1$. Training stop: after 300,000 training examples.

Results. 4 trials led to similar results. Only 3 to 5 of the 25 HUS survive and become obvious feature detectors, now of a different kind: they detect whether their receptive field covers a dark stripe to the left, to the right, or in the middle.

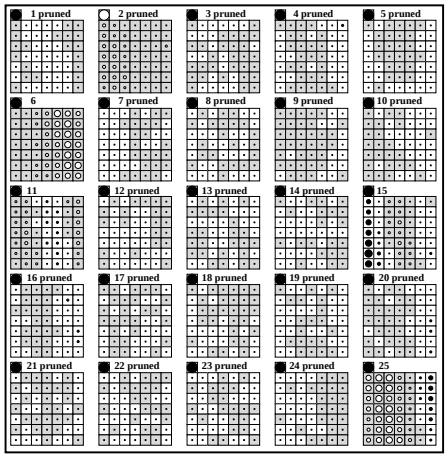

Fig. 5. *Task 3.3 (stripes): weights from input to hidden units after 300,000 examples. 4 units survive.*

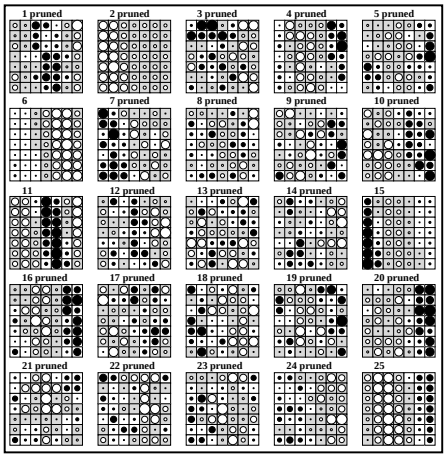

Fig. 6. *Task 3.3: weights from hidden to output units after 300,000 examples.*

Figures 5 and 6 depict typical weights on connections to and from HUs.

Conclusion. Unlike standard BP-trained AAs, FMS-trained AAs generate highly structured sensory codes. FMS automatically prunes superfluous units. Taking into account statistical properties of the visual input data, it generates appropriate feature detectors such as the familiar on-center-off-surround and bar detectors. It also produces biologically plausible sparse codes. FMS's objective function, however, does *not* contain explicit terms enforcing such codes (this contrasts previous methods, e.g., [20]).

3.4 EXPERIMENT 4: vowel recognition

In separate publications [13,14] we successfully use LOCOCODE as a preprocessor for a standard vowel recognition benchmark problem [23]. The training data is

first coded using an FMS AA. Then the AA's weights are frozen. From now on, the vowel codes across all nonconstant HUs are used as inputs for a conventional supervised BP classifier, which is trained to recognize the vowels from the code.

In our comparison, this simple approach outperforms (1) various neural nets, (2) Nearest neighbor, (3) linear discriminant analysis, (4) Softmax, (5) quadratic discriminant analysis, (6) CART, (7) flexible discriminant analysis (FDA) using additive models with adaptive selection of terms and splines smoothing parameters, (8) Softmax with a set of basis functions for better class separation, (9) FDA using multivariate adaptive regression splines (MARS), (10) Softmax/MARS. Although we made no attempt at preventing 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 method for data preprocessing.

4 CONCLUSION

LOCOCODE's notion of 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 be sparse and exhibit *low but not minimal* redundancy — if the complexity costs of generating minimal redundancy are too high. Lococodes tend towards binary, informative feature detectors, but occasionally there are trinary or continuous-valued code components (where complexity considerations suggest such alternatives).

A general principle? Our results indicate that local, factorial, or sparse codes actually are natural, input-dependent by-products of LOCOCODE, although biologically plausible sparseness is the most common case. Unlike the objective functions of previous methods (e.g., [20]), however, LOCOCODE's does *not* contain an explicit term enforcing, say, sparse codes. This seems to suggest that LOCOCODE's objective may embody a general principle of unsupervised learning going beyond previous, more specialized ones. We would like to emphasize, however, that our focus on low mapping complexity (LMC) is meant to complement previous views rather than to replace them. We see LMC as one fragment of a puzzle whose solution would correspond to a better understanding of the yet unknown, "true" goal of sensory coding.

Limitations. FMS' order of computational complexity depends on the number of output units. For typical classification tasks (requiring few output units) it equals standard backprop's. In the AA case, however, the output's dimensionality grows with the input's. That's why large scale FMS-trained AAs seem to require a parallel implementation.

Outlook. We suspect that LOCOCODE reduces to "principal component analysis" (PCA, e.g., [19]) and "independent component analysis" (ICA, e.g., [1,5]) in case of certain simple linear signal mixtures. LOCOCODE appears to go beyond ICA, however, because it (a) is not inherently limited to the linear case, and (b) does not need *a priori* information about the number of independent data

sources. The precise relation between ICA and LOCOCODE remains to be studied though. We also intend to look at alternative LOCOCODE implementations besides FMS-based AAs. Finally we would like to improve our understanding of the relationship between low-complexity codes, low-complexity art [28] and informal notions such as "beauty".

5 ACKNOWLEDGMENTS

We would like to thank Peter Dayan and Nic Schraudolph for helpful discussions and comments. Results for realistic visual data were obtained with a modification of the code written by M. Baumgartner for his diploma thesis. This work was supported by *DFG grant SCHM 942/3-1* from "Deutsche Forschungsgemeinschaft".

References

1. S. Amari, A. Cichocki, and H.H. Yang. A new learning algorithm for blind signal separation. In David S. Touretzky, Michael C. Mozer, and Michael E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8*, pages 757–763. The MIT Press, Cambridge, MA, 1996.
2. P. Baldi and K. Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks*, 2:53–58, 1989.
3. H. B. Barlow, T. P. Kaushal, and G. J. Mitchison. Finding minimum entropy codes. *Neural Computation*, 1(3):412–423, 1989.
4. S. Becker. Unsupervised learning procedures for neural networks. *International journal of Neural Systems*, 2(1 & 2):17–33, 1991.
5. A. J. Bell and T. J. Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neural Computation*, 7(6):1129–1159, 1995.
6. P. Dayan and R. Zemel. Competition and multiple cause models. *Neural Computation*, 7:565–579, 1995.
7. D. DeMers and G. Cottrell. Non-linear dimensionality reduction. In J. D. Cowan S. J. Hanson and C. L. Giles, editors, *Advances in Neural Information Processing Systems 5*, pages 580–587. Morgan Kaufmann, San Mateo, CA, 1993.
8. D. J. Field. What is the goal of sensory coding? *Neural Computation*, 6:559–601, 1994.
9. P. Földiák and M. P. Young. Sparse coding in the primate cortex. In M. A. Arbib, editor, *The Handbook of Brain Theory and Neural Networks*, pages 895–898. The MIT Press, Cambridge, Massachusetts, 1995.
10. 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**.
11. G. E. Hinton and R. S. Zemel. Autoencoders, minimum description length and Helmholtz free energy. In J. D. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems 6*, pages 3–10. Morgan Kaufmann, San Mateo, CA, 1994.
12. S. Hochreiter and J. Schmidhuber. Flat minima. *Neural Computation*, 9(1):1–42, 1997.

13. S. Hochreiter and J. Schmidhuber. LOCOCODE. Technical Report FKI-222-97, Fakultät für Informatik, Technische Universität München, 1997.
14. S. Hochreiter and J. Schmidhuber. Unsupervised coding with LOCOCODE. In *Proc. ICANN 97, Lausanne, Switzerland*. Springer, 1997. To appear.
15. T. Kohonen. *Self-Organization and Associative Memory*. Springer, second ed., 1988.
16. M. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE journal*, 37:233–243, 1991.
17. Z. Li. A theory of the visual motion coding in the primary visual cortex. *Neural Computation*, 8(4):705–730, 1995.
18. R. Linsker. Self-organization in a perceptual network. *IEEE Computer*, 21:105–117, 1988.
19. E. Oja. Neural networks, principal components, and subspaces. *International journal of Neural Systems*, 1(1):61–68, 1989.
20. B. A. Olshausen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, 1996.
21. G. Palm. On the information storage capacity of local learning rules. *Neural Computation*, 4(2):703–711, 1992.
22. J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.
23. A. J. Robinson. *Dynamic Error Propagation Networks*. PhD thesis, Trinity Hall and Cambridge University Engineering Department, 1989.
24. D. E. Rumelhart and D. Zipser. Feature discovery by competitive learning. In *Parallel Distributed Processing*, pages 151–193. MIT Press, 1986.
25. E. Saund. A multiple cause mixture model for unsupervised learning. *Neural Computation*, 1995.
26. J. Schmidhuber. Learning factorial codes by predictability minimization. *Neural Computation*, 4(6):863–879, 1992.
27. J. Schmidhuber. Discovering neural nets with low Kolmogorov complexity and high generalization capability. *Neural Networks*, 1997. In press.
28. J. Schmidhuber. Low-complexity art. *Leonardo, Journal of the International Society for the Arts, Sciences, and Technology*, 30(2):97–103, 1997.
29. J. Schmidhuber, M. Eldracher, and B. Foltin. Semilinear predictability minimization produces well-known feature detectors. *Neural Computation*, 8(4):773–786, 1996.
30. N. N. Schraudolph and T. J. Sejnowski. Unsupervised discrimination of clustered data via optimization of binary information gain. In J. D. Cowan S. J. Hanson and C. L. Giles, editors, *Advances in Neural Information Processing Systems 5*, pages 499–506. San Mateo, CA: Morgan Kaufmann, 1993.
31. R.J. Solomonoff. A formal theory of inductive inference. Part I. *Information and Control*, 7:1–22, 1964.
32. C. S. Wallace and D. M. Boulton. An information theoretic measure for classification. *Computer journal*, 11(2):185–194, 1968.
33. S. Watanabe. *Pattern Recognition: Human and Mechanical*. Willey, New York, 1985.
34. R. S. Zemel and G. E. Hinton. Developing population codes by minimizing description length. In J. D. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems 6*, pages 11–18. San Mateo, CA: Morgan Kaufmann, 1994.