LOCOCODE PERFORMS NONLINEAR ICA WITHOUT KNOWING THE NUMBER OF SOURCES

Sepp Hochreiter

Fakultät für Informatik
Technische Universität München
80290 München, Germany
hochreit@informatik.tu-muenchen.de

Jürgen Schmidhuber

IDSIA
Corso Elvezia 36
6900 Lugano, Switzerland
juergen@idsia.ch

ABSTRACT

*Lo*w-*co*mplexity *co*ding and *de*coding (LOCOCODE), a novel approach to sensory coding, trains autoassociators (AAs) by *Flat Minimum Search* (FMS), a recent general method for finding low-complexity networks with high generalization capability. FMS works by minimizing both training error and required weight precision. We find that as a by-product LOCOCODE separates nonlinear superpositions of sources without knowing their number. Assuming that the input data can be reduced to few simple causes (this is often the case with visual data), according to our theoretical analysis the hidden layer of an FMS-trained AA tends to code each input by a sparse code based on as few simple, independent features as possible. In experiments LOCOCODE extracts optimal codes for difficult, nonlinear versions of the "noisy bars" benchmark problem, while traditional ICA and PCA do not.

1. INTRODUCTION

Blind source separation and Independent Component Analysis (ICA) refer to recovering original source signals from observed mixtures of them. This belongs to the area of unsupervised learning and has become one of the most active areas in signal processing. ICA favors near-factorial, minimally redundant codes of the input data. Recent efforts have focused on linear source mixtures and fall into two major categories: Maximum Entropy (ME) [3, 19] and Minimum Mutual Information (MMI) [1, 5, 4, 15, 17]. Both are based on "code component-oriented objective functions (COCOFs)": ME maximizes code entropy, MMI minimizes the mutual information between code components. Most current variants require *a priori* knowledge of the number of independent sources.

There has been work on COCOFs for nonlinear ICA given unknown source numbers, e.g., [2, 21, 7, 22]. In particular, to the best of our knowledge reference [21] represents the first "neural" approach to nonlinear ICA. Here, however, we shift the point of view away from COCOF-based approaches and instead focus on the information-theoretic costs of code generation. We use a novel approach to unsupervised learning called "*lo*w-*co*mplexity *co*ding and *de*coding" (LOCOCODE [14]). In the spirit of research on minimum description length (MDL), 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 an LCM.

To implement LOCOCODE we regularize an autoassociator (AA) whose hidden layer activations represent the code. The hidden layer is forced to code information about the input data by minimizing training error; the regularizer reduces coding/decoding costs. Our regularizer of choice will be *Flat Minimum Search* (FMS) [13].

We will see that nonlinear ICA actually occurs as a by-product of LOCOCODE's more general complexity-minimizing strategy: we find that LOCOCODE encourages *sparse codes* based on *few, separated, simple component functions* (the functions determining the activation of a code component in response to a given input).

This also establishes a connection to extensive recent work on biologically plausible, sparse distributed codes [18, 9, 23, 8, 6, 20, 11, 16]. In fact, we find that LOCOCODE is appropriate for extracting independent sources if single inputs (with many input components) are determined by few sources computable by simple functions. Hence, assuming that visual data usually can be reduced to few simple causes, LOCOCODE is appropriate for visual coding. Unlike recent linear ICA methods, LOCOCODE (a) is not inherently limited to the linear case, and (b) does not need *a priori* information about the number of independent data sources

- it simply prunes superfluous code components.

2. FLAT MINIMUM SEARCH: REVIEW AND ANALYSIS

FMS is a general gradient-based 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. Previous FMS applications focused on supervised learning [12, 13].

Notation. Let O, H, I denote index sets for output, hidden, and 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 error term:

$$B = \sum_{i,j:\ i \in O \cup H} \log \sum_{k \in O} \left(\frac{\partial y^k}{\partial w_{ij}} \right)^2 + \qquad (1)$$
$$W \log \sum_{k \in O} \left(\sum_{i,j: i \in O \cup H} \frac{\left| \frac{\partial y^k}{\partial w_{ij}} \right|}{\sqrt{\sum_{k \in O} \left(\frac{\partial y^k}{\partial w_{ij}} \right)^2}} \right)^2 .$$

$E = E_q + \lambda B$ is minimized by gradient descent, where E_q is the training set mean squared error (MSE), and λ a positive "regularization constant" scaling B's influence. B measures the weight precision (number of bits needed to describe all weights in the net). Given a constant number of output units, FMS can be implemented efficiently, namely, with standard backprop's order of computational complexity [13].

2.1. FMS Analysis

Simple component functions (CFs). Minimizing B's term

$$T1 := \sum_{i,j:\ i \in O \cup H} \log \sum_{k \in O} \left(\frac{\partial y^k}{\partial w_{ij}} \right)^2$$

obviously reduces output sensitivity with respect to weights (and therefore units). $T1$ is responsible for pruning weights (and, therefore, units). $T1$ is one reason why low-complexity (or simple) CFs are preferred: weight precision (or complexity) is mainly determined by $\frac{\partial y^k}{\partial w_{ij}}$. The chain rule allows for rewriting

$$\frac{\partial y^k}{\partial w_{ij}} = \frac{\partial y^k}{\partial y^i} \frac{\partial y^i}{\partial w_{ij}} = \frac{\partial y^k}{\partial y^i}\ f_i'(s_i)\ y^j, \qquad (2)$$

where $f_i'(s_i)$ is the derivative of the activation function of unit i with activation y^i. We obtain

$$\begin{aligned} T1 \quad = \quad & 2 \sum_{i \in O \cup H} \text{fan-in}(i) \log f_i'(s_i)\ + \\ & 2 \sum_{j \in H \cup I} \text{fan-out}(j) \log y^j\ + \\ & \sum_{i \in O \cup H} \text{fan-in}(i) \log \sum_{k \in O} \left(\frac{\partial y^k}{\partial y^i} \right)^2 , \end{aligned}$$

where fan-in(i) (fan-out(i)) denotes the number of incoming (outgoing) weights of unit i.

$T1$ makes (1) unit activations decrease to zero in proportion to their fan-outs, (2) first-order derivatives of activation functions decrease to zero in proportion to their fan-ins, and (3) the influence of units on the output decrease to zero in proportion to the unit's fan-in. For a detailed analysis see Hochreiter and Schmidhuber (1997a).

Sparseness. Point (1) above favors sparse hidden unit activations (here: few active components); point (2) favors non-informative hidden unit activations hardly affected by small input changes. Point (3) favors sparse hidden unit activations in the sense that "few hidden units contribute to producing the output". In particular, sigmoid hidden units with activation function $\frac{1}{1+\exp(-x)}$ favor near-zero activations.

B's second term

$$T2 := W \log \sum_{k \in O} \left(\sum_{i,j \in O \cup H} \frac{\left| \frac{\partial y^k}{\partial w_{ij}} \right|}{\sqrt{\sum_{k \in O} \left(\frac{\partial y^k}{\partial w_{ij}} \right)^2}} \right)^2$$

punishes units with similar influence on the output.

Using equation (2) and for $i \in O$

$$\frac{\left| \frac{\partial y^k}{\partial y^i} \right|}{\sqrt{\sum_{k \in O} \left(\frac{\partial y^k}{\partial y^i} \right)^2}} = \delta_{ki}\ ,$$

where δ is the Kronecker delta ($\delta_{ki} = 1$ if $k = i$ and 0 otherwise), we obtain

$$T2 = W \log \left(|O| \; |O \times H|^2 + |I|^2 \sum_{k \in O} \sum_{i \in H} \sum_{u \in H} \frac{\left| \frac{\partial y^k}{\partial y^i} \right| \; \left| \frac{\partial y^k}{\partial y^u} \right|}{\sqrt{\sum_{k \in O} \left(\frac{\partial y^k}{\partial y^i} \right)^2} \sqrt{\sum_{k \in O} \left(\frac{\partial y^k}{\partial y^u} \right)^2}} \right)$$

See [14] for intermediate reformulation steps of $T2$.

We observe: (1) an output unit that is very sensitive with respect to two given hidden units will heavily contribute to $T2$ (compare the numerator in the last term of $T2$). (2) This large contribution can be reduced by making both hidden units have large impact on other output units (see denominator in the last term of $T2$).

Few separated component functions. Hence FMS tries to figure out a way of using (1) as few CFs as possible for determining the activation of each output unit, while simultaneously (2) using the same CFs for determining the activations of as many output units as possible (common CFs). (1) and $T1$ separate the CFs: the force towards simplicity (see $T1$) prevents input information from being channelled through a single CF; the force towards few CFs per output makes them non-redundant. (1) and (2) cause few CFs to determine all outputs.

Summary. Collectively $T1$ and $T2$ (which make up B) encourage *sparse codes* based on *few separated simple component functions* producing all outputs. Due to space limitations a more detailed analysis (e.g. linear output activation) had to be left to a TR [14] (on the WWW).

3. EXPERIMENTS

Nonlinear noisy bars adapted from [10, 11]. The input is a 5×5 pixel grid with horizontal and vertical bars at random positions. The task is to extract the independent features (the bars). Each of the 10 possible bars appears with probability $\frac{1}{5}$. Bar intensities vary in $[0.1, 0.5]$; input units that see a pixel of a bar are activated correspondingly others adopt activation -0.5. We add Gaussian noise with variance 0.05 and mean 0 to each pixel. In contrast to [10, 11] we allow for mixing of vertical and horizontal bars — *this makes the task harder,* because the bars do not add linearly, thus exemplifying a major characteristic of real visual inputs.

Comparison. We compare Lococode to PCA and ICA, which is realized by Cardoso's JADE algorithm based on whitening and subsequent joint diagonalization of 4th-order cumulant matrices. To measure the information conveyed by resulting codes we train a standard backprop net on the training set used for code generation. Its inputs are the code components; its task is to reconstruct the original input. The test set consists of 500 off-training set exemplars. *Coding efficiency* is the average number of bits needed to code a test set input pixel. The code components are scaled to the interval $[0, 1]$ and partitioned into discrete intervals. Assuming independence of the code components we estimate the probability of each discrete code value by Monte Carlo sampling on the training set. To obtain the test set codes' bits per pixel (Shannon's optimal value) the average sum of all negative logarithms of code component probabilities is divided by the number of input components.

Source numbers. For ICA and PCA we have to provide information about the number (ten) of independent sources (tests with n assumed sources will be denoted by ICA-n and PCA-n). Lococode does not require this — using 25 hidden units (HUs) we expect Lococode to *prune* the 15 superfluous HUs. The nonlinearly added sources make the task hard for PCA and ICA. All details necessary for reimplementation are given in [14].

Results. See Table 1. Lococode finds independent sources that *exactly* mirror the pattern generation process. PCA codes and ICA-15 codes, however, are unstructured and dense. While ICA-10 codes are almost sparse and do recognize some sources, the sources are not clearly separated like with Lococode. While the reconstruction errors of all methods are similar, Lococode has the best coding efficiency. 15 of its 25 HUs are indeed automatically pruned.

	#	rec.	code	bits per pixel		
	c.	err.	type	10	50	100
LOC	10	1.05	sparse	0.58	1.16	1.37
ICA	10	1.02	sparse	0.81	1.45	1.68
PCA	10	1.03	dense	0.80	1.42	1.66
ICA	15	0.71	dense	1.19	2.14	2.50
PCA	15	0.72	dense	1.17	2.11	2.47

Table 1: *Results: coding method, number of relevant code components (code size), reconstruction error, nature of code observed on the test set. PCA's and ICA's code sizes need to be prewired.* Lococode*'s, however, are found automatically: we always start with 25 HUs but eventually end up with the optimal number of 10. The final 3 columns show the coding efficiency measured in bits per pixel, assuming the real-valued HU activations are partitioned into 10, 50, and 100 discrete intervals.* Lococode *codes most efficiently.*

For each of the 25 HUs, Figure 1 shows a 5×5 square depicting 25 typical post-training weights on connections from 25 inputs. Figure 2 shows the according post-training weights on connections to 25 outputs. White (black) circles on gray (white) background are positive (negative) weights. The circle radius is proportional to the weight's absolute value. Figure 1 also shows the bias weights (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).

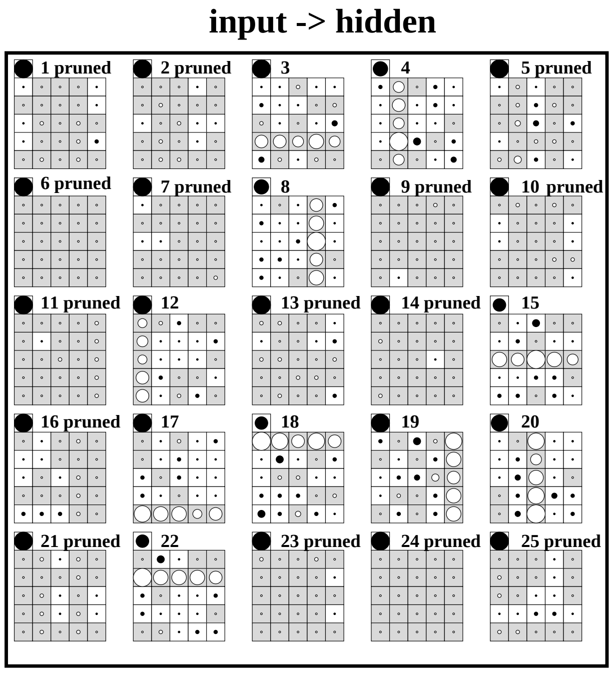

Figure 1: LOCOCODE*'s input-to-hidden weights (see text for explanation). Despite noise and nonlinearity,* LOCOCODE exactly *extracts the independent sources and prunes the 15 superfluous code components.*

LOCOCODE can exploit the advantages of sigmoid output functions and is applicable to nonlinear signal mixtures. PCA and ICA, however, are limited to linear source superpositions. See Figure 3 for PCA results. See Figure 4 for ICA-10 result with *a priori* information about the number of sources. See Figure 5 for ICA-15 results.

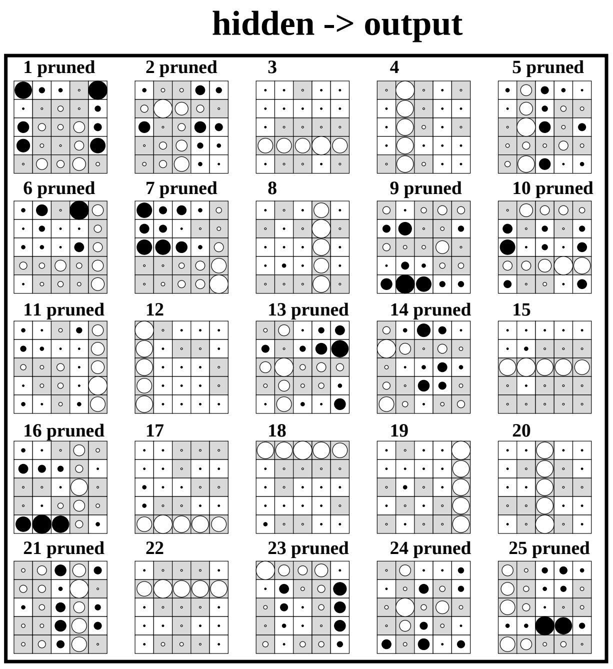

Figure 2: LOCOCODE*'s hidden-to-output weights.*

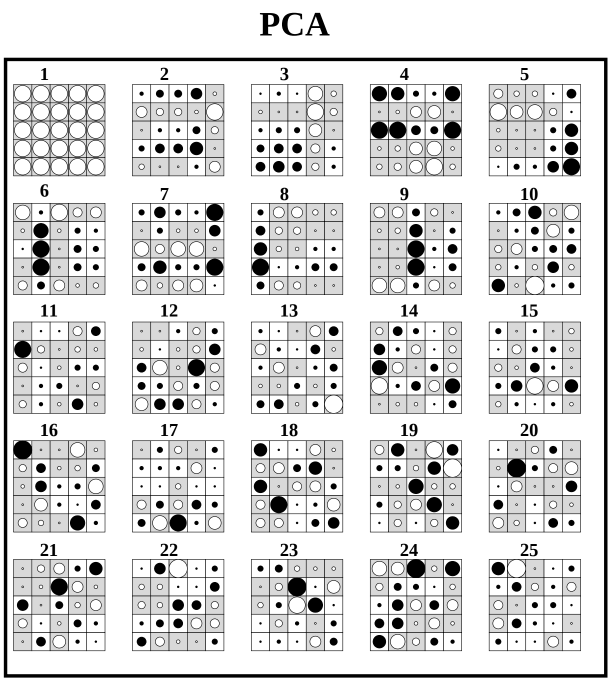

Figure 3: *PCA weights to code components. PCA does not extract the true independent sources at all.*

4. CONCLUSION

According to our analysis LOCOCODE attempts to describe single inputs with as few and as simple sources as possible. Given the statistical properties of many visual inputs (with few defining features), this typically results in sparse codes. Since LOCOCODE minimizes the information-theoretic complexity of the mappings used for coding and decoding, the resulting codes typically compromise between conflicting goals. They tend to be

ICA 10

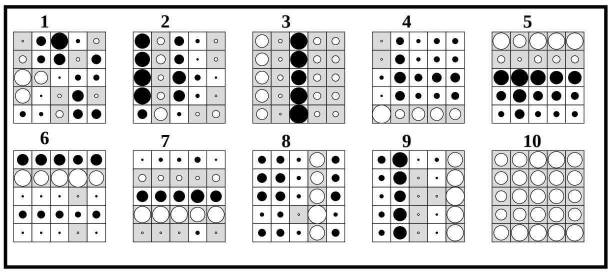

Figure 4: *ICA with 10 components: weights to code components. We observe that some of the sources are partially reflected by some of ICA-10's feature detectors. The results are better than those of ICA-15, but by far not as convincing as* LOCOCODE*'s.*

ICA 15

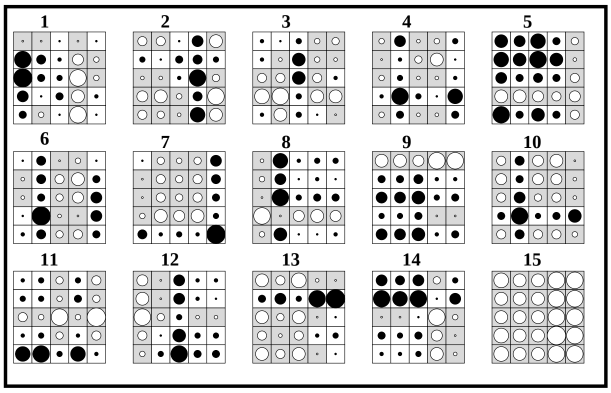

Figure 5: *ICA with 15 components: weights to code components. ICA-15 codes fail to represent the true sources.*

sparse and exhibit *low but not minimal* redundancy — if the cost of minimal redundancy is too high.

Our results suggest that LOCOCODE's objective may embody a general principle of unsupervised learning going beyond previous, more specialized, COCOF-based ones. We see that there is at least one representative (FMS) of a broad class of algorithms (regularizers that reduce network complexity) which (1) can do optimal feature extraction as a by-product, (2) outperforms traditional ICA and PCA on nonlinear visual source separation tasks, and (3) unlike ICA does not even need to know the number of independent sources in advance. This reveals an interesting, previously ignored connection between regularization and ICA research, and may represent a first step towards unification of regularization and unsupervised learning.

More. Due to space limitations, much additional theoretical and experimental analysis had to be left to a tech report (29 pages, 20 figures) on the WWW: see [14].

Acknowledgements. This work was supported by and *DFG grant BR 609/10-2* from "Deutsche Forschungsgemeinschaft."

5. 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] H. B. Barlow, T. P. Kaushal, and G. J. Mitchison. Finding minimum entropy codes. *Neural Computation*, 1(3):412–423, 1989.

[3] A. J. Bell and T. J. Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neural Computation*, 7(6):1129–1159, 1995.

[4] J.-F. Cardoso and A. Souloumiac. Blind beamforming for non Gaussian signals. *IEE Proceedings-F*, 140(6):362–370, 1993.

[5] P. Comon. Independent component analysis – a new concept? *Signal Processing*, 36(3):287–314, 1994.

[6] P. Dayan and R. Zemel. Competition and multiple cause models. *Neural Computation*, 7:565–579, 1995.

[7] G. Deco and W. Brauer. Nonlinear higher-order statistical decorrelation by volume-conserving neural architectures. *Neural Networks*, 8(4):525–535, 1995.

[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, P. Dayan, B. J. Frey, and R. M. Neal. The wake-sleep algorithm for unsupervised neural networks. *Science*, 268:1158–1161, 1995.

[11] G. E. Hinton and Z. Ghahramani. Generative models for discovering sparse distributed representations. *Philosophical Transactions of the Royal Society* **B**, 352:1177–1190, 1997.

[12] S. Hochreiter and J. Schmidhuber. Simplifying nets by discovering flat minima. In G. Tesauro, D. S. Touretzky, and T. K. Leen, editors, *Advances in Neural Information Processing Systems 7*, pages 529–536. MIT Press, Cambridge MA, 1995.

[13] S. Hochreiter and J. Schmidhuber. Flat minima. *Neural Computation*, 9(1):1–42, 1997.

[14] S. Hochreiter and J. Schmidhuber. LOCOCODE. Technical Report FKI-222-97, Revised Version, Fakultät für Informatik, Technische Universität München, 1998.

[15] C. Jutten and J. Herault. Blind separation of sources, part I: An adaptive algorithm based on neuromimetic architecture. *Signal Processing*, 24(1):1–10, 1991.

[16] M. S. Lewicki and B. A. Olshausen. Inferring sparse, overcomplete image codes using an efficient coding framework. In M. I. Jordan, M. J. Kearns, and S. A. Solla, editors, *Advances in Neural Information Processing Systems 10*, 1998. To appear.

[17] L. Molgedey and H. G. Schuster. Separation of independent signals using time-delayed correlations. *Phys. Reviews Letters*, 72(23):3634–3637, 1994.

[18] M. C. Mozer. Discovering discrete distributed representations with iterative competitive learning. In R. P. Lippmann, J. E. Moody, and D. S. Touretzky, editors, *Advances in Neural Information Processing Systems 3*, pages 627–634. San Mateo, CA: Morgan Kaufmann, 1991.

[19] J.-P. Nadal and N. Parga. Non-linear neurons in the low noise limit: a factorial code maximises information transfer. *Network*, 5:565–581, 1904.

[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] J. Schmidhuber. Learning factorial codes by predictability minimization. *Neural Computation*, 4(6):863–879, 1992.

[22] J. Schmidhuber, M. Eldracher, and B. Foltin. Semilinear predictability minimization produces well-known feature detectors. *Neural Computation*, 8(4):773–786, 1996.

[23] 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.