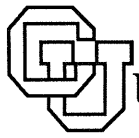
LEARNING FACTORIAL CODES BY PREDICTABILITY MINIMIZATION

JURGEN SCHMIDHUBER

CU-CS-565-91 December 1991



University of Colorado at Boulder
DEPARTMENT OF COMPUTER SCIENCE

LEARNING FACTORIAL CODES BY PREDICTABILITY MINIMIZATION
Technical Report CU-CS-565-91

Jürgen Schmidhuber
Department of Computer Science
University of Colorado
Campus Box 430, Boulder, CO 80309, USA
yirgan@cs.colorado.edu

Dec. 18, 1991

Abstract

I present a novel general principle for unsupervised learning of distributed non-redundant internal representations of input patterns or input sequences. With a given set of representational units, each unit tries to react to the environment such that it minimizes its predictability by an adaptive predictor that sees all the other units. This encourages each unit to filter 'abstract concepts' out of the environmental input such that these concepts are statistically independent of those upon which the other units focus. I discuss various simple yet potentially powerful implementations of the principle which aim at finding binary factorial codes [1], i.e. codes where the probability of the occurrence of a particular input is simply the product of the probabilities of the corresponding code symbols. None of these implementations requires the explicit maximization of the determinant of the output covariance matrix. Unlike the latter method, however, and unlike the more conventional linear methods for principal component analysis the principle allows for removing not only linear but also non-linear output redundancy. Methods for finding non-redundant codes automatically embed Occam's razor for finding codes using a minimal number of units. The final part of this paper describes an entirely local algorithm that has a potential for learning unique representations of extended sequences. Experiments show that algorithms based on the principle of predictability minimization are practically feasible.

1 INTRODUCTION

Consider a perceptual system being exposed to an unknown (usually dynamic) environment. The system has some kind of internal 'state' to represent external events. We consider the general case where the state is an n-dimensional distributed representation y^p (a vector of real-valued or binary code symbols) created in response to the p-th input vector x^p. Later I will address the case of input sequences.

Typical questions asked in the context of unsupervised learning are: Are there any *useful* goal-independent internal representations reflecting environmental 'regularities'? What is a 'regularity'? How can the system detect and represent it?

Strictly speaking, the answer to the first question is no – if the system's goals do not depend on the environmental input, then there is no point in searching for regularities there. But in all realistic and interesting cases, reasonable goals have something to do with the input. Experience tells us that the achievement of many typical goals in typical environments (like emission of target values at certain times, or reinforcement maximization) is facilitated by building upon the same internal representations of inputs (or more generally speaking, of input histories). This experience flows into the models that we build. It lets us define the kind of regularities we are looking for.

Usually, the term 'regularity' is approached from a statistical perspective[1]. Various performance criteria and algorithms for capturing statistical regularities have been defined, most of them essentially implementing principal component analysis in the case of linear units, e.g. [8][11][10].

A more ambitious and potentially more useful objective of unsupervised learning, however, is to represent the environment such that the various parts of the representation are *statistically independent* of each other. In other words, we would like to have methods for decomposing the environment into entities that belong together and do not have much to do with other entities[2]. This notion is captured by the concept of 'factorial codes' [1].

The aim of 'factorial coding' is the following: Given the statistical properties of the inputs from the environment, find invertible internal representations such that the occurrence of the i-th code symbol y_i is independent of any of the others [1]. Such internal representations are called factorial codes because they have a remarkable and unique property: The probability of the occurrence of a

[1] However, the notion of a 'regularity' comes in many forms. To appreciate that there is no single accepted approach to defining 'regular structure', note that algorithmic information theory [6][4] defines regular patterns in a way different from statistical information theory [17]. The algorithmic information of a pattern is essentially the length of the shortest program to generate it. Since there is no general way to find such a program (essentially because there is no upper limit for its run-time), algorithmic information theory is usually not considered as a practical tool for detecting regularities.

[2] The G-Max algorithm [9] aims at a related objective: It tries to discover features that account for input redundancy. G-Max, however, is designed for single output units only.

particular input is simply the product of the probabilities of the corresponding code symbols.

Among the advantages of non-redundant codes are as follows: As Barlow, Kaushal, and Mitchison point out [1], with such codes the detection of dependence between two symbols indicates hitherto undiscovered associations (novelty detection). As Becker observes [2], if a representation with uncorrelated components is used as the input to a higher-level linear supervised learning network, then the Hessian of its error function is diagonal, thus allowing efficient methods for speeding up learning (note that statistical independence is a stronger criterion than the mere absence of statistical correlation). Non-linear networks ought to profit as well. But there is more to non-redundant codes: An efficient method for discovering mutually independent features would have consequences for many segmentation tasks. For instance, consider the case where the inputs are given by retinal images of various objects whose positions are independent of each other. At any given time, the activations of nearby 'pixels' caused by the same object are highly dependent on each other. Therefore a factorial code would not represent them by different parts of the internal storage, thus filtering 'abstract objects' out of the input.

Finally, any method for finding factorial codes automatically implements Occam's razor (see section 9) which prefers simpler models over more complex ones, where simplicity is defined as the number of storage cells necessary to represent the environment in a factorial fashion. This implies storage efficiency, as well as a potential for better generalization capabilities.

Since factorial codes have to meet the invertibility criterion, in noise-free environments they have to obey Linsker's Infomax principle [7]. The latter aims at maximizing the mutual information between input and internal representation. Let $P(x^p)$ denote the probability of input x^p. Given a finite number of distinguishable internal representations y^q, the mutual information is given by

$$R = < log \frac{P(y^q \mid x^p)}{P(y^q)} >,$$

where $< ... >$ denotes the ensemble average. In the case of single output cells, there are efficient and straight forward rules for minimizing R when binary outputs are used and, under appropriate Gaussian assumptions, when real-valued outputs are used [7][3]. With high-dimensional internal representations, however, the situation becomes more complicated. For instance, assuming simplifying and often not very realistic Gaussian distributions of the output and input signals, maximization of the Shannon information rate implies maximization of the determinant D of the $n \times n$-covariance matrix of the output activations [16]. In the absence of noise, this method decorrelates the output activations. It tends to remove linear dependencies between the output units, but not necessarily non-linear ones. In different contexts, previous approaches explicitly calculated the derivatives of D [7][20]. For high-dimensional internal representations, this is clumsy and biologically rather implausible.

Neither Linsker nor Zemel and Hinton present an efficient general method for maximizing information in the case of multi-dimensional outputs, and Barlow et al. do not present an efficient general method for finding factorial codes.

The main contribution of this paper is a simple but general principle useful for finding such codes. There are a number of straight forward 'neural' implementations of the principle that do not involve the explicit calculation the derivatives of D. Due to the presence of non-linearities, the methods described above go beyond previous work which applies only to linear units [8][11][5][10][7][18].

I would not be surprised, however, if the general problem of finding factorial codes turned out to be NP-hard. In that case, gradient-based procedures as described herein could not be expected to always find factorial codes. The paper at hand focuses on the novel basic principle without trying to provide solutions for the old problem of local maxima. Also, the purpose of this report is not to compare the performance of algorithms based on the novel principle to the performance of existing sequential 'non-neural' heuristic methods [1]. The experiments described below are merely for illustrative purposes.

2 OUTLINE OF PAPER

Section 3 states the problem by formulating three criteria that qualify a code as a binary factorial code: The invertibility criterion, the binary criterion and, most notably, the mutual independence criterion. Section 4 contains the central idea of this paper: It describes the fundamental predictor-based architecture for removing intra-representational redundancy. This architecture forms the basis for all following sections. The essential idea is: For each unit in the internal representation ('representational unit') there is an adaptive predictor that sees all the other units. Each predictor is trained to predict the current activation of a representational unit from the current activations of the other units. In turn, *all representational units try to transform the environmental inputs such that they take on values that make them as unpredictable as possible*, while at the same time conveying maximal information about the input. With the help of the predictors one can define various objective functions for achieving this idea of mutual predictability minimization; various possibilitites are discussed in sections 5, 6, and 7: Section 5 defines three objective functions, each designed for achieving one of the three criteria, the most notable being a predictor-based function for minimizing mutual dependence. Section 6 discusses various combinations of these three particular functions as well as a non-predictor-based error function for minimizing the sum of bit-entropies (as opposed to explicit predictability minimization). A parameter tuning problem is identified, which leads to section 7. In section 7 an objective function designed for 'local conditioned variance maximization' is described which appears not to suffer from the disadvantage mentioned in section 6. With local conditioned variance maximization the internal state is trained to maximize exactly the same function

the predictors try to minimize. Section 8 describes a 'neural' gradient ascent based algorithm for predictability minimization. This section also considers modifications that become necessary if binary *stochastic* units instead of semi-linear units are employed. Section 9 considers the relationship between factorial codes and Occam's razor. Section 10 discusses a local algorithm for learning unique representations of extended *sequences* based on intra-representational predictability minimization, as well as the relationship of this algorithm to recent sequence-chunking methods. Section 11 mentions how goal-directed learning and unsupervised intra-representational predictability minimization can be made compatible. Section 12 describes a number of experiments which demonstrate the practical applicability of the basic principle.

3 FORMULATING THE PROBLEM

Let us assume n different adaptive input processing modules which see a single input at a time. The output of each module can be implemented as a set of neuron-like units. Throughout this paper I focus on the simplest case: One output unit (also called a representational unit) per module. The i-th module (or unit) produces an output value $y_i^p \in [0, 1]$ in response to the current external input vector $\boldsymbol{x}^p$. In what follows, $P(A)$ denotes the probability of event A, $P(A \mid B)$ denotes the conditional probability of event A given B, $\bar{y}_i$ denotes the mean of the activations of unit i, and E denotes the expectation operator.

The methods described in this paper are primarily devoted to finding binary or at least *quasi-binary* codes. Each code symbol participating in a quasi-binary code is either 0 or 1 in response to a given input pattern or emits a constant value in response to every input pattern. Therefore, binary codes are a special case of quasi-binary codes. Most of our quasi-binary codes will be created by starting out from real-valued codes.

Recall that there are three criteria that a binary factorial code must fulfill:

1. *The binary criterion*: Each code-symbol should be either 1 or 0 in response to a given input pattern.

2. *The invertibility criterion*: It must be possible to reconstruct the input from the code. In cases where the environment is too complex (or too noisy) to be fully coded into limited internal representations (i.e., in the case of binary codes where there are more than $2^{dim(y)}$ input patterns), we want to relax the invertibility criterion. In that case, we still want the internal representations to convey maximal information about the inputs. The focus of this paper, however, is on situations like the ones studied in [1]: Noise-free environments and sufficient representational capacity in the representational units.

3. *The independence criterion*: The occurrence of each code symbol ought to be independent from all other code symbols. For binary codes we may rewrite this criterion by requiring that

$$E(y_i \mid \{y_k, k \neq i\}) = P(y_i = 1 \mid \{y_k, k \neq i\}) = P(y_i = 1) = E(y_i).$$

The latter condition implies that y_i does not depend on $\{y_k, k \neq i\}$. In other words, $E(y_i \mid \{y_k, k \neq i\})$ is computable from a constant. With real-valued codes this criterion does not necessarily imply that the y_k are independent (see also section 6.1).

4 THE BASIC PRINCIPLE AND ARCHITECTURE

For each representational unit i there corresponds an adaptive predictor P_i, which, in general, is non-linear. With the p-th input pattern x^p, P_i's input is the concatenation of the outputs y_k^p of all units $k \neq i$. P_i's one-dimensional output P_i^p is trained to equal the expectation $E(y_i \mid \{y_k^p, k \neq i\})$. It is well-known that this can be achieved by letting P_i minimize[3]

$$\sum_p (P_i^p - y_i^p)^2. \tag{1}$$

With the help of the n predictors one can define various objective functions for the representational modules to enforce the 3 criteria listed above. Common to these methods (to be described in the following sections) is that all units are trained to take on values that *minimize mutual predictability* via the predictors: Each unit tries to extract features from the environment such that no combination of $n-1$ units conveys information about the remaining unit. In other words, no combination of $n-1$ units should allow better predictions of the remaining unit than a prediction based on a constant. I call this the *principle of intra-representational predictability minimization* or, somewhat shorter, the *principle of predictability minimization.*

A major novel aspect of this principle which makes it different from previous work is that it uses adaptive sub-modules (the predictors) to define the objective functions for the subjects of interest, namely, the representational units themselves.

Following the principle of predictability minimization, each processing module tries to use the statistical properties of the environment to protect itself from being predictable. This forces each processing module to focus on aspects of the environment that are independent from environmental properties upon which the other modules focus.

[3] Cross-entropy is another objective function for achieving the same goal. In the experiments, however, the mean squared error based function above led to satisfactory results.

5 OBJECTIVE FUNCTIONS FOR THE THREE CRITERIA

The only novel objective function presented in this section is the one of subsection 5.3 - sections 5.1 and 5.2 present nothing new.

5.1 AN OBJECTIVE FUNCTION FOR THE BINARY CRITERION

An objective function V for enforcing binary codes is given by

$$V = \sum_i \sum_p (\bar{y}_i - y_i^p)^2.$$

Maximizing this term encourages each unit to take on binary values. The contribution of each unit i is maximized if $E(y_i)$ is as close to 0.5 as possible. This implies maximal entropy for unit i under the binary constraint, i.e., i wants to become a binary unit that conveys maximal information about its input.

5.2 AN ERROR FUNCTION FOR THE INVERTIBILITY CRITERION

The following is a simple, well-known method for enforcing invertibility: With pattern p, a reconstructor module receives the concatenation of all y_i^p as an input and is trained to emit as an output the reconstruction z^p of the external input x^p. The basic structure is an auto encoder. The auto encoder's objective function, to be minimized, is defined as

$$I = \sum_p (z^p - x^p)^T (z^p - x^p). \tag{2}$$

5.3 AN ERROR FUNCTION FOR THE INDEPENDENCE CRITERION

For the sake of argument, let us assume that at all times each P_i is as good as it can be, meaning that P_i always predicts the expectation of y_i conditioned on the outputs of the other modules, $E(y_i \mid \{y_k^p, k \neq i\})$. (In practice, the predictors will have to be retrained continually.) In the case of quasi-binary codes the following objective function H is zero if the independence criterion is met:

$$H = \sum_i \sum_p [P_i^p - \bar{y}_i]^2. \tag{3}$$

This term for mutual predictability minimization aims at making the outputs independent – similar to the goal of a term for maximizing the determinant

of the covariance matrix under Gaussian assumptions [7]. The latter method, however, tends to remove only linear predictability, while the former can remove non-linear predictability as well (even without Gaussian assumptions), due to possible non-linearities learnable by non-linear predictors.

6 COMBINING ERROR TERMS

A straight forward way of combining V, I, and H is to maximize the total objective function

$$T = \alpha V - \beta I - \gamma H, \tag{4}$$

where α, β, γ are positive constants determining the relative weighting of the opposing error terms. Maximization of (4) with non-zero α tends to force the representational units to take on binary values that *maximize* independence in addition to *minimizing* the reconstruction error.

6.1 REMOVING THE VARIANCE TERM: REAL-VALUED CODES

If with a specific application we want to make use of the representational capacity of real-valued codes and if we are satisfied with decorrelated (instead of independent) representational units, then we might remove the V-Term from (4) by setting $\alpha = 0$. In this case, we want to minimize

$$\beta I + \gamma H.$$

Note that with real-valued units the invertibility criterion theoretically can be achieved with a single unit. In that case, the independence criterion would force all other units to take on constant values in response to all input patterns. In noisy environments, however, it may turn out to be advantageous to code the input into more than one representational unit. This has already been noted by Linsker in the context of his Infomax principle [7].

6.2 REMOVING THE GLOBAL INVERTIBILITY TERM

Theoretically it is sufficient to do without the auto encoder and set $\beta = 0$ in (4). In this case, we simply want to maximize

$$T = \alpha V - \gamma H.$$

The H-Term counteracts the possibility that different (near-) binary units convey the same information about the input. Setting $\beta = 0$ means to maximize information locally for each unit while at the same time trying to force each

unit to focus on different pieces of information from the environment [4]. Unlike with auto-associators, there is *no* global invertibility term.

Note that this method seemingly works diametrically opposite to the sequential, heuristic, non-neural methods described by Barlow *et al.* in [1], where the sum of bit entropies is *minimized* instead of being maximized. How can both methods pursue the same goal? One may put it this way: Among all invertible codes, Barlow et. al. try to find those closest to something similar to the independence criterion. In contrast, among all codes fulfilling the independence criterion (ensured by sufficiently strong γ), the above methods try to find the invertible ones.

6.3 AN APPROACH TO FINDING BINARY FACTORIAL CODES WITHOUT USING PREDICTORS

In passing I would like to mention another 'neural' possibility for finding factorial codes based on a procedure for finding minimum entropy codes. If the code is represented by a set of storage cells that can take on either the value 1 or 0 then the sum of the bit entropies of code b, $e(b)$, is given by

$$e(b) = -\sum_i E(y_i) log E(y_i) - \sum_i (1 - E(y_i)) log(1 - E(y_i)).$$

As pointed out in [1], if there are one or more factorial codes then finding the minima of $e(b)$ over the set of all possible codes b is equivalent to finding one of them. Barlow et al. give some (non-neural) heuristics for minimizing sums of bit-entropies, but do not present an efficient general method. I propose starting with continuous-valued units and maximizing the following total objective function:

$$T = \alpha \sum_i \sum_p (y_i^p - \bar{y}_i)^2 - \beta \sum_p (z^p - x^p)^T (z^p - x^p) + \sum_i (\bar{y}_i - \frac{1}{2})^2.$$

[4] LOCAL INFOMAX FOR REAL-VALUED CODES. A linear *real-valued* unit with a Gaussian output distribution also conveys maximal information if it has maximal variance, as noted by Linsker in [7]. In the course of describing a more complex system for maximizing mutual information between neighbouring modules, Hinton and Becker [3] maximize variance in the case of non-linear mappings between input and output. In the general case, however, such procedures seem to be at least questionable, because Gaussian assumptions are not very realistic in practical applications. And, of course, telling the learning system about Gaussian constraints has the effect of supplying it with a priori knowledge. It is important to realize that variance maximization usually does not implement the Infomax principle in the general case of units with real-valued outputs.

This footnote is based on the assumption of a hypothetical, efficient method for maximizing mutual information between two one-dimensional real-valued variables. Instead of using mean squared error terms in the application of the basic principle, we simply use terms for mutual information between single units. Define $I(a, b)$ as the mutual information between units a and b. Each predictor P_i with output unit o^i is trained to maximize $I(i, o_i)$. In turn, each unit i is trained to take on values that *minimize* $I(i, o_i)$ while at the same time *maximizing* i's entropy.

Here z^p is defined as in the section 5.2 on auto encoders (again, α and β are positive constants). The first term forces each unit to be either on or off in response to a given input pattern. The second term enforces invertibility. The third term forces the mean of each unit to be close to either 0 or 1, thus enforcing minimal bit-entropy.

A problem with this approach is that the first term and the third term try to enforce contradictory objectives – this creates a need for parameter tuning and makes the method less attractive than, say, the method of section 7.

Again, it makes sense to use cross-entropy instead of mean squared error.

6.4 A DISADVANTAGE OF THE ABOVE METHODS

Note that a factorial code causes *non-maximal* V and therefore *non-maximal* T for all methods with $\alpha > 0$ except for rare cases (such as if there are 2^n equally probable different input patterns). This means that with a given problem there is some need for parameter tuning of the relative weighting factors (with all methods of section 6, only H is predictor-based). The method in the next section avoids this necessity for parameter tuning by replacing the term for variance maximization by a predictor-based term for *conditioned* variance maximization.

7 LOCAL CONDITIONED VARIANCE MAXIMIZATION

This is the author's preferred method for implementing the principle of predictability minimization. It does not (presumably) suffer from the parameter tuning problems involved with the V-term above. It is extremely straight forward and reveals a striking symmetry between opposing forces.

Let us define

$$V_C = \sum_i \sum_p (P_i^p - y_i^p)^2. \tag{5}$$

Recall that P_i^p is supposed to be equal to $E(y_i \mid \{y_k^p, k \neq i\})$, and note that (5) is formally equivalent to the error function of the predictors (equation (1)).

Like in section 6.2 we drop the global invertibility term and redefine the total objective function T to be maximized by the representational modules as

$$T = V_C - \gamma H. \tag{6}$$

I conjecture that if there exists a quasi-binary factorial code for a given pattern ensemble, then among all possible (real-valued or binary) codes T is maximized with a factorial code, *even if* $\gamma = 0$.

If this conjecture is true, then we may forget about the H-term in (9). In this case, *all representational units simply try to maximize the same function that the predictors try to minimize*, namely, V_C. In other words, this generates

a symmetry between two forces that fight each other – one trying to predict, the other one trying to escape the predictions.

Although the conjecture above seems to be intuitively plausible (see the following footnote), it remains unproven for all possible real-valued codes[5]. However, algorithms based solely on V_C-maximization performed well in the experiments to be described below.

8 'NEURAL' IMPLEMENTATIONS

8.1 STARTING WITH REAL-VALUED UNITS

In a realistic application, of course, it is implausible to assume that the errors of all P_i are minimal at all times. After having modified the functions computing the internal representations, the P_i must be trained for some time to assure that they can adapt to the new situation.

Each of the n predictors, the n modules, and the potentially available auto-associator can be implemented as a feed-forward back-propagation network (e.g. [19]). There are two alternating passes:

[5]Note that

$$E[E(y_i \mid \{y_k, k \neq i\}) - y_i]^2 \leq E\,[E(y_i) - y_i]^2\,, \tag{7}$$

and that equality holds only if the following equality always holds: $E(y_i \mid \{y_k, k \neq i\}) = E(y_i)$. Maximizing V_C is equivalent to maximizing $Q_C = \sum_i \sum_p P(x^p)(P_i^p - y_i^p)^2$, where p ranges over all *different* patterns instead of *all* patterns. In the quasi-binary case Q_C can be rewritten as

$$Q_C = \sum_{i\ binary,\ j} P(\alpha_j^i) E(y_i \mid \alpha_j^i)(1 - E(y_i \mid \alpha_j^i)). \tag{8}$$

Here α_j^i is the j-th different event $\{y_k, k \neq i\}$. In the case of a factorial code this becomes

$$\sum_{i\ binary} E(y_i)(1 - E(y_i)). \tag{9}$$

(Interestingly, if we define a very similar objective function $\bar{Q}_C = \sum_i \sum_p P(x^p) \mid P_i^p - y_i^p \mid$ based on absolute values instead of mean squared error then we find that $\bar{Q}_C = 2Q_C$, in the case of a binary factorial code.) The maximization of Q_C encourages quasi-binary codes. Let us consider a quasi-binary factorial code F. It is $Q_C^F = \sum_{i\ binary} E^F(y_i)(1 - E^F(y_i))$, where additional superscripts denote correspondence to a particular code. Every code B for which $\sum_{i\ binary} E^B(y_i)(1 - E^B(y_i)) \leq Q_C^F$ cannot cause greater total error than F, essentially because of equations (7) and (9). But what if $\sum_{i\ binary} E^B(y_i)(1-E^B(y_i)) > Q_C^F$? Intuitively, this seems to imply that the code-capacity exceeds the information contained in the input ensemble, which in turn causes intra-representational redundancy, which in turn causes smaller Q_C^B. I tried to put this formally, using the fact that $E(y_i \mid \alpha_j^i) = \frac{P(\alpha_j^i \mid y_i=1)}{P(\alpha_j^i)} E(y_i)$, but I did not arrive at the desired general conclusion. Perhaps I am just too blind to see the obvious.

Finally, it should be noted that D. Prelinger (personal communication) has made some progress towards proving the conjecture, by observing what happens if (with a given binary code) a single unit changes its response to a single input pattern.

PASS 1 (minimizing prediction error): Repeat for a "sufficient" number of time steps: Select an input pattern x^p according to input probability distribution. Compute all y_i^p. Compute all P_i^p. Train all P_i to predict the y_i^p, using conventional back-propagation. Change only the weights of the P_i during this pass.

PASS 2: Select an input pattern x^p according to its probability. Compute all y_i^p. Compute all P_i^p. If an auto-associator is involved, compute z^p. Change each non-predictor weight w in proportion to $\frac{\partial}{\partial w} T(x^p)$, where $T(x^p)$ is the contribution of the current presentation of x^p to the global objective function T. The weights of the P_i do not change during this pass, but all other weights do change. Note that PASS 2 requires back-propagation of error signals through the predictors (without changing their weights) and then through their $n-1$ input units (which are the output units of the representational modules) down to the weights of the representational modules.

As with all gradient ascent procedures, the method is subject to the problem of local maxima.

It should be mentioned that some or all of the representational modules may share hidden units. The same holds for the predictors. Predictors sharing hidden units, however, will have to be updated sequentially: No representational unit may be used to predict its own activity.

What does the word "sufficient" in PASS 1 mean? To be on the safe side, an "off-line" procedure would sweep a few times through the entire set of training patterns. This solution is perhaps not as appealing as one where computing time is distributed evenly between PASS 2 and PASS 1.

Near-simultaneous updates of the representations and the predictors, however, will introduce on-line effects: Both the predictors and the representational modules will perform gradient descent (or gradient ascent) in changing functions. Given a particular implementation of the basic principle, experiments are needed to find out how much on-line interaction is permittable. With the experiments reported below, on-line learning did not cause major problems.

8.2 BINARY STOCHASTIC UNITS

A binary stochastic unit is a unit that can adopt only two output activations, namely 1 or 0. At any given time, it sums its weighted input, passes it through a squashing function, and interprets the result as the probability of adopting the output value 1.

To what extent can the ideas of the last subsection be transferred to binary stochastic units?

Each predictor P_i can still be trained to model $E(y_i^p \mid \{y_k^p, k \neq i\})$, using essentially the same old error function (the only difference being that the

same pattern may cause different outputs at different presentations, due to the randomness inherent in the stochastic units).

However, care has to be taken if we want to transfer PASS 2 of section 8.1 to modules with binary stochastic units. Only in the linear case is the expectation of the output of the predictors (and the auto encoder) equal to the weighted sum of the expectations of the outputs of the representation units. In the linear case we may apply PASS 2 without much modification, interpreting $o_i(t)$ as the probability that unit i is active at time t.

In the general case, however, back-propagation of errors through random number generators with discrete (in this case binary) outputs does not ensure proper gradient descent.

9 FACTORIAL CODES AND OCCAM'S RAZOR

Any method for finding factorial codes automatically implements Occam's razor which prefers simpler models over more complex ones, where simplicity is defined as the number of storage cells necessary to represent the environment in a factorial fashion. If there are more storage cells than necessary to implement a factorial code, then the independence criterion is met by letting all superfluous units emit constant values in response to all inputs.

It is interesting to note that with non-factorial codes predictability minimization prefers to follow Occam's razor instead of minimizing the sum of bit-entropies *à la* [1]. This can be seen by looking at an example described by Mitchison in the appendix of [1]. This example shows a case where the minimal sum of bit-entropies can be achieved with an expansive local coding of the input. Local representations, however, maximize mutual predictability: With local representations, each unit can always be predicted from all the others. Predictability minimization tries to avoid this by creating non-local, non-expansive codings.

In the case of logistic activation functions, it would be nice if all unused units emitted 0.5 as a constant value in response to all input patterns. Then these units would become more sensitive to new unseen patterns than the other (near-) binary units, due to the maximal derivative of the activation function at 0.5. I define an additional error term M to be

$$M = \delta \sum_i (\bar{y_i} - \frac{1}{2})^2, \tag{10}$$

where δ is another positive constant. This term penalizes mean output values $E(y_i) \neq \frac{1}{2}$. With sufficiently small δ, this term tends to pull unused units with constant activations towards the point in their activation functions where the derivative is maximal. In the lucky case where the number of input patterns is a power of 2 and all patterns are equally probable, M can be zero.

In some experiments it was found that M can also help to escape certain cases of local minima of T.

10 PREDICTABILITY MINIMIZATION AND TIME

Let us now consider the case of input *sequences*. This section describes an entirely local method designed to find unambiguous, non-redundant, reduced sequence descriptions.

The initial state vector $y^p(0)$ is the same for all sequences p. The input at time $t > 0$ of sequence p is the concatenation $x^p(t) \circ y^p(t-1)$ of the input $x^p(t)$ and the last internal state $y^p(t-1)$. The output is $y^p(t)$ itself.

We minimize and maximize essentially the same objective functions as described above. That is, for the i-th module *which now needs recurrent connections to itself and the other modules*, there is again an adaptive predictor P_i *which need not be recurrent.* P_i's input at time t is the concatenation of the outputs $y_k^p(t)$ of all units $k \neq i$. P_i's one-dimensional output $P_i^p(t)$ is trained to equal the expectation of the output y_i, given the outputs of the other units, $E(y_i \mid \{y_k(t), k \neq i\})$, by defining P_i's error function as

$$\sum_p \sum_t (P_i^p(t) - y_i^p(t))^2.$$

In addition, all units are trained to take on values that *maximize*

$$\bar{E} = \sum_t T(t),$$

where $T(t)$ is defined analoguously to the respective stationary cases.

The basic lines of reasoning in sections 5, 6, 7, 8, and 9 hold for this time-processing architecture as well. The only way a unit can protect itself from being predictable from the other units is to store aspects of the input sequences that are independent of aspects stored by the other units. In other words, this method will tend to throw away redundant temporal information much as the systems in [13], [14], and [15]. For computing weight changes, each module looks back only to the last time step. In the on-line case, this implies an *entirely local* learning algorithm. Still, even when there are long time lags, the algorithm theoretically may learn unique representations of *extended* sequences – as can be seen by induction over the length of the longest training sequence:

1. y can learn unique representations of all beginnings of all sequences.

2. Suppose all sequences and sub-sequences with length $< k$ are uniquely represented in y. Then, by looking back only one time step at a time, y can learn unique representations of all sub-sequences with length k.

The argument neglects all on-line effects and possible cross-talk.

10.1 PREDICTABILITY MINIMIZATION AND CHUNKING

The chunking systems described in [13] and [15] are based on the following principle: *If, at a given time step, it is possible to predict the new input from the combination of the last input and the last state, then the new input conveys no non-redundant information and therefore is not incorporated into the new state.* This principle allows for bridging long time lags, because only unexpected events are stored.

This principle can be readily applied to the system above. The only modification necessary is the following:

P_i's input at time t is the concatenation of the outputs $y_k^p(t)$ of all units $k \neq i$ and $y_k^p(t-1) \circ x_k^p(t-1)$.

Maximization of $T(t)$ will force y not to represent and store aspects of the inputs that are predictable from previous inputs. This may speed up learning and reduce the minimal necessary dimension of y.

In noisy environments y will prefer to store events that are not noise-like. This is opposed to the history compression technique of [13] which considers noise as having high information.

11 PREDICTABILITY MINIMIZATION AND GOAL-DIRECTED LEARNING

Any reinforcement or supervised learning system can receive $y^p(t)$ or $x^p(t) \circ y^p(t)$ as an input and can be trained in the standard way to complete the task at hand. For instance, a supervised feed-forward net can be trained to emit desired outputs whenever an external teacher so requires. A reinforcement learner with a non-Markovian interface to its environment [12] will be potentially able to build a Markovian interface with the method of section 10.

In some cases we want to make sure that a goal-directed learning system does not only depend on the internal representations generated by the process of predictability minimization (sometimes these representations might carry less information than the input, due to local minima encountered during their formation). In this case we may simply use the conventional input as an additional input for the goal-directed learner, thus reducing the role of the representational modules to merely an assisting role.

Note that goal-directed learning algorithms, based on unique history representations learned by the local, unsupervised systems of section 10 are entirely local, also. This represents an important potential of the method, which has not yet been experimentally exploited.

12 EXPERIMENTS

All experiments described below were based on T defined as in section 7, with $\gamma = 0$. In other words, the representational units try to maximize the same objective function V_C that the predictors try to minimize. All representational modules and predictors were implemented as 3-layer back-propagation networks. All hidden and output units used a logistic activation function and were connected to a bias-unit with constant activation. Parameters such as learning rates and number of hidden units were not chosen to optimize performance – there was no systematic attempt to improve learning speed.

An *on-line* system based on section 8 was implemented and tested Daniel Prelinger (a student at the Technische Universität München). Jeff Rink (a student at the University of Colorado at Boulder) independently implemented and tested an *off-line* version. The purpose of this section is not to compare off-line and on-line versions but to show that both can lead to satisfactory results.

With the off-line version, PASS 1 was slightly modified. There were 5 consecutive 'epochs' during which each pattern of the whole training ensemble was presented to the system. Weight changes for the predictors were executed after each epoch. With PASS 2 there was 1 epoch after which weight changes for the representational units were executed. The learning rates of all predictors were 0.2, the learning rates of all representational modules were 0.3. Within the representational modules there were direct connections from the input units to the representational units. There were no direct input-output connections in the predictors. Weights were initialized randomly between -0.5 and 0.5.

With the on-line system, at any given time, the same input pattern was used in both PASS 1 and PASS 2; the "sufficient" number of time steps for PASS 1 was assumed to be 1. There were direct connections between the input and output units of the representational modules and the predictors. Prelinger introduced an additional method for escaping certain cases of local minima: The activations of the output units of the representational modules were forced to lie between 0.05 and 0.95. With pattern p each predictor P_i locally tried to minimize

$$(P_i^p - (y_i^p + 0.1(0.5 - y_i^p))^2 \tag{11}$$

instead of simply minimizing $(P_i^p - y_i^p)^2$. The effect of this was that the predictors were trained to 'overshoot' a little bit, which caused the expectations of the representational units to tend to move away from the 'corners' of their range. With the experiments reported below, it was found that during PASS 2 no back-propagation through the predictors' input units down to the representational units was necessary. The learning rates of all predictors were 1.0, the learning rates of all representational modules were 0.1. Weights were initialized randomly between -0.3 and 0.3.

With all experiments, a unit was considered to be binary if the absolute difference between its possible activations and either the maximal or the minimal

activation permitted by its activation function never exceeded 0.05. A code was considered to be quasi-binary if each unit was either binary or the absolute difference between its possible activations and its expectation never exceeded 0.05.

The next subsections list some selected experiments with both the on-line and the off-line method. In what follows, the term 'local input representation' means that there are $dim(x)$ different binary inputs, each with only one non-zero bit. The term 'distributed input representation' means that there are $2^{dim(x)}$ different binary inputs.

12.1 UNIFORMLY DISTRIBUTED INPUTS

With the experiments described in this subsection there are $2^{dim(y)}$ uniformly distributed input patterns. This means that the desired factorial codes are the full binary codes.

Experiment 1: off-line, $dim(y) = 2$, $dim(x) = 4$, local input representation, 3 hidden units per predictor, 4 hidden units shared among the representational modules. 10 test runs with 20,000 epochs for the representational modules were conducted. In 8 cases this was sufficient to find a binary factorial code.

Experiment 2: on-line, $dim(y) = 2$, $dim(x) = 2$, distributed input representation, 2 hidden units per predictor, 4 hidden units shared among the representational modules. 10 test runs were conducted. Less than 3,000 pattern presentations (equivalent to ca. 700 epochs) were always sufficient to find a binary factorial code.

Experiment 3: on-line, $dim(y) = 3$, $dim(x) = 3$, distributed input representation, 4 hidden units per predictor, 6 hidden units shared among the representational modules. 10 test runs were conducted. Less than 20,000 pattern presentations (equivalent to ca. 2,000 epochs) were always sufficient to find a binary factorial code. In fact, in most cases less than 4,000 pattern presentations (ca. 500 epochs) were sufficient.

Experiment 4: off-line, $dim(y) = 4$, $dim(x) = 16$, local input representation, 3 hidden units per predictor, 16 hidden units shared among the representational modules. 10 test runs with 20,000 epochs for the representational modules were conducted. In 1 case the system found an invertible factorial code. In 4 cases it created only 15 different output patterns in response to the 16 input patterns. In 3 cases it created only 13 different ouput patterns. In 2 cases it created only 12 different ouput patterns.

Experiment 5: on-line, $dim(y) = 4$, $dim(x) = 4$, distributed input representation (16 patterns), 6 hidden units per predictor, 8 hidden units shared among the representational modules. 10 test runs were conducted. In all cases but one the system found a factorial code within less than 4,000 pattern presentations (corresponding to less than 300 epochs).

12.2 OCCAM'S RAZOR AT WORK

The experiments in this section are meant to verify the effectiveness of Occam's razor mentioned in section 9.

Experiment 1: off-line, $dim(y) = 3$, $dim(x) = 4$, local input representation, 3 hidden units per predictor, 4 hidden units shared among the representational modules. 10 test runs with 10,000 epochs for the representational modules were conducted. In 7 cases the system found a binary factorial code: In the end, one of the output units always emitted a constant value. In the remaining 3 cases, the code was at least binary and invertible.

Experiment 2: off-line, $dim(y) = 4$, $dim(x) = 4$, local input representation, 3 hidden units per predictor, 4 hidden units shared among the representational modules. 10 test runs with 10,000 epochs for the representational modules were conducted. In 5 cases the system found a binary factorial code: In the end, two of the output units always emitted a constant value. In the remaining cases, the code did not use the minimal number of output units but was at least binary and invertible.

Experiment 3: on-line, $dim(y) = 3$, $dim(x) = 2$, distributed input representation, 2 hidden units per predictor, 4 hidden units shared among the representational modules. 10 test runs with 100,000 pattern presentations (corresponding to 25,000 epochs) were conducted. This was always sufficient for finding a quasi-binary factorial code. After training, the unused unit always emitted a value close to 0.5. This is because minimization of (11) has an effect similar to the one caused by minimization of (10).

Experiment 4: on-line, $dim(y) = 4$, $dim(x) = 2$, distributed input representation, 2 hidden units per predictor, 4 hidden units shared among the representational modules. 10 test runs with 250,000 pattern presentations were conducted. This was sufficient to always find a quasi-binary factorial code: In the end, two of the output units always emitted a constant value. In 7 out of 10 cases, less than 100,000 pattern presentations (corresponding to 25,000 epochs) were necessary.

12.3 NON-UNIFORMLY DISTRIBUTED INPUTS

The input ensemble considered in this subsection consists of four different patterns denoted by x_a, x_b, x_c, and x_d, respectively. The probabilities of the patterns were

$$P(x^a) = \frac{1}{9}, P(x^b) = \frac{2}{9}, P(x^c) = \frac{2}{9}, P(x^d) = \frac{4}{9}.$$

This ensemble allows for binary factorial codes, one of which is denoted by the following

code F: $y^a = 00$, $y^b = 01$, $y^c = 10$, $y^d = 11$.

With code F, the total objective function V_C^F becomes $V_C^F = \frac{4}{9}$. A non-factorial but invertible (information-preserving) code is given by

code B: $y^a = 01$, $y^b = 00$, $y^c = 10$, $y^d = 11$.

With code B, $V_C^B = \frac{19}{45}$, which is only $\frac{1}{45}$ below V_C^F. This already indicates that certain local maxima of the internal state's objective function may be very close to the global maxima. This is reflected in the experiment described next.

Experiment 1: off-line, $dim(y) = 2$, $dim(x) = 4$, local input representation, 3 hidden units per predictor, 4 hidden units shared among the representational modules. 10 test runs with 10,000 epochs for the representational modules were conducted. Here one epoch consisted of the presentation of 9 patterns – x^a was presented once, x^b was presented twice, x^c was presented twice, x^d was presented four times.

In 5 cases, the system found a global maximum corresponding to a factorial code. In 2 remaining cases the code was invertible, and in 3 remaining cases it was not. Similar results were obtained with the on-line method.

Experiment 2 (Occam's Razor): Like experiment 1, but with $dim(y) = 3$. In 2 out of 10 test runs the system found a factorial code (including one unused unit). In the remaining 8 test runs it found always invertible codes. This reflects a trade-off between redundancy and invertibility: Superfluous degrees of freedom among the representational units increase the probability that an invertible (information-preserving) code is found, but decrease the probability of finding a factorial code.

12.4 EXPERIMENTS WITH STOCHASTIC UNITS

Predictability minimization with stochastic representational units (section 8.1) led to similar results as predictability minimization with deterministic units. An approximation to proper gradient descent was used: It was possible to obtain satisfactory results without propagating through the input units of the predictors down to the representational modules. Here is just one example:

Experiment: on-line, $dim(y) = 3$, $dim(x) = 2$, distributed input representation, 2 hidden units per predictor, 4 hidden units shared among the representational modules. A unit was considered to be 'near-deterministic' if its probability of being switched on was either less than 0.05 or higher than 0.95 for any given input pattern. 10 test runs with 100,000 pattern presentations (corresponding to 25,000 epochs) were conducted. This was always sufficient for finding a 'near-determininstic' binary factorial code. After training, an unused unit was either always switched on or always switched off with high probability.

12.5 RECURRENT SYSTEMS

On-line variants of the system described in section 10 were implemented by Daniel Prelinger. Preliminary experiments were conducted with the resulting recurrent systems. These experiments demonstrated that entirely local methods based on section 10 allow for learning unique representations of all subsequences

of non-trivial sequences (like a sequence consisting of 8 consecutive presentations of the same input pattern). It is intended to elaborate on sequence learning by predictability minimization in a separate paper.

13 CONCLUDING REMARKS, OUTLOOK

Although gradient methods based on the principle of predictability minimization can not always be expected to find factorial codes – due to local minima and the possibility that the problem of finding factorial codes may be NP-hard – they have a potential for removing kinds of redundancy that previous linear methods were not able to remove. This holds even if the conjecture in section 7 ultimately proves to be false.

In many realistic cases, approximations of non-redundant codes should be satisfactory. It remains to be seen whether predictability minimization can be useful to find *nearly* non-redundant representations of real-world inputs, such as retinal images. In ongoing research it is intended to apply the methods described herein to problems of unsupervised image segmentation (in the case of multiple objects), as well as to unsupervised sequence segmentation.

There is a relationship of predictability minimization to more conventional 'competitive' learning schemes: In a certain sense, units compete for representing certain 'abstract' transformations of the environmental input. The competition is not based on a physical 'neighbourhood' criterion but on mutual predictability. Unlike with most previous schemes based on 'winner-take-all' networks, output representations formed by predictability minimization may have multiple 'winners', as long as they stand for independent features extracted from the environment.

One might speculate about whether the brain uses a similar principle based on 'representational neurons' trying to escape the predictions of 'predictor neurons'. Since the principle allows for entirely local sequence learning algorithms (in space and time), it seems to be biologically more plausible than methods such as 'back-propagation through time' etc.

Predictability minimization also might be useful in cases where different representational modules see *different* inputs. For instance, if a binary feature of one input 'patch' is predictable from features extracted from neighbouring 'patches', then representations formed by predictability minimization would tend to not use additional storage cells for representing the feature.

The paper at hand adopts a general viewpoint on predictability minimization by focussing on the general case of non-linear nets. In some cases, however, it might be desireable to restrict the computational power of the representational modules and/or the predictors by making them linear or semi-linear. For instance, a hierarchical system with successive stages of computationally limited modules may be useful for reflecting the hierarchical structure of certain environments.

14 ACKNOWLEDGEMENTS

Thanks to Daniel Prelinger and Jeff Rink for conducting the experiments. Thanks to Daniel Prelinger, Mike Mozer, Radford Neal, Luis Almeida, Sue Becker, Rich Zemel, Peter Dayan, and Clayton McMillan for valuable comments and suggestions which helped to improve the paper. This research was supported by NSF PYI award IRI–9058450, grant 90–21 from the James S. McDonnell Foundation, and DEC external research grant 1250 to Michael C. Mozer.

References

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

[2] S. Becker. Unsupervised learning procedures for neural networks. *International Journal of Neural Systems*, 2(1 & 2):17–33, 1991.

[3] S. Becker and G. E. Hinton. Spatial coherence as an internal teacher for a neural network. Technical Report CRG-TR-89-7, Department of Computer Science, University of Toronto, Ontario, 1989.

[4] G.J. Chaitin. A theory of program size formally identical to information theory. *Journal of the ACM*, 22:329–340, 1965.

[5] F. Földiák. Forming sparse representations by local anti-hebbian learning. *Biological Cybernetics*, 64:165–170, 1990.

[6] A.N. Kolmogorov. Three approaches to the quantitative definition of information. *Problems of Information Transmission*, 1:1–11, 1965.

[7] R. Linsker. Self-organization in a perceptual network. *IEEE Computer*, 21:105–117, 1988.

[8] E. Oja. Neural networks, principal components, and subspaces. *International Journal of Neural Systems*, 1(1):61–68, 1989.

[9] B. A. Pearlmutter and G. E. Hinton. G-maximization: An unsupervised learning procedure for discovering regularities. In J. S. Denker, editor, *Neural Networks for Computing: American Institute of Physics Conference Proceedings 151*, volume 2, pages 333–338, 1986.

[10] J. Rubner and K. Schulten. Development of feature detectors by self-organization: A network model. *Biological Cybernetics*, 62:193–199, 1990.

[11] T. D. Sanger. An optimality principle for unsupervised learning. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems 1*, pages 11–19. San Mateo, CA: Morgan Kaufmann, 1989.

[12] J. H. Schmidhuber. Reinforcement learning in markovian and non-markovian environments. In D. S. Lippman, J. E. Moody, and D. S. Touretzky, editors, *Advances in Neural Information Processing Systems 3*, pages 500–506. San Mateo, CA: Morgan Kaufmann, 1991.

[13] J. H. Schmidhuber. Learning complex, extended sequences using the principle of history compression. Accepted for publication in *Neural Computation*, 1992.

[14] J. H. Schmidhuber. Learning unambiguous reduced sequence descriptions. In J. E. Moody, S. J. Hanson, and R. P. Lippman, editors, *Advances in Neural Information Processing Systems 4, to appear.* San Mateo, CA: Morgan Kaufmann, 1992.

[15] J. H. Schmidhuber, M. C. Mozer, D. Prelinger, R. Blumenthal, and D. Mathis. Continuous history compression. Technical report, Dept. of Comp. Sci., University of Colorado at Boulder, 1992.

[16] C. E. Shannon. A mathematical theory of communication (part III). *Bell System Technical Journal*, XXVII:623–656, 1948.

[17] C. E. Shannon. A mathematical theory of communication (parts I and II). *Bell System Technical Journal*, XXVII:379–423, 1948.

[18] F. M. Silva and L. B. Almeida. A distributed decorrelation algorithm. In Erol Gelenbe, editor, *Neural Networks, Advances and Applications.* North-Holland, 1991, to appear.

[19] P. J. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences.* PhD thesis, Harvard University, 1974.

[20] R. S. Zemel and G. E. Hinton. Discovering viewpoint-invariant relationships that characterize objects. In D. S. Lippman, J. E. Moody, and D. S. Touretzky, editors, *Advances in Neural Information Processing Systems 3*, pages 299–305. San Mateo, CA: Morgan Kaufmann, 1991.