LEARNING FACTORIAL CODES BY PREDICTABILITY MINIMIZATION (compact version of TR CU-CS-565-91)

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

December 18, 1991

Abstract

I propose 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. Unlike previous methods the novel principle has a potential for removing not only linear but also non-linear output redundancy. Methods for finding factorial codes automatically embed Occam's razor for finding codes using a minimal number of units. Illustrative experiments show that algorithms based on the principle of predictability minimization are practically feasible. The final part of this paper describes an entirely local algorithm that has a potential for learning unique representations of extended sequences.

1 INTRODUCTION

Note: This is a more compact and less tedious version of [9].

Consider a perceptual system being exposed to an unknown 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.

An ambitious and potentially powerful objective of unsupervised learning 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[1]. 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 particular input is simply the product of the probabilities of the corresponding code symbols.

Among the advantages of factorial 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 factorial 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 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

[1]The G-Max algorithm [6] 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.

to implement a factorial code, then the independence criterion is met by letting all superfluous units emit constant values in response to all inputs. This implies storage efficiency, as well as a potential for better generalization capabilities.

Barlow et al. do not present an efficient general method for finding factorial codes (but they propose a few sequential 'non-neural' heuristic methods). Existing 'neural' methods for decreasing output redundancy (e.g. [4][15][5][8][3][7][4][13]) are restricted to the linear case and do not aim at the ambitious general goal of statistical independence. In addition, some of these methods require Gaussian assumptions about the input and output signals, as well as the explicit calculation of the derivative of the determinant of the output covariance matrix [12].

The main contribution of this paper is a simple but general 'neural' architecture (plus the appropriate objective functions) for finding factorial codes.

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 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 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.

3 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[2]

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

With the help of the n predictors one can define various objective functions for the representational modules to enforce the 3 criteria listed above (see section 4 and section 5). Common to these methods 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.

[2] 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.

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.

4 OBJECTIVE FUNCTIONS FOR THE THREE CRITERIA

Sections 4.1, 4.2, 4.3 provide objective functions for the three criteria. Sections 4.4, 4.5, 4.6 describe various combinations of these objective functions. Section 4.7 hints at a parameter tuning problem. A way to overcome it is presented in section 5.

4.1 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:

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

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 [4]. 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.

4.2 AN OBJECTIVE FUNCTION FOR THE BINARY CRITERION

A well-known 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.

4.3 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

$$I = \sum_p (z^p - x^p)^T (z^p - x^p). \tag{3}$$

4.4 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.

4.5 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 [4].

4.6 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. 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.

4.7 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. 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.

5 LOCAL CONDITIONED VARIANCE MAXIMIZATION

This is the author's preferred method for implementing the principle of predictability minimization. It does not 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 objective function of the predictors (equation (1)).

Like in section 4.6 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 quasi-binary 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.

The conjecture remains unproven for the general case. The long version of this paper [9], however, mathematically justifies the conjecture for certain special cases and provides some intuitive justification for the general case. In addition, algorithms based solely on V_C-maximization performed well in the experiments to be described below.

6 'NEURAL' IMPLEMENTATION

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. [14]). There are two alternating passes:

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 the input probability distribution. 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.

7 EXPERIMENTS

All the illustrative experiments described below were based on T defined as in section 5, 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.

Daniel Prelinger and Jeff Rink implemented *on-line* and *off-line* systems based on section 6 (see details in [9]). 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. Learning rates around 0.3 for both the predictors and representational modules led to satisfactory results.

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. An additional modification for escaping certain cases of local minima was introduced (see [9]). The learning rates of all predictors were 1.0, the learning rates of all representational modules were 0.1 (therefore the predictors 'learned faster' than the representational modules).

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.

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.

7.1 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. In the case of a factorial code all predictors have to emit 0.5 in response to every input pattern.

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: 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 a near-factorial code with 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 4: 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).

7.2 OCCAM'S RAZOR AT WORK

The experiments in this section are meant to verify the effectiveness of Occam's razor mentioned in the introduction. It is interesting to note that with non-factorial codes predictability minimization prefers to reduce the number of used units 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.

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, $dim(y) = 4$, $dim(x) = 2$, 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.

7.3 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.

8 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 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 [10] and [11]. 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.

On-line variants of the system described above were implemented by Daniel Prelinger. Preliminary experiments were conducted with the resulting recurrent systems. These experiments demonstrated that there are entirely local sequence learning methods that 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.

9 CONCLUDING REMARKS, OUTLOOK

Although gradient methods based on 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 5 ultimately proves to be false.

In many realistic cases, however, 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. 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.

Among the additional topics covered by the longer version of this report [9] are: General remarks on unsupervised learning and information-theoretic aspects, a 'neural' approach to finding binary factorial codes *without* using predictors, implementations of predictability minimization using binary stochastic units, the relationship of predictability minimization to recent sequence chunking methods, combinations of goal-directed learning and unsupervised predictability minimization.

10 ACKNOWLEDGEMENTS

Thanks to Daniel Prelinger and Jeff Rink for conducting the experiments. Thanks to Daniel Prelinger, Mike Mozer, Radford Neal, Luis Almeida, Peter Dayan, Sue Becker, Rich Zemel, 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] F. Földiák. Forming sparse representations by local anti-hebbian learning. *Biological Cybernetics*, 64:165–170, 1990.

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

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

[6] 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.

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

[8] 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.

[9] J. H. Schmidhuber. Learning factorial codes by predictability minimization. Technical Report CU-CS-565-91, Dept. of Comp. Sci., University of Colorado at Boulder, December 1991.

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

[11] 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.

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

[13] 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.

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

[15] 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.