DISCOVERING PREDICTABLE CLASSIFICATIONS
Technical Report CU-CS-626-92

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

Daniel Prelinger
Institut für Informatik
Technische Universität München
Arcisstr. 21, 8000 München 2, Germany
prelinge@informatik.tu-muenchen.de

November 11, 1992

Abstract

Prediction problems are among the most common learning problems for neural networks (e.g. in the context of time series prediction, control, etc.). With many such problems, however, perfect prediction is inherently impossible. For such cases we present novel unsupervised systems that learn to *classify* patterns such that the classifications are predictable while still being as specific as possible. The approach can be related to the IMAX method of Hinton, Becker and Zemel (1989, 1991). Experiments include Becker's and Hinton's stereo task, which can be solved more readily by our system.

1 MOTIVATION AND BASIC APPROACH

Many neural net systems (e.g. for control, time series prediction, etc.) rely on adaptive submodules for learning to predict patterns from other patterns. Perfect prediction, however, is often inherently impossible. In this paper we study the problem of finding pattern *classifications* such that the classes are predictable, while still being as specific as possible.

To grasp the basic idea, let us discuss several examples.

Example 1: Hearing the first two words of a sentence "Henrietta eats ..." allows you to infer that the third word probably indicates something to eat but you cannot tell what. The *class* of the third word is predictable from the previous words – the particular instance of the class is not. The class "food" is not only predictable but also non-trivial and specific in the sense that it does not include *everything* – "John", for instance, is not an instance of "food".

The problem is to classify patterns from a set of training examples such that the classes are both predictable *and* not too general. A general solution to this problem would be useful for discovering higher level structure in sentences generated by unknown grammars, for instance. Another application would be the unsupervised classification of different pattern instances belonging to the same class, as will be seen in the next example.

Example 2 (stereo task; due to Becker and Hinton, 1989): There are two binary images called the 'left' image and the 'right' image. Each image consists of two 'strips' – each strip being a binary vector. The right image is purely random. The left image is generated from the right image by choosing, at random, a single global shift to be applied to each strip of the right image. An input pattern is generated by concatenating a strip from the right image with the corresponding strip from the left image. "So the input can be interpreted as a fronto-parallel surface at an integer depth. The only local property that is invariant across space is the depth (i.e. shift)." (Becker and Hinton, 1989). With a given pair of different input patterns, the task is to extract a non-trivial classification of whatever is common to both patterns – which happens to be the stereoscopic shift.

Example 1 is an instance of the so-called *asymmetric* case: There we are interested in a predictable non-trivial classification of one pattern (the third word), given some other patterns (the previous words). Example 2 is an instance of the so-called *symmetric* case: There we are interested in the non-trivial common properties of two patterns from the same class.

In its simplest form, our basic approach to unsupervised discovery of predictable classifications is based on two neural networks called T_1 and T_2. Both can be implemented as standard back-prop networks (Werbos, 1974)(LeCun, 1985)(Parker, 1985)(Rumelhart et al., 1986). With a given pair of input patterns, T_1 sees the first pattern, T_2 sees the second pattern. Let us first focus on the asymmetric case. For instance, with the example 1 above T_1 may see a representation of the words "Henrietta eats", while T_2 may see a representation of the word "vegetables". T_2's task is to classify its input. T_1's task is not to predict T_2's raw environmental input but to predict T_2's output instead.

Both networks have q output units. Let $p \in \{1, \ldots, m\}$ index the input patterns. T_2 produces as an output the classification $y^{p,2} \in [0, \ldots, 1]^q$ in response to an input vector $x^{p,2}$. T_1's output in response to its input vector $x^{p,1}$ is the prediction $y^{p,1} \in [0, \ldots, 1]^q$ of the current classification $y^{p,2}$ emitted by T_2.

We have two conflicting goals which in general are not simultaneously satisfiable: (A) All predictions $y^{p,1}$ should match the corresponding classifications $y^{p,2}$. (B) The $y^{p,2}$ should be discriminative – different inputs $x^{p,2}$ should lead to different classifications $y^{p,2}$.

We express the trade-off between (A) and (B) by means of two opposing costs.

(A) is expressed by an error term M (for '*M*atch'):

$$M = \sum_{p=1}^{m} ||y^{p,1} - y^{p,2}||^2. \tag{1}$$

Here $||v||$ denotes the Euclidean norm.

(B) is enforced by an additional error term D_2 (for '*D*iscrimination') to be minimized by T_2 only. D_2 will be designed to encourage significant Euclidean distance between classifications of different input patterns. D_2 can be defined in more than one reasonable way. The next section will list four *alternative* possibilities with mutual advantages and disadvantages. These alternatives include (a) a novel method for constrained variance maximization, (b) auto-encoders, and (c) a recent technique called "predictability minimization" (Schmidhuber, 1992).

The total error to be minimized by T_2 is

$$\epsilon M + (1 - \epsilon) D_2, \tag{2}$$

where $0 < \epsilon < 1$ determines the relative weighting of the opposing error terms. In the asymmetric case, the total error to be minimized by T_1 is just

$$\epsilon M. \tag{3}$$

The error functions are minimized by gradient descent. This forces the predictions and classifications to be more like each other, while at the same time forcing the classifications not to be too general but to tell something about the current input. The procedure is *unsupervised* in the sense that no teacher is required to tell T_2 how to classify its inputs.

With the symmetric case (see example 2 above), both T_1 and T_2 are naturally treated in a symmetric manner. They share the goal of uniquely representing as many of their input patterns as possible – under the constraint of emitting equal (and therefore mutually predictable) classifications in response to a pair of input patterns. Such classifications represent whatever abstract properties are common to both patterns of a typical pair. For handling such symmetric tasks in a natural manner, we only slightly modify T_1's error function for the asymmetric case, by introducing an extra 'discriminating' error term D_1 for T_1. Now both $T_l, l = 1, 2$ minimize

$$\epsilon M + (1 - \epsilon) D_l, \tag{4}$$

where alternative possibilities for defining the D_l will be defined in the next section. Figure 1 shows a system based on (4) and a particular implementation of D_l (to be explained in section 2.4).

The assumption behind our basic approach is that a prediction that closely (in the Euclidean sense) matches the corresponding classification is a nearly accurate prediction. Likewise, two very similar (in the Euclidean sense) classifications emitted by a particular network are assumed to have very similar 'meaning'. It should be mentioned that, in theory, even the slightest differences between classifications of different patterns are sufficient to convey all (maximal) Shannon information about the patterns (assuming noise-free data). But then close matches between predictions and classifications could not necessarily be interpreted as accurate predictions. The alternative designs of D_l (to be described below), however, will have the tendency to emphasize differences between different classifications by increasing the Euclidean distance between them (sometimes under certain constraints, see section 2). There is another reason why this is a reasonable thing to do: In a typical application, a classifier will function as a pre-processor for some higher-level network. We usually do not want higher-level input representations with different 'meaning' to be separated by tiny Euclidean distance.

Weight sharing. If both T_1 and T_2 are supposed to provide the same outputs in response to the same inputs (this holds for the stereo task but does not hold in the general case) then we need only one set of weights for both classifiers. This reduces the number of free parameters (this may improve generalization performance).

Outline. The current section motivated and explained our basic approach. Section 2 presents various instances of the basic approach (based on various possibilities for defining D_l). Section 3 mentions previous related work. Section 4 presents illustrative experiments and experimentally demonstrates advantages of our approach.

2 ALTERNATIVE DEFINITIONS OF D_l

This section lists four *different* approaches for defining D_l, the term which enforces non-trivial discriminative classifications. Section 2.1 presents a novel method that encourages locally represented classes (like with winner-take-all networks). The advantage of this method is that the class representations are orthogonal to each other and easy to understand, its disadvantage is the low representation capacity. In contrast, the remaining methods can generate distributed class representations. Section 2.2 defines D_l with the help of auto-encoders. One advantage of this straight-forward method is that it is easy to implement. Section 2.3 mentions the Infomax approach for defining D_l and explains why we do not pursue this approach.

Section 2.4 finally defines D_l by the recent method for *predictability minimization* (Schmidhuber, 1992). An advantage of this method is its potential for creating distributed class representations with statistically independent components.

2.1 MAXIMIZING CONSTRAINED OUTPUT VARIANCE

With this alternative for defining D_l, the goal is to obtain local class representations. One advantage of local class representations is that they are orthogonal to each other a well as easy to understand. Their disadvantage is the low representation capacity.

We write

$$D_l = -\frac{1}{2}\sum_p \sum_i (y_i^{p,l} - \bar{y_i}^l)^2 + \frac{\lambda}{2}\sum_i [\frac{1}{q} - \bar{y_i}^l]^2 \tag{5}$$

and minimize D_l subject to the constraint

$$\forall p : \sum_i y_i^{p,l} = 1. \tag{6}$$

Here, as well as throughout the remainder of this paper, subscripts of symbols denoting vectors denote vector components: v_i denotes the i-th element of some vector v. λ is a positive constant, and $\bar{y_i}^l$ denotes the mean of the i-th output unit of T_l. It is possible to show that the first term on the right hand side of (5) is maximized subject to (6) if each input pattern is locally represented (just like with winner-take-all networks) by exactly one corner of the q-dimensional hypercube spanned by the possible output vectors, if there are sufficient output units (Prelinger, 1992) [1]. Maximizing the second negative term encourages each local class representation to become active in response to only $\frac{1}{q}$-th of all possible input patterns.

Constraint (6) is enforced by setting

$$y_i^{p,j} = \frac{u_i^{p,j}}{\sum_i u_i^{p,j}},$$

where $u^{p,j}$ is the activation vector (in response to $x^{p,j}$) of a q-dimensional layer of hidden units of T_j which can be considered as its unnormalized output layer.

This novel method is easy to implement – it achieves an effect similar to the one of the recent entropy-based method by Bridle and MacKay (1992).

2.2 AUTO-ENCODERS

Another simple alternative for defining D_l is based on auto-encoders. Auto-encoders are easy to implement and allow us to obtain distributed class representations (as opposed to local representations, see section 2.1). With pattern p and classifier T_l a reconstructor module A_l (another back-prop network) receives $y^{p,l}$ as an input. The combination of T_l and A_l functions as an auto-encoder. The auto-encoder is trained to emit the reconstruction $h^{p,l}$ of T_l's external input $x^{p,l}$, thus forcing $y^{p,l}$ to tell something about $x^{p,l}$. D_l is defined as

$$D_l = \frac{1}{2}\sum_p ||h^{p,l} - x^{p,l}||^2. \tag{7}$$

2.3 INFOMAX

Following Linsker's Infomax approach (Linsker, 1988), we might think of defining $-D_l$ explicitly as the mutual information between the inputs and the outputs of T_l.

We did not use Infomax methods in our experiments for the following reasons:

[1] Simply maximizing the variance of the output units without obeying constraint (6) will not necessarily maximize the number of different classifications. Example: Consider a set of four different four-dimensional input patterns 1000, 0100, 0010, 0001. Suppose the classifier maps the first two input patterns to the four-dimensional output pattern 1100 and the other two to 0011. This will yield a variance of 4. A 'more discriminative' response would map each pattern to itself, but this will yield a lower variance of 3.

(a) There is no efficient and general method for maximizing mutual information. (b) With our basic approach from section 1, Infomax makes sense only in situations where it automatically enforces high variance of the outputs of the T_l (possibly under certain constraints). This holds for the simplifying Gaussian noise models studied by Linsker, but it does not hold for the general case. (c) Even under appropriate Gaussian assumptions, with more than one-dimensional representations, Infomax implies maximization of functions of the determinant DET of the covariance matrix of the output activations (Shannon, 1948). In a small application, Linsker explicitly calculated DET's derivatives. In general, however, this is clumsy.

2.4 PREDICTABILITY MINIMIZATION

Schmidhuber (1992) shows how D_l can be defined with the help of *intra-representational* adaptive predictors that try to predict each output unit of some T_l from its remaining output units, while each output unit in turn tries to extract properties of the environment that allow it to *escape* predictability. This was called the *principle of predictability minimization.* This principle encourages each output unit of T_l to represent environmental properties that are statistically independent from environmental properties represented by the remaining output units. The procedure aims at generating binary 'factorial codes' (Barlow et al., 1989). It is our preferred method, because (unlike the methods used by Linsker (1988), Becker and Hinton (1989), and Zemel and Hinton (1991)) it has a potential for removing even non-linear statistical dependencies[2] among the output units of some classifier.

Let us define

$$\bar{D}_l = -\frac{1}{2}\sum_i (s_i^{p,l} - y_i^{p,l})^2, \tag{8}$$

where the $s_i^{p,l}$ are the outputs of S_l^i, the i-th additional so-called *intra-representational* predictor network of T_l (one such additional predictor network is required for each output unit of T_l). S_l^i is trained to predict $y_i^{p,l}$ from $\{y_k^{p,l}, \;\; k \neq i\}$.

To encourage even distributions in output space, we slightly modify $\bar{D}_l$ by introducing a term similar to the one in equation (5), subsection 2.1 and obtain

$$D_l = -\frac{1}{2}\sum_i (s_i^{p,l} - y_i^{p,l})^2 + \frac{\lambda}{2}\sum_i (0.5 - \bar{y}_i{}^l)^2. \tag{9}$$

3 PREVIOUS WORK

Becker and Hinton (1989) solve symmetric problems (like the one of example 2, see section 1) by maximizing the mutual information between the outputs of T_1 and T_2 (IMAX). This corresponds to the notion of finding mutually predictable yet informative input transformations.

Note that our approach does not only enforce mutual predictability but also equality of $y^{p,1}$ and $y^{p,2}$. This does not at all affect the generality of the approach. Note that one could introduce additional 'predictor networks' – one for learning to predict $y^{p,2}$ from $y^{p,1}$ and another one for learning to predict $y^{p,1}$ from $y^{p,2}$. Then one could design error functions enforcing mutual predictability (instead of using the essentially equivalent error function M used in this paper). However, this would not increase the power of the approach but would only introduce unnecessary additional complexity. In fact, one advantage of our simple approach is that it makes it trivial to decide whether the outputs of both networks essentially represent the same thing.

One variation of the IMAX approach assumes that T_1 and T_2 have *single binary* probabilistic output units. In another variation, T_1 and T_2 have single *real-valued* output units. The latter case, however, requires certain (not always realistic) Gaussian assumptions about the input and output signals (see also section 2.3 on Infomax).

In the case of *vector-valued* output representations, Zemel and Hinton (1991) again make simplifying Gaussian assumptions and maximize functions of the determinant D of the $q \times q$-covariance matrices (DETMAX) of the output activations (Shannon, 1948) (see again section 2.3). In addition, DETMAX

[2] Steve Nowlan has described an alternative non-predictor based approach for finding non-redundant codes (Nowlan, 1988).

can remove only *linear* redundancy among the output units. (It should be mentioned, however, that with Zemel's and Hinton's approach the outputs may be non-linear functions of the inputs).

The following section includes an experiment that compares IMAX to our approach.

4 ILLUSTRATIVE EXPERIMENTS

All networks used below were trained by back-propagation (Werbos, 1974) (LeCun, 1985) (Parker, 1985) (Rumelhart et al., 1986) – in all cases we used the activation dynamics of (Rumelhart et al., 1986), as well as 'on-line' learning: Weight changes took place immediately after each presentation of some randomly chosen input pattern. Approximations of mean values $\bar{y}_i^l$ were updated by the formula

$$\hat{y}_i^l \leftarrow 0.95\hat{y}_i^l + 0.05y_i^l,$$

where $\hat{y}_i^l$ is the approximation of $\bar{y}_i^l$ after observing the current input pattern y^l. In the case of local output representations (section 2.1) $\hat{y}_i^l$ was initially set to $\frac{1}{q}$, otherwise $\hat{y}_i^l$ was initially set to 0.5.

4.1 FINDING PREDICTABLE LOCAL CLASS REPRESENTATIONS

Motivated by example 1, section 1, a simple 'language generator' emitting 'sentences' consisting of two symbols was defined. Each alternative in the following grammar occurred with equal probability. S is the start symbol, A, B, C, D are non-terminals, and a, a^1, a^2, b, b^1, b^2, c, c^1, c^2, d, d^1, d^2, are terminal symbols.

$$S \rightarrow A|B|C|D, \quad A \rightarrow aa^1|aa^2, \quad B \rightarrow bb^1|bb^2, \quad C \rightarrow cc^1|cc^2, \quad D \rightarrow dd^1|dd^2.$$

During training, T_1 saw the first symbol of some randomly chosen sentence (out of 8 possible equally probable sentences), while T_2 saw the second symbol. T_1 needed 2 input units for its 4 possible input symbols represented as 2-dimensional binary vectors. T_2 needed 3 input units for its 8 possible input symbols represented as 3-dimensional binary vectors.

Both T_1 and T_2 had 4 hidden units and 6 output units, two more than necessary to locally represent the 4 predictable classes

$$\{a^1, a^2\}, \quad \{b^1, b^2\}, \quad \{c^1, c^2\}, \quad \{d^1, d^2\}.$$

10 test runs with $\epsilon = 0.25, \lambda = 1$, predictability maximization according to equations (2) and (3), constrained variance maximization according to (5) and (6), a learning rate of 1.0, and 15000 pattern presentations were conducted. No attempt was made to optimize learning speed. All experiments were successful – in the end T_2 emitted 4 different localized representations in response to members of the 4 predictable classes, while the two superfluous output units always remained switched off.

4.2 FINDING PREDICTABLE DISTRIBUTED REPRESENTATIONS

The following task is meant to test the system's capability for extracting a distributed representation of more than one non-trivial binary feature from patterns belonging to the same class.

The task. Two properties of a 4-dimensional binary input vector are the truth values of the following expressions:

1) There are more 'ones' on the 'right' side of the input vector than on the 'left' side.

2) The input vector consists of more 'ones' than 'zeros'.

Input vectors with equal numbers of ones and zeros as well as input vectors with equal numbers of ones on both sides are excluded. This leaves 2 input vectors for each possible feature combination.

During one learning cycle, a randomly chosen legal input vector was presented to T_1, another input vector randomly chosen among those with the feature combination of the first one was presented to T_2. T_1 and T_2 were constrained to have the same weights (see section 1) – both had 4 input units, 4 hidden units and 2 output units.

With predictability maximization according to (4), D_l defined by an auto-encoder (equation (7)), the reconstructor modules (see section 2.2) having 4 hidden units, and $\epsilon = 0.1$, ten test runs involving 15,000 pattern presentations were conducted. The learning rates of all networks were 0.5. The classifiers always came up with a distributed representation of the 4 possible feature combinations – the final output responses were near-binary with an error margin of 0.1.

With D_l being defined by modified predictability *minimization* (equation (9)), simultaneous training of both predictors and classifiers, 2 hidden units per predictor, 4 hidden units per classifier, and with $\lambda = 1$, $\epsilon = 0.5$, ten test runs involving 10,000 pattern presentations were conducted. The classifier learning rates were 0.5, the predictor learning rates were 1.0. Again, the system always extracted the two features.

4.3 STEREO TASK

The stereo experiment described in (Becker and Hinton, 1989) was used to compare IMAX to our approach.

Details of the task. There are two binary images called the 'left' image and the 'right' image. Each image consists of 2 'strips' – each strip being a binary input vector with 4 components. There are two classifiers with single output units and non-overlapping inputs: Each classifier has 8 input units and 'sees' an 8-dimensional input vector consisting of a strip from the right image and a corresponding strip from the left image generated as follows: The right image is purely random. The left image is generated from the right image by choosing, at random, a single global shift to be applied to each strip of the right image. The shift can be either one bit to the right or one bit to the left – 'overflow bits' generated by shifting some bit of a strip taken from the right image beyond the strip boundaries reappear on the opposite side of the corresponding 'shifted' strip of the left image ('wraparound'). Ambiguous shifts are excluded. The input may be interpreted as a fronto-parallel surface at an integer depth. The only common non-trivial property of both classifier inputs is the depth (i.e. the shift) (Becker and Hinton, 1989).

To implement D_l, we used predictability *minimization* according to section 2.4. Since the feature to be extracted is one-dimensional, only one predictor per classifier was necessary to predict the single output unit from a bias unit with constant activation.

The intra-representational predictors and the classifiers learned simultaneously. Each of the two classifiers T_1 and T_2 had 12 hidden units – the predictors had none. The learning rate of the predictors was 1.0, the classifier's learning rate was 0.5. Parameter settings were $\epsilon = 0.5$, $\lambda = 1.0$. The task was considered to be solved (the shift was considered to be extracted) if (1) the outputs of both classifiers were always equal (with an error margin of 0.1) and (2) each classifier emitted different binary outputs (again with an error margin of 0.1) in response to input patterns with different shifts. This corresponds to 1 bit of mutual information between the outputs and the shift.

With a first experiment, we employed a separate set of weights for each classifier. With ten test runs involving 100,000 training patterns and predictability maximization according to (4), D_l defined by modified *intra*-representational predictability *minimization* (equation (9)), the classifiers always learned to extract the shift.

Becker and Hinton report that their system (based on binary probabilistic units) was able to extract the shift only if IMAX was applied in successive layer by layer 'bootstrap' stages. In addition, they heuristically tuned the learning rate during learning. Finally they introduced a maximal weight change for each weight during gradient ascent.

In contrast, our method (based on continuous-valued units) does not rely on successive training stages, bootstrap learning, or learning rate adjustments. Once the learning phase is started, no external mechanism influences the behavior of the system. The performance of our system, however, is comparable to the performance of Becker's and Hinton's bootstrapped system.

With a second experiment, we used only one set of classifier weights shared by both classifiers (this leads to a reduction of free parameters, as mentioned at the end of section 1). The result was a significant decrease of learning time – with ten test runs the system needed between 20,000 and 50,000 training patterns to learn to extract the shift.

Again, no systematic attempt was made to optimize learning speed.

5 CONCLUSION

In contrast to IMAX, our methods (1) tend to be simpler (e.g., do not require sequential layer by layer 'bootstrapping' or learning rate adjustments – the stereo task can be solved more readily by our system),

(2) do not require Gaussian assumptions about the input or output signals, (3) do not require something like $DETMAX$, (4) partly have (unlike $DETMAX$) a potential for creating classifications with statistically independent components (this holds for D_l defined according to section 2.4).

The experiments above show that the alternative methods of section 2 can be useful techniques for implementing the D_l terms in (4) to obtain predictable informative input transformations. More experiments are needed, however, to become clear about their mutual advantages and disadvantages. It also remains to be seen how well the methods of this paper scale to larger problems.

6 ACKNOWLEDGEMENTS

Thanks to Mike Mozer for fruitful discussions. Thanks to Mike Mozer, Sue Becker, Rich Zemel, and an unknown referee, for helpful comments on drafts of this paper. This research was supported in part by a DFG fellowship to J. Schmidhuber, as well as by NSF award IRI–9058450, grant 90–21 from the James S. McDonnell Foundation.

References

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

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

Bridle, J. S. and MacKay, D. J. C. (1992). Unsupervised classifiers, mutual information and 'phantom' targets. In Lippman, D. S., Moody, J. E., and Touretzky, D. S., editors, *Advances in Neural Information Processing Systems 4,* to appear. San Mateo, CA: Morgan Kaufmann.

LeCun, Y. (1985). Une procédure d'apprentissage pour réseau à seuil asymétrique. *Proceedings of Cognitiva 85, Paris*, pages 599–604.

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

Nowlan, S. J. (1988). Auto-encoding with entropy constraints. In *Proceedings of INNS First Annual Meeting, Boston, MA.* Also published in special supplement to Neural Networks.

Parker, D. B. (1985). Learning-logic. Technical Report TR-47, Center for Comp. Research in Economics and Management Sci., MIT.

Prelinger, D. (1992). Diploma thesis, in preparation. Institut für Informatik, Technische Universität München.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning internal representations by error propagation. In *Parallel Distributed Processing*, volume 1, pages 318–362. MIT Press.

Schmidhuber, J. H. (1992). Learning factorial codes by predictability minimization. *Neural Computation, in press.*

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

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

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

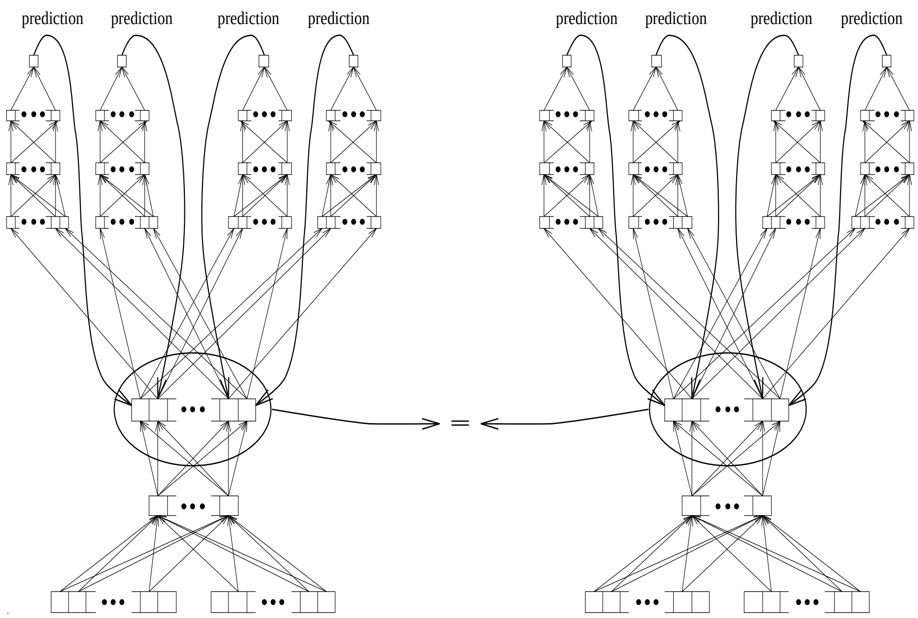

Figure 1: Two networks try to transform their *different* inputs to obtain the same representation. Each network is encouraged to tell something about its input by means of the recent technique for 'predictability minimization'. This technique requires additional *intra*-representational predictors (8 of them shown above) for detecting redundancies among the output units of the networks. Alternatives are provided in the text.