SEMILINEAR PREDICTABILITY MINIMIZATION PRODUCES WELL-KNOWN FEATURE DETECTORS

Neural Computation 8(4):773–786, 1996

Jürgen Schmidhuber*
IDSIA, Corso Elvezia 36
6900 Lugano, Switzerland

Martin Eldracher
IDSIA, Corso Elvezia 36
6900 Lugano, Switzerland

Bernhard Foltin
Fakultät für Informatik
TUM, 80290 München, Germany

Abstract

Predictability minimization (PM — Schmidhuber, 1992) exhibits various intuitive and theoretical advantages over many other methods for unsupervised redundancy reduction. So far, however, there were only toy applications of PM. In this paper, we apply semilinear PM to static real world images and find: without a teacher and without any significant pre-processing, the system automatically learns to generate distributed representations based on well-known feature detectors, such as orientation sensitive edge detectors and off-center-on-surround-like structures, thus extracting simple features related to those considered useful for image pre-processing and compression.

1 INTRODUCTION

Redundancy reduction is widely regarded as an important goal of unsupervised learning. See e.g. (Barlow et al., 1989; Atick et al., 1992; Schmidhuber, 1994); compare also (Linsker, 1988; Földiák, 1990; Deco and Obradovic, 1996; Miller, 1994; Field, 1994). But how to achieve this goal in a massively parallel, local, efficient, and perhaps even biologically plausible way?

Predictability minimization (PM). The simple approach in this paper is based on the recent principle of *predictability minimization* (PM) (Schmidhuber,

*juergen@idsia.ch http://www.idsia.ch/~juergen

1992). A feedforward network with n output units (or code units) sees input patterns with redundant components. Its goal is to respond with informative but less redundant output patterns, ideally by creating a factorial (statistically nonredundant) code of the input ensemble (Barlow et al., 1989). The central idea of PM is: *for each code unit, there is a predictor network that tries to predict the code unit from the remaining $n-1$ code units. But each code unit tries to become as unpredictable as possible, by representing environmental properties that are independent from those represented by other code units.* Predictors and code units co-evolve by fighting each other. See details in section 2.

Potential advantages of PM over other methods are: (1) Unlike certain inherently sequential methods (e.g. Rubner and Schulten, 1990), PM can be implemented in a parallel way. (2) Unlike e.g. with Barrow's model (1987), there may be *many* simultaneously active code units (multiple "winners" instead of single "winners"), as long as they represent *different* aspects of the environment (distributed coding instead of local coding). (3) Unlike e.g. with Linsker's INFOMAX (1988), there is no need to compute the derivatives of determinants of covariance matrices. (4) Unlike, e.g., Deco's and Obradovic's system (1996), Földiák's system (1990), Rubner and Tavan's system (1989), and anti-Hebbian systems in general, PM does neither require time consuming settling phases (due to recurrent connections), nor analytic computation of weight vectors. (5) Unlike almost all other methods, PM has a potential to discover *nonlinear* redundancy in the input data, and to generate appropriate redundancy-free codes. (6) Unlike most other "neural" methods (see references above), existing variants of PM create *binary* codes as opposed to *continuous* codes. This (a) allows for easier post training analysis and (b) facilitates the creation of *statistically independent* code components as opposed to merely *decorrelated* code components. Note that statistical independence implies decorrelation. But decorrelation does not imply statistical independence. **Why are statistically independent code components of interest?** One of many important reasons is this: for efficiency reasons, most statistical classifiers (e.g. Bayesian pattern classifiers) assume statistical independence of their input variables (corresponding to the pattern components). If we had a method that takes an arbitrary pattern ensemble and generates an equivalent factorial code, the latter could be fed into an efficient conventional classifier, which in turn could achieve its theoretically optimal performance.

Purpose of paper. Despite its potential advantages, PM has been tested on artificial data only (Lindstädt, 1993; Schmidhuber, 1993, 1994). To start a more thorough experimental analysis, in this paper we study the question: what happens if we apply a computationally simple, entirely local, highly parallel, and even biologically plausible variant of PM to real world images? An intuitively reasonable first step towards representing images in a less redundant way (and one adopted by standard image processing techniques, but apparently also by early visual processing stages of biological systems) is to build compact representations based on information about boundaries (*edges*) between areas

with nonvarying, redundant pixel activations. Since PM aims at generating codes with reduced redundancy, we may expect it to discover related ways of coding visual scenes, by creating feature detectors responsive to edges or similar informative features in the input scenes. Moreover, since edge detectors (as well as other, related useful feature detectors such as on-center-off-surround detectors) can be implemented with a single layer of neuronal units, we already may expect a single layer system to come up with such detectors. This paper reports a confirmation of this expectation, thus demonstrating that PM makes sense not only intuitively and in theory, but also in practical applications. The results encourage us to expect that the method also will be beneficial for large scale applications, by extracting more sophisticated, nonlinear, useful features in deeper layers. Due to our current hardware limitations, however, a test of this hypothesis is left for future research.

Outline. Section 2 briefly reviews the principles of PM in more detail. Section 3 applies the technique to real world images and presents results.

2 PREDICTABILITY MINIMIZATION: DETAILS

In its most simple form, PM is based on a feedforward network with n sigmoid output units (or code units). See Figure 1. The i-th code unit produces a

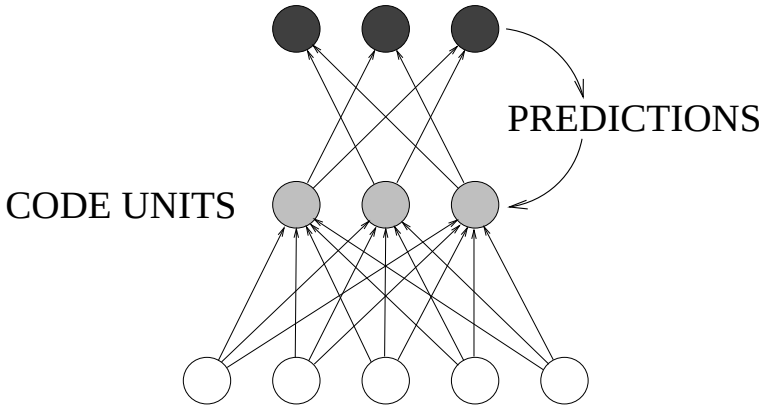

Figure 1: *Predictability minimization (PM): input patterns with redundant components are coded across n code units (grey). Code units are also input units of n predictor networks. Each predictor (output units black) attempts to predict its code unit (which it cannot see). But each code unit tries to escape the predictions, by representing environmental properties that are independent from those represented by other code units. This encourages high information throughput and redundancy reduction. Predictors and code generating net may have hidden units. In this paper, however, they don't. See text for details.*

real-valued output value $y_i^p \in [0,1]$ (the unit interval) in response to the p-th external input vector x^p (later we will see that training tends to make the ouput values near-binary). There are n additional feedforward nets called predictors, each having one output unit and $n-1$ input units. The predictor for code unit i is called P_i. Its real-valued output in response to the $\{y_k^p : \ k \neq i\}$ is called P_i^p. P_i is trained (in our experiments by conventional online backprop) to *minimize*

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

thus learning to approximate the conditional expectation $E(y_i \mid \{y_k : k \neq i\})$ of y_i, given the activations of the remaining code units. Of course, this conditional expectation typically will be very different from the actual activations of the code unit. For instance, assume that a certain code unit will be switched on in one third of all cases within a given context (defined by the activations of the remaining code units), while it will be switched off in two thirds of all such cases. Then, given this context, the predictor will predict a value of 0.3333.

The clue is: the code units are trained (in our experiments by online backprop) to *maximize essentially the same* objective function (Schmidhuber, 1992) the predictors try to *minimize*:

$$V_C = \sum_{i,p} (P_i^p - y_i^p)^2. \tag{2}$$

Predictors and code units co-evolve by fighting each other.

Justification. Let us assume that the P_i never get trapped in local minima and always perfectly learn the conditional expectations. It then turns out that the objective function V_C is essentially equivalent to the following one (also given in Schmidhuber, 1992):

$$\sum_i VAR(y_i) - \sum_{i,p} (P_i^p - \bar{y}_i)^2, \tag{3}$$

where $\bar{y}_i$ denotes the mean activation of unit i, and VAR denotes the variance operator. The equivalence of (2) and (3) was observed by Peter Dayan, Richard Zemel and Alex Pouget (personal communication, SALK Institute, 1992 — see (Schmidhuber, 1993) for details). (3) gives some intuition about what is going on while (2) is maximized. Mazimizing the first term of (3) tends to enforce binary units, and also **local** maximization of information throughput (given the binary constraint). Maximizing the second (negative) term (or minimizing the corresponding unsigned term) tends to make the conditional expectations equal to the unconditional expectations, thus encouraging mutual statistical independence (zero mutual information) and **global** maximization of information throughput.

3 APPLICATION: IMAGE PROCESSING

We apply PM to static black and white images of driving cars (see Figure 2). Each image is divided into 566×702 square pixels. Each pixel can take on 16 different grey levels represented as integers scaled around zero.

Input generation. There is a circular "input area". Its diameter is 64 pixel widths. There are 32 code units. For each code unit, there is a "bias input unit" with constant activation 1.0, and a circular receptive field of 81 evenly distributed additional input units. The diameter of each receptive field is 20 pixel widths. Receptive fields partly overlap. The positions of code units and receptive fields relative to the input area are fixed. See Figure 3. The rotation of the input area is chosen randomly. Its position is chosen randomly within the boundaries of the image. The activation of an input unit is the average grey level value of the closest pixel and the four adjacent pixels (see Figure 4).

Learning: heuristic simplifications. To achieve extreme computational simplicity (and also biological plausibility), we simplify the general method from section 2. Heuristic simplifications are: (1) No error signals are propagated through the predictor input units down into the code network. (2) We focus on semilinear networks as opposed to general nonlinear ones (no hidden units within predictors and code generating net – see Figure 1). (3) Predictors and code units learn simultaneously (also, each code unit sees only part of the total input). These simplifications make the method local in both space and time — to change the weight of any predictor connection, we can use the simple delta rule, which needs to know only the current activations of the two connected units, and the current activation of the unit to be predicted: in response to input pattern x_p, each weight w of predictor P_i changes according to

$$w \leftarrow -\eta_P \frac{\partial (P_i^p - y_i^p)^2}{\partial w},$$

where η_P is a positive constant. Likewise, to change the weight of any connection to a code unit, we need to know only the current activations of the two connected units, and the current activation of the corresponding predictor output unit: in response to input pattern x_p, each weight v leading to code unit i changes according to

$$v \leftarrow \eta_C \frac{\partial (P_i^p - y_i^p)^2}{\partial v},$$

where η_C is a positive constant.

Measuring information throughput. Unsupervised learning is occasionally switched off. Then the number N of pairwise different output patterns in response to 5000 randomly generated input patterns is determined (the activation of each output unit is taken to be 0 if below 0.05, 1 if above 0.95, and 0.5 otherwise). The **success rate** is defined by $\frac{N}{5000}$. Clearly, a success rate close to 1.0 implies high information throughput.

Results. Figure 5 plots success rate against number of training pattern presentations. Results are shown for various pairs of predictor learning rates η_P and code unit learning rates η_C. For instance, with η_P close to 1.0 and η_C being one or two orders of magnitude smaller, high success rates are obtained. Although the learning rates do have an influence on learning speed, the basic shapes of the learning curves are similar.

Edge detectors. With the set-up described above, to maximize information throughput, the system tends to create orientation sensitive edge detectors in an unsupervised manner. Weights corresponding to a typical receptive field (after 5000 pattern presentations) are shown in Figure 6. The connections are divided into two groups, one with inhibitory connections, the other one with excitatory connections. Both groups are separated by a "fuzzy" axis through the center of the receptive field. Its rotation angle determines the alignment of the edge provoking maximal response. In general, receptive fields of different code units exhibit different rotation angles. See Figure 7. Obviously, to represent the inputs in an informative but compact, efficient, redundancy-poor way, the creation of feature detectors specializing on certain rotated edges proves useful.

It is noticeable that in Figure 6 there appears to be a smooth gradient of weight strengths from strongly positive to strongly negative as one moves perpendicular to the positive/negative border. This is different from, e.g., MacKay and Miller (1990), where all weights tend to zero at the edge of the receptive field. At the moment, we don't have a good explanation for this effect.

On-center-off-surround / Off-center-on-surround. The nature of the receptive fields partly depends on receptive field size and degree of overlap. For instance, with nearly 200 input units per field and a more symmetric arrangement of receptive field centers (essentially, on the circular boundary of each field there are 6 other field centers), the system tends to generate another well-known kind of feature detector: the weight patterns become either on-center-off-surround-like or off-center-on-surround-like structures. See Figure 8 for an example.

4 DISCUSSION

We do not claim that PM is the only parallel method (as opposed to sequential methods, e.g. Rubner and Schulten, 1990) that can lead to well-known feature detectors. For instance, in case of Gaussian input distributions, Linsker's linear approach (Linsker, 1986b; Linsker, 1986a) for single output units also generates certain kinds of orientation sensitive fields (see also MacKay and Miller, 1990). This holds for more structured input data as well (Linsker, personal communication, 1994). In case of multiple code units, however, to prevent different code units from representing the same information, Linsker's INFOMAX approach (1988) requires to compute the derivatives of determinants of covariance matrices, which is computationally expensive, and also biologically

implausible (the multiple cell approach presented in Linsker (1986b) does not have certain problems of his later infomax approach, but it also does not have that nice theoretical foundation). Finally, it is conceivable that Földiák's system (1990), Rubner and Tavan's system (1989), and Deco and Obradovic's system (1996), might come up with similar edge detectors when applied to real world images. Unlike these approaches (and unlike other similar systems), however, our feedforward net does neither require time consuming settling phases (due to recurrent "anti-Hebbian" connections for lateral inhibition) nor analytic computation of the weight vectors.

Future research. We implemented a hierarchy of processing stages, each consisting of code modules and predictors as above. Each stage computes the input to the next stage. Preliminary tests again led to feature detectors causing *high* information throughput. However, unlike receptive fields of feature detectors observed in the first layer, receptive fields in higher layers appeared rather complex and did *not* exhibit any obvious structure. We would like to test the system on large data sets of real world scenes. We expect that this will lead to successively more complex and more specialized feature detectors, hopefully not only potentially useful for technical applications but also qualitatively related to those observed in biological systems. Another interesting experiment will be to add neighborhood relationships between the code units, to see whether this leads to automatic development of a smoothly-varying map of orientation preference. Unfortunately, however, our current hardware equipment does not permit large scale applications.

5 ACKNOWLEDGMENTS

Thanks to Ralf Linsker, Gustavo Deco, and Peter Dayan, for valuable comments.

References

Atick, J. J., Li, Z., and Redlich, A. N. (1992). Understanding retinal color coding from first principles. *Neural Computation*, 4:559–572.

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

Barrow, H. G. (1987). Learning receptive fields. In *Proceedings of the IEEE 1st Annual Conference on Neural Networks*, volume IV, pages 115–121. IEEE.

Deco, G. and Obradovic, D. (1996). *An information-theoretic approach to neural computing.* Springer, New York.

Field, D. J. (1994). What is the goal of sensory coding? *Neural Computation*, 6:559–601.

Földiák, P. (1990). Forming sparse representations by local anti-Hebbian learning. *Biological Cybernetics*, 64:165–170.

Lindstädt, S. (1993). Comparison of two unsupervised neural network models for redundancy reduction. In Mozer, M. C., Smolensky, P., Touretzky, D. S., Elman, J. L., and Weigend, A. S., editors, *Proc. of the 1993 Connectionist Models Summer School*, pages 308–315. Hillsdale, NJ: Erlbaum Associates.

Linsker, R. (1986a). From basic network principles to neural architecture: Emergence of orientation-selective cells. *Proc. Natl. Acad. Sci. USA*, 83:8779–8783.

Linsker, R. (1986b). From basic network principles to neural architecture: Emergence of spatial-opponent cells. *Proc. Natl. Acad. Sci. USA*, 83:8390–8394.

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

MacKay, D. J. C. and Miller, K. D. (1990). Analysis of Linsker's simulation of Hebbian rules. *Neural Computation*, 2:173–187.

Miller, K. D. (1994). A model for the development of simple cell receptive fields and the ordered arrangement of orientation columns through activity-dependent competition between on- and off-center inputs. *Journal of Neuroscience*, 14(1):409–441.

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

Rubner, J. and Tavan, P. (1989). A self-organization network for principal-component analysis. *Europhysics Letters*, 10:693–698.

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

Schmidhuber, J. (1993). Netzwerkarchitekturen, Zielfunktionen und Kettenregel. Habilitationsschrift, Institut für Informatik, Technische Universität München.

Schmidhuber, J. (1994). Neural predictors for detecting and removing redundant information. In Cruse, H., Dean, J., and Ritter, H., editors, *Adaptive Behavior and Learning*, number 9, pages 135–145. Center for Interdisciplinary Research, Universität Bielefeld.

Figure 2: *A typical image from the image data base.*

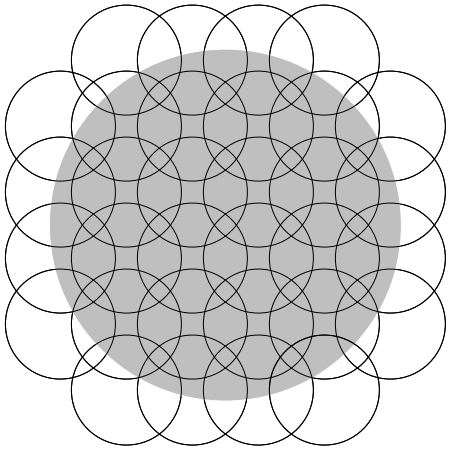

Figure 3: *Small circles represent partly overlapping receptive fields of code units. Their positions are shown relative to the input area (grey). See Figure 4 for details.*

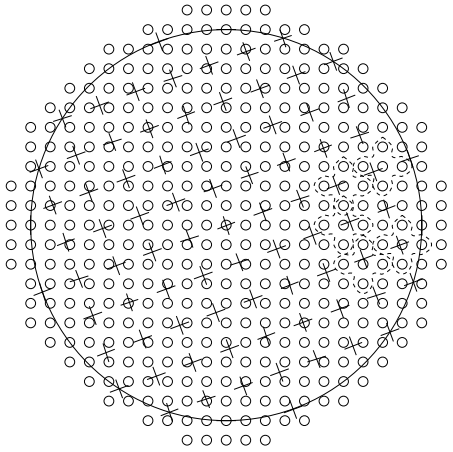

Figure 4: *Details of a receptive field inside the input area. Small circles indicate pixel positions. Crosses indicate positions of rotated input units. The activation of each input unit is the average grey level value of the closest pixel and the four adjacent pixels (indicated by dotted lines).*

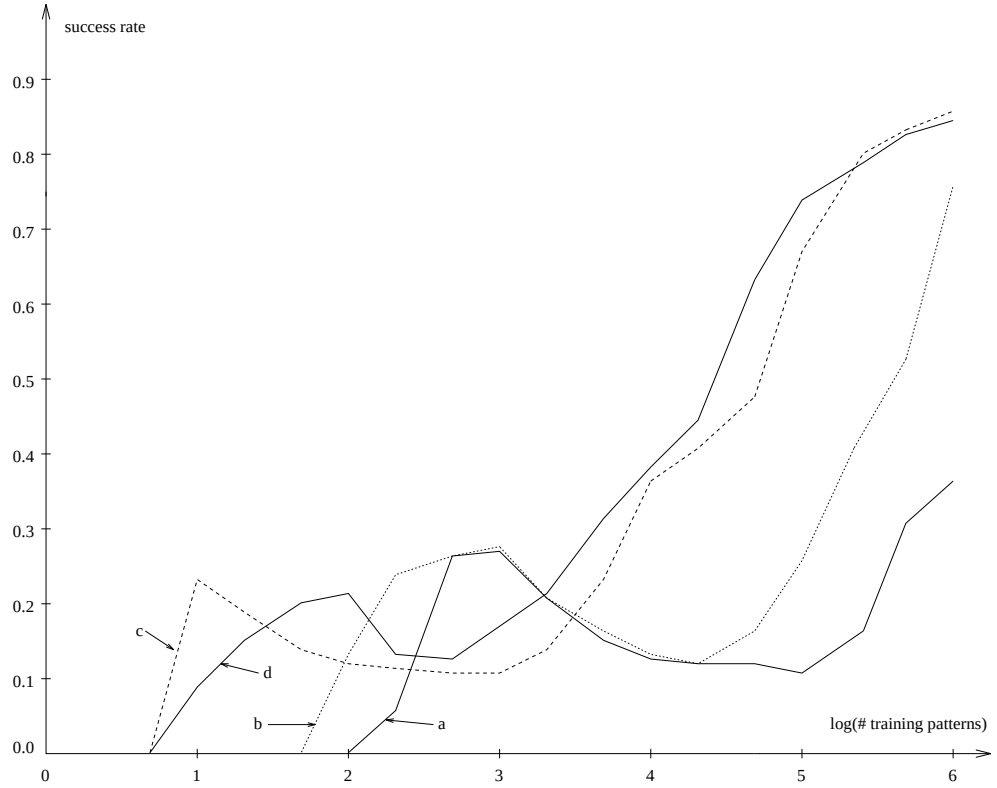

Figure 5: *Success rate (measuring information throughput) plotted against number of training pattern presentations (logarithmic scale). Results are shown for various pairs of predictor learning rates η_P and code unit learning rates η_C. a : $\eta_P = 0.001$, $\eta_C = 0.00004$. b : $\eta_P = 0.01$, $\eta_C = 0.00011$. c : $\eta_P = 0.1$, $\eta_C = 0.005$. d : $\eta_P = 1.0$, $\eta_C = 0.0042$.*

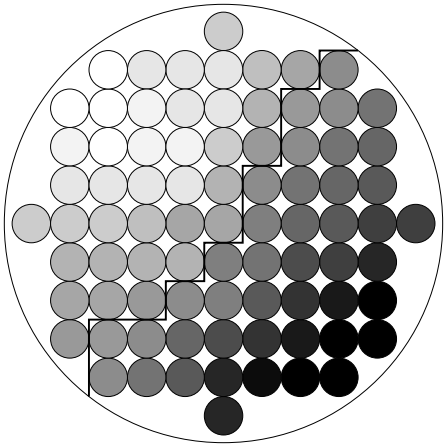

Figure 6: *Weights corresponding to a typical receptive field (after 5000 pattern presentations). Bright (dark) circles represent positive (negative) weights. The connections are divided into two groups, one with inhibitory connections, the other one with excitatory connections. Both groups are separated by a "fuzzy axis" (black division line) through the center of the receptive field. Its rotation angle determines the alignment of the edge provoking maximal response.*

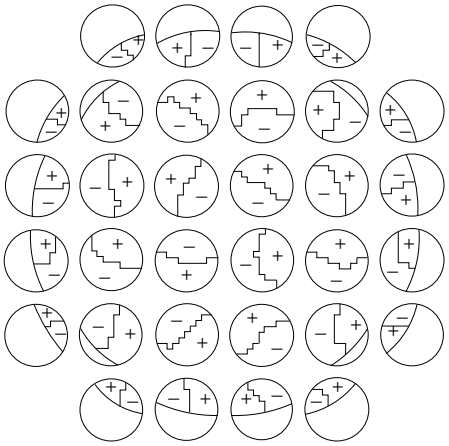

Figure 7: *For all receptive fields, typical post training boundaries between inhibitory and excitatory weights are shown. Distances between field centers are "blown up" to avoid confusion caused by overlaps.*

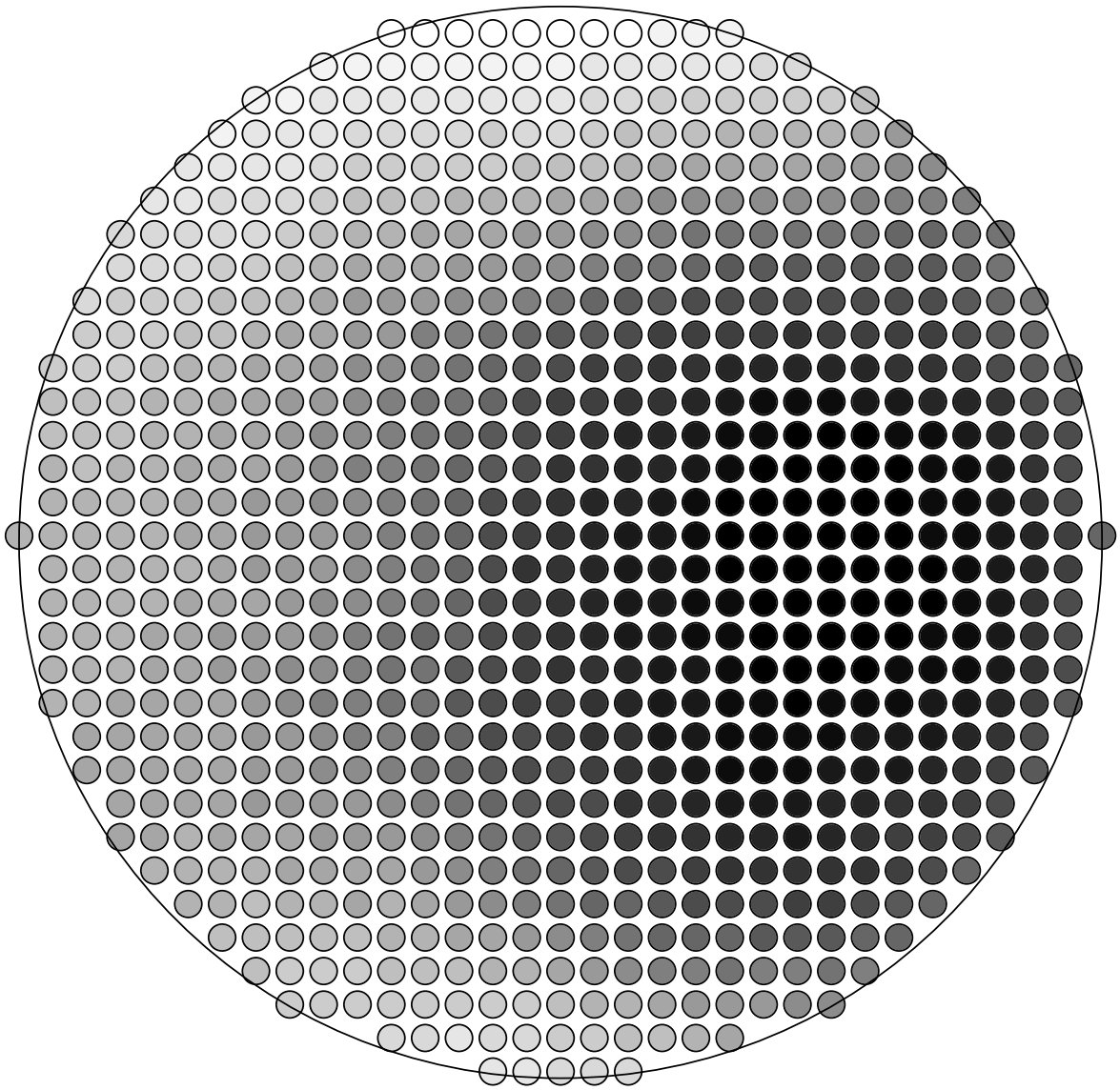

Figure 8: *Bright (dark) circles represent positive (negative) weights. With nearly 200 input units per field and a symmetric arrangement of receptive field centers (essentially, on the circular boundary of each field there are 6 other field centers), the weight patterns generated by the system tend to be either off-center-on-surround-like (see figure) or on-center-off-surround-like.*