The final version of this article has been published in
Neural Computation, 14(4):715-770 (2002) published by The MIT Press.
This version does not differ significantly from the final version.

Slow Feature Analysis: Unsupervised Learning of Invariances

Laurenz Wiskott[1,4,5] and Terrence Sejnowski[1,2,3]

[1]Computational Neurobiology Laboratory
[2]Howard Hughes Medical Institute
The Salk Institute for Biological Studies
San Diego, CA 92186-5800
http://www.cnl.salk.edu/CNL

[3]Department of Biology
University of California, San Diego
La Jolla, CA 92093

[4]Institute for Advanced Studies
Wallotstraße 19, D-14193 Berlin, Germany

[5]Innovationskolleg Theoretische Biologie
Institute for Biology, Humboldt-University Berlin
Invalidenstraße 43, D-10115 Berlin, Germany
http://itb.biologie.hu-berlin.de/
l.wiskott@biologie.hu-berlin.de

Abstract

Invariant features of temporally varying signals are useful for analysis and classification. Slow feature analysis (SFA) is a new method for learning invariant or slowly varying features from a vectorial input signal. SFA is based on a non-linear expansion of the input signal and application of principal component analysis to this expanded signal and its time derivative. It is guaranteed to find the optimal solution within a family of functions directly and can learn to extract a large number of decorrelated features, which are ordered by their degree of invariance. SFA can be applied hierarchically to process high dimensional input signals and to extract complex features. Slow feature analysis is applied first to complex cell tuning properties based on simple cell output including disparity and motion. Then, more complicated input-output functions are learned by repeated application of SFA. Finally, a hierarchical network of SFA-modules is presented as a simple model of the visual system. The same unstructured network can learn translation, size, rotation, contrast, or, to a lesser degree, illumination invariance for one-dimensional objects, depending only on the training stimulus. Surprisingly, only a few training objects sufficed to achieve good generalization to new objects. The generated representation is suitable for object recognition. Performance degrades, if the network is trained to learn multiple invariances simultaneously.

1 Introduction

Generating invariant representations is one of the major problems in object recognition. Some neural network systems have invariance properties built into the architecture. In both the Neocognitron (Fukushima et al., 1983) and the weight sharing back-propagation network (e.g. LeCun et al., 1989), for instance, translation invariance is implemented by replicating a common but plastic synaptic weight pattern at all shifted locations and then spatially sub-sampling the output signal, possibly blurred over a certain region. Other systems employ a matching dynamics to map representations onto each other in an invariant way (e.g. Olshausen et al., 1993; Konen et al., 1994). These two types of systems are trained and/or applied to static images and the invariances, e.g. translation and/or size invariance, need to be known in advance by the designer of the system (dynamic link matching (e.g. Konen et al., 1994) is less strict in specifying the invariance in advance). The approach described in this paper belongs to a third class of systems which are based on learning invariances from temporal input sequences (e.g. Földiák, 1991; Mitchison, 1991; Becker, 1993; Stone, 1996). The assumption is that primary sensory signals, which in general code for local properties, vary quickly while the perceived environment changes slowly. If one succeeds in extracting slow features from the quickly varying sensory signal, one is likely to obtain an invariant representation of the environment.

The general idea is illustrated in Figure 1. Assume three different objects in the shape of striped letters move straight through the visual field with different direction and speed, e.g. first an S, then an F, and then an A. On a high level this stimulus can be represented by three variables changing over time. The first one indicates object identity, i.e. which of the letters is currently visible, assuming that only one object is visible at a time. This is the *what*-information. The second and third variable indicate vertical and horizontal location of the object, respectively. This is the *where*-information. This representation would be particularly convenient, because it is compact and the important aspects of the stimulus are directly accessible.

The primary sensory input, i.e. the activities of the photoreceptors in this example, is distributed over many sensors, which respond only to simple localized features of the objects, such as local grey values, dots, or edges. Since the sensors respond to localized features, their activities will change quickly, even if the stimulus moves slowly. Consider, for example, the response of the first photoreceptor to the F. Because of the stripes, as the F moves across the receptor by just two stripe widths, the activity of the receptor rapidly changes from low to

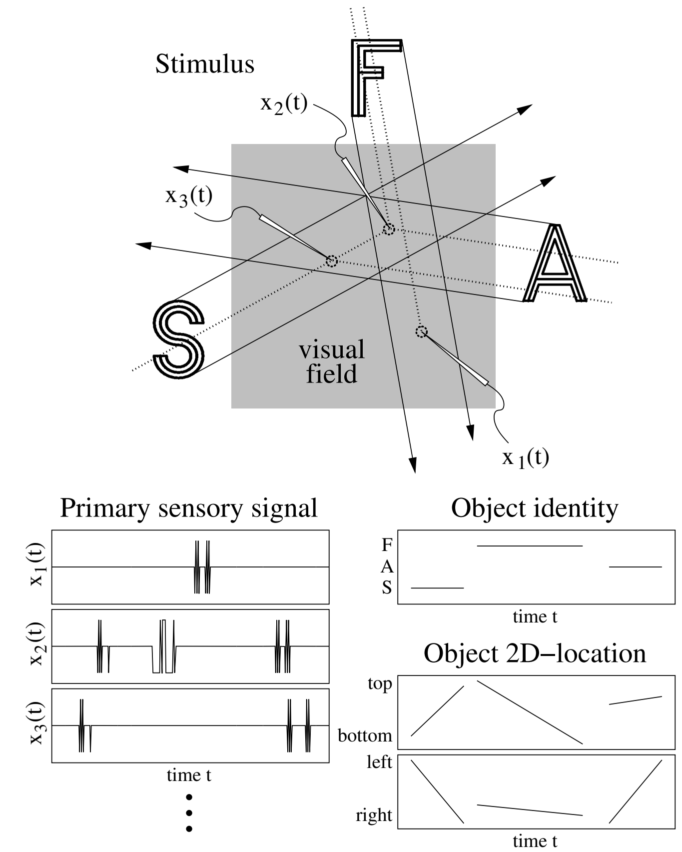


Figure 1: Relation between slowly varying stimulus and quickly varying sensor activities. **Top:** Three different objects, the letters S, F, and A, move straight through the visual field one by one. The responses of three photoreceptors to this stimulus at different locations are recorded and correspond to the grey value profiles of the letters along the dotted lines. **Bottom left:** Activities of the three (out of many) photoreceptors over time. The receptors respond vigorously when an object moves through their localized receptive field and are quiet otherwise. High values indicate white, low values indicate black. **Bottom right:** These three graphs show a high-level representation of the stimulus in terms of object identity and object location over time. See text for more explanations.

high and back again. This primary sensory signal is a low-level representation and contains the relevant information, such as object identity and location, only implicitly. However, if receptors cover the whole visual field, the visual stimulus is mirrored by the primary sensory signal and presumably there exists an input-output function that can extract the relevant information and compute a high-level representation like the one described above from this low-level representation.

What can serve as a general objective to guide unsupervised learning to find such an input-output function? One main difference between the high level representation and the primary sensory signal is the time scale on which they vary. Thus, a slowly varying representation can be considered to be of higher abstraction level than a quickly varying one. It is important to note here that the input-output function computes the output signal instantaneously, only on the basis of the current input. Slow variation of the output signal can therefore not be achieved by temporal low-pass filtering, but must be based on extracting aspects of the input signal that are inherently slow and useful for a high level representation. The vast majority of possible input-output functions would generate quickly varying output signals and only a very small fraction will generate slowly varying output signals. The task is to find some of these rare functions.

The primary sensory signal in Figure 1 is quickly varying compared to the high level representation, even though the components of the primary sensory signal have extensive quiet periods due to the blank background of the stimulus. The sensor of x_1, for instance, responds only to the F, because its receptive field is at the bottom right of the visual field. However, this illustrative example assumes an artificial stimulus and under more natural conditions the difference in temporal variation between the primary sensory signals and the high level representation should be even more pronounced.

The graphs for object identity and object location in Figure 1 have gaps between the linear sections representing the objects. These gaps need to be filled in somehow. If that is done, for example with some constant value, and the graphs are considered as a whole, then both the object identity and location vary on similar time scales, at least in comparison to the much faster varying primary sensory signals. This means that object location *and* object identity can be learned based on the objective of slow variation, which sheds new light on the problem of learning invariances. The common notion that object location changes quickly while object identity changes slowly, or rarely, and that the recognition system has to learn to represent only object identity and to ignore object location. However, another interpretation of this situation is that object translation induces quick changes in the primary sensory signal, in comparison to which object identity *and* object location vary slowly. Both of these aspects can be extracted as slow features from the primary sensory signal. While it is conceptionally convenient to call object identity the *what*-information of the stimulus and object location the *where*-information, they are of similar nature in the sense that they may vary on similar time scales compared to the primary sensory signal.

In the next two sections a formal definition of the learning problem is given and a new algorithm to solve it is presented. The subsequent sections describe several sample applications. The first shows how complex cell behavior found in visual cortex can be inferred from simple cell outputs. This is extended to include disparity and direction of motion in the second example. The third example illustrates that more complicated input-output functions can be approximated by applying the learning algorithm repeatedly. The fourth and fifth example show how a hierarchical network can learn translation and other invariances. In the final discussion, the algorithm is compared with previous approaches to learning invariances.

2 The Learning Problem

The first step is to give a mathematical definition for the learning of invariances. Given a vectorial input signal $\mathbf{x}(t)$, the objective is to find an input-output function $\mathbf{g}(\mathbf{x})$ such that the output signal $\mathbf{y}(t) := \mathbf{g}(\mathbf{x}(t))$ varies as slowly as possible while still conveying some information about the input to avoid the trivial constant response. Strict invariances are not the goal, but rather approximate ones that change slowly. This can be formalized as follows:

Learning Problem: *Given an I-dimensional input signal $\mathbf{x}(t) = [x_1(t), ..., x_I(t)]^T$ with $t \in [t_0, t_1]$ indicating time and $[...]^T$ indicating the transpose of $[...]$. Find an input-output function $\mathbf{g}(\mathbf{x}) = [g_1(\mathbf{x}), ..., g_J(\mathbf{x})]^T$ generating the J-dimensional output signal $\mathbf{y}(t) = [y_1(t), ..., y_J(t)]^T$ with $y_j(t) := g_j(\mathbf{x}(t))$ such that for each $j \in \{1, ..., J\}$*

$$\Delta_j := \Delta(y_j) := \langle \dot{y}_j^2 \rangle \quad \textit{is minimal} \tag{1}$$

under the constraints

$$\begin{aligned} \langle y_j \rangle &= 0 \quad \textit{(zero mean)}\,, & (2)\\ \langle y_j^2 \rangle &= 1 \quad \textit{(unit variance)}\,, & (3)\\ \forall\, j' < j: \quad \langle y_{j'}\, y_j \rangle &= 0 \quad \textit{(decorrelation)}\,, & (4) \end{aligned}$$

where the angle brackets indicate temporal averaging, i.e. $\langle f \rangle := \frac{1}{t_1 - t_0} \int_{t_0}^{t_1} f(t)\, dt$.

Equation (1) expresses the primary objective of minimizing the temporal variation of the output signal. Constraints (2) and (3) help avoiding the trivial solution $y_j(t) = \text{const.}$ Constraint (4) guarantees that different output signal components carry different information and do not simply reproduce each other. It also induces an order, so that $y_1(t)$ is the optimal output signal component, while $y_2(t)$ is a less optimal one, since it obeys the additional constraint $\langle y_1\, y_2 \rangle = 0$. Thus, $\Delta(y_{j'}) \leq \Delta(y_j)$ if $j' < j$.

The zero mean constraint (2) was added for convenience only. It could be dropped, in which case constraint (3) should be replaced by $\langle (y_j - \langle y_j \rangle)^2 \rangle = 1$ to avoid the trivial solutions $y_1(t) = \pm 1$. One can also drop the unit variance constraint (3) and integrate it into the objective, which would then be to minimize $\langle \dot{y}_j^2 \rangle / \langle (y_j - \langle y_j \rangle)^2 \rangle$. This is the formulation used in (Becker & Hinton, 1992). However, integrating the two constraints (Eqs. 2 and 3) in the objective function leaves the solution undetermined by an arbitrary offset and scaling factor for y_j. The explicit constraints make the solution less arbitrary.

This learning problem is an optimization problem of variational calculus and in general difficult to solve. However, for the case that the input-output function components g_j are constrained to be a linear combination of a finite set of non-linear functions, the problem simplifies significantly. An algorithm for solving the optimization problem under this constraint is given in the following section.

3 Slow Feature Analysis (SFA)

Given an I-dimensional input signal $\mathbf{x}(t) = [x_1(t), ..., x_I(t)]^T$, consider an input-output function $\mathbf{g}(\mathbf{x}) = [g_1(\mathbf{x}), ..., g_J(\mathbf{x})]^T$ each component of which is a weighted sum over a set of K non-linear functions $h_k(\mathbf{x})$, i.e. $g_j(\mathbf{x}) := \sum_{k=1}^{K} w_{jk} h_k(\mathbf{x})$. Usually $K > \max(I, J)$. Applying $\mathbf{h} = [h_1, ..., h_K]^T$ to the input signal yields the non-linearly expanded signal $\mathbf{z}(t) := \mathbf{h}(\mathbf{x}(t))$.

After this non-linear expansion the problem can be treated as linear in the expanded signal components $z_k(t)$. This is a common technique to turn a non-linear problem into a linear one. A well known example is the support vector machine (e.g. VAPNIK, 1995). The weight vectors $\mathbf{w}_j = [w_{j1}, ..., w_{jK}]^T$ are subject to learning and the jth output signal component is given by $y_j(t) = g_j(\mathbf{x}(t)) = \mathbf{w}_j^T \mathbf{h}(\mathbf{x}(t)) = \mathbf{w}_j^T \mathbf{z}(t)$. The objective (Eq. 1) is to optimize the input-output function and thus the weights such that

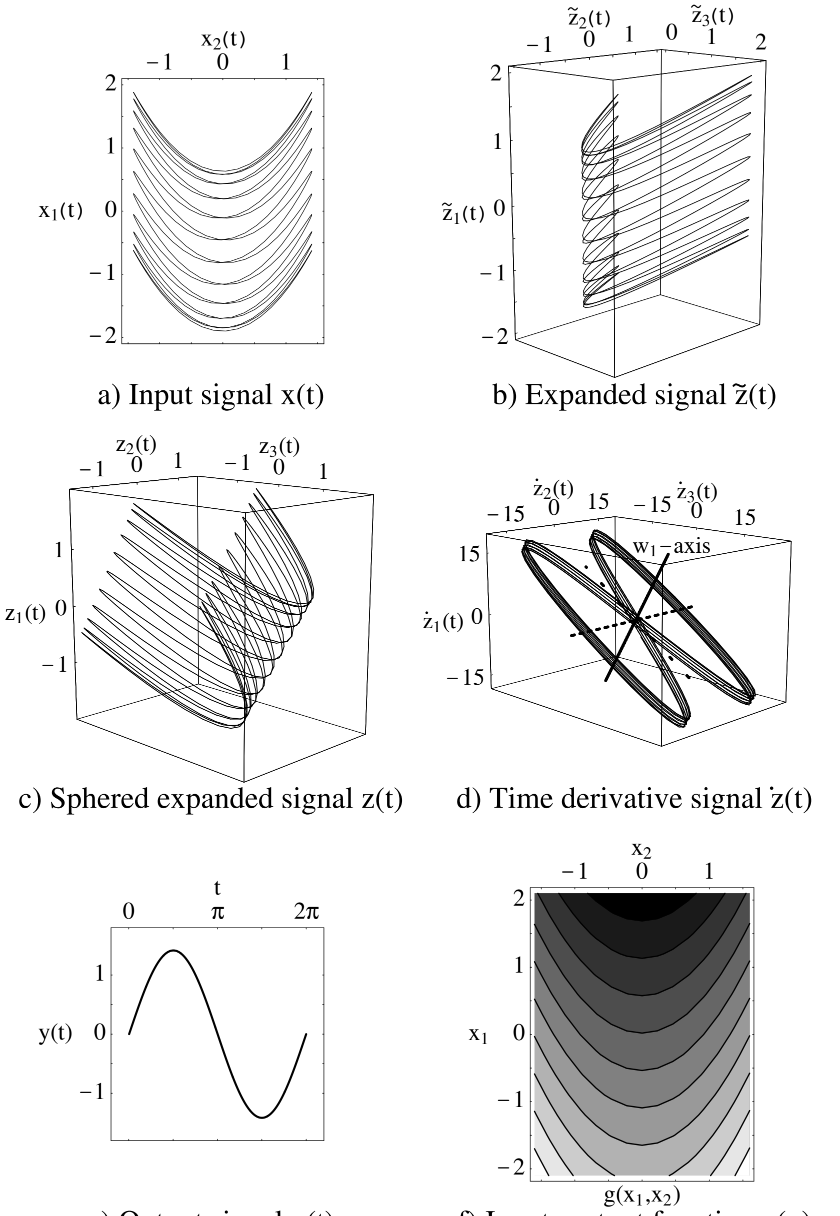
$$\Delta(y_j) = \langle \dot{y}_j^2 \rangle = \mathbf{w}_j^T \langle \dot{\mathbf{z}}\dot{\mathbf{z}}^T \rangle \mathbf{w}_j \tag{5}$$

is minimal. Assume the non-linear functions h_k are chosen such that the expanded signal $\mathbf{z}(t)$ has zero mean and a unit covariance matrix. Such a set h_k of non-linear functions can be easily derived from an arbitrary set h'_k by a sphering stage, as will be explained below. Then we find that the constraints (Eqs. 2-4),

$$\begin{aligned} \langle y_j \rangle &= \mathbf{w}_j^T \underbrace{\langle \mathbf{z} \rangle}_{=\mathbf{0}} = 0 \;, && (6) \\ \langle y_j^2 \rangle &= \mathbf{w}_j^T \underbrace{\langle \mathbf{z}\mathbf{z}^T \rangle}_{=\mathbf{I}} \mathbf{w}_j = \mathbf{w}_j^T \mathbf{w}_j = 1 \;, && (7) \\ \forall\, j' < j : \quad \langle y_{j'}\, y_j \rangle &= \mathbf{w}_{j'}^T \underbrace{\langle \mathbf{z}\mathbf{z}^T \rangle}_{=\mathbf{I}} \mathbf{w}_j = \mathbf{w}_{j'}^T \mathbf{w}_j = 0 && (8) \end{aligned}$$

are automatically fulfilled if and only if we constrain the weight vectors to be an orthonormal set of vectors. Thus for the first component of the input-output function the optimization problem reduces to finding the normed weight vector that minimizes $\Delta(y_1)$ of Equation (5). The solution is the normed eigenvector of matrix $\langle \dot{\mathbf{z}}\dot{\mathbf{z}}^T \rangle$ that corresponds to the smallest eigenvalue (cf. MITCHISON, 1991). The eigenvectors of the next higher eigenvalues produce the next components of the input-output function with the next higher Δ-values. This leads to an algorithm for solving the optimization problem stated above.

It is useful to make a clear distinction between raw signals, exactly normalized signals derived from training data, and approximately normalized signals derived from test data. Let $\tilde{\mathbf{x}}(t)$ be a raw input signal that can have any mean and variance. For computational convenience and display purposes the signals are normalized to zero mean and unit variance. This normalization is exact for the training data $\mathbf{x}(t)$. Correcting test data by the same offset and factor will in general yield an input signal $\mathbf{x}'(t)$ that is only approximately normalized, since each data sample has a slightly different mean and variance, while the normalization is always done with the offset and factor determined from the training data. In the following, raw signals have a tilde and test data have a dash; symbols with neither a tilde nor a dash usually (but not always) refer to normalized training data.



a) Input signal x(t)

b) Expanded signal $\tilde{z}(t)$

c) Sphered expanded signal z(t)

d) Time derivative signal $\dot{z}(t)$

e) Output signal y(t)

f) Input−output function g(x)

See next page for figure caption

Figure 2: Illustration of the learning algorithm by means of a simplified example. **a)** Input signal $\tilde{\mathbf{x}}(t)$ is given by $\tilde{x}_1(t) := \sin(t) + \cos^2(11\ t)$, $\tilde{x}_2(t) := \cos(11\ t)$, $t \in [0, 2\pi]$, where $\sin(t)$ constitutes the slow feature signal. Shown is the normalized input signal $\mathbf{x}(t)$. **b)** Expanded signal $\tilde{\mathbf{z}}(t)$ is defined as $\tilde{z}_1(t) := x_1(t)$, $\tilde{z}_2(t) := x_2(t)$, and $\tilde{z}_3(t) := x_2^2(t)$. $x_1^2(t)$ and $x_1(t)x_2(t)$ are left out for easier display. **c)** Sphered signal $\mathbf{z}(t)$ has zero mean and unit covariance matrix. Its orientation in space is algorithmically determined by the principal axes of $\tilde{\mathbf{z}}(t)$ but otherwise arbitrary. **d)** Time derivative signal $\dot{\mathbf{z}}(t)$. The direction of minimal variance determines the weight vector $\mathbf{w}_1$. This is the direction in which the sphered signal $\mathbf{z}(t)$ varies most slowly. The axes of next higher variance determine the weight vectors $\mathbf{w}_2$ and $\mathbf{w}_3$, shown as dashed lines. **e)** Projecting the sphered signal $\mathbf{z}(t)$ onto the $\mathbf{w}_1$-axis yields the first output signal component $y_1(t)$, which is the slow feature signal $\sin(t)$. **f)** The first component $\mathbf{g}_1(x_1, x_2)$ of the input-output function derived by the steps a) to e) is shown as a contour plot.

The algorithm now has the following form (see also Fig. 2):

1. **Input signal:** For training, an I-dimensional input signal is given by $\tilde{\mathbf{x}}(t)$.

2. **Input signal normalization:** Normalize the input signal to obtain

$$\begin{aligned} \mathbf{x}(t) \;&:=\; [x_1(t), ..., x_I(t)]^T && (9)\\ \text{with}\quad x_i(t) \;&:=\; \frac{\tilde{x}_i(t) - \langle \tilde{x}_i \rangle}{\sqrt{\langle (\tilde{x}_i - \langle \tilde{x}_i \rangle)^2 \rangle}} \,, && (10)\\ \text{so that}\quad \langle x_i \rangle \;&=\; 0 \,, && (11)\\ \text{and}\quad \langle x_i^2 \rangle \;&=\; 1 \,. && (12) \end{aligned}$$

3. **Non-linear expansion:** Apply a set of non-linear functions $\tilde{\mathbf{h}}(\mathbf{x})$ to generate an expanded signal $\tilde{\mathbf{z}}(t)$. Here all monomials of degree one (resulting in linear SFA sometimes denoted by SFA^1) or of degree one and two including mixed terms such as $x_1 x_2$ (resulting in quadratic SFA sometimes denoted by SFA^2) are used, but any other set of functions could be used as well. Thus, for quadratic SFA

$$\begin{aligned} \tilde{\mathbf{h}}(\mathbf{x}) \;&:=\; [x_1, ..., x_I, x_1 x_1, x_1 x_2, ..., x_I x_I]^T \,, && (13)\\ \tilde{\mathbf{z}}(t) := \tilde{\mathbf{h}}(\mathbf{x}(t)) \;&=\; [x_1(t), ..., x_I(t), x_1(t)x_1(t), x_1(t)x_2(t), ..., x_I(t)x_I(t)]^T \,. && (14) \end{aligned}$$

   Using first- and second-degree monomials, $\tilde{\mathbf{h}}(\mathbf{x})$ and $\tilde{\mathbf{z}}(t)$ are of dimensionality $K = I + I(I+1)/2$.

4. **Sphering:** Normalize the expanded signal $\tilde{\mathbf{z}}(t)$ by an affine transformation to generate $\mathbf{z}(t)$ with zero mean and identity covariance matrix $\mathbf{I}$,

$$\begin{aligned} \mathbf{z}(t) \;&:=\; \mathbf{S}\,(\tilde{\mathbf{z}}(t) - \langle \tilde{\mathbf{z}} \rangle) \,, && (15)\\ \text{with}\quad \langle \mathbf{z} \rangle \;&=\; \mathbf{0} && (16)\\ \text{and}\quad \langle \mathbf{z}\,\mathbf{z}^T \rangle \;&=\; \mathbf{I} \,. && (17) \end{aligned}$$

   This normalization is called sphering (or whitening). Matrix $\mathbf{S}$ is the sphering matrix and can be determined with the help of principal component analysis (PCA) on matrix $(\tilde{\mathbf{z}}(t) - \langle \tilde{\mathbf{z}} \rangle)$. It therefore depends on the specific training data set. This also defines

$$\mathbf{h}(\mathbf{x}) := \mathbf{S}\,(\tilde{\mathbf{h}}(\mathbf{x}) - \langle \tilde{\mathbf{z}} \rangle) \,, \qquad (18)$$

which is a normalized function, while $\mathbf{z}(t)$ is the sphered data.

5. **Principal Component Analysis (PCA):** Apply PCA to matrix $\langle \dot{\mathbf{z}}\,\dot{\mathbf{z}}^T \rangle$. The J eigenvectors with lowest eigenvalues λ_j yield the normalized weight vectors

$$\mathbf{w}_j: \quad \langle \dot{\mathbf{z}}\,\dot{\mathbf{z}}^T \rangle \mathbf{w}_j = \lambda_j \mathbf{w}_j \tag{19}$$

$$\text{with} \quad \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_J \;, \tag{20}$$

which provide the input-output function

$$\mathbf{g}(\mathbf{x}) \;:=\; [g_1(\mathbf{x}), ..., g_J(\mathbf{x})]^T \tag{21}$$

$$\text{with} \quad g_j(\mathbf{x}) \;:=\; \mathbf{w}_j^T \mathbf{h}(\mathbf{x}) \tag{22}$$

and the output signal

$$\mathbf{y}(t) \;:=\; \mathbf{g}(\mathbf{x}(t)) \tag{23}$$

$$\text{with} \quad \langle \mathbf{y} \rangle \;=\; \mathbf{0} \;, \tag{24}$$

$$\langle \mathbf{y}\,\mathbf{y}^T \rangle \;=\; \mathbf{I} \;, \tag{25}$$

$$\text{and} \quad \Delta(y_j) = \langle \dot{y}_j^2 \rangle \;=\; \lambda_j \;. \tag{26}$$

The components of the output signal have exactly zero mean, unit variance, and are uncorrelated.

6. **Repetition:** If required, use the output signal $\mathbf{y}(t)$ (or the first few components of it or a combination of different output signals) as an input signal $\mathbf{x}(t)$ for the next application of the learning algorithm. Continue with Step 3.

7. **Test:** In order to test the system on a test signal, apply the normalization and input-output function derived in Steps 2 to 6 to a new input signal $\tilde{\mathbf{x}}'(t)$. Notice that this test signal needs to be normalized with the same offsets and factors as the training signal to reproduce the learned input-output relation accurately. Thus the training signal is normalized only approximately to yield

$$\mathbf{x}'(t) \;:=\; [x'_1(t), ..., x'_I(t)]^T \tag{27}$$

$$\text{with} \quad x'_i(t) \;:=\; \frac{\tilde{x}'_i(t) - \langle \tilde{x}_i \rangle}{\sqrt{\langle (\tilde{x}_i - \langle \tilde{x}_i \rangle)^2 \rangle}} \;, \tag{28}$$

$$\text{so that} \quad \langle x'_i \rangle \;\approx\; 0 \;, \tag{29}$$

$$\text{and} \quad \langle x'^2_i \rangle \;\approx\; 1 \;. \tag{30}$$

The normalization is accurate only to the extent the test signal is representative for the training signal. The same is true for the output signal

$$\mathbf{y}'(t) \;:=\; \mathbf{g}(\mathbf{x}'(t)) \tag{31}$$

$$\text{with} \quad \langle \mathbf{y}' \rangle \;\approx\; \mathbf{0} \;, \tag{32}$$

$$\text{and} \quad \langle \mathbf{y}'\,\mathbf{y}'^T \rangle \;\approx\; \mathbf{I} \;. \tag{33}$$

For practical reasons, singular value decomposition is used in Steps 4 and 5 instead of principal component analysis. Singular value decomposition is preferable for analyzing degenerate data in which some eigenvalues are very close to zero, which are then discarded in Step 4. The non-linear expansion sometimes leads to degenerate data, since it produces a highly redundant representation where some components may have a linear relationship. In general signal components with eigenvalues close to zero typically contain noise, e.g. rounding errors, which after normalization is very quickly fluctuating and would not be selected by SFA in Step 5 in any case. Thus the decision as to which small components should be discarded is not critical.

It is useful to measure the invariance of signals not by the value of Δ directly but by a measure that has a more intuitive interpretation. A good measure may be an index η defined by

$$\eta(y) := \frac{T}{2\pi}\sqrt{\Delta(y)} \tag{34}$$

for $t \in [t_0, t_0 + T]$. For a pure sine wave $y(t) := \sqrt{2}\sin(n\,2\pi\,t/T)$ with an integer number of oscillations n the index $\eta(y)$ is just the number of oscillations[1], i.e. $\eta(y) = n$. Thus the index η of an arbitrary signal indicates what the number of oscillations would be for a pure sine wave of same Δ-value, at least for integer values of η. Low η-values indicate slow signals. Since output signals derived from test data are only approximately normalized, $\eta(y')$ is meant to include an exact normalization of y' to zero mean and unit variance, to make the η-index independent of an accidental scaling factor.

Neural Implementation

The SFA-algorithm is formulated for a training input signal $\mathbf{x}(t)$ of definite length. During learning, it processes this signal as a single batch in one shot and does not work incrementally, such as online learning rules do. However, SFA is a computational model for neural processing in biological systems and can be related to two standard network architectures (Fig. 3). The non-linear basis functions $\tilde{h}_k(\mathbf{x})$ can, for instance, be considered as synaptic clusters on the dendritic tree, which locally perform a fixed non-linear transformation on the input data and can be weighted independently of other synaptic clusters performing other but also fixed non-linear transformations on the input signal (Fig. 3, top). Sigma-pi units (Rumelhart et al., 1986), for instance, are of this type. In another interpretation the non-linear basis functions could be realized by fixed non-linear units in a hidden layer. These then provide weighted inputs to linear output units which can be trained (Fig. 3, bottom). The radial basis function network with non-adaptable basis functions is an example of this interpretation (cf.

[1]For symmetry reasons and since $(\sqrt{2}\sin(x))^2 + (\sqrt{2}\cos(x))^2 = 2$, it is evident that $\langle y\rangle = 0$ and $\langle y^2\rangle = 1$ if averaging $y(t) := \sqrt{2}\sin(n\,2\pi\,t/T)$ over $t \in [t_0, t_0 + T]$, i.e. over an integer number of oscillations n. Setting $t_0 = 0$ without loss of generality, we find that

$$\Delta(y) = \langle \dot{y}^2\rangle = \frac{1}{T}\int_0^T 2\frac{n^2 4\pi^2}{T^2}\cos^2(n\,2\pi\,t/T)\,dt = \frac{n^2 4\pi^2}{T^2}\,\frac{1}{n\,2\pi}\int_0^{n\,2\pi} 2\cos^2(t')\,dt' = \frac{n^2 4\pi^2}{T^2}$$

and

$$\eta(y) = \frac{T}{2\pi}\sqrt{\Delta(y)} = \frac{T}{2\pi}\sqrt{\frac{n^2 4\pi^2}{T^2}} = n\,.$$

(Becker & Hinton, 1995); see also (Bishop, 1995) for an introduction to radial basis function networks). Depending on the type of non-linear functions used for $\tilde{h}_k(\mathbf{x})$, the first or the second interpretation is more appropriate. Lateral connections between the output units g_j, either only from lower to higher units as shown in the figure or between all units, are needed to decorrelate the output signal components by some variant of anti-Hebbian learning (see (Becker & Plumbley, 1996) for an overview). Each of these two networks forms a functional unit performing SFA. In the following we will refer to such a unit as an *SFA-module*, modelled by the algorithm described above.

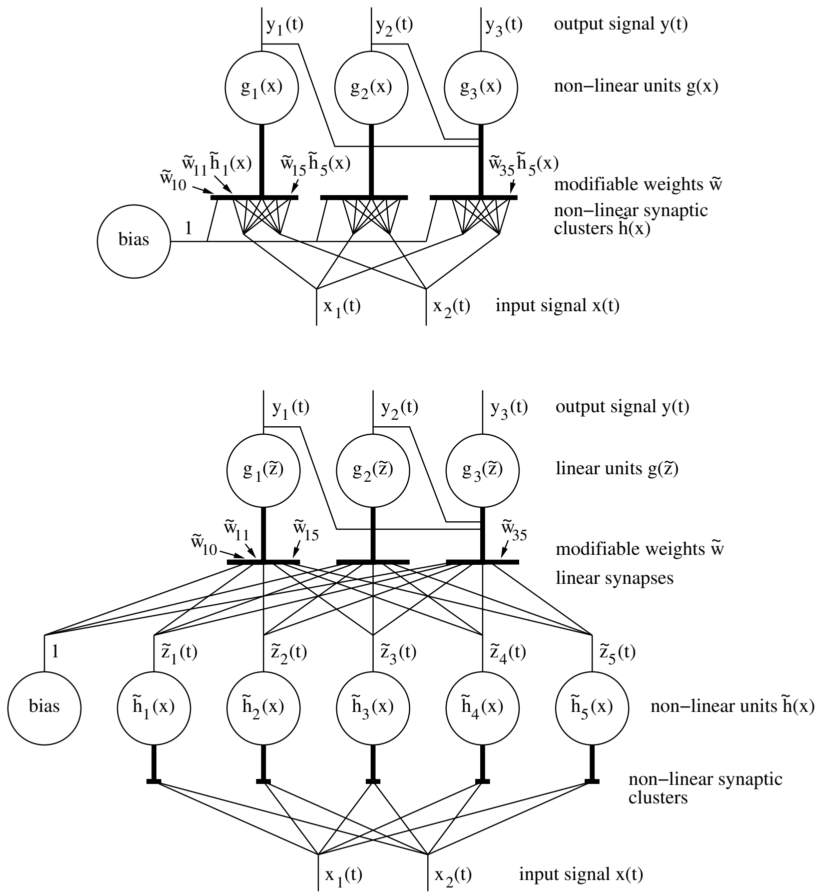


Figure 3: Two possible network structures for performing slow feature analysis. **Top:** Interpretation as a group of units with complex computation on the dendritic trees (thick lines), e.g. sigma-pi units. **Bottom:** Interpretation as a layered network of simple units with fixed non-linear units in the hidden layer, e.g. radial basis function networks with non-adaptable hidden units. In both cases the input-output function components are given by $g_j(\mathbf{x}) = \mathbf{w}_j^T \mathbf{h}(\mathbf{x}) = \tilde{w}_{j0} + \tilde{\mathbf{w}}_j^T \tilde{\mathbf{h}}(\mathbf{x})$, with appropriate raw weight vectors $\tilde{\mathbf{w}}_j$. The input signal components are assumed to be normalized here.

4 Examples

The properties of the learning algorithm are now illustrated by several examples. The first example is about learning response behaviour of complex cells based on simple cell responses (simple and complex cells are two types of cells in primary visual cortex). The second one is similar but also includes estimation of disparity and motion. One application of slow feature analysis is sufficient for these two examples. The third example is more abstract and requires a more complicated input-output function, which can be approximated by three SFAs in succession. This leads over to the fourth example, which shows a hierarchical network of SFA-modules learning translation invariance. This is generalized to other invariances in example five. Each example illustrates a different aspect of SFA; all but the third example also refer to specific learning problems in the visual system and present possible solutions on a computational level based on SFA, although these examples do not claim to be biologically plausible in any detail. All simulations were done with Mathematica.

4.1 Examples 1 and 2: Complex Cells, Disparity, and Motion Estimation

The first two examples are closely related and use subsets of the same data. Consider five monocular simple cells for the left and right eyes with receptive fields as indicated in Figure 4. The simple cells are modeled by spatial Gabor wavelets (JONES & PALMER, 1987), whose responses $\tilde{x}(t)$ to a visual stimulus smoothly moving across the receptive field is given by a combination of non-negative amplitude $\tilde{a}(t)$ and phase $\tilde{\phi}(t)$ both varying in time, i.e. $\tilde{x}(t) := \tilde{a}(t)\ \sin(\tilde{\phi}(t))$.

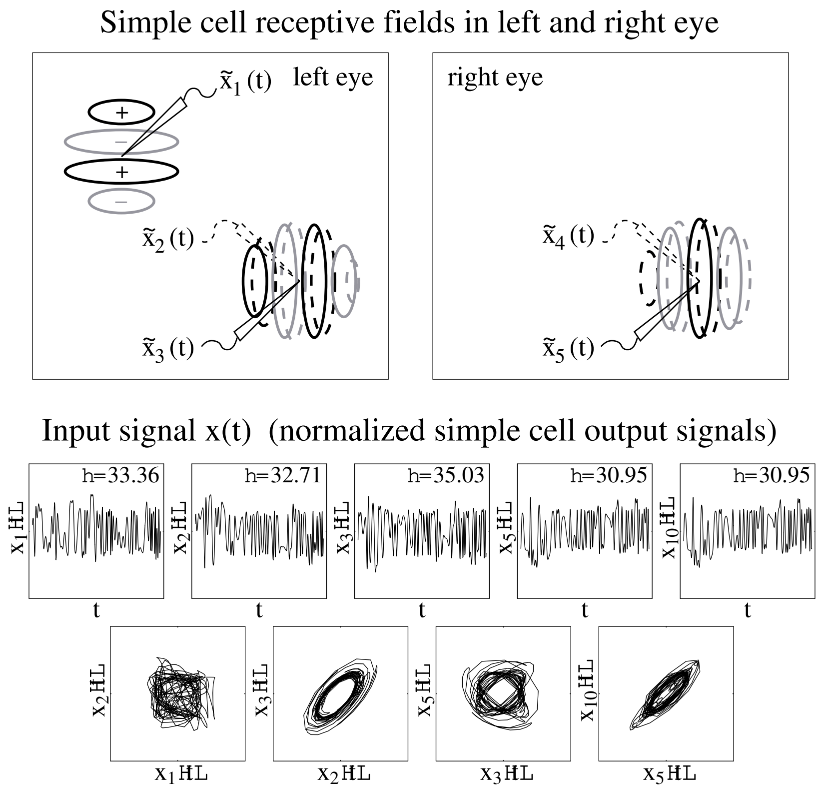


Figure 4 (ad Examples 1 and 2): **Top:** Receptive fields of five simple cells for left and right eye which provide the input signal $\tilde{\mathbf{x}}(t)$. **Bottom:** Selected normalized input signal components $x_i(t)$ ploted vs. time and vs. each other. All graphs range from -4 to $+4$; time axes range from 0 to 4π.

The output signals of the five simple cells shown in Figure 4 are modeled by

$$\begin{aligned}
\tilde{x}_1(t) &:= (4 + a_0(t)) \sin(t + 4\,\phi_0(t)) \,, && (35)\\
\tilde{x}_2(t) &:= (4 + a_1(t)) \sin(t + 4\,\phi_1(t)) \,, && (36)\\
\tilde{x}_3(t) &:= (4 + a_1(t)) \sin(t + 4\,\phi_1(t) + \pi/4) \,, && (37)\\
\tilde{x}_4(t) &:= (4 + a_1(t)) \sin(t + 4\,\phi_1(t) + \pi/2 + 0.5\phi_D(t)) \,, && (38)\\
\tilde{x}_5(t) &:= (4 + a_1(t)) \sin(t + 4\,\phi_1(t) + 3\pi/4 + 0.5\phi_D(t)) \,, && (39)
\end{aligned}$$

$t \in [0, 4\pi]$. All signals have a length of 512 data-points, resulting in a step size of $\Delta t = 4\pi/512$ between successive data-points. The amplitude and phase modulation signals are low-pass filtered Gaussian white noise normalized to zero mean and unit variance. The width σ of the Gaussian low-pass filters is 30 data-points for $\phi_D(t)$ and 10 for the other four signals, $a_0(t)$, $a_1(t)$, $\phi_0(t)$, and $\phi_1(t)$. Since the amplitude signals $a_0(t)$ and $a_1(t)$ have positive and negative values, an offset of 4 was added to shift these signals to a positive range, as required for amplitudes. The linear ramp t within the sine ensures that all phases occur equally often, otherwise a certain phase would be overrepresented because the phase signals $\phi_0(t)$ and $\phi_1(t)$ are concentrated around zero. The factor 4 in front of the phase signals ensures that phase changes more quickly than amplitude. The raw signals $\tilde{x}_1(t), ..., \tilde{x}_5(t)$ are normalized to zero mean and unit variance yielding $x_1(t), ..., x_5(t)$. Five additional signal components are obtained by a time delay of 1 data-point, namely $x_6(t) := x_1(t - \Delta t), ..., x_{10}(t) := x_5(t - \Delta t)$. Some of these ten signals are shown in Figure 4, bottom, together with some trajectory plots of one component versus another. The η-value of each signal as given by Equation (34) is shown in the upper right corner of the signal plots.

Since the first simple cell has a different orientation and location than the others, it also has a different amplitude and phase modulation. This makes it independent, as can be seen from trajectory plot $x_2(t)$ versus $x_1(t)$, which does not show any particular structure. The first simple cell only serves as a distractor which needs to be ignored by SFA.

The second and third simple cell have the same location, orientation, and frequency. They therefore have the same amplitude and phase modulation. However, the positive and negative subregions of the receptive fields are slightly shifted relative to each other. This results in a constant phase difference of $45°(\pi/4)$ between these two simple cells. This is reflected in the trajectory plot $x_3(t)$ versus $x_2(t)$ by the elliptic shape (Fig. 4). Notice that a phase difference of $90°(\pi/2)$ would be computationally more convenient, but is not necessary here. If the two simple cells had a phase difference of 90°, like sine- and cosine-Gabor wavelets, the trajectory would describe circles and the square of the desired complex cell response $a_1(t)$ would be just the square sum $x_2^2(t) + x_3^2(t)$. Since the phase difference is 45°, the transformation needs to be slightly more complex to represent the elliptic shape of the $x_3(t)$-$x_2(t)$-trajectory.

The fourth and fifth simple cells have the same relationship as the second and third one do, but for the right eye instead of the left eye. If the two eyes received identical input, the third and and fifth simple cell would also have this relationship, i.e. same amplitude and phase modulation with a constant phase difference. However, disparity induces a shift of the right image versus the left image which results in an additional slowly varying phase difference $\phi_D(t)$. This leads to the slowly varying shape of the ellipses in trajectory plot $x_5(t)$ versus $x_3(t)$, varying back and forth between slim left-oblique ellipses over circles to slim right-oblique ellipses. This phase difference between simple cell responses of different eyes can be used to estimate disparity (Theimer & Mallot, 1994).

Since $x_{10}(t)$ is a time delayed version of $x_5(t)$, the vertical distance from the diagonal in the trajectory plot $x_{10}(t)$ versus $x_5(t)$ in Figure 4 is related to the time derivative of $x_5(t)$. Just as the phase difference between corresponding simple cells of different eyes can be used to estimate disparity, phase changes over time can be used to estimate direction and speed of motion of the visual stimulus (Fleet & Jepson, 1990).

In the first example $x_1(t), x_2(t)$, and $x_3(t)$ serve as an input signal. Only the common amplitude modulation $a_1(t)$ of $x_2(t)$ and $x_3(t)$ represents a slow feature and can be extracted from this signal. Example 2 uses all five normalized simple cell responses, $x_1(t), ..., x_5(t)$, plus the time delayed versions of them, $x_6(t) := x_1(t - \Delta t), ..., x_{10}(t) := x_5(t - \Delta t)$. Several different slow features, such as motion and disparity, can be extracted from this richer signal.

Example 1: Consider the normalized simple cell responses $x_1(t), x_2(t)$, and $x_3(t)$ as an input signal for SFA. Figure 5, top, shows the input signal trajectories already seen in Figure 4, cross-sections through the first three components of the learned input-output function $\mathbf{g}(\mathbf{x})$ (arguments not varied are not listed and set to zero, e.g. $g_1(x_2, x_3)$ means $g_1(0, x_2, x_3)$), and the first three components of the extracted output signal $\mathbf{y}(t)$. The first component of the input-output function represents the elliptic shape of the $x_3(t)$-$x_2(t)$-trajectory correctly and ignores the first input signal component $x_1(t)$. It therefore extracts the amplitude modulation $a_1(t)$ (actually the square of it) as desired and is insensitive to the phase of the stimulus (cf. Fig. 5, bottom right). The correlation[2] r between $a_1(t)$ and $-y_1(t)$ is 0.99. The other input-output function and output signal components are not related to slow features and can be ignored. This becomes clear from the η-values of the output signal components (Fig. 5, bottom left). Only $\eta(y_1)$ is significantly lower than those of the input signal. Phase cannot be extracted, because it is a cyclic variable. A reasonable representation of a cyclic variable would be its sine and cosine value, but this would be almost the original signal and is therefore not a slow feature. When trained and tested on signals of length 16π (2048 data-points), the correlation is 0.981 ± 0.004 between $a_1(t)$ and $\pm y_1(t)$ (training data) and 0.93 ± 0.04 between $a_1'(t)$ and $\pm y_1'(t)$ (test data) (means over 10 runs $\pm$ standard deviation).

Example 2: The previous example can be extended to binocular input and the time domain. The input signal is now given by all normalized simple cell responses described above and the time delayed versions of them, i.e. $x_1(t), ..., x_{10}(t)$. Since the input is 10-dimensional, the output signal of a second-degree polynomial SFA can be potentially 65-dimensional. However, singular value decomposition detects two dimensions with zero variance, i.e. the non-linearly expanded signal has two redundant dimensions, so that only 63 dimensions remain. The η-values of the 63 output signal components are shown in Figure 6, top left. Interestingly, they grow almost linearly with index value j. Between 10 and 16 components vary significantly more slowly than the input signal. This means that in contrast to the previous example there are now several slow features being extracted from the input signal. These include elementary features, such as amplitude (or complex cell response) $a_1(t)$ and disparity $\phi_D(t)$, but also more complex ones, such as phase change $\dot{\phi}_1(t)$ (which is closely related to the velocity of stimulus movement), the square of disparity $\phi_D^2(t)$, or even the product $\phi_D(t)\dot{a}_1(t)$ and other nonlinear combinations of the phase and amplitude signals that were used to generate the input signal. If so many slow features are extracted, the

[2] A correlation coefficient r between two signals $a(t)$ and $b(t)$ is defined as $r(a, b) := \langle (a - \langle a \rangle)(b - \langle b \rangle) \rangle / \sqrt{\langle (a - \langle a \rangle)^2 \rangle \langle (b - \langle b \rangle)^2 \rangle}$. Since the signs of output signal components are arbitrary, they will be corrected if required to obtain positive correlations, which is particularly important for averaging over several runs of an experiment.

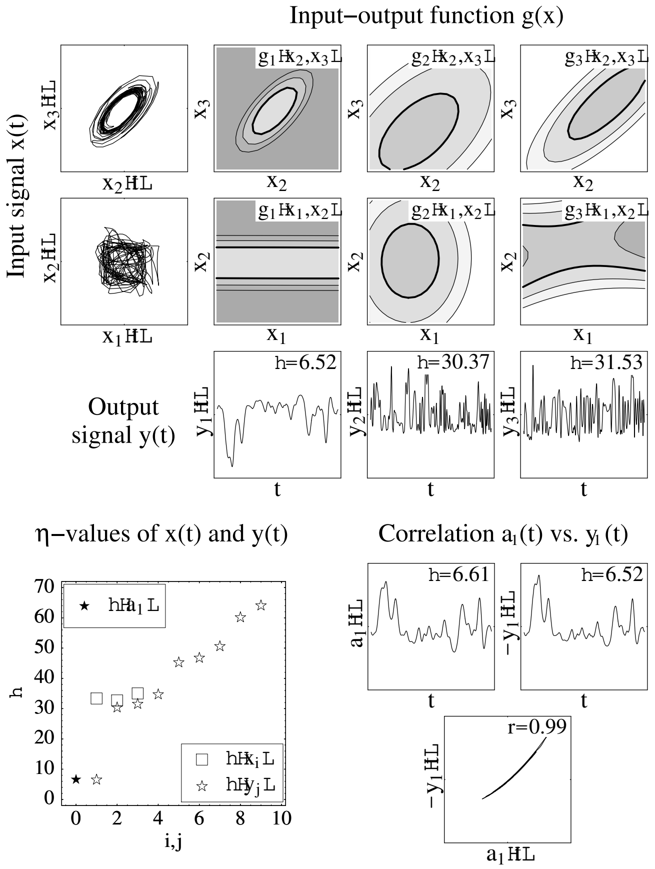


Figure 5 (ad Example 1): Learning to extract complex cell response $a_1(t)$ from normalized simple cell responses $x_1(t), x_2(t)$, and $x_3(t)$. **Top:** First three components of the learned input-output function $\mathbf{g}(\mathbf{x})$ which transforms the input signal $\mathbf{x}(t)$ into the output signal $\mathbf{y}(t)$. All signal components have unit variance and all graphs range from -4 to $+4$, including the grey-value scale of the contour plots. The thick contour lines indicate the value 0, the thin ones indicate ± 2 and ± 4; white is positive. Time axes range from 0 to 4π. **Bottom left:** η-values of the input and the output signal components. Only y_1 is a relevant invariant. $\eta(a_1)$ is shown at $i, j = 0$ for comparison. **Bottom right:** $-y_1(t)$ correlates well with amplitude $a_1(t)$.

decorrelation constraint is not sufficient to isolate the slow features into separate components. For instance, even though the motion signal $\dot{\phi}_1(t)$ is mainly represented by $-y_8(t)$ with a correlation of 0.66, part of it is also represented by $-y_5(t)$ and $-y_6(t)$ with correlations of 0.34 and 0.33, respectively. This is inconvenient but in many cases not critical. For example, for a repeated application of SFA, such as in the following examples, the distribution of slow features over output signal components is irrelevant as long as they are concentrated in the first components. We can therefore ask how well the slow features are encoded by the first 14 output components, for instance. Since the output signals obtained from training data form an orthonormal system, the optimal linear combination to represent a normalized slow signal $s(t)$ is given by

$$\tilde{Y}_{14}[s](t) := \sum_{j=1}^{14} \langle s\, y_j \rangle \; y_j(t) \ , \tag{40}$$

where $\tilde{Y}_l$ indicates a projection operator onto the first l output signal components. Normalization to zero mean and unit variance yields the normalized projected signal $Y_{14}[s](t)$. The correlation of $\dot{\phi}_1(t)$ with $Y_{14}[\dot{\phi}_1](t)$ is 0.94.

Figure 6 and Table 1 give an overview over some slow feature signals and their respective correlations r with their projected signals. Since the test signals have to be processed without explicit knowledge about the slow feature signal s' to be extracted, the linear combinations for test signals are computed with the coefficients determined on the training signal and we write $\tilde{Y}'_{14}[s](t) := \sum_{j=1}^{14} \langle s\, y_j \rangle \; y'_j(t)$. Normalization yields $Y'_{14}[s](t)$.

4.2 Example 3: Repeated SFA

The first two examples were particularly easy because a second-degree polynomial was sufficient to recover the slow features well. The third example is more complex. First generate two random time series, one slowly varying $x_s(t)$ and one fast varying $x_f(t)$ (Gaussian white noise, low-pass filtered with $\sigma = 10$ data-points for x_s and $\sigma = 3$ data-points for x_f). Both signals have a length of 512 data-points, zero mean, and unit variance. They are then mixed to provide the raw input signal $\tilde{\mathbf{x}}(t) := [x_f, \sin(2x_f) + 0.5x_s]^T$ which is normalized to provide $\mathbf{x}(t)$. The task is to extract the slowly varying signal $x_s(t)$.

The input-output function required to extract the slow feature x_s cannot be well approximated by a polynomial of degree two. One might therefore use polynomials of third or higher degree. However, one can also repeat the learning algorithm, applying it with second degree polynomials, leading to input-output functions of degree two, four, eight, sixteen etc. To avoid an explosion of signal dimensionality, only the first few components of the output signal of one SFA are used as an input for the next SFA. In this example only three components were propagated from one SFA to the next. This cuts down the computational cost of this iterative scheme significantly compared to the direct scheme of using polynomials of higher degree, at least for high-dimensional input signals, for which polynomials of higher degree become so numerous that they would be computationally prohibitive.

Data			$s=\phi_D$	$s=\phi_D^2$	$s=\dot\phi_D$	$s=a_1$	$s=\phi_1$	$s=\dot\phi_1$	$s=\phi_D\dot a_1$
Test.		$\eta(s')$	22 2	32 3	37 1	66 4	66 2	113 2	114 5
Test.	SFA^1	$r(s',Y'_{10}[s])$	-.03 .07	-.03 .07	-.02 .04	-.00 .08	.01 .05	-.01 .04	-.00 .06
Train.	SFA^2	$j^* \leq 14$	1.0 0.0	2.1 0.3	2.8 0.4	4.0 0.0	6.0 4.2	7.9 1.2	8.7 2.1
Test.	SFA^2	$r(s',\pm y'_{j^*})$	.87 .02	.80 .07	.78 .11	.96 .01	.04 .02	.53 .12	.58 .12
Test.	SFA^2	$r(s',Y'_{14}[s])$	.88 .01	.89 .03	.89 .01	.98 .00	.02 .04	.85 .02	.89 .03
Train.	SFA^2	$r(s,Y_{14}[s])$	.90 .01	.91 .02	.90 .01	.99 .00	.22 .06	.87 .02	.91 .03

Table 1 (ad Example 2): η-values and correlations r for various slow feature signals s under different experimental conditions. SFA^1 and SFA^2 indicate linear and quadratic SFA, respectively. j^* is the index of that output signal component y_{j^*} among the first fourteen which is best correlated with s on the training data. $Y_l[s]$ is an optimal linear combination of the first l output signal components to represent s (see Eq. 40). All figures are means over 10 simulation runs with the standard deviation given in small numerals. Signal length was always 4096. We were particularly interested in the signals ϕ_D (disparity), a_1 (complex cell response), and $\dot\phi_1$ (indicating stimulus velocity), but several other signals get extracted as well, some of which are shown in the table. Notice that linear SFA with ten input signal components can only generate ten output signal components and is obviously not sufficient to extract any of the considered feature signals. For some feature signals, such as ϕ_D and a_1, it is sufficient to use only the single best correlated output signal component. Others are more distributed over several output signal components, e.g. $\dot\phi_1$. Phase ϕ_1 is given as an example of a feature signal that cannot be extracted, since it is a cyclic variable.

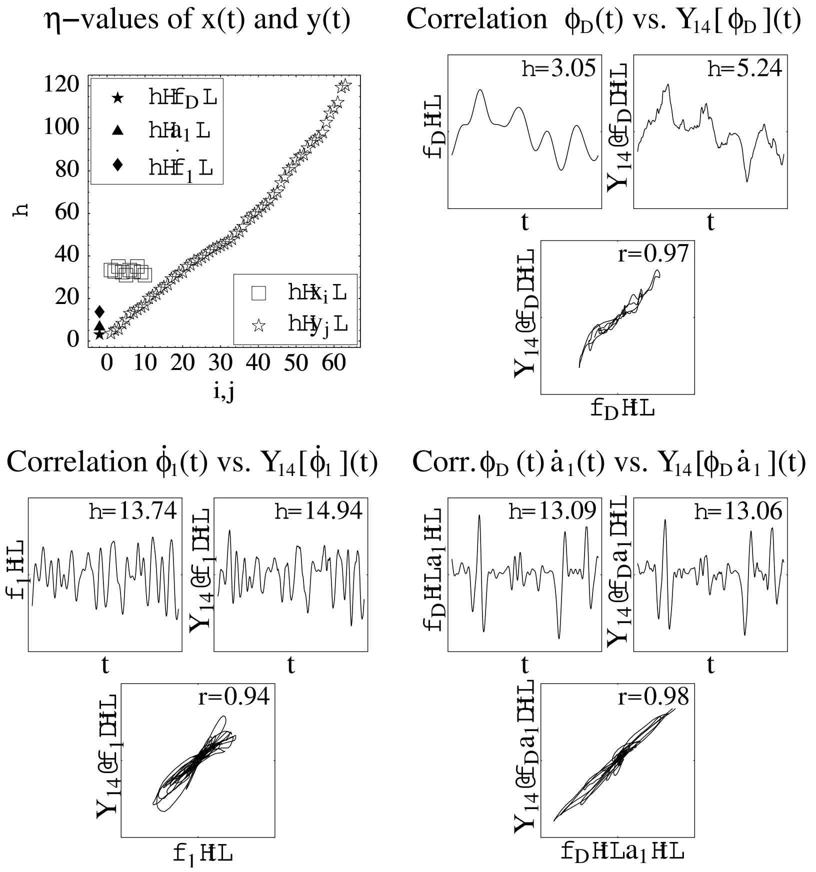
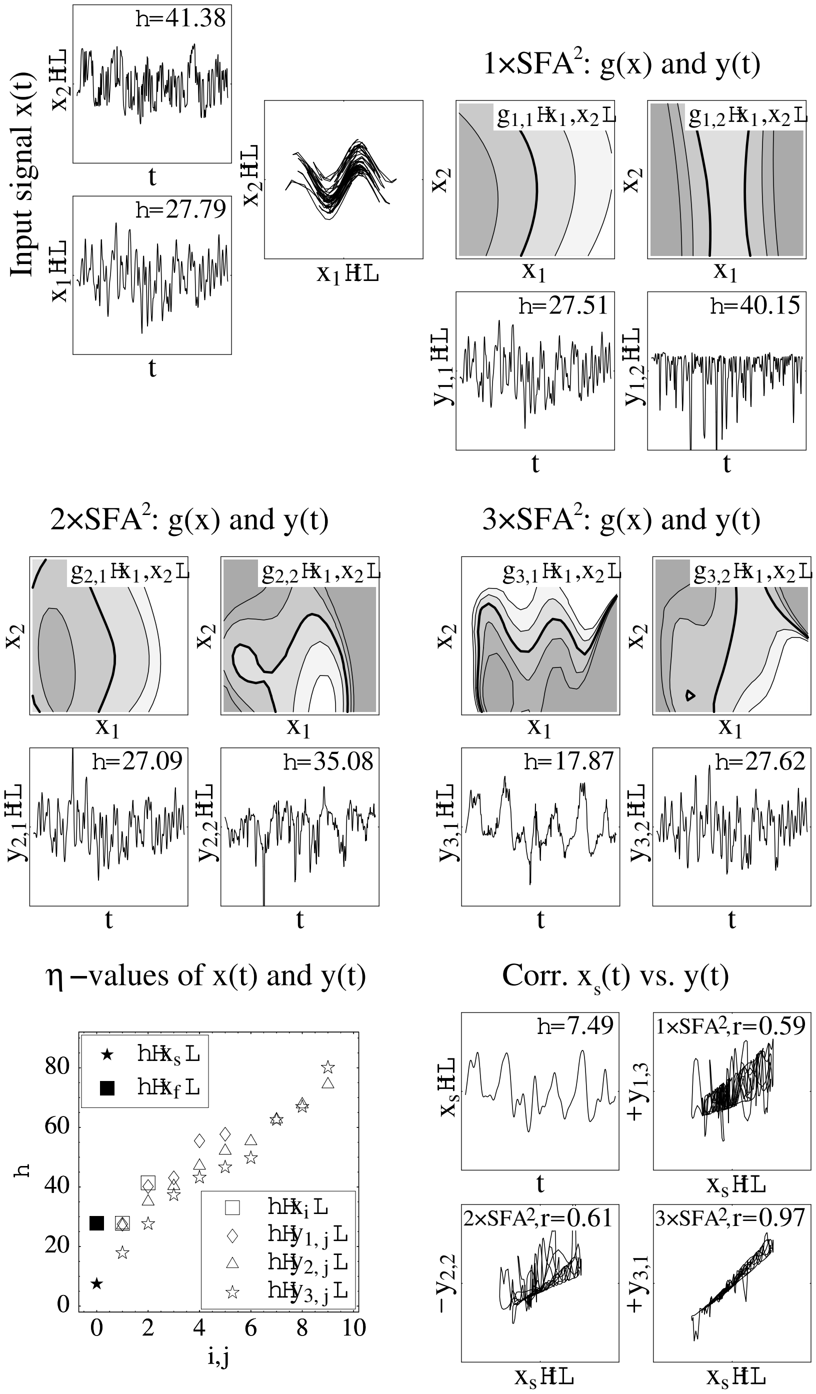


Figure 6 (ad Example 2): Learning to extract several slow features from normalized simple cell responses $x_1, x_2, ..., x_{10}$. **Top left:** η-values of the input and the output signal components. The first 14 output components were assumed to carry relevant information about slow features. **Top right, bottom left and right:** The first 14 output components contain several slow features: elementary ones, such as disparity ϕ_D and phase variation $\dot{\phi}_1$ (which is closely related to motion velocity), and combinations of these, such as the product of disparity and amplitude change $\phi_D \dot{a}_1$. Shown are the slow feature signals, the optimal linear combinations of output signal components, and the correlation trajectories of the two.



See next page for figure caption

Figure 7 (ad Example 3): Hidden slowly varying feature signal discovered by three SFAs in succession. **Top left:** Input signal. **Top right** and **middle left and right**: One, two, and three SFAs in succession and their corresponding output signals. First subscript refers to the number of the SFA, second subscript refers to its component. The slow feature signal $x_s(t)$ is the slow up and down motion of the sine curve in the trajectory plot $x_2(t)$ vs. $x_1(t)$. **Bottom left:** η-values of the input signal components and the various output signal components. $\eta(x_s)$ and $\eta(x_f)$ are shown at $i, j = 0$ for comparison. Only $y_{3,1}$ can be considered to represent a slow feature. **Bottom right:** Slow feature signal $x_s(t)$ and its relation to some output signal components. The correlation of $x_s(t)$ with $+y_{1,3}(t)$, $-y_{2,2}(t)$, and $+y_{3,1}(t)$ is 0.59, 0.61, and 0.97, respectively.

Figure 7 shows input signal, input-output functions and output signals for three SFAs in succession (only the first two components are shown). The plotted input-output functions always include the transformations computed by previous SFAs. The approximation of $\mathbf{g}(\mathbf{x})$ to the sinusoidal shape of the trajectories becomes better with each additional SFA; compare, for instance, $g_{2,2}$ with $g_{3,1}$. The η-values shown at the bottom left indicate that only the third SFA extracts a slow feature. The first and second SFA did not yield an η-value lower than that of the input signal. For the first, second, and third SFA, $x_s(t)$ was best correlated with the third, second (with inverted sign), and first output signal component, with correlation coefficients of 0.59, 0.61, and 0.97, respectively. The respective trajectory plots are shown at the bottom right of Figure 7. Notice that each SFA was trained in an unsupervised manner and without back-propagating error signals. Results for longer signals and multiple simulation runs are given in Table 2.

The algorithm can extract not only slowly varying features but also rarely varying ones. To demonstrate this, we tested the system with a binary signal which occasionally switches between two possible values. To generate a rarely changing feature signal $x_r(t)$ we applied the sign function to low-pass filtered Gaussian white noise and normalized the signal to zero mean and unit variance. The σ of the Gaussian low-pass filter was chosen to be 150 to make the η-values of $x_r(t)$ similar to those of $x_s(t)$ in Table 2. Figure 8 shows the rarely changing signal $x_r(t)$ and single best correlated output signal components of the three SFAs in succession. For the first, second, and third SFA, $x_r(t)$ was best correlated with the third, third, and first output signal component, with corresponding correlation coefficients of 0.43, 0.69, and 0.97. This is similar to the results obtained with the slowly varying feature signal $x_s(t)$.

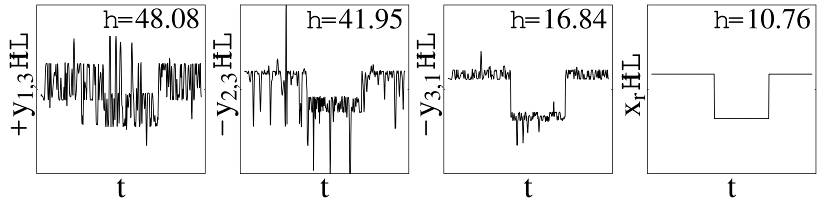


Figure 8 (ad Example 3): Hidden rarely changing feature signal discovered by three SFAs in succession.

Table 2 shows results on training and test data for slowly and rarely varying feature signals of length 4096. They confirm that there is no significant difference between slowly and rarely varying signals, given a comparable η-value. In both cases, even two quadratic SFAs in succession do not perform much better than one linear one. It is only the third quadratic SFA, which extracts the feature signal with a high correlation coefficient. As in the previous example, results improve if a linear combination of several (three) output signal components is used instead of just the single best one (this is strictly true only for training data, since on test data the linear combination from the training data is used).

Data		$1\times$SFA^1	$1\times$SFA^2	$2\times$SFA^2	$3\times$SFA^2	$4\times$SFA^2
Slowly varying signal $x_s(t)$ with $\bar{\eta} = 66$ 1						
Train.	$j^* \leq 3$	2.0 0.0	3.0 0.0	2.7 0.5	1.6 0.5	1.4 0.5
Test.	$r(x'_s, \pm y'_{j^*})$	.61 .02	.59 .02	.65 .08	.81 .10	.85 .11
Test.	$r(x'_s, Y'_3[x_s])$	.60 .02	.60 .02	.69 .07	.87 .06	.86 .10
Train.	$r(x_s, Y_3[x_s])$	.62 .01	.62 .01	.74 .05	.89 .05	.91 .05
Rarely changing signal $x_r(t)$ with $\bar{\eta} = 66$ 8						
Train.	$j^* \leq 3$	2.0 0.0	3.0 0.0	2.7 0.5	1.6 0.5	1.3 0.5
Test.	$r(x'_r, y'_{j^*})$	.60 .01	.57 .06	.59 .10	.84 .08	.81 .17
Test.	$r(x'_r, Y'_3[x_r])$	.60 .01	.58 .06	.67 .10	.87 .08	.85 .17
Train.	$r(x_r, Y_3[x_r])$	.61 .01	.62 .02	.73 .07	.91 .04	.94 .03

Table 2 (ad Example 3): Correlations r for slowly (cf. Fig. 7) and rarely (cf. Fig. 8) varying feature signals $x_s(t)$ and $x_r(t)$ with output signal components of several SFAs in succession. SFA^1 and SFA^2 indicate linear and quadratic SFA, respectively. j^* is the index of that output signal component y_{j^*} among the first three which is best correlated with s on the training data (in all cases j^* was also optimal for the test data). $Y_3[s]$ is an optimal linear combination of the first three output signal components to represent s (see Eq. 40). All figures are means over 10 simulation runs with the standard deviation given in small numerals. Signal length was always 4096.

4.3 Example 4: Translation Invariance in a Visual System Model

Network architecture

Consider now a hierarchical architecture as illustrated in Figure 9. Based on a one-dimensional model retina with 65 sensors, layers of linear SFA-modules with convergent connectivity alternate with layers of quadratic SFA-modules with direct connectivity. This division into two types of sub-layers allows us to separate the contribution of simple spatial averaging from the contribution of non-linear processing to the generation of a translation invariant object representation. It is clear from the architecture that the receptive field size increases from bottom to top and therefore the units become potentially able to respond to more complex features, two properties characteristic for the visual system (Oram & Perrett, 1994).

Training the network

In order to train the network to learn translation invariance, the network must be exposed to objects moving translationally through the receptive field. We assume that these objects create fixed one-dimensional patterns moving across the retina with constant speed and same direction. This ignores scaling, rotation, etc., which will be investigate in the next

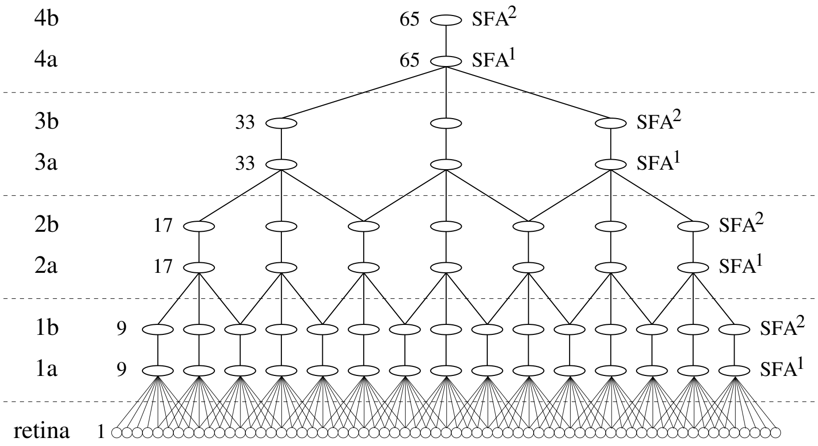


Figure 9 (ad Examples 4 and 5): A hierarchical network of SFA-modules as a simple model of the visual system learning translation and other invariances. Different layers correspond to different cortical areas. For instance, one could associate layers 1, 2, 3, and 4 with areas V1, V2, V4, and PIT, respectively. The retinal layer has 65 input units, representing a receptive field which is only a part of the total retina. The receptive field size of SFA-modules is indicated at the left of the leftmost modules. Each layer is split into an a and a b sub-layer for computational efficiency and to permit a clearer functional analysis. The a-modules are linear and receive convergent input, either from nine units in the retina or from three neighboring SFA-modules in the preceding layer. The b-modules are quadratic and receive input from only one SFA-module. Thus, receptive field size increases only between a b-module and an a-module but not vice versa. The number of units per SFA-module is variable and the same for all modules. Only the layer $4b$ SFA-module always has 9 units to make output signals more comparable. Notice that each module is independent of the neighboring and the succeeding ones; there is no weight sharing constraint and no backpropagation of error.

section. The effects of additive noise are studied here. The training and test patterns were low-pass filtered Gaussian white noise. The size of the patterns was randomly chosen from a uniform distribution between 15 and 30 units. The low-pass filter was a Gaussian with a width σ randomly chosen from a uniform distribution between 2 and 5. Each pattern was normalized to zero mean and unit variance to eliminate trivial differences between patterns, which would make object recognition easier at the end. The patterns were always moved across the retina with a constant speed of 1 spatial unit per time unit. Thus, for a given pattern and without noise the sensory signal of a single retinal unit is an accurate image of the spatial grey-value profile of the pattern. The time between one pattern being in the center of the retina and the next one was always 150 time units. Thus there was always a pause between the presentation of two patterns and there were never two patterns visible simultaneously. The high symmetry of the architecture and the stimuli made it possible to compute the input-output function only for one SFA-module per layer. The other modules in the layer would have learned the same input-output function, since they saw the same input, just shifted by a time delay determined by the spatial distance of two modules. This cut down computational costs significantly. Notice that this symmetry also results in an implicit weight sharing constraint, although this was not explicitly implemented; see the next section for more general examples. The sensory signals for some training and test patterns are shown in Figure 10.

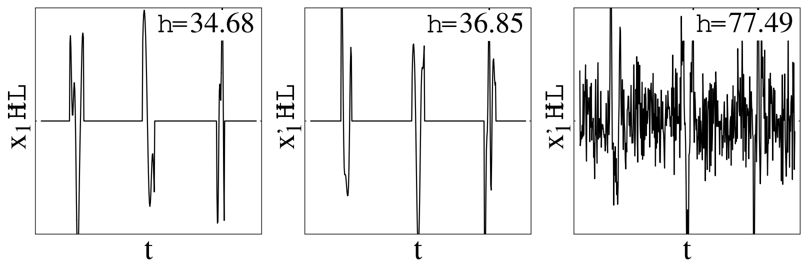


Figure 10 (ad Example 4): Sensory signal of the central retinal unit in response to three patterns. Since all patterns move with the same constant speed across the retina, the sensory signals have the same shape in time as the patterns have in space, at least for the noise-free examples. **Left:** First three 'patterns' of the training sensory signal. Pattern characteristics are (29, 4.0), (25, 4.5), and (16, 3.1), where the first numbers indicate pattern size and the second ones the width of the Gaussian low-pass filter used. **Middle:** First three 'patterns' of the test sensory signal without noise. Pattern characteristics are (22, 3.1), (26, 3.8), and (24, 2.5). **Right:** Same test sensory signal but with a noise level of 1. With noise the sensory signals of different retinal units differ in shape and not only by a time delay.

In the standard parameter setting each SFA-module had 9 units; a stimulus with 20 patterns was used for training and a stimulus with 50 patterns was used for testing; no noise was added. To improve generalization to test patterns, all signals transfered from one SFA-module to the next were clipped at ± 3.7 to eliminate extreme negative and positive values. The limit of ± 3.7 was not optimized for best generalization but chosen such that clipping became visible in the standard display with a range of ± 4, which is not relevant for the figures presented here.

What does the network learn?

Figure 11 shows the first four components of the output signals generated by the trained network in response to three patterns of the training and test stimulus.

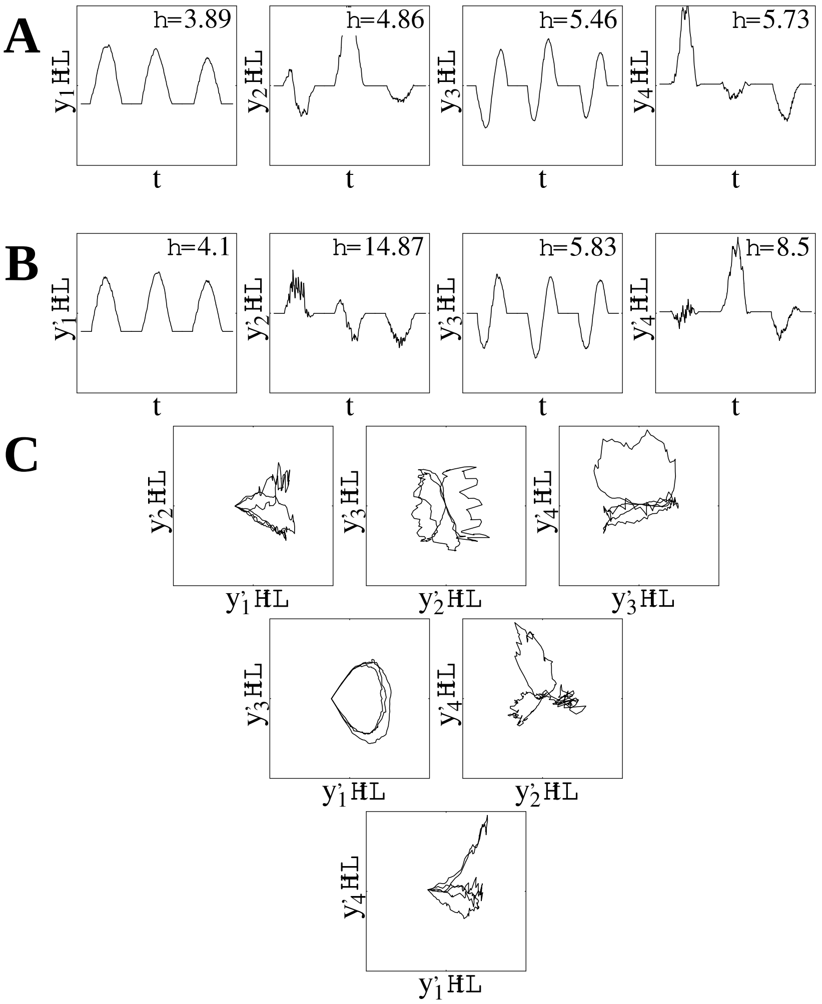


Figure 11 (ad Example 4): First four output signal components in response to three patterns of (**A**) the training signal and (**B**) the test signal. (**C**) shows all respective trajectory plots for the test signal. Time axes range from 0 to 3×150, all other axes range from -4 to +4. All signals are approximately normalized to zero mean and unit variance.

- **Instantaneous feed-forward processing:** Once the network is trained, processing is instantaneous. The output signal can be calculated moment by moment and does not require a continuously changing stimulus. This is important, because it permits processing also of briefly flashed patterns. However, it is convenient for display purposes to show the response to moving patterns.

- **No overfitting:** The test output signal does not differ qualitatively from the training output signal, although it is a bit noisier. This indicates that 20 patterns are sufficient for training to avoid overfitting. However, clipping the signals as indicated above is crucial here. Otherwise, signals that get slightly out of the usual working range are quickly amplified to extreme values.

- **Longer responses:** The response of a layer 4b unit to a moving pattern is longer than the response of a retinal sensor (compare Figs. 10 and 11). This is due to the larger receptive field size. For a pattern of size 20, a retinal sensor response has a length of 20, given a stimulus speed of 1 spatial unit per time unit. A layer 4b unit responds to this pattern as soon as it moves into the receptive field and until it leaves it, resulting in a response of length 84 = 65 (receptive field size) + 20 (pattern size) - 1.

- **Translation invariance:** The responses to individual patterns have a shape similar to a half or a full sine wave. It may be somewhat surprising that the most invariant response should be half a sine wave and not a constant as suggested in Figure 1. But the problem with a constant response would be that it would require a sharp onset and offset as the pattern moves into and out of the receptive field. Half a sine wave is a better compromise between signal constancy in the center and smooth onsets and offsets. Thus the output signal tends to be as translation invariant as possible under the given constraints, invariance being defined by Equation (1).

- ***Where*-information:** Some components, such as the first and the third one, are insensitive to pattern identity. Component 1 can thus be used to determine whether a pattern is in the center or periphery of the receptive field. Similarly, Component 3 can be used to determine whether a pattern is more on the left or the right side. Taken together, Components 1 and 3 represent pattern location, regardless of other aspects of the pattern. This becomes particularly evident from trajectory plot $y_3'(t)$ vs. $y_1'(t)$. These two components describe a loop in phase space, each point on this loop corresponding to a unique location of the pattern in the receptive field. $y_1(t)$ and $y_3(t)$ therefore represent *where*-information.

- ***What*-information:** Some components, such as the second and fourth one, do distinguish well between different patterns, despite the translation invariance. The response to a certain pattern can be positive or negative, strong or weak. A reasonable representation for pattern identity can therefore be constructed as follows. Take Components 1, 2, and 3, and subtract the baseline response. This yields a 3-dimensional response vector for each moment in time, which is zero if no pattern is present in the receptive field. As a pattern moves through the receptive field the amplitude of the response vector increases and decreases but its direction tends to change little. The direction of the response vector is therefore a reasonable representation of pattern identity. This can be seen in the three trajectory plots $y_2'(t)$ vs. $y_1'(t)$, $y_4'(t)$ vs. $y_1'(t)$, and $y_4'(t)$ vs. $y_2'(t)$. Idealy the response vector should describe a straight line going from the baseline origin to some extreme point and then back again to the origin. This is what was observed on training data if only a few patterns were used for training. When training was done with 20 pattern, the response to test data formed noisy loops rather than straight lines. We will later investigate quantitatively to what extent this representation permits translation invariant pattern recognition.

Two aspects of the output signal were somewhat surprising. First, why does the network generate a representation that distinguishes between different patterns, even though the only objective of slow feature analysis is to generate a slowly varying output? Secondly, why do *where*- and *what*-information get represented in separate components, even though this distinction has neither been built in the algorithm nor in the network architecture? This question is particularly puzzling since we know from the second example that SFA tends to distribute slow features over several components. Another apparent paradox is the fact that *where*-information can be extracted at all by means of the invariance objective, even though the general notion is, that one wants to ignore pattern location when learning translation invariance (cf. Sec. 1).

To give an intuitive answer to the first question assume that the optimal output components would not distinguish between different patterns. The experiments suggest that the first component would then have approximately half sine wave responses for all patterns; the second component would have full sine wave responses for all patterns, since a full sine wave is uncorrelated to a half sine wave and still slowly varying. It is obvious that a component with half sine wave responses with different positive and negative amplitudes for different patterns can also be uncorrelated to the first component but is more slowly varying than the component with full sine wave responses only, which is in contradiction to the assumption. Thus the objective of slow variation in combination with the decorrelation constraint leads to components differentiating between different patterns.

Why *where*- and *what*-information gets represented in separate components is a more difficult issue. The first *where*-component, $y_1(t)$, with half sine wave responses is probably distinguished by its low η-value, because it is easier for the network to generate smooth responses if they do not distinguish between different patterns, at least for larger numbers of training patterns. Notice that the *what*-components, $y_2(t)$ and $y_4(t)$, are noisier. It is unclear why the second *where*-component, $y_3(t)$, emerges so reliably (although not always as the third component) even though its η-value is comparable to that of other *what*-components with half sine wave responses. It should also be mentioned that for some parameter regimes, e.g. fewer patterns or smaller distances between patterns, no explicit *where*-components emerge. However, more important than the concentration of *where*-information in isolated components is the fact that the *where*-information gets extracted at all by the same mechanism as the *what*-information, regardless of whether explicit *where*-components emerged.

It is interesting to compare the pattern representation of the network with the one sketched in Figure 1. We have mentioned above that the sharp onsets and offsets of the sketched representation are avoided by the network, leading to typical responses in the shape of a half or full sine wave. Interestingly, however, if one divides Components 2 or 4 by Component 1 (all components taken minus their resting value), one obtains signals similar to that suggested in Figure 1 for representing pattern identity, namely signals that are fairly constant and pattern specific if a pattern is visible and undetermined otherwise. If one divides Component 3 by Component 1, one obtains a signal similar to those suggested in Figure 1 for representing pattern location, namely one that is monotonically related to pattern location and undetermined if no pattern is visible.

How translation invariant is the representation?

We have argued above that the direction of the response vector is a good representation of the patterns. How translation invariant is this representation? To address this question, we have measured the angle between the response vector of a pattern at a reference location and at all other valid locations. The location of a pattern is defined by the location of its center (or the right center pixel if the pattern has an even number of pixels). The retina has a width of 65, ranging from -32 to +32 with the central sensor serving as the origin, i.e. location 0. The largest patterns of size 30 are at least partially visible from location -46 up to +47. The standard location for comparison is -15. The response vector is defined as a subset of output components minus their resting values. For the output signal of Figure 11, for instance, the response vector may be defined as $\mathbf{r}(t) := [y_1(t) - y_1(0), y_2(t) - y_2(0), y_4(t) - y_4(0)]^T$, if Components 1, 2, and 4 are taken into account and if at time $t = 0$ no pattern was visible and the output at resting level. If not stated otherwise, all components out of the nine output signal components that were useful for recognition were taken for the response vectors. In this Example 4 the useful components were always determined on the training data based on recognition performance. Since at a given time t_{pl} the stimulus shows a unique pattern p at a unique location l we can also parameterize the response vector by pattern index p and location l and write $\mathbf{r}_p(l) := \mathbf{r}(t_{pl})$. The angle between the response vectors of pattern p at the reference location -15 and pattern p' at a test location l is defined as

$$\vartheta(\mathbf{r}_p(-15), \mathbf{r}_{p'}(l)) := \arccos(\,(\mathbf{r}_p(-15) \cdot \mathbf{r}_{p'}(l))\,/\,(\|\,\mathbf{r}_p(-15)\,\|\,\|\,\mathbf{r}_{p'}(l)\,\|)\,)\,, \tag{41}$$

where $r \cdot r'$ indicates the usual inner product and $\|\, r \,\|$ indicates the Euclidean norm. Figure 12 shows percentiles for the angles of the response vectors at a test location relative to a reference location for the 50 test patterns.

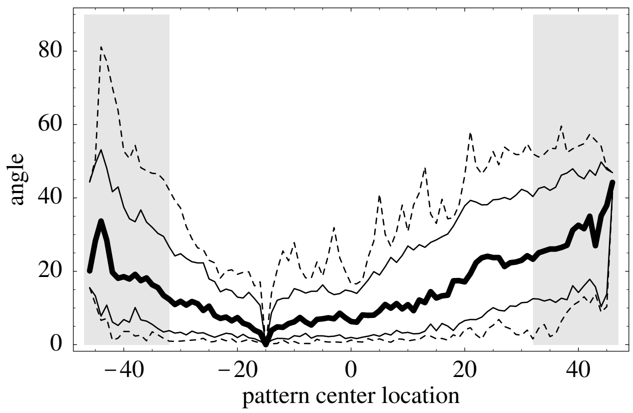


Figure 12 (ad Example 4): Percentiles of the angles $\vartheta(\mathbf{r}_p(-15), \mathbf{r}_p(l))$ between the response vectors at a test location l and the reference location -15 for 50 test patterns p. For this graph Components 1, 2, and 4 were used (cf. Fig. 11). The thick line indicates the median angle or 50%-percentile. The dashed lines indicate the smallest and largest angles. The bottom and top thin line indicate the 10%- and 90%-percentiles, respectively. The white area indicates the size of the retina (65 units). Gray areas indicate locations outside the retina. Patterns presented at the edge between white and grey are only half visible to the retina. The average angle over the white area is 12.6°. If the vectors were drawn randomly, the average and median angle would be 90° (except for the reference location where angles are always zero). Thus, the angles are relatively stable over different pattern locations. Results are very similar for the training patterns with an average angle of 11.3°.

What is the recognition performance?

If the ultimate purpose of the network is learning a translation invariant representation for pattern recognition, we should characterize the network in terms of recognition performance. This can be done by using the angles between response vectors as a simple similarity measure. But instead of considering the raw recognition rates we will characterize the performance in terms of the ranking of the patterns induced by the angles, since that is a more refined measure. Take the response vectors of all patterns at the reference location -15 as the stored models, which should be recognized. Then compare a response vector $\mathbf{r}_{p'}(l)$ of a pattern p' at a test location l with all stored models $\mathbf{r}_p(-15)$ in terms of the enclosed angles $\vartheta(\mathbf{r}_p(-15), \mathbf{r}_{p'}(l))$, which induce a ranking. If $\vartheta(\mathbf{r}_{p'}(-15), \mathbf{r}_{p'}(l))$ is smaller than all other angles $\vartheta(\mathbf{r}_p(-15), \mathbf{r}_{p'}(l))$ with $p \neq p'$, then pattern p' can be recognized correctly at location l and it is on rank 1. If 2 other patterns have a smaller angle than pattern p', then it is on rank 3, etc. Figure 13 illustrates how the ranking is induced and Figure 14 shows rank percentiles for all 20 training patterns and 50 test patterns over all valid locations.

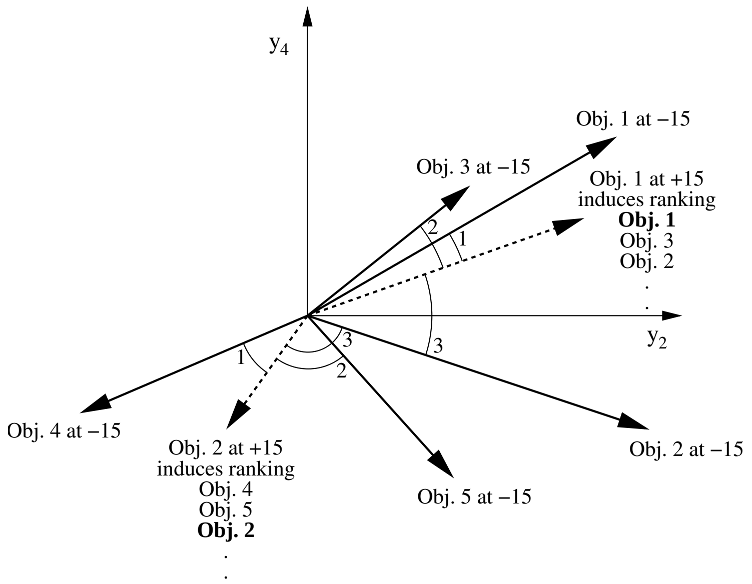


Figure 13 (ad Examples 4 and 5): Pattern recognition based on the angles of the response vectors. Solid arrows indicate the response vectors for objects at the reference location. Dashed arrows indicate the response vectors for objects shown at a test location. The angles between a dashed vector and the solid vectors induce a ranking of the stored objects. If the correct object is at rank one, the object is recognized correctly, as in case of Object 1 in the figure.

For the training patterns there is a large range within which the median rank is 1, i.e. at least 50% of the patterns would be correctly recognized. This even holds for location +15, where there is no overlap between the pattern at the test location and the pattern at the reference location. Performance degrades slightly for the test patterns. The average ranks over all test locations within the retina (the white area in the graph) is 1.9 for the training patterns and 6.9 for the test patterns. Since these average ranks depend on the number of patterns, we will in the following normalize them by subtracting 1 and dividing by the number of patterns minus 1. This gives a normalized average rank between 0 and 1, 0 indicating perfect performance (always rank 1) and 1 indicating worst performance (always last rank). The normalized average ranks for the graphs in Figure 14 are 0.05 and 0.12 for the training and test patterns, respectively; chance level were about 0.5. Notice that the similarity measure used here is simple and gives only a lower bound for the performance; a more sophisticated similarity measure can only do better.

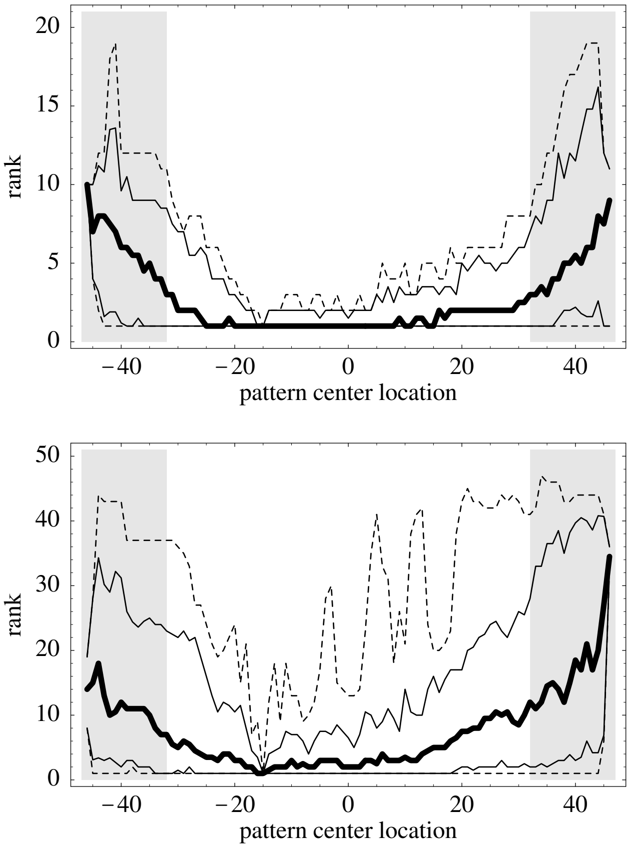


Figure 14 (ad Example 4): Rank percentiles for the 20 training patterns (top) and 50 test patterns (bottom) depending on the test location. For this graph the Components 1, 2, and 4 were used (cf. Fig. 11). Conventions are as for Figure 12. Performance is, of course, perfect at the reference location -15, but also relatively stable over a large range. The average rank over the white area is 1.9 for the training patterns and 6.2 for the test patterns. Perfect performance would be 1 in both cases and chance level would be approximately 10.5 and 25.5 for training and test patterns, respectively.

How does performance depend on number of training patterns, network complexity, and noise level?

Using the normalized average rank as a measure of performance for translation invariant pattern recognition, we can now investigate the dependence on various parameters. Figure 15 shows the dependence on the number of training patterns. One finds that 20 training patterns are enough to achieve good generalization; performance does not degrade dramatically down to five training patterns. This was surprising, since translation invariant recognition appears to be a fairly complex task.

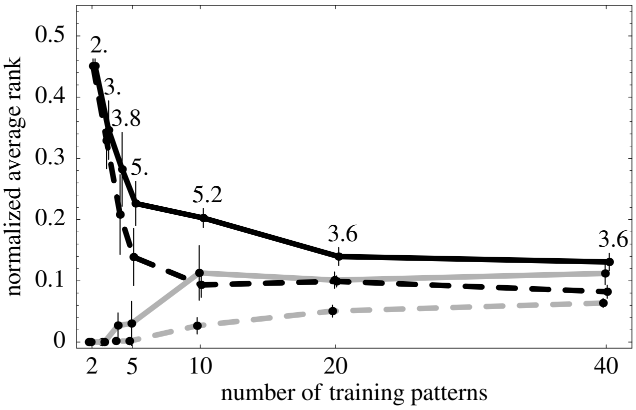


Figure 15 (ad Example 4): Dependence of network performance on the number of training patterns. The standard number was 20. The ordinate indicates normalized average rank; the abscissa indicates the considered parameter, i.e. the number of training patterns in this case. Gray and black curves indicate performance on training and test data, respectively. Solid curves indicate results obtained with two layer $4b$ output signal components only; dashed curves indicate results obtained with as many components out of the first nine components as were useful for recognition. The useful components were determined on the training data based on recognition performance. The average number of components used for the dashed curves are shown above. Each point in the graph represents the mean over five different simulation runs; the standard deviation is indicated by small bars. The curves are slightly shifted horizontally to let the standard deviation bars not overlap. 20 training patterns were sufficient and even five training patterns produced reasonable performance.

Figure 16 shows the dependence of network performance on the number of components used in the SFA-modules. The standard was nine for all modules. In this experiment the number of components propagated from one module to the next one was stepwise reduced down to just one. At the top level, however, usually nine components were used to make results more comparable. In case of 1 and 2 components per module, however, only 2 and 5 components were available in the layer $4b$ module, respectively. Performance degraded slowly down to 4 components per module; below that performance degraded quickly. With one component the normalized average rank was at chance level. One can expect that performance would improve slightly with more than nine components per module. With 1 and 2 components per module there were always two clear components in the output signal useful for recognition. With 3 components per module the useful information began to mix with components not useful, so that performance actually degraded from 2 to 3 components per module.

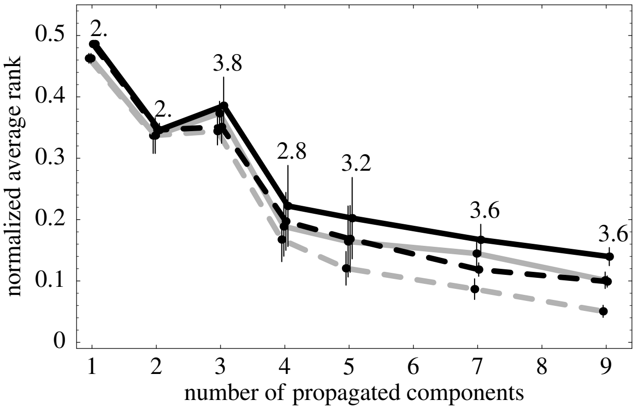


Figure 16 (ad Example 4): Dependence of network performance on the number of components propagated from one module to the next. Standard value was 9. Conventions are as for Figure 15. Performance degraded slowly down to 4 components per module and more quickly for fewer components.

SFA should not be too sensitive to noise, because noise would yield quickly varying components that are ignored. To test this, we added Gaussian white noise with different variance to the training and test signals. A sensory signal with a noise level of 1 is shown in Figure 10. The noise was independent only for nine adjacent sensors feeding into one SFA-module. Because in the training procedure only one module per layer was being trained, all modules in one layer effectively saw the same noise, but delayed by 4, 8, 12, etc. time units. However, since the network processed the input instantaneously, it is unlikely that the delayed reoccurence of noise could be used to improve the robustness. Figure 17 shows the dependence of performance on noise level. The network performance degraded gracefully.

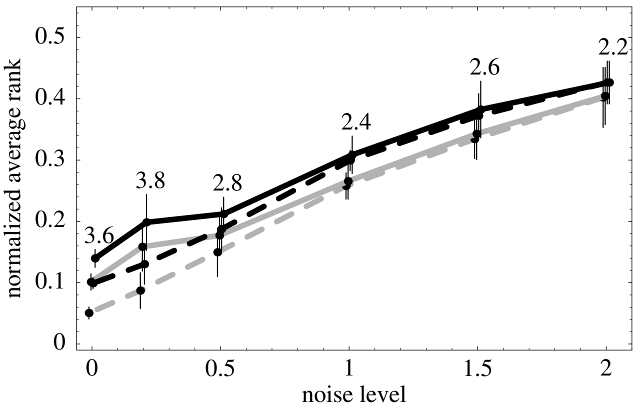


Figure 17 (ad Example 4): Dependence of network performance on the added noise level. Standard value was 0. Conventions are as for Figure 15. Performance degrades gracefully until it approaches chance level at a noise level of 2. A sensory signal with noise level 1 is shown in Figure 10

What is the computational strategy for achieving translation invariance?

How does the network achieve translation invariance? This question can be answered at least for a layer 1b SFA-module by visualizing its input-output function (not including the clipping operation from the 1a to the 1b module). At that level the input-output function is a polynomial of nine input components, e.g. $x_{-4}, x_{-3}, ..., x_4$, and degree 2, i.e. it is a weighted sum over all first and second degree monomials. Let us now define a center location and a spread for each monomial. For second degree monomials the spread is the difference between the indices of the two input components; its center location is the average over the two indices. For first degree monomials the spread is 0 and the center location is the index. Thus, $x_{-3}x_0$ has a spread of 3 and center location -1.5, x_3x_3 has a spread of 0 and center location 3, and x_{-2} has a spread of 0 and center location -2. Figure 18 shows the weights or coefficient values of the monomials for the first two components of the input-output function learned by the central layer $1b$ SFA-module including the contribution of the preceding $1a$ SFA-module. All other modules in the same layer learn the same input-output function just shifted by a multiple of 4 units. The graphs show that second degree monomials contributed significantly (signals generated by first and second degree monomials have similar variance, so that coefficient values of first and second degree monomials can be compared), which indicates that second order features were extracted. In general, coefficient values related to monomials of same degree and spread increase smoothly from perifery to center, forming bell-shaped or half sine wave-shaped curves. This results in a fade-in/fade-out mechanism applied to the extracted features as patterns move through the receptive field. The general mechanism by which the network learns approximate translation invariance is closely related to higher-order networks in which translation invariance is achieved by weight sharing among monomials of same degree and spread (see Bishop, 1995, 8.7.4 and references therein). The difference is that in those networks the invariance is built in by hand and without any fade-in/fade-out.

What is the relative contribution of the linear a-sublayers and the quadratic b-sublayers to the translation invariance? This can be inferred from the η-values plotted across layers. One can see in Figure 19 that a- and b-sublayers both contribute to the slow variation of the output signals although the linear a-sublayers seem to be slightly more important, at least beyond layer $1b$. This holds particularly for the first component, which is usually the half sine wave responding *where*-component, for which simple linear averaging seems to be a good strategy at higher levels.

Is there an implicit similarity measure between patterns?

The representation of a given pattern is similar at different locations: This is the learned translation invariance. But how does the network compare different patterns at the same location? What is the implicit similarity measure between patterns given that we compare the representations generated by the network in terms of the angles between the response vectors? Visual inspection did not give an obvious insight into what the implicit similarity measure would be and also comparisons with several other similarity measures were not conclusive. For example, the correlation coefficient between the maximum correlation between pairs of patterns over all possible relative shifts on the one hand and the angle between the response vectors (presenting the patterns in the center of the retina and using Components 1, 2, and 4) on the other hand was only -0.47 and the respective scatter plot was quite diffuse. There was no obvious similarity measure implicit in the network.

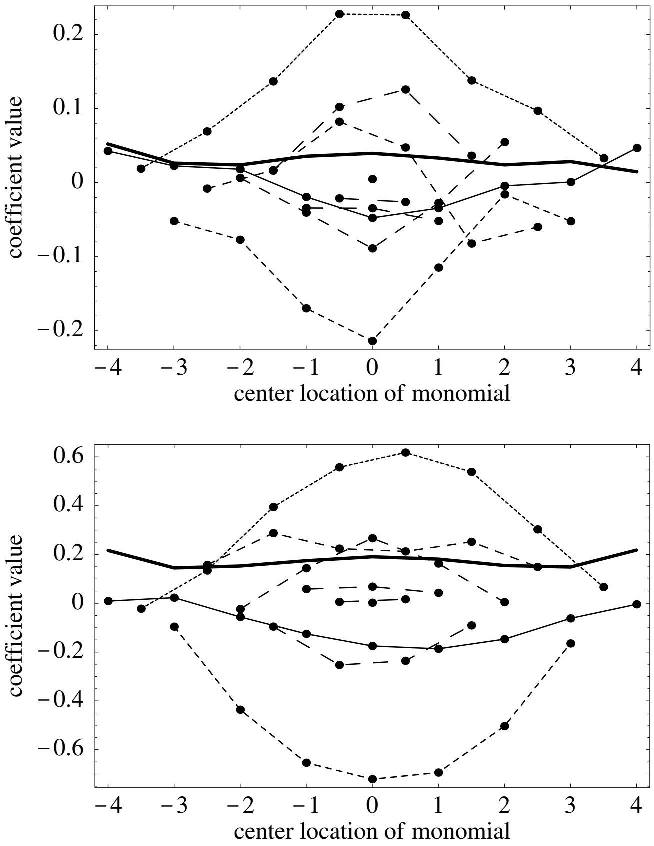


Figure 18 (ad Example 4): Visualization of the first two components, $g_1(\mathbf{x})$ (top) and $g_2(\mathbf{x})$ (bottom), of the input-output function realized by the central SFA-module of layer 1b. Black dots indicate the coefficient values of the second degree monomials at their center location. Dots related to monomials of same spread are connected by dashed lines, dense dashing indicating a small spread and wide dashing indicating a large spread. The spread can also be inferred from the number of dots connected, since there are nine monomials with spread 0, eight monomials with spread 1, etc. The thick line indicates the coefficient values of the first degree monomials. One can see that second order features get extracted and that coefficient values related to same spread vary smoothly with center location to achieve a slowly varying output signal, i.e. approximate translation invariance.

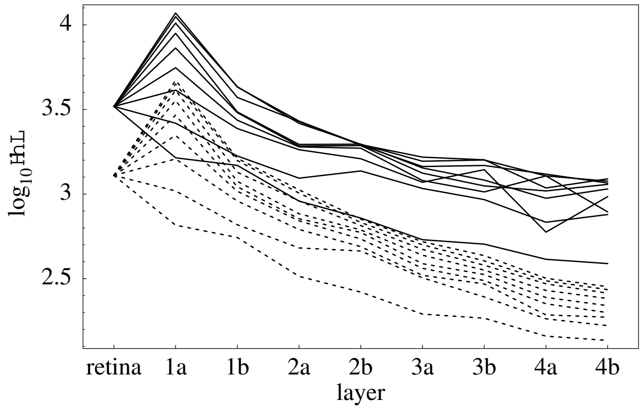


Figure 19 (ad Example 4): η-values of the nine signal components in the different layers averaged over five simulation runs. Dashed and solid curves indicate the training and test cases, respectively. The bottom curve in each of the two bunches indicates the first component with the lowest η-value, the top one the ninth component. A difference of $0.397 = \log_{10}(2.5)$ between the dashed and solid curves can be explained by the fact that there were 2.5 times more test than training patterns. The remaining difference and the fact that the solid curves are not monotonic and cross each other reflects limited generalization. The step from retina to layer $1a$ is linear and performes only decorrelation; no information is lost. Linear as well as nonlinear processing contribute to the translation invariance. For the first *where*-component, the linear convergent processing seems to be more important.

How well does the system generalize to other types of patterns?

We have addressed this question with an experiment using two different types of patterns, high-frequency patterns generated with a Gaussian low-pass filter with $\sigma = 2$ and low-frequency patterns generated with $\sigma = 10$ (see Figure 10 for patterns with different values of σ). Otherwise the training and testing procedures were unchanged. Average angles and normalized average ranks are given in Table 3 and indicate: (i) translation invariance generalized well to the pattern type not used for training, since the average angles did not differ significantly for the two different testing conditions given the same training condition; (ii) the invariance was in general better learned with low-frequency patterns than with high-frequency patterns (although at the cost of high variance), since the average angles were smaller for all testing conditions if the network was trained with low-frequency patterns; (iii) low-frequency patterns were more easily discriminated than high-frequency patterns, since the normalized average rank was on average lower by 0.04 for the test patterns with $\sigma = 10$ than for those with $\sigma = 2$; (iv) pattern recognition generalized only to a limited degree, since the normalized average rank increased on average by 0.06 if training was not done on the pattern type used for testing. Notice that recognition performance dropped (point iv) even though the invariance generalized well (point i). Response vectors for the pattern type not used for training seemed to be invariant but similar for different patterns so that discrimination degraded.

Testing ⇒ ⇓ Training	$\sigma = 2$ Train.	$\sigma = 2$ Test.	$\sigma = 10$ Train.	$\sigma = 10$ Test.
	Average angles			
$\sigma = 2$	**15.2** 1.5	**18.1** 2.2		19.0 1.6
$\sigma = 10$		11.5 7.0	**6.4** 4.3	**8.9** 5.3
	Normalized average ranks			
$\sigma = 2$	**.08** .02	**.13** .03		.15 .01
$\sigma = 10$		.19 .01	**.06** .04	**.09** .02

Table 3 (ad Example 4): Average angles and normalized average ranks for the networks trained for one pattern type ($\sigma = 2$ or $\sigma = 10$) and tested on the training data or on test data of the same or the other pattern type. All figures are means over five simulation runs with the standard deviation given in small numerals. Bold face numbers indicate performances if a network was tested on the same pattern type for which it was trained. The components useful for recognition were determined on the training data. Training (testing) was done with identical patterns for all networks within a row (column).

4.4 Example 5: Other Invariances in the Visual System Model

Training the network

In this section we want to extend our analysis to other invariances. To be able to consider not only geometrical transformations but also illumination variations, patterns are now derived from object profiles by assuming a Lambertian surface reflectance (e.g. Brooks & Horn, 1989). Let the depth profile of a one-dimensional object be given by $q(u)$ and its surface normal vector by $\mathbf{n}(u) := [-q'(u), 1]^T/\sqrt{q'^2(u) + 1^2}$ with $q'(u) := \mathrm{d}q(u)/\mathrm{d}u$. If a light source shines with unit intensity from direction $\mathbf{l} = (\sin(\alpha), \cos(\alpha))$ (where unit vector $(0, 1)$ would point away from the retina and $(1, 0)$ would point towards the right of the receptive field), the pattern of reflected light is $i(u) = \max(0, \mathbf{n}(u) \cdot \mathbf{l})$. To create an object's depth profile, take Gaussian white noise of length 30 pixels, low-pass filter it with cyclic boundary conditions by a Gaussian with a width σ randomly drawn from the interval $[2, 5]$, and then normalize it to zero mean and unit variance. This is then taken to be the depth derivative $q'(u)$ of the object. The depth profile itself could be derived by integration but is not needed here. Applying the formula for the Lambertian reflectance given above yields the grey-value pattern $i(u)$.

Varying α (here between -60° and +60°) yields different patterns for the same object. In addition the object can be placed at different locations (centered anywhere in the receptive field), scaled (to a size between 2 and 60 pixels), rotated (which results in a cyclic shift up to a full rotation), and varied in contrast (or light intensity, between 0 and 2). Thus, beside the fact that each pattern is derived from a randomly drawn object profile, it has specific values for the five parameters location, size, cyclic shift, contrast, and illumination angle. If a certain parameter is not varied, it usually takes a standard value which is the center of the receptive field for the location, a size of 30 pixels, no cyclic shift for rotation, a contrast of 1, and -30° for the light source. Notice that location, size, and contrast have values that can be uniquely inferred from the stimulus if the object is entirely within the receptive field. Cyclic shift and illumination angle are not unique, since identical patterns could be derived for different cyclic shift and illumination angle if the randomly drawn object profiles happened to be different but related in a specific way, e.g. if two objects are identical except for a cyclic shift. Between the presentation of two objects, there is usually a pause of 60 time units with no stimulus presentation. Some examples of stimulus patterns changing with respect to one of the parameters are given in Figure 20, top.

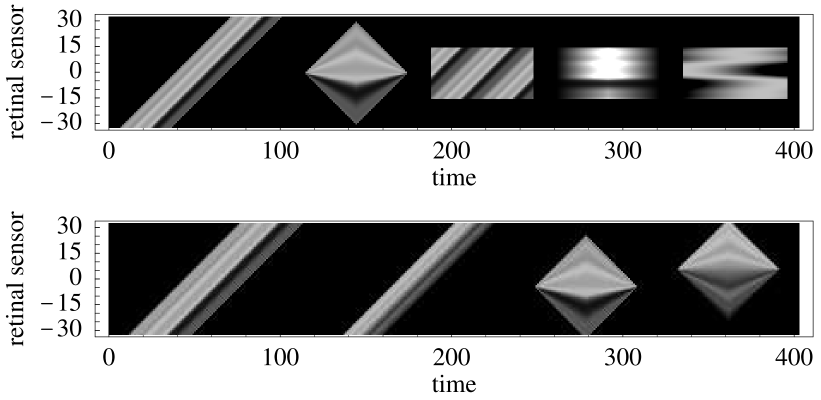


Figure 20 (ad Example 5): **Top:** Stimulus patterns of the same object generated by changing object location from the extreme left to the extreme right in steps of 1, object size from 2 to 30 and back to 2 in steps of 2, cyclic shift from 0 to twice the object size in steps of 1 (resulting in two full rotations), contrast (or light intensity) from 1/15 to 2 and back again in steps of 1/15, and illumination angle from -60° to +60° in steps of 2°. **Bottom:** Stimulus for training translation and size invariance. Patterns derived from two different objects are shown. The first two patterns have constant but randomly choosen size and vary in location. The second two patterns have constant but randomly choosen location and vary in size. The first and third pattern were derived from the same object and so were the second and fourth pattern. Cyclic shift (rotation), contrast, and illumination angle have their standard values. The pause between two presentations of an object is only 14 or 24 time units instead of 60 in the graphs for display purposes.

First, the network was trained for only a single invariance at a time. As in the previous section, 20 patterns were usually used for training and 50 patterns were used for testing. The patterns varied in only one parameter and had standard values for all other parameters. In a second series, the network was trained for two or three invariances simultaneously. Again the patterns of the training stimulus varied in just one parameter at a time. The other parameters either had their standard values, if they belonged to an invariance not trained for, or they had a random value within the valid range, if they belonged to an invariance trained for. For example, if the network was to be trained for translation and size invariance, each pattern varied either in location or in size while having a randomly choosen but constant value for size or location, respectively (cf. Figure 20, bottom). All patterns had standard parameter values for rotation, contrast, and illumination angle. This training procedure (i) comes close to the natural situation in which each pattern occurs many times with different sets of parameters, (ii) is computationally convenient, and (iii) permits a direct comparison between different experiments, because identical stimuli are used in different combinations. In some experiments, all non-varied parameters had their standard value, i.e. in the example given above, the patterns varying in location had standard size and the patterns varying in size had standard location. In that case, however, the network can only learn translation invariance for patterns of standard size and, simultaneously, size invariance for patterns of standard location. No invariances can be expected for patterns of non-standard size *and* location.

What does the network learn?

Figure 21 shows the first four components of the output signal generated by a network trained for size invariance in response to ten objects of the training stimulus. In this case the stimuli consisted of patterns of increasing and decreasing size like the second example in Figure 20, top. The response in this case and also for the other invariances is similar to that in case of translation invariance (see Fig. 11) with the difference that no explicit *where*-information is visible. Thus object recognition should also be possible for these other invariances based on the response vectors.

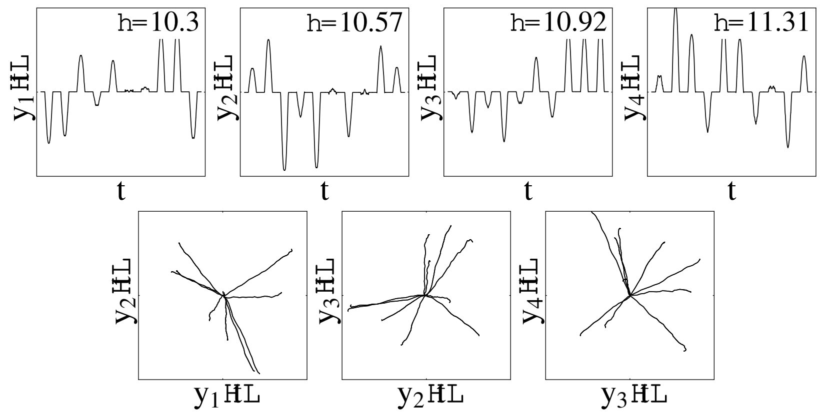


Figure 21 (ad Example 5): First four output signal components of the network trained for size invariance in response to ten objects of the training stimulus with patterns changing in size only. At the bottom are also shown some trajectory plots. Time axes range from 0 to 10×119, where 119 corresponds to the presentation of a single pattern including a pause of 60 time units with no stimulation. All other axes range from -4 to +4. All signals are approximately normalized to zero mean and unit variance.

How invariant are the representations?

Figure 22 shows rank percentiles for the five different invariances. For each but the top right graph a network was trained with 20 patterns varying in one viewing parameter and having standard values for the remaining four ones. 50 test patterns were used to determine which of the output components were useful for recognition. 50 different test patterns were then used to determine the rank percentiles shown in the graphs. Notice that the graph shown for translation invariance differs from the bottom graph in Figure 14 because of differences in the reference location, the number of components used for recognition, and the characteristics of the patterns.

The recognition performance was fairly good over a wide range of parameter variations, indicating that the networks have learned invariances well. For varying illumination angle, however, the network did not generalize if the angle had a different sign, i.e. if the light comes from the other side. The top right graph shows the performance of a network trained for translation and size invariance simultaneously and tested for translation invariance.

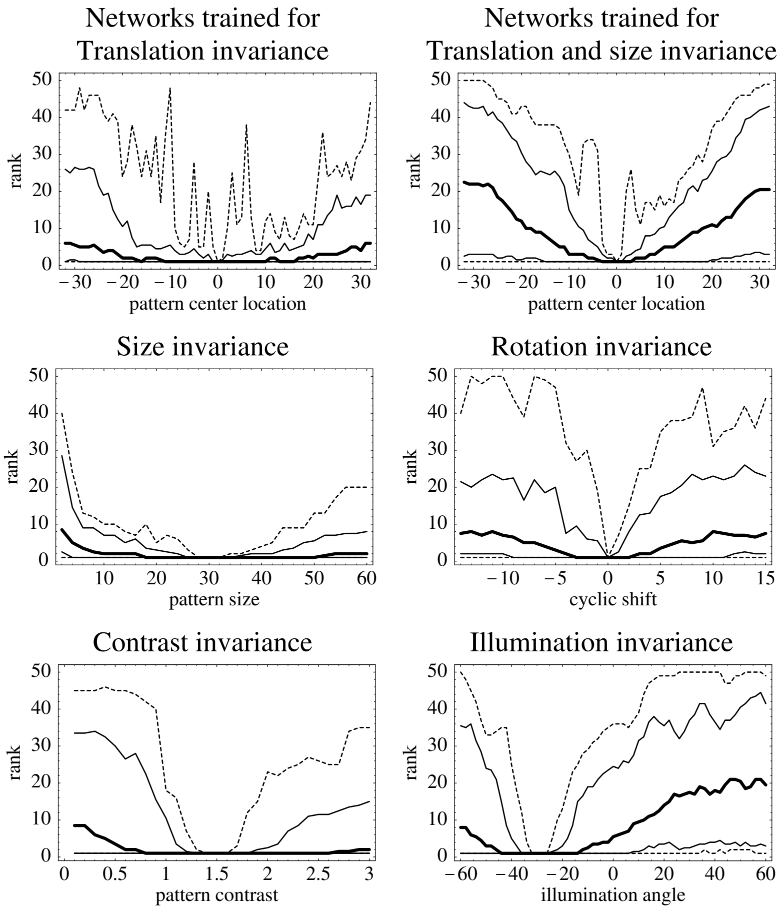


Figure 22 (ad Example 5): Rank percentiles for different networks and invariances and for 50 test patterns. The top right graph shows percentiles for a network that has been trained for translation and size invariance simultaneously and was tested for translation invariance; all other networks have been trained only for one invariance and tested on the same invariance. The thick lines indicate the median ranks or 50%-percentiles. The dashed lines indicate the smallest and largest ranks. The bottom and top thin lines indicate the 10%- and 90%-percentiles, respectively. Chance level would be a rank of 25. Performance is, of course, perfect for the standard view, but also relatively stable over a wide range.

What are the recognition performances?

The normalized average ranks were used as a measure of invariant recognition performance, as for translation invariance in Section 4.3. Normalized average ranks are listed in Table 4 for single and multiple invariances. The numbers of output signal components used in each experiment are shown in Table 5.

Testing ⇒ / ⇓ Training	Location		Size		Rotation		Contrast		Illumination	
	Train.	Test.	Train.	Test.	Train.	Test.	Train.	Test.	Train.	Test.
Location	**.04** .01	**.06** .01		.15 .02		.32 .03		.36 .02		.37 .01
Size		.46 .01	**.01** .00	**.03** .01		.42 .00		.32 .02		.34 .01
Rotation		.37 .01		.33 .03	**.03** .01	**.12** .01		.38 .02		.31 .02
Contrast		.49 .01		.42 .04		.48 .01	**.00** .00	**.06** .03		.34 .05
Illumination		.49 .00		.44 .03		.47 .01		.18 .02	**.06** .01	**.22** .01
Loc. & Size	**.14** .04	**.18** .03	**.04** .01	**.13** .01		.26 .04		.38 .05		.40 .02
Loc. & Rot.	**.21** .06	**.27** .06		.22 .04	**.03** .01	**.20** .03		.39 .01		.35 .04
Size & Illumin.		.47 .01	**.07** .03	**.17** .05		.47 .01		.35 .02	**.13** .04	**.28** .03
Rot. & Illumin.		.38 .02		.33 .02	**.13** .04	**.22** .02		.41 .01	**.22** .03	**.26** .02
Loc., Size & Rot.	**.26** .02	**.31** .00	**.06** .00	**.19** .02	**.03** .01	**.24** .02		.43 .02		.38 .01
Rot., Contr. & Ill.		.41 .04		.34 .01	**.14** .06	**.31** .08	**.19** .04	**.28** .03	**.12** .02	**.30** .04

See next page for table caption.

Table 4 (ad Example 5): Normalized average ranks for networks trained for one, two, or three of the five different invariances and tested with patterns varying in one of the five different parameters. All figures are means over three simulation runs with the standard deviation given in small numerals. Bold face numbers indicate performances if the network was tested on an invariance for which it was trained. Training performance was based on the same output signal components as used for testing. Training (testing) was done with identical patterns for all networks within a row (column).

Tr. ⇓ Test. ⇒	Location	Size	Rotation	Contrast	Illumin.
Location	**6.0** 0.0	6.0 1.0	4.3 0.6	5.3 0.6	5.0 1.7
Size	3.7 1.2	**6.7** 0.6	3.3 0.6	5.3 2.1	4.0 1.0
Rotation	6.7 0.6	5.7 2.3	**8.0** 1.0	4.3 1.5	4.7 1.5
Contrast	3.7 2.9	6.0 2.6	2.3 0.6	**8.7** 0.6	5.3 2.5
Illumination	4.7 1.2	5.7 3.1	3.7 1.2	8.7 0.6	**6.7** 0.6
Loc. & Size	**8.0** 1.0	**7.3** 0.6	4.3 0.6	4.3 0.6	4.3 1.2
Loc. & Rot.	**6.3** 1.2	6.3 0.6	**5.0** 1.7	4.3 1.5	4.3 0.6
Size & Illumin.	3.3 0.6	**7.3** 1.5	4.0 1.0	7.0 1.7	**5.7** 0.6
Rot. & Illumin.	7.0 0.0	6.0 1.0	**6.0** .00	6.0 1.0	**5.0** 2.0
Loc., Size & Rot.	**6.3** 2.3	**6.7** 0.6	**6.7** 1.5	4.0 1.0	4.3 0.6
Rot., Contr. & Ill.	5.0 1.0	5.3 0.6	**4.0** 1.0	**5.7** 0.6	**4.7** 1.5

Table 5 (ad Example 5): Numbers of output signal components useful for recognition. All figures are means over three simulation runs with the standard deviation given in small numerals. Networks were trained for one, two, or three of the five different invariances and tested with patterns varying in one of the five different parameters. Bold face numbers indicate experiments where the network was tested on an invariances for which it was trained.

Even though normalized average ranks for different invariances cannot be directly compared, the figures suggest that contrast and size invariance are easier to learn than translation invariance, and the latter in turn is easier to learn than rotation and illumination invariance.

As shown below, contrast invariance can be learned perfectly if enough training patterns are given. This is not surprising, since the input patterns themselves already form vectors which simply move away from the origin in a straight line and back again, as required above for invariant object recognition. That contrast requires many training patterns for good generalization is probably because a single pattern changing in contrast only spans one dimension. Thus, at least 30 patterns are required to span the relevant space (since patterns have a size of 30).

The performance is better for size invariance than for translation invariance, even though translation is mathematically simpler, probably because individual components of the input signal change more drastically with translation than with scaling, at least around the point with respect to which objects are being rescaled.

Illumination invariance is by far the most difficult of the considered invariances to learn, in part because there is no unique relationship between illumination angle, the object's depth profile, and the light intensity pattern. This also holds to some degree for rotation invariance, on which the network performs second worst. Illumination invariance seems to be particularly difficult to learn if the illumination angle changes sign, i.e. if the light comes from the other side; cf. Figure 22.

It is also interesting to look at how well invariances have been implicitly learned for which the network has not been trained. If trained for translation invariance, for instance, the network also learns size invariance fairly well. Notice, that this is not symmetric. If trained for size invariance, the network does not learn translation invariance well. A similar relationship holds for contrast and illumination invariances. Learning illumination invariance teaches the network also some contrast invariance but not vice versa.

A comparison of the results listed in the bottom of Table 4 with those listed in the top shows that performance degrades when the network is trained on multiple invariances simultaneously. Closer inspection shows that this is due to at least two effects. First, if trained on translation invariance, for instance, patterns vary only in location and have standard size. If trained on translation and size invariance, training patterns that vary in location do not have standard size but a random size. Thus the network has to learn translation invariance not only for standard-size patterns but also for non-standard-size patterns. Thus size becomes a parameter of the patterns the network can represent and the space of patterns is much larger now. However, since testing is done with patterns of standard size (this is important to prevent patterns from being recognized based on their size), pattern size cannot be used during testing. This effect can be found already in a network trained only for translation invariance, but with patterns of random size (compare entries ⎅:⎅ (or ⊟:⊟) with entries ▥:⎅ (or ▤:⊟) in Table 6).

Secondly, the computational mechanisms by which different invariances are achieved may not be compatible and so would interfere with each other (compare entries ▥:⎅ (or ▤:⊟) with entries ▦:⎅ (or ▦:⊟) in Table 6). Further research is required to investigate whether this interference between different invariances is an essential problem or one that can be overcome by using more training patterns and networks of higher complexity, more computational power, and by propagating more components from one SFA-module to the next.

Location (⎅ , ▥) and Size (⊟ , ▤)							
⎅:⎅	**.06** .01	▥:⎅	**.10** .03	⎅:⊟	.15 .02	▥:⊟	.22 .02
⊞:⎅	**.16** .02	▦:⎅	**.18** .03	⊞:⊟	**.05** .01	▦:⊟	**.13** .01
⊟:⎅	.46 .01	▤:⎅	.40 .00	⊟:⊟	**.03** .01	▤:⊟	**.14** .04
Rotation (⎅ , ▥) and Illumination (⊟ , ▤)							
⎅:⎅	**.12** .01	▥:⎅	**.14** .02	⎅:⊟	.31 .02	▥:⊟	.30 .01
⊞:⎅	**.17** .03	▦:⎅	**.22** .02	⊞:⊟	**.28** .03	▦:⊟	**.26** .02
⊟:⎅	.47 .01	▤:⎅	.45 .03	⊟:⊟	**.22** .01	▤:⊟	**.24** .01

See next page for table caption.

Table 6 (ad Example 5): Normalized average ranks for the network trained on one or two invariances and tested on one invariance. The icons represent the two-dimensional parameter space of two invariances considered; all other parameters have standard values. In the top part of the table, for instance, the vertical dimension refers to location and the horizontal dimension refers to size. The icons then have the following meaning:

⎅ indicates 20 input patterns with standard size and varying location,
⊟ indicates 20 input patterns with standard location and varying size,
▥ indicates 20 input patterns with random but fixed size and varying location,
▤ indicates 20 input patterns with random but fixed location and varying size,
⊞ indicates input patterns ⎅ and ⊟ together, i.e. 40 input patterns in total,
▦ indicates input patterns ▥ and ▤ together, i.e. 40 input patterns in total.

The same scheme also holds for any other pair of two invariances. Two icons separated by a colon indicate a training and a testing stimulus. The experiments of the top half of Table 4 would correspond to icons ⊟:⊟ (or ⎅:⎅) if the tested invariance is the same as the trained one (boldface figures) and ⊟:⎅ (or ⎅:⊟) otherwise. The experiments of the bottom half of Table 4, where multiple invariances were learned, would correspond to icons ▦:⊟ (or ▦:⎅) if the tested invariance is one of the trained ones (boldface figures). The remaining experiments in the bottom half cannot be indicated by the icons used here, because the networks are trained on two invariances and tested on a third one.

Table 6 shows that the amount of degradation due to each of the two effects discussed above can vary considerably. Translation invariance does not degrade significantly if training patterns do not have standard size, but it does if size invariance has to be learned simultaneously. Size invariance on the other hand does not degrade if translation invariance has to be learned simultaneously, but it degrades if the patterns varying in size are not at the same (standard) location. Some of these differences can be understood intuitively. For instance, it is clear that the learned translation invariance is largly insensitive to object size but that the learned size invariance is specific to the location at which it has been learned. Evaluating the interdependencies between different invariances in detail could be subject of further research.

How does performance depend on the number of training patterns and network complexity?

The dependencies on the number of training patterns and network complexity are illustrated in Figure 23. It shows normalized average ranks on training and test stimuli for the five different invariances and different numbers of training patterns as well as different numbers of components propagated from one SFA-module to the next. As can be expected, training performance degrades and testing performance improves with the number of training patterns. One can extrapolate that for many training patterns the performance would be about .050 for location, .015 for size, .055 for rotation, .000 for contrast, and .160 for illumination angle, which confirms the general results we found in Table 4 as to how well invariances can be learned, with the addition that contrast can be learned perfectly given enough training patterns. Testing performance also improves with network complexity, i.e. with the number of propagated components, except when overfitting occurs, which is apparently the case at the bottom ends of the dashed curves. Contrast invariance generally degrades with network complexity. This is due to the fact that the input vectors already form a perfect representation for invariant recognition (see above) which can only be degraded by further processing, and the fact that each pattern only spans one dimension, which means that overfitting occurs easily.

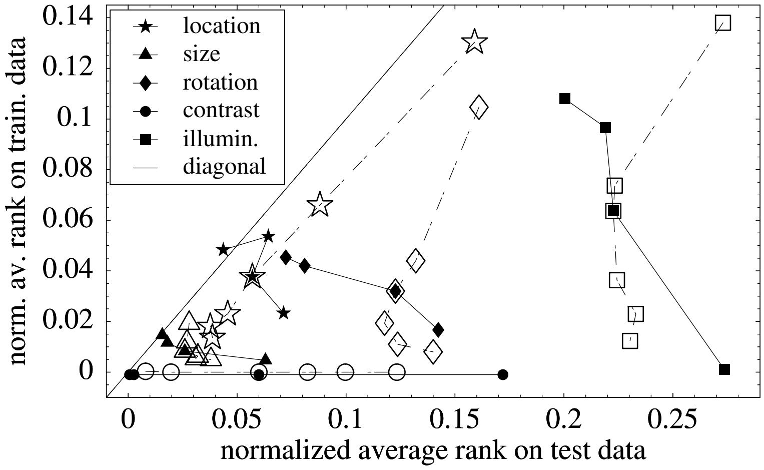


Figure 23 (ad Example 5): Normalized average ranks on training and test stimuli for the five different invariances. Solid lines and filled symbols indicate the dependency on the number of training patterns, which was 10, 20, 40, and 80 from the lower right to the upper left of each curve. Dashed lines and empty symbols indicate the dependency on the number of components propagated from one SFA-module to the next one, which was 5, 7, 9, 11, 13, and 15 from the upper end of each curve to the lower end. The direction of increasing number of components can also be infered from the polarity of the dashing, which follows the pattern 5 - — - — 7 - — ... - — 15, i.e. long dashes point towards large numbers of propagated components. If not varied, the number of training patterns was 20 and the number of propagated components was 9. Notice that the curves cross for these standard values. The solid curve for contrast invariance is shifted slightly downwards to make it distinguishable from the dashed curve. Each point is an average over three simulation runs; standard deviations for points with standard parameter sets can be taken from Table 4.

5 Discussion

The new unsupervised learning algorithm, called *slow feature analysis* (SFA), presented here yields a high-dimensional non-linear input-output function which extracts slowly varying components from a vectorial input signal. Since the learned input-output functions are non-linear, the algorithm can be applied repeatedly, so that complex input-output functions can be learned in a hierarchical network of SFA-modules with limited computational effort.

Slow feature analysis is somewhat unusual in that directions of minimal variance rather than maximal variance are extracted. One might expect that this is particularly sensitive to noise. Noise can enter the system in two ways. First, there may be an additional input component carrying noise. Assume an input signal as in Example 1 plus a component $x_4(t)$ which has a constant value plus some small noise. $x_4(t)$ would seem to be most invariant. However, normalizing it to zero mean and unit variance would amplify the noise such that it becomes a highly fluctuating signal which would be discarded by the algorithm. Secondly, noise could be superimposed on the input signal which carries the slowly varying features. This could change the Δ-values of the extracted signals significantly. However, a slow signal corrupted by noise will usually be slower than a fast signal corrupted by noise of the same amount, so that the slow signal will still be the first one extracted. Only if the noise is unevenly distributed over the potential output signal components can one expect the slow components not to be correctly discovered. Temporal low-pass filtering might help dealing with noise of this kind to some extent.

The apparent lack of an obvious similarity measure in the network of Section 4.3 might reflect an inability to find it. For instance, the network might focus on features that are not well captured by correlation and which are not easily detectable by visual inspection. Alternatively, no particular similarity measure may be realized and similarity may be more a matter of chance. On the negative side, this contradicts the general goal that similar inputs should generate similar representations. However, on the positive side, the network might have enough degrees of freedom to learn any kind of similarity measure in addition to translation invariance if trained appropriately. This could be valuable, because simple correlation is actually not the best similarity measure in the real world.

5.1 Models for unsupervised learning of invariances

Földiák (1991) presented an online learning rule for extracting slowly varying features based on models of classical conditioning. This system associated the present stimulus input with past response activity by means of memory traces. This implemented a tendency to respond similarly to stimuli presented successively in time, i.e. a tendency to generate slow feature responses. A weight growth rule and normalization kept the weights finite and allowed some useful information to be extracted. A simple form of decorrelation of output units was achieved by winner-take-all competition. This, however, did not lead to the extraction of different slow features but rather to a one-of-n code, where each output unit codes for a certain value of the slow feature. The activity of a unit was linear in the inputs and the output was a non-linear function of the total input and the activities of neighboring units. Closely related models were also presented in (Barrow & Bray, 1992; O'Reilly & Johnson, 1994; Fukushima, 1999) and applied to learning lighting and orientation invariances in face recognition in (Stewart Bartlett & Sejnowski, 1998). These kinds of learning rules have also been applied to hierarchical networks with several layers, most clearly in (Wallis & Rolls, 1997). A more detailed biological model of learning invariances also based on memory traces was presented by Eisele (1997). He introduced an additional memory trace to permit association between patterns not only in the backward direction but also in the forward direction.

Another family of online learning rules for extracting slowly varying features has been derived from information theoretic principles. Becker & Hinton (1992) trained a network to discover disparity as an invariant variable of random-dot stereograms. Two local networks, called modules, received input from neighboring but non-overlapping parts of the stereogram, where each module received input from both images. By maximizing the mutual information between the two modules the system learned to extract the only common aspect of their input, namely the disparity, given that disparity changed slowly. This is an example of spatial invariance, in contrast to temporal invariance considered so far. Spatial and temporal invariance are closely related concepts and algorithms can usually be interchangably applied to one or the other domain. An application to the temporal domain, for instance, can be found in (Becker, 1993). For an overview see (Becker, 1996).

The information theoretic approach is appealing because the objective function is well motivated. This becomes particularly obvious in the binary case, where the consequent application of the principles is computationally feasible. However, in case of continuous variables, several approximations need to be made in order to make the problem tractable. The resulting objective function to be maximized is

$$I := 0.5 \log \frac{V(a+b)}{V(a-b)}, \tag{42}$$

where a and b are the outputs of the two modules and $V(\cdot)$ indicates the variance of its argument. If we set $a := y(t)$ and $b := y(t-1)$, assuming discretized time, this objective function is almost identical to the objectives formalized in Equations (1–4). Thus the information theoretic approach provides a systematic motivation but not a different optimization problem. (Zemel & Hinton, 1992) have generalized this approach to extract several invariant variables. a and b became vectors and $V(\cdot)$ became the determinant of the covariance matrix. The output units then preferentially produced decorrelated responses.

Stone & Bray (1995) have presented a learning rule that is based on an objective function similar to that of Becker & Hinton (1992) or Equation (1) and which includes a memory trace mechanism like in (Földiák, 1991). They define two memory traces

$$\tilde{y} := \frac{\sum_{t'=1}^{\infty} \exp(-t'/\tau_s)\ y(t-t')}{\sum_{t''=1}^{\infty} \exp(-t''/\tau_s)}\ , \quad \bar{y} := \frac{\sum_{t'=1}^{\infty} \exp(-t'/\tau_l)\ y(t-t')}{\sum_{t''=1}^{\infty} \exp(-t''/\tau_l)}\ , \tag{43}$$

one on a short time scale ($\tilde{y}$ with a small τ_s) and one on a long time scale ($\bar{y}$ with a large τ_l). The objective function is defined as

$$F := 0.5 \log \frac{\langle (y-\bar{y})^2 \rangle}{\langle (y-\tilde{y})^2 \rangle}, \tag{44}$$

which is equivalent to Equation (42) in the limit $\tau_s \to 0$ and $\tau_l \to \infty$. The derived learning rule performs gradient ascent on this objective function. The examples in (Stone & Bray, 1995) are linearly solvable so that only linear units are used. They include an example where two output units are trained. Inhibitory input from the first to the second output unit enforces decorrelation. Examples in (Stone, 1996) are concerned with disparity estimation, which is not linearly solvable and requires a multilayer network. Backpropagation was used for training with the error signal given by $-F$. A similar system derived from an objective function was presented in (Peng et al., 1998).

Mitchison (1991) presented a learning rule for linear units that is also derived from an objective function like that of Equation (1). He pointed out that the optimal weight vector is given by the last eigenvector of matrix $\langle \dot{\mathbf{x}}\dot{\mathbf{x}}^T \rangle$. In the online learning rule, weights were prevented from decaying to zero by an explicit normalization, such as $\sum w_i^2 = 1$. The extracted output signal would therefore depend strongly on the range of the individual input components, which may be arbitrarily manipulated by rescaling the input components. For instance, if there were one zero input component $x_i = 0$, an optimal solution would be $w_i = 1$ and $w_{i'} = 0$ for all $i' \neq i$, which would be undesirable. Therefore, it seems to be preferable to prevent weight decay by controlling the variance of the output signal and not the sum over the weights directly. The issue of extracting several output components was addressed by introducing a different bias for each output component, which would break the symmetry, if the weight space for slowest output components were more than one-dimensional. However, redundancy can probably be better reduced by the decorrelation constraint of Equation (4) also used in other systems, with the additional advantage that also suboptimal output-signal components can be extracted in an ordered fashion.

Many aspects of the learning algorithm presented here can be found in these previous models. Particularly the optimization problem considered by Becker and colleagues is almost identical to the objective function and constraints used here. The novel aspect of the system presented here is its formulation as a closed-form learning algorithm rather than an online learning rule. This is possible because the input signal is expanded non-linearly, which makes the problem linear in the expanded signal. The solution can therefore be found by sphering and applying principal component analysis to the time derivative covariance matrix. This has several consequences that distinguish this algorithm from others.

- The algorithm is simple and guaranteed to find the optimal solution in one shot. Becker & Hinton (1995) have reported problems in finding the global maxima and they propose several extensions to avoid this problem, such as switching between different learning rules during training. This is not necessary here.

- Several slow features can be easily extracted simultaneously. The learning algorithm automatically yields a large set of decorrelated output signals, extracting decorrelated slow features. (This is different from having several output units representing only one slow feature at a time by a one-of-n code.) This makes it particularly easy to consider hierarchical networks of SFA-modules, since enough information can be propagated from one layer to the next.

- The learning algorithm presented here suffers from the curse of dimensionality. The non-linear expansion makes it necessary to compute large covariance matrices, which soon becomes computationally prohibitive. The system is therefore limited to input signals of moderate dimensionality. This is a serious limitation compared to the online learning rules. However, this problem might be alleviated in many cases by hierarchical networks of SFA-modules, where each module has only a low-dimensional input, e.g. up to 12 dimensions. Example 4 shows such a hierarchical system where a 65 dimensional input signal is broken down by several small SFA-modules.

5.2 Future perspectives

There are several directions in which slow feature analysis as presented here can be investigated and developed further.

- Comparisons with other learning rules should be extended. In particular the scaling with input and output dimensionality needs to be quantified and compared.

- The objective function and the learning algorithm presented here are amenable to analysis. It would be interesting to investigate the optimal responses of an SFA-module and the consequences and limitations of using SFA-modules in hierarchical networks.

- Example 2 demonstrates how several slowly varying features can be extracted by SFA. It also shows that the decorrelation constraint is insufficient to extract slow features in a pure form. It would be interesting to investigate whether SFA can be combined with independent component analysis (e.g. Bell & Sejnowski, 1995) to extract the truly independent slow features.

- Example 5 demonstrates how SFA-modules can be used in a hierarchical network for learning various invariances. It seems, however, that learning multiple invariances simultaneously leads to a significant degradation of performance. It needs to be investigated, whether this is a principal limitation or can be compensated for by more training patterns and more complex networks.

- In the network of Example 4 there was no obvious implicit measure of pattern similarity. More research is necessary to clarify whether there is a hidden implicit measure or whether the network has enough capacity to learn a specific measure of pattern similarity in addition to the invariance.

- Finally, it has to be investigated whether invariances can be learned also from real world image sequences, e.g. from natural scenes, within a reasonable amount of time. Some work in this direction has been done by Wallis & Rolls (1997). They trained a network for translation and other invariances on grey-value images of faces, but they did not test for generalization to new images. The number of images was so small that good generalization could not be expected.

It is clear that SFA can only be one component in a more complex self-organizational model. Aspects such as attention, memory, and recognition (more sophisticated than implemented here) need to be integrated to form a more complete system.

Acknowledgment

We are grateful to James Stone for very fruitful discussions about this project. Many thanks go also to Michael Lewicki and Jan Benda for useful comments on the manuscript. At the Salk Institute LW was partially supported by a Feodor-Lynen fellowship by the Alexander von Humboldt-Foundation, Bonn, Germany; at the Innovationskolleg LW has been supported by HFSP (RG 35-97) and the Deutsche Forschungsgemeinschaft (DFG).

References

Barrow, H. G. and Bray, A. J. (1992). A model of adaptive development of complex cortical cells. In Aleksander, I. and Taylor, J., editors, *Artificial Neural Networks II: Proc. of the Intl. Conf. on Artificial Neural Networks*, pages 881–884, Amsterdam. Elsevier.

Becker, S. (1993). Learning to categorize objects using temporal coherence. In Giles, C. L., Hanson, S. J., and Cowan, J. D., editors, *Advances in Neural Information Processing Systems*, volume 5, pages 361–368, San Mateo, CA. Morgan Kaufmann.

Becker, S. (1996). Mutual information maximization: Models of cortical self-organization. *Network: Computation in Neural Systems*, 7(1):7–31.

Becker, S. and Hinton, G. E. (1992). A self-organizing neural network that discovers surfaces in random-dot stereograms. *Nature*, 355(6356):161–163.

Becker, S. and Hinton, G. E. (1995). Spatial coherence as an internal teacher for a neural network. In Chauvin, Y. and Rumelhart, D. E., editors, *Backpropagation: Theory, Architecture and Applications.*, pages 313–349. Lawrence Erlbaum Associates, Hillsdale, NJ.

Becker, S. and Plumbley, M. (1996). Unsupervised neural network learning procedures for feature extraction and classification. *J. of Applied Intelligence*, 6(3):1–21.

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

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition.* Oxford University Press.

Brooks, M. J. and Horn, B. K. P., editors (1989). *Shape from Shading.* MIT.

Eisele, M. (1997). Unsupervised learning of temporal constancies by pyramidal-type neurons. In Ellacott, S. W., Mason, J. C., and Anderson, I. J., editors, *Mathematics of Neural Networks*, pages 171–175. Kluwer Academic.

Fleet, D. J. and Jepson, A. D. (1990). Computation of component image velocity from local phase information. *Intl. J. of Computer Vision*, 5(1):77–104.

Földiák, P. (1991). Learning invariance from transformation sequences. *Neural Computation*, 3:194–200.

Fukushima, K. (1999). Self-organization of shift-invariant receptive fields. *Neural Networks*, 12(6):791–801.

Fukushima, K., Miyake, S., and Ito, T. (1983). Neocognitron: A neural network model for a mechanism of visual pattern recognition. *IEEE Trans. on Systems, Man, and Cybernetics*, 13:826–834. Reprinted in *Neurocomputing*, J. A. Anderson and E. Rosenfeld, Eds., MIT Press, Massachusetts, pp. 526–534.

Jones, J. P. and Palmer, L. A. (1987). An evaluation of the two-dimensional Gabor filter model of simple receptive fields in cat striate cortex. *J. of Neurophysiology*, 58:1233–1258.

Konen, W., Maurer, T., and von der Malsburg, C. (1994). A fast dynamic link matching algorithm for invariant pattern recognition. *Neural Networks*, 7(6/7):1019–1030.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551.

Mitchison, G. (1991). Removing time variation with the anti-Hebbian differential synapse. *Neural Computation*, 3(3):312–320.

Olshausen, B. A., Anderson, C. H., and Van Essen, D. C. (1993). A neurobiological model of visual attention and invariant pattern recognition based on dynamic routing of information. *J. of Neuroscience*, 13(11):4700–4719.

ORAM, M. W. AND PERRETT, D. I. (1994). Modeling visual recognition from neurobiological constraints. *Neural Networks*, 7(6/7):945–972.

O'REILLY, R. C. AND JOHNSON, M. H. (1994). Object recognition and sensitive periods: A computational analysis of visual imprinting. *Neural Computation*, 6(3):357–389.

PENG, H. C., SHA, L. F., GAN, Q., AND WEI, Y. (1998). Energy function for learning invariance in multilayer perceptron. *Electronics Letters*, 34(3):292–294.

RUMELHART, D. E., HINTON, G. E., AND MCCLELLAND, J. L. (1986). A general framework for parallel distributed processing. In RUMELHART, D. E. AND MCCLELLAND, J. L., editors, *Parallel Distributed Processing*, volume 1, chapter 2, pages 45–76. MIT Press, Cambridge, MA.

STEWART BARTLETT, M. AND SEJNOWSKI, T. J. (1998). Learning viewpoint invariant face representations from visual experience in an attractor network. *Network: Computation in Neural Systems*, 9(3):399–417.

STONE, J. V. (1996). Learning perceptually salient visual parameters using spatiotemporal smoothness constraints. *Neural Computation*, 8(7):1463–1492.

STONE, J. V. AND BRAY, A. J. (1995). A learning rule for extracting spatio-temporal invariances. *Network: Computation in Neural Systems*, 6(3):429–436.

THEIMER, W. M. AND MALLOT, H. A. (1994). Phase-based binocular vergence control and depth reconstruction using active vision. *CVGIP: Image Understanding*, 60(3):343–358.

VAPNIK, V. (1995). *The Nature of Statistical Learning Theory.* Springer-Verlag, New York.

WALLIS, G. AND ROLLS, E. (1997). Invariant face and object recognition in the visual system. *Progress in Neurobiology*, 51:167–194.

ZEMEL, R. S. AND HINTON, G. E. (1992). Discovering viewpoint-invariant relationships that characterize objects. In LIPPMANN, R. P., MOODY, J. E., AND TOURETZKY, D. S., editors, *Advances in Neural Information Processing Systems*, volume 3, pages 299–305, San Mateo, CA. Morgan Kaufmann.