The final version of this article has been published in Neural Computation, 24(11):2994-3024 (2012) published by The MIT Press. This version does not differ significantly from the final version.

Incremental Slow Feature Analysis: Adaptive Low-Complexity Slow Feature Updating from High-Dimensional Input Streams

Varun Raj Kompella, **Matthew Luciw**, **Juergen Schmidhuber**
IDSIA, SUPSI, USI, Galleria 2, Manno-Lugano 6928, Switzerland
{varun, matthew, juergen}@idsia.ch

Keywords: SFA (Slow Feature Analysis), Incremental Learning, PCA (Principal Component Analysis), MCA (Minor Component Analysis)

Abstract

We introduce here an incremental version of Slow Feature Analysis (IncSFA), combining Candid Covariance-Free Incremental Principal Components Analysis (CCIPCA) and Covariance-Free Incremental Minor Components Analysis (CIMCA). IncSFA's feature updating complexity is linear with respect to the input dimensionality, while batch SFA's (BSFA) updating complexity is cubic. IncSFA does not need to store, or even compute, any covariance matrices. The drawback to IncSFA is data-efficiency: it does not use each data point as effectively as BSFA. But IncSFA allows SFA to be tractably applied, with just a few parameters, directly on high-dimensional input streams (e.g., visual input of an autonomous agent), while BSFA has to resort to hierarchical receptive-field based architectures when the input dimension is too high. Further, IncSFA's updates have simple Hebbian and anti-Hebbian forms, extending the biological plausibility of SFA. Experimental results show IncSFA learns the same set of features as BSFA and can handle a few cases where BSFA fails.

1 Introduction

Slow feature analysis (SFA; Wiskott and Sejnowski (2002); Wiskott et al. (2011)) is an unsupervised learning technique that extracts features from an input stream with the objective that the feature responses must change over time, but should change as slowly as possible. A *temporal stability* objective has been used in some other unsupervised learning techniques (Hinton, 1989; Földiák, 1991; Mitchison, 1991; Schmidhuber, 1992a; Bergstra and Bengio, 2009). SFA is different: the slow features are the lowest order eigenvectors of an eigensystem based on the covariance matrix of input derivative measurements. The eigensystem formulation guarantees that its solution methods reliably converge to a best solution, avoiding entirely the issue of local minima. SFA has shown success in many problems and scenarios — extraction of driving forces of a dynamical system (Wiskott, 2003), nonlinear blind source separation (Sprekeler et al., 2010), as a preprocessor for reinforcement learning (Legenstein et al., 2010; Kompella et al., 2011b), and learning of place-cells, head-direction cells, grid-cells, and spatial view cells from high-dimensional visual input (Franzius et al., 2007): such representations also exist in biological agents (O'Keefe and Dostrovsky, 1971; Taube et al., 1990; Rolls, 1999; Hafting et al., 2005).

A typical SFA implementation occurs in several stages: the data must all be collected, processed (non-linearly expanded, outliers removed, etc.), whitened, the covariance matrix of the derivative measurements constructed, its eigenvectors are solved and these eigenvectors, in reverse order, are the slowest features. We present here Incremental Slow Feature Analysis (IncSFA; Kompella et al. (2011a)), which interleaves all the stages so that the slow features can be *updated* after each derivative measurement. A few earlier techniques with temporal continuity objective were incremental as well (Hinton, 1989; Bergstra and Bengio, 2009), but IncSFA follows the SFA formulation and can track solutions that would be uncovered by batch SFA (BSFA), over which it has the following advantages:

- **Adaptation to Changing Input Statistics.** In the BSFA paradigm, new data cannot be used to modify already learned slow features. If input statistics change, IncSFA can adapt existing features without outside intervention, while BSFA has to discard previous features to process the new data.

- **Computational Efficiency.** BSFA techniques rely upon batch Principal Component Analysis (PCA; Jolliffe (1986)). For input observations in a I-dimensional space, the computational complexity of PCA using the Jacobi method (Forsythe and Henrici, 1958) is of the order $O(I^3)$. IncSFA's updating complexity scales linearly with dimensionality $(O(I))$, thus it has an advantage when the input dimension is large.

- **Space Complexity.** First, IncSFA can discard each observation immediately after an update. Second, we note that IncSFA uses *covariance-free* techniques, where the data covariance matrices never need to be computed, even in passing. In a covariance-free technique, the features are updated directly from the new data. The $I(I+1)/2$ parameters in the covariance matrix do not have to be estimated.

- **Simplicity.** For extracting features from high-dimensional image sequences, IncSFA presents a simpler solution method than the alternate technique of deep receptive-field based BSFA networks. By simpler, we mean that IncSFA has just a handful of parameters, instead of the multitude of parameters associated with the deep nets.

- **Reduced Sensitivity to Outliers.** Outlier observations in a dataset can cause problems for BSFA, as these outliers can corrupt the slow features. In some cases, a feature may even become sensitive to an outlier. In a typical batch implementation, each observation has the same amount of influence on the features. In IncSFA, the influence of a single observation fades as newer observations are experienced. The learning rate implicitly controls this forgetting factor. Different learning rate settings can lead to different features — features that emerge from a high learning rate setting are biased to detect slowly-changing phenomena that occur with more regularity than if a lower learning rate were to be used.

- **Biological Plausibility.** IncSFA adds further biological plausibility to SFA. SFA itself has been linked to biological systems due to the results in deriving place cell, grid cells, etc., but it is difficult to see how BSFA could be realized in the brain. IncSFA's updates can be described in incremental Hebbian and anti-Hebbian forms (discussed in detail in Sec. 5.4).

The disadvantage of IncSFA is that BSFA is better in terms of data efficiency — more input samples are needed for learning as compared to BSFA.

The remainder of this paper is organized as follows. Section 2 reviews SFA and its batch solution. Section 3 walks through the new incremental SFA. Section 4 has some illustrative experiments and results. Section 5 discusses supplementary issues, including convergence, parameter setting and biological plausibility. Section 6 concludes the paper.

2 Background

2.1 SFA: Intuition

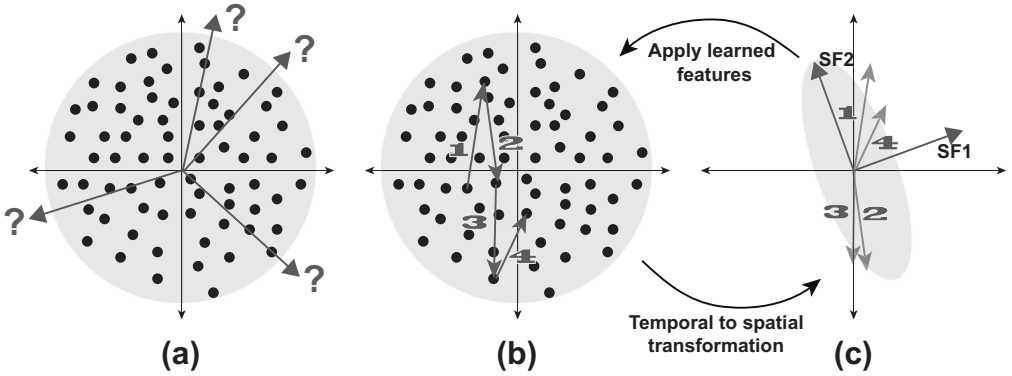


Figure 1: A toy example to explain what a slow feature is. (A): Consider a zero-mean input signal that spatially resembles white noise. Input points (the black dots) are drawn from within the gray circle area. Linear spatial feature extractors (such as PCA) will not prefer any direction over any other (since, for PCA, the variance in all directions is the same). (B): If we recode the data in terms of how it changes between subsequent time instants, certain directions can be more informative than others. Here, the arrows show a short representative sequence of input. All difference vectors (not just the four shown) create the space shown in (C). In this space, the first principal component gives the (linear) direction of quickest change. The second — the minor component — gives the direction of slowest change (the slowest feature).

In this section, we briefly review SFA in an intuitive sense and present its formulation and batch solution method (those who are already familiar with SFA can skip this section). SFA is a form of unsupervised learning (UL). Like some other feature extraction techniques, it searches for a set of mappings g_i from data $\mathbf{x} \in \mathcal{R}^I$ to output components $y_i = g_i(\mathbf{x})$ that are *separate* from each other in some sense and express information that is in some sense *relevant*. In SFA the features are separated via mutual decorrelation of their outputs, while relevance is defined as *minimal but nonzero change over time* (slowness). Ordering the functions $g_1, g_2, ..., g_I$ by slowness, we can discard all but the $J < I$ slowest, getting rid of irrelevant information such as quickly changing noise assumed to be useless. See Fig. 1 for a visual example of the meaning of a slow feature.

SFA-based UL learns *instantaneous* features from sequential data (Hinton, 1989; Wiskott and Sejnowski, 2002; Doersch et al.). Relevance cannot be *uncovered* without taking time into account, but once it is known, each input frame can be encoded on its own. Due to this, SFA differs from both 1. many well-known unsupervised feature extractors (Abut, 1990; Jolliffe, 1986; Comon, 1994; Lee and Seung, 1999; Kohonen, 2001; Hinton, 2002), which ignore dynamics, and 2. Other UL systems that both learn and apply features to sequences (Schmidhuber, 1992a,c,b; Lindstädt, 1993; Klapper-Rybicka et al., 2001; Jenkins and Matarić, 2004; Lee et al., 2010; Gisslen et al., 2011), thus assuming that the state of the system itself can depend on past information.

The compact relevant encodings uncovered by SFA reduce the search space for downstream goal-directed learning procedures (Schmidhuber, 1999; Barlow, 2001), especially reinforcement learning. As an example, consider a robot sensing with an onboard camera. Reinforcement learning algorithms applied directly to pixels can be quite inefficient due to the size of the search space. Slow features can encode each image into a small set of useful state variables, and the robot can use these few state variables to quickly develop useful control policies. The state variables from SFA are approximations of low-order eigenvectors of the graph Laplacian (Sprekeler, 2011), i.e., proto-value functions (Mahadevan and Maggioni, 2007). This is why they are typically more useful as features in reinforcement learning instead of other types of features, such as principal components.

2.2 SFA: Formulation

SFA's optimization problem (Wiskott and Sejnowski, 2002; Franzius et al., 2007) is formally written as follows:

Given an I-dimensional sequential input signal $\mathbf{x}(t) = [x_1(t), ..., x_I(t)]^T$*, find a set of* J *instantaneous real-valued functions* $\mathbf{g}(x) = [g_1(\mathbf{x}), ..., g_J(\mathbf{x})]^T$*, which together generate a* 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

$$\langle y_j \rangle = 0 \quad \textit{(zero mean),} \tag{2}$$

$$\langle y_j^2 \rangle = 1 \quad \textit{(unit variance),} \tag{3}$$

$$\forall i < j : \langle y_i y_j \rangle = 0 \quad \textit{(decorrelation and order),} \tag{4}$$

with $\langle \cdot \rangle$ *and* $\dot{y}$ *indicating temporal averaging and the derivative of* y*, respectively.*

The problem is to find instantaneous functions g_j that generate different output signals varying as slowly as possible. The constraints (2) and (3) together avoid a trivial constant output solution. The decorrelation constraint (4) ensures that different functions g_j do not code for the same features.

2.3 Batch SFA

Solving this learning problem involves variational calculus optimization. But it is simplified through an eigenvector approach. If the g_j are linear combinations of a finite set of nonlinear functions $\mathbf{h}$, then

$$y_j(t) = g_j(\mathbf{x}(t)) = \mathbf{w}_j^T \, \mathbf{h}(\mathbf{x}(t)) = \mathbf{w}_j^T \, \mathbf{z}(t), \tag{5}$$

and the SFA problem now becomes to find weight vectors $\mathbf{w}_j$ to minimize the rate of change of the output variables,

$$\Delta(y_j) = \langle \dot{y}_j^2 \rangle = \mathbf{w}_j^T \, \langle \dot{\mathbf{z}} \dot{\mathbf{z}}^T \rangle \, \mathbf{w}_j, \tag{6}$$

subject to the constraints (2-4). The slow feature learning problem has become linear on the derivative signal $\dot{\mathbf{z}}$.

If the functions of $\mathbf{h}$ are chosen such that $\mathbf{z}$ has unit covariance matrix and zero mean, the three constraints will be fulfilled if and only if the weight vectors $\mathbf{w}_j$ are orthonormal. Eq. 6 will be minimized, and the orthonormal constraint satisfied, with the set of J normed eigenvectors of $\langle \dot{\mathbf{z}}\dot{\mathbf{z}}^T \rangle$ with the J smallest eigenvalues (for any $J \leq I$).

The BSFA technique implements this solution by using batch principal component analysis (PCA) (Jolliffe, 1986) twice. Referring back to Eq. 6, to select $\mathbf{h}$ appropriately, a well-known process called whitening (or sphering), is used to map $\mathbf{x}$ to a $\mathbf{z}$ with zero mean and identity covariance matrix, thus decorrelating signal components and scaling them so that there is unit variance along each principal component (PC) direction. Whitening serves as a bandwidth normalization, so that slowness can truly be measured (slower change will not simply be due to a low variance direction). Whitening requires the PCs of the input signal (PCA #1). The orthonormal basis that minimizes the rate of output change are the minor components – principal components with smallest eigenvalues – in the derivative space. So, another PCA (#2) on $\dot{\mathbf{z}}$ yields the slow features (eigenvectors) and their order (via eigenvalues).

3 Incremental SFA

Like BSFA, IncSFA employs the eigenvector tactic, but uses incremental algorithms for the two required PCAs. Therefore, IncSFA can update existing slow feature estimates on any amount of new data, even a single data point $\mathbf{x}(t)$.

To replace PCA $\#1$, IncSFA needs to incrementally whiten the input $\mathbf{x}$. We use the state-of-the-art Candid Covariance-Free Incremental (CCI) PCA (Weng et al., 2003). CCIPCA incrementally updates both the eigenvectors and eigenvalues necessary for whitening, and does not keep an estimate of the covariance matrix. It has been proven to converge to the true PCs (Zhang and Weng, 2001). CCIPCA is optionally used to reduce dimensionality at this intermediate stage, by only computing the K highest order eigenvectors.

Except for low-dimensional derivative signals $\dot{\mathbf{z}}$, CCIPCA cannot replace PCA $\#2$.

It will take a long time to converge to the slow features, since they correspond to the least significant components. Minor Components Analysis (MCA) (Oja, 1992) incrementally extracts principal components, but with a reversed preference: it finds it easiest to extract the components with the smallest eigenvalues. We use a modified version of Peng's low complexity MCA updating rule (Peng et al., 2007). Peng proved its convergence even for constant learning rates—good for open-ended learning. MCA with sequential addition (Chen et al., 2001; Peng and Yi, 2006) will extract multiple slow features in parallel. In IncSFA, this method is modified to be covariance-free.

A high-level formulation of IncSFA is

$$(\mathbf{W}(t+1), \mathbf{V}(t+1)) = IncSFA(\mathbf{W}(t), \mathbf{V}(t), \mathbf{x}(t), \theta(t)), \tag{7}$$

where $\mathbf{W} = (\mathbf{w}_1, ..., \mathbf{w}_J)$ is the matrix of existing slow feature vector estimates for J slow features, where each feature is a column vector in the matrix, $\mathbf{V} = (\mathbf{v}_1, ..., \mathbf{v}_K)$ is the matrix of K principal component vector estimates used to construct the whitening matrix and for dimensionality reduction, $\mathbf{x}(t) \in \mathcal{R}^I$ is the input observation, and θ contains parameters about setting learning rates, which we will discuss later. In general $K < I$ and $J < K$.

Next, we'll walk through the components of the algorithm.

3.1 Principal Components for Whitening

Given zero-mean data $\mathbf{u} = \mathbf{x} - \mathrm{E}[\mathbf{x}]$, a PC is a normed eigenvector $\mathbf{v}_i^*$ of the data covariance matrix $\mathrm{E}[\mathbf{u}\mathbf{u}^T]$. Eigenvalue λ_i^* is the variance of the samples along $\mathbf{v}_i^*$. By definition, an eigenvector and eigenvalue satisfy

$$\mathrm{E}[\mathbf{u}\mathbf{u}^T]\mathbf{v}_i^* = \lambda_i^* \mathbf{v}_i^*, \tag{8}$$

The set of eigenvectors are orthonormal, and ordered such that $\lambda_1^* \geq \lambda_2^* \geq ... \geq \lambda_K^*$.

The whitening matrix is generated by multiplying the matrix of principal component length-one eigenvectors $\mathbf{V}^*$ by the diagonal matrix $\mathbf{D}^*$, where component $\hat{d}_{i,i} = \frac{1}{\sqrt{\lambda_i}^*}$. After whitening via $\mathbf{z}(t) = \mathbf{D}^*\mathbf{V}^{*T}\mathbf{u}(t)$, the data will be normalized in scale and decorrelated, as seen by the fact that the covariance matrix will be the identity matrix: $\mathrm{E}[\mathbf{z}\mathbf{z}^T] = I$.

The procedure to generate a whitening matrix is outlined in Algorithm 1. We note that, via CCIPCA, the magnitudes of the eigenvector estimates are the eigenvalue estimates.

Algorithm 1: CONSTRUCTWHITENINGMATRIX($\mathbf{V}$)

$\hat{\mathbf{V}} \leftarrow \left(\frac{\mathbf{v}_1}{\|\mathbf{v}_1\|}, ..., \frac{\mathbf{v}_K}{\|\mathbf{v}_K\|} \right)$ `//` $I \times K$ `matrix`

$\mathbf{D} \leftarrow \mathbf{0}$ `//` $K \times K$ `matrix`

for *i* $\leftarrow$ *1 to K* **do**

  $D_{i,i} = 1/\sqrt{\|\mathbf{v}_i\|}$

end

$\mathbf{S} \leftarrow \hat{\mathbf{V}}\mathbf{D}$ `//` $I \times K$ `matrix`

return $\mathbf{S}$

3.2 CCIPCA Updating

CCIPCA updates estimates of eigenvalues and eigenvectors from each sample. For inputs $\mathbf{u}_i$, the first PC is the expectation of the normalized response-weighted inputs. Eq 8 can be rewritten as

$$\lambda_i^* \, \mathbf{v}_i^* = \mathrm{E}\left[(\mathbf{u}_i \cdot \mathbf{v}_i^*) \, \mathbf{u}_i \right], \tag{9}$$

The corresponding incremental updating equation, where $\lambda_i^* \, \mathbf{v}_i^*$ is estimated by $\mathbf{v}_i(t)$, is

$$\mathbf{v}_i(t) = (1 - \eta^{PCA}) \, \mathbf{v}_i(t-1) + \eta^{PCA} \left[\frac{\mathbf{u}_i(t) \cdot \mathbf{v}_i(t-1)}{\|\mathbf{v}_i(t-1)\|} \, \mathbf{u}_i(t) \right]. \tag{10}$$

where $0 < \eta^{PCA} < 1$ is the learning rate. In other words, both the eigenvector and eigenvalue of the first PC of $\mathbf{u}_i$ can be found through the sample mean-type updating in Eq. 9. The estimate of the eigenvalue is given by $\lambda_i = \|\mathbf{v}_i(t)\|$. Using both a learning rate η^{PCA} and retention rate $(1 - \eta^{PCA})$ automatically makes this algorithm invariant to the magnitude of the input vectors.

3.3 Lower-Order Principal Components

Any component $i > 1$ not only must satisfy Eq. 8 but must also be orthogonal to the higher-order components. The *residual method* (Kreyszig, 1988; Sanger, 1989) generates observations in a complementary space so that lower-order eigenvectors can be found by the update rule of Eq. 10.

Denote $\mathbf{u}_i(t)$ as the observation for component i. When $i = 1$, $\mathbf{u}_1(t) = \mathbf{u}(t)$. When $i > 1$, $\mathbf{u}_i$ is a residual vector, which has the "energy" of $\mathbf{u}(t)$ from the higher-order components removed. Solving for the first PC in this residual space solves for the i-th component overall. To create a residual vector, $\mathbf{u}_i$ is projected onto $\mathbf{v}_i$ to get the energy of $\mathbf{u}_i$ that $\mathbf{v}_i$ is responsible for. Then, the energy-weighted $\mathbf{v}_i$ is subtracted from $\mathbf{u}_i$ to obtain $\mathbf{u}_{i+1}$:

$$\mathbf{u}_{i+1}(t) = \mathbf{u}_i(t) - \left(\mathbf{u}_i^T(t)\frac{\mathbf{v}_i(t)}{\|\mathbf{v}_i(t)\|}\right)\frac{\mathbf{v}_i(t)}{\|\mathbf{v}_i(t)\|}. \tag{11}$$

Together, Eq. 10 and Eq. 11 constitute the CCIPCA technique described in Algorithm 2.

Algorithm 2: CCIPCA-UPDATE($\mathbf{V}, K, \mathbf{u}, \eta$)

```
//Candid Covariance-Free Incremental PCA
```

1 $\mathbf{u}_1 \leftarrow \mathbf{u}$

2 **for** *i ← 1 to K* **do**

```
    //Principal component update
```

3 $\mathbf{v}_i \leftarrow (1-\eta)\,\mathbf{v}_i + \eta\,\left[\frac{\mathbf{u}_i \cdot \mathbf{v}_i}{\|\mathbf{v}_i\|}\,\mathbf{u}_i\right]$

```
    //Residual
```

4 $\mathbf{u}_{i+1} = \mathbf{u}_i - \left(\mathbf{u}_i^T\frac{\mathbf{v}_i}{\|\mathbf{v}_i\|}\right)\frac{\mathbf{v}_i}{\|\mathbf{v}_i\|}$

5 **end**

6 **return** $\mathbf{V}$

3.4 MCA Updating

After using CCIPCA components to generate an approximately whitened signal $\mathbf{z}$, the derivative is approximated by (for example) $\dot{\mathbf{z}}(t) = \mathbf{z}(t) - \mathbf{z}(t-1)$. In this derivative space, the minor components on $\dot{\mathbf{z}}$ are the slow features.

To find the minor component, Peng's MCA (Peng et al., 2007) is used, the updating equation of which is

$$\begin{aligned} \mathbf{w}_i(t) &= 1.5\mathbf{w}_i(t-1) - \eta^{MCA}\,\mathbf{C}_i\,\mathbf{w}_i(t-1) \\ &- \eta^{MCA}\,[\mathbf{w}_i^T(t-1)\mathbf{w}_i(t-1)]\,\mathbf{w}_i(t-1), \end{aligned} \tag{12}$$

where, for the first minor component, $\mathbf{C}_1 = \dot{\mathbf{z}}(t)\dot{\mathbf{z}}^T(t)$.

For "lower-order" minor components, the *sequential addition* technique (Chen et al., 2001) shifts each observation into a space where the minor component of the current space will be the first PC, and all other PCs are reduced in order by one. Sequential addition allows IncSFA to extract more than one slow feature in parallel. Sequential addition updates the matrix $\mathbf{C}_i,\ \forall i > 1$ as follows:

$$\mathbf{C}_i(t) = \mathbf{C}_{i-1}(t) + \gamma(t)\ \left(\mathbf{w}_{i-1}(t)\mathbf{w}_{i-1}^T(t)\right) / \left(\mathbf{w}_{i-1}^T(t)\mathbf{w}_{i-1}(t)\right) \tag{13}$$

Note Eq. 13 introduces parameter γ, which must be larger than the largest eigenvalue of $\mathrm{E}[\dot{\mathbf{z}}(t)\dot{\mathbf{z}}^T(t)]$. To automatically set γ, we compute the greatest eigenvalue of the derivative signal through another CCIPCA rule to update only the first PC. Then, let $\gamma = \lambda_1(t) + \epsilon$ for small ϵ.

3.5 Covariance-Free MCA

We can avoid the potentially costly outer products via the same trick that made CCIPCA covariance-free: $(\dot{\mathbf{z}}\dot{\mathbf{z}}^T)\ \mathbf{w}_i = (\dot{\mathbf{z}} \cdot \mathbf{w}_i)\dot{\mathbf{z}}$. Considering only the first slow feature for now, Eq. 12 can be re-written (switching to arrow notation from now on for clarity) as:

$$\begin{aligned} \mathbf{w}_1 &\leftarrow 1.5\mathbf{w}_1 - \eta^{MCA}\,\dot{\mathbf{z}}\,[\dot{\mathbf{z}}^T\,\mathbf{w}_i] - \eta^{MCA}\,[\mathbf{w}_i^T\mathbf{w}_i]\,\mathbf{w}_i, \\ &\leftarrow \left(1.5 - \eta^{MCA}\,\|\mathbf{w}_1\|^2\right)\,\mathbf{w}_1 - \eta^{MCA}\,(\dot{\mathbf{z}} \cdot \mathbf{w}_1)\,\dot{\mathbf{z}}, \end{aligned} \tag{14}$$

as shown in Sec. 5.4.

When dealing with nonstationary input, as we do in IncSFA due to the simultaneously learning CCIPCA components, it is acceptable[1] to normalize the magnitude of the

[1] Peng: personal communication.

slow feature vectors: $\mathbf{w}_i \leftarrow \mathbf{w}_i / \|\mathbf{w}_i\|$. Normalization at least ensures non-divergence (see Section 5.1). If we normalize, Eq. 14 can be rewritten in an even simpler form with retention rate and learning rate,

$$\mathbf{w}_1 \leftarrow (1 - \eta^{MCA})\mathbf{w}_1 - \eta^{MCA}(\dot{\mathbf{z}} \cdot \mathbf{w}_1)\, \dot{\mathbf{z}}, \tag{15}$$

$$\mathbf{w}_1 \leftarrow \mathbf{w}_1 / \|\mathbf{w}_1\|. \tag{16}$$

Now, for all other slow features $i > 1$, the update can be written so sequential addition shows itself to be a Gram-Schmidt procedure.

$$\mathbf{w}_i \leftarrow (1 - \eta^{MCA})\mathbf{w}_i - \eta^{MCA} \left((\dot{\mathbf{z}} \cdot \mathbf{w}_i)\, \dot{\mathbf{z}} + \gamma \sum_{j}^{i-1} (\mathbf{w}_j \cdot \mathbf{w}_i)\mathbf{w}_j \right). \tag{17}$$

The covariance-free MCA is outlined in algorithm 3.

Algorithm 3: CIMCA-UPDATE($\mathbf{W}, J, \dot{\mathbf{z}}, \gamma, \eta$)

```
//Covariance-Free Incremental MCA
```

$\mathbf{l}_1 \leftarrow 0$

for *i ← 1 to J* **do**

```
//Minor component update
```

$\mathbf{w}_i \leftarrow (1 - \eta)\mathbf{w}_i - \eta \left[(\dot{\mathbf{z}} \cdot \mathbf{w}_i)\, \dot{\mathbf{z}} + \mathbf{l}_i \right].$

```
//Normalize
```

$\mathbf{w}_i \leftarrow \mathbf{w}_i / \|\mathbf{w}_i\|.$

```
//Lateral competition from ``lower'' components
```

$\mathbf{l}_{i+1} \leftarrow \gamma \sum_j^i (\mathbf{w}_j \cdot \mathbf{w}_i)\mathbf{w}_j$

end

return $\mathbf{W}$

Now, we can introduce the overall framework of IncSFA, seen in Algorithm 4.

Algorithm 4: INCSFA(J, K, θ)

//Autonomously learn J slow features from samples $\mathbf{x} \in \mathcal{R}^I$

// $\mathbf{V}$: Matrix of $K \leq I$ principal component column vectors

// $\mathbf{W}$: Matrix of $J \leq K$ slow feature column vectors

// $\mathbf{v}^\gamma$: First PC in whitened difference space

// $\bar{\mathbf{x}}$: Mean of $\mathbf{x}$

$\{\mathbf{V}, \mathbf{W}, \mathbf{v}^\gamma, \bar{\mathbf{x}}\} \leftarrow$ INITIALIZE ()

for $t \leftarrow 1$ *to* ∞ **do**

  $\check{\mathbf{x}} \leftarrow$ SENSE($worldstate$)

  $\mathbf{x} \leftarrow$ EXPAND ($\check{\mathbf{x}}$) //non-linear expansion

  $\{\eta_t^{PCA}, \eta_t^{MCA}\} \leftarrow$ LEARNINGRATESCHEDULE (θ, t)

  $\bar{\mathbf{x}} \leftarrow (1 - \eta_t^{PCA})\, \bar{\mathbf{x}} + \eta_t^{PCA}\, \mathbf{x}$ //Update mean

  $\mathbf{u} \leftarrow (\mathbf{x} - \bar{\mathbf{x}})$ //Centering

  //Candid Covariance-Free Incremental PCA

  $\mathbf{V} \leftarrow$ CCIPCA-UPDATE ($\mathbf{V}, K, \mathbf{u}, \eta_t^{PCA}$)

  $\mathbf{S} \leftarrow$ CONSTRUCTWHITENINGMATRIX ($\mathbf{V}$)

  If $t > 1$ **then** $(\mathbf{z}_{prev} \leftarrow \mathbf{z}_{curr})$ //Store prev.

  $\mathbf{z}_{curr} \leftarrow \mathbf{S}^T \mathbf{u}$ //Whitening and dim. reduction

  if $t > 1$ **then**

    $\dot{\mathbf{z}} \leftarrow (\mathbf{z}_{curr} - \mathbf{z}_{prev})$ //Approx. derivative

    //To learn sequential addition parameter γ

    $\mathbf{v}^\gamma \leftarrow$ CCIPCA-UPDATE ($\mathbf{v}^\gamma, 1, \dot{\mathbf{z}}, \eta_t^{PCA}$)

    $\gamma \leftarrow \mathbf{v}^\gamma / \|\mathbf{v}^\gamma\|$

    //Covariance-free Incremental MCA

    $\mathbf{W} \leftarrow$ CIMCA-UPDATE ($\mathbf{W}, J, \dot{\mathbf{z}}, \gamma, \eta_t^{MCA}$)

  end

  $\mathbf{y} \leftarrow \mathbf{z}_{curr}^T \mathbf{W}$ //Slow feature output

end

A few items to note on the algorithm:

3.1 Intermediate dimensionality reduction

Since often only a relatively small number of principal components of $\mathbf{x}$ are needed to explain most of the variance in the data, the other components do not even have to be estimated. With IncSFA, dimensionality reduction can be done during PC estimation, and no time needs to be wasted on computing insignificant lower-order PCs. The whitening output dimension K must be set by hand[2]. However, some prior problem knowledge seems necessary: the insignificant lower-order PCs may contain data corresponding to the slowest varying signal in the input. It would be unwise to remove them in this case, since discarding these might eliminate an important slow feature.

3.2 On complexity

From the algorithm, it can be seen that IncSFA complexity with respect to input dimension is $O(I)$.

With respect to the K eigenvectors after the CCIPCA step, and J slow feature eigenvectors, every IncSFA update is of complexity $O(K + J^2)$. The quadratic complexity on J is due to the Gram-Schmidt procedure in the CIMCA algorithm. CCIPCA uses the residual method, which has linear complexity. However, we expect $J < K$ and $K < I$, so the quadratic complexity on J should not hurt much.

3.3 Parameters

There are just a handful. One must choose K and J and set a learning rate schedule. We provide some discussion on learning rates in Sec. 5.2.

4 Experiments and Results

Experiments were done either using Python (based on the MDP toolbox (T. Zito and Berkes, 2008)) or Matlab[3].

[2] One might set K such that 95% of the estimated total data variance is kept, for example.

[3] Python code is available at `www.idsia.ch/~kompella/codes/incsfa.html`. Matlab code is available at `www.idsia.ch/~luciw/incsfa.html`

4.1 Proof of Concept

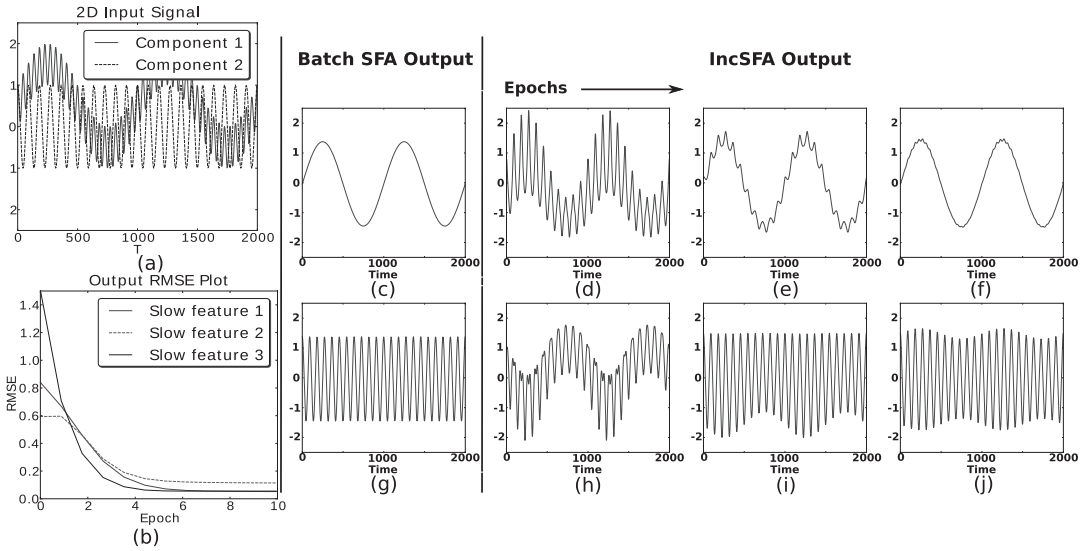


Figure 2: Extracting slow features incrementally from a simple non-linear input signal. (a). Input Signal (b). Output RMSE plot showing convergence of the first three IncSFA features to the corresponding BSFA features. (c). Batch SFA output of the first slow feature (d)-(f). IncSFA output of feature 1 at t = 2, 5, 10 epochs. (g). Batch SFA output of the second slow feature (h)-(j). IncSFA output of feature 2 at t = 2, 5, 10 epochs.

As a basic proof of concept, IncSFA is applied to introductory problem from the original SFA paper (Wiskott and Sejnowski, 2002). We want to show that IncSFA can derive the same set of features as BSFA. The input signal is

$$\breve{x}_1(t) = \sin(t) + \cos(11\,t)^2, \tag{18}$$

$$\breve{x}_2(t) = \cos(11\,t),\ t \in [0, 2\pi], \tag{19}$$

Both input components vary quickly over time (see Figure 2(a)). The slowest feature hidden in the signal is $y_1(t) = \breve{x}_1(t) - \breve{x}_2(t)^2 = \sin(t)$. The second slowest feature is $y_2(t) = \breve{x}_2(t)^2$.

Each epoch contains a total of $2,000$ discrete datapoints, over the entire range of t, are used for learning. A quadratic input expansion is done. A learning rate of $\eta^{MCA} = 0.08$ is used.

Both BSFA and IncSFA extract the correct features. Figure 2(b) shows the Root Mean Square Error (**RMSE**) of three IncSFA feature outputs compared to the corresponding BSFA outputs, over multiple epochs of training, showing that the IncSFA features converge to the correct ones. Figures 2(c) and (g) show feature outputs of batch SFA, and (to the right) IncSFA outputs at $2,\ 5$, and 10 epochs. Figures 2(g)-(j) show this comparison for the second feature. This basic result shows that it is indeed possible to extract multiple slow features in an online way without storing covariance matrices.

4.2 Feature Adaptation to a Changing Environment

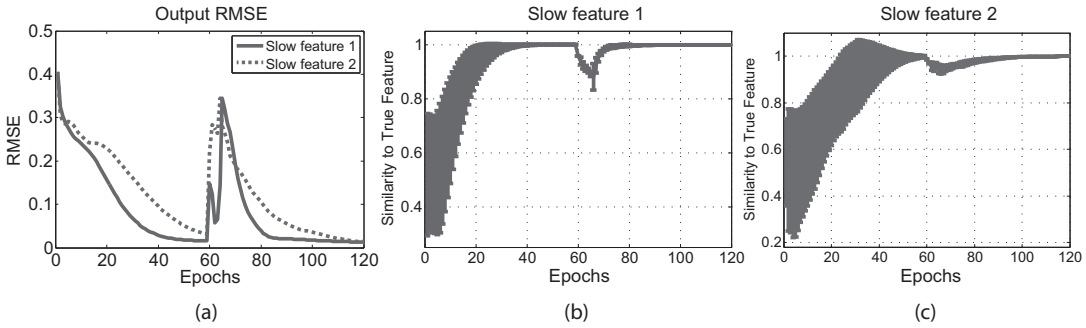


Figure 3: (a) RMSE of IncSFA's first two output functions with respect to the true functions for original signal (epochs 1-59), and switched signal (epochs 60-120). (b) Normalized similarity (direction cosine) of the first slow feature to the true first slow feature of the current process, over 25 independent runs. (c) Normalized similarity of the second incremental slow feature.

The purpose of this experiment is to illustrate how IncSFA's features *adapt* to an unpredicted sudden shift in the input process. The input used is the same signal as in Experiment #1, but broken into two partitions. At epoch 60, the two input lines x_1 and x_2 are switched such that the x_1 signal suddenly carries what x_2 used to, and vice versa. IncSFA can first learn the slow features in the first partition, then is able to adapt to learn the slow features in the second partition.

Here, the signal is sampled 500 times per epoch. The CCIPCA learning rate parameters, also used to set the learning rate of the input average $\bar{\mathbf{x}}$, were set to $t_1 = 20, t_2 = 200, c = 4, r = 5000$ (See Sec.5.2). The MCA learning rate is a constant $\eta_{mca} = 0.01$.

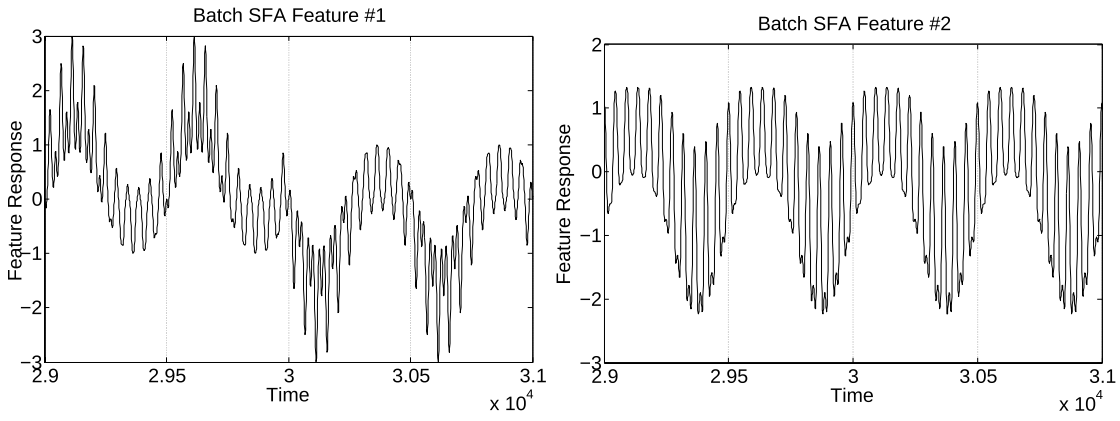


Figure 4: Outputs of first two slow features, from epoch 59 through 61, extracted by batch SFA over the input sequence.

Results of IncSFA are shown in Fig. 3, demonstrating successful adaptation. To measure convergence accuracy, we use the direction cosine (Chatterjee et al., 2000) between the estimated feature $\mathbf{w}(t)$ and true (unit length) feature $\mathbf{w}^*$,

$$DirectionCosine(t) = \frac{|\mathbf{w}^T(t) \cdot \mathbf{w}^*|}{\|\mathbf{w}^T(t)\| \cdot \|\mathbf{w}^*\|}, \tag{20}$$

The direction cosine equals one when the directions align (the feature is correct) and zero when they are orthogonal.

BSFA results are shown in Fig. 4. The first batch feature somewhat catches the meta-dynamics and could actually be used to roughly sense the signal switch. However, the dynamics within each partition are not extracted. The BSFA result might be improved by generating embedding-vector time series (Wiskott, 2003) and increasing the non-linear expansion. But due to long duration of the signals and the unpredicted nature of the signal switch, time-embedding with a fixed delay might not be able to recover the dynamics appreciably.

4.3 Recovery from Outliers

Now, we show the effect of a single extreme outlier on both BSFA and IncSFA. Again, the learning rate setup and basic signal from the previous experiments are used, with 500 samples per epoch, over 150 epochs. A single outlier point is inserted at time 100 (only in the first epoch!): $x_1(100) = x_2(100) = 2000$.

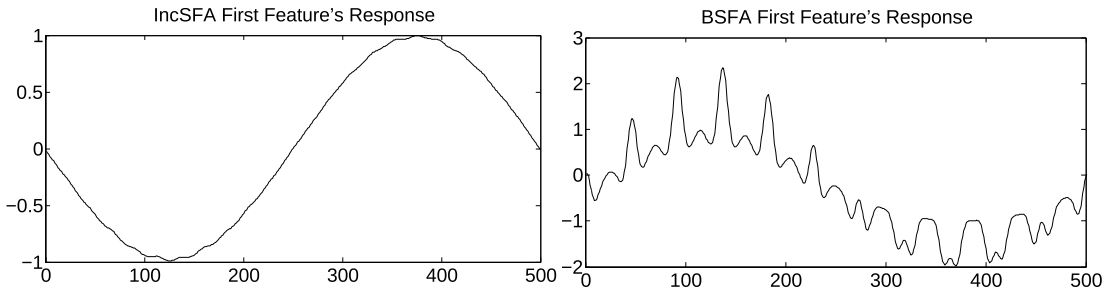


Figure 5: First output signals of IncSFA and BSFA on the simple signal with a single outlier.

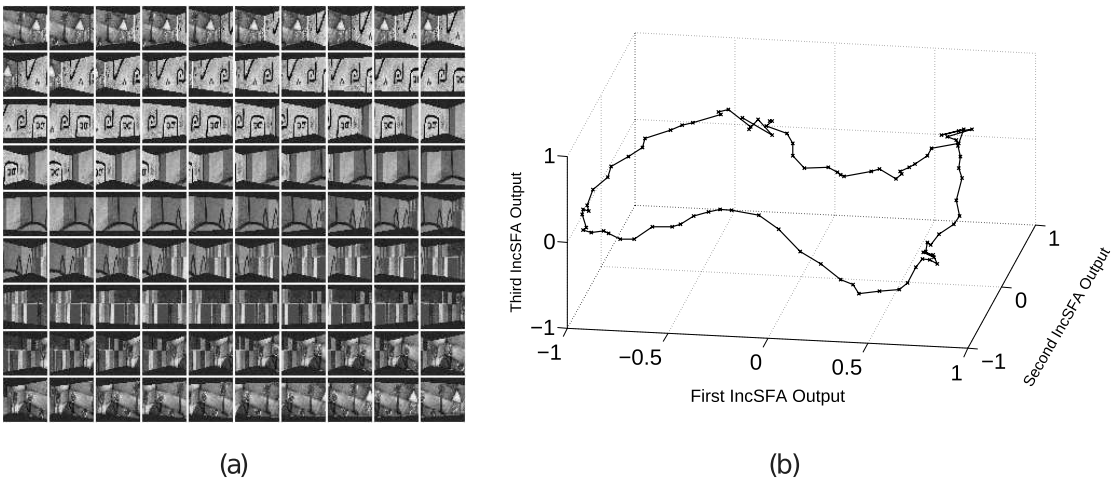
Figure 5 shows the first output signal of BSFA and IncSFA. The one outlier point at time 100 (out of 75,000) is enough to corrupt the first feature of BSFA, whereas IncSFA recovers. It is possible to include clipping Franzius et al. (2007) in BSFA, so that the effect of the outliers that have different variance statistics compared to the signal can be overcome.

Outliers that are generated from another signal source lying within the variance of the main signal can affect the BSFA output in a different way. We refer to a real-world experiment (Kompella et al., 2011b), using AutoIncSFA (where the input to IncSFA is the output at a bottleneck layer of an autoencoder neural net — for image compression) on an image sequence, in which a person moves back and forth in front of a stable camera. At only one point in the training sequence, a door in the background is opened. The BSFA hierarchical network's first slow feature became sensitive to this event. Yet, the AutoIncSFA network's first slow feature encodes the relative distance of the moving interactor.

4.4 High-Dimensional Video with Linear IncSFA

Via IncSFA, SFA can be utilized in some high-dimensional video processing applications without using deep receptive-field based networks. CCIPCA provides an intermediate dimensionality reduction, which, when low enough compared to the input dimension, can greatly reduce the computational and space complexities as well as the search space for the slow features via MCA.

As a first experiment to show this, we extract SFs from a rotating vision-based agent in a square room. The room has four complex-textured walls. See Fig. 6(a). Each image



(a) (b)

Figure 6: (a) Stream of 90 $41 \times 41 \times 3$ images as the agent completes one rotation (360 degrees). There are 10 subsequent images per row, starting from the top-left. The image after that at the far right of a row starts at the far left of the lower row. (b) All 90 images (noise-free) projected onto the first three features learned by IncSFA. We can easily see the 1D and circular nature of the agent's movement within the environment. This embedding can be used as a compact encoding of the agent's state.

is dimension $41 \times 41 \times 3$.

In each episode[4], starting from a different orientation, the agent rotates slowly (4 degree shifts from one image to the next) by 360 degrees, each episode. At any time, a slight amount of Gaussian noise is added to the image ($\sigma = 8$).

Each $5,043$ dimensional image is fed into a linear IncSFA directly. Only the 40 most significant principal components are computed by CCIPCA, using learning rate parameters $t_1 = 20,\ t_2 = 200,\ c = 4,\ r = 5000$ (See Sec.5.2). Computation of the covariance matrix and its full eigendecomposition (over 5000 eigenvectors and eigenvalues) is therefore avoided. On the 40 dimensional whitened difference signal, only the first 5 slow features are computed via CIMCA.

500 epochs through the data took approximately 15 minutes using Matlab on a machine with an Intel i3 CPU and 4 GB RAM. This is a framerate of about $50fps$.

[4]IncSFA can be readily extended to episodic tasks, with a minor modification: The derivative signal, which is computed as a difference over a single time step, is simply not computed for the starting sample of each episode. The first data point in each episode is used for updating the PCs, but not the slow feature vectors.

The result of projecting the (noise-free) data onto the first three slow features are shown in Fig. 6(b). A single linear IncSFA has incrementally compressed this high-dimensional noisy sequence to a nearly unambiguous compact form, learning to ignore the details at the pixel level and attend to the true cyclical nature underlying the image sequence. We say "nearly" since a few subsequences have somewhat ambiguous encodings, probably because certain images associated with slightly different angles are very similar.

4.5 iCub Experiment

Again, we use a single layer of IncSFA on high-dimensional vision sequences. Two plastic cups are placed in the iCub robot's field of view. The robot performs motor babbling in one joint of its right arm, using a movement paradigm derived from Franzius *et al.* During the course of babbling, it happens to topple both cups, in one of two possible orders. The episode ends a few frames after it has knocked both down. A new episode begins with the cups upright again and the arm in the beginning position. A total of 50 separate episodes were recorded and the images used as training data.

IncSFA updates from each 80×60 (grayscale) image. Only the 20 most significant principal components are computed by CCIPCA, using learning rate parameters $t_1 = 20,\ t_2 = 200,\ c = 2,\ r = 10000$ (See Sec.5.2). Only the first 5 slow features are computed via CIMCA with learning rate 0.001. The MCA vectors are normalized after each update during the first 10 episodes, but not thereafter (for faster convergence). The algorithm runs for 400 randomly-selected (of the 50 possible) episodes. We replicated it 25 times.

Results are shown in Fig. 7. The slowness of the feature outputs were measured on three "testing" episodes, after each episode of training. The upper left plot shows that all five features get slower as they train over the 400 episodes. Figure 8 shows the average mutual direction cosine between non-identical pairs of slow features, and we can see the features quickly become nearly decorrelated.

After training completes, we embed the images in a lower dimension using the learned features. The embedding of trajectories of 20 different episodes are shown with respect to the first two PCs as well as the first two slow features. Since the cups being

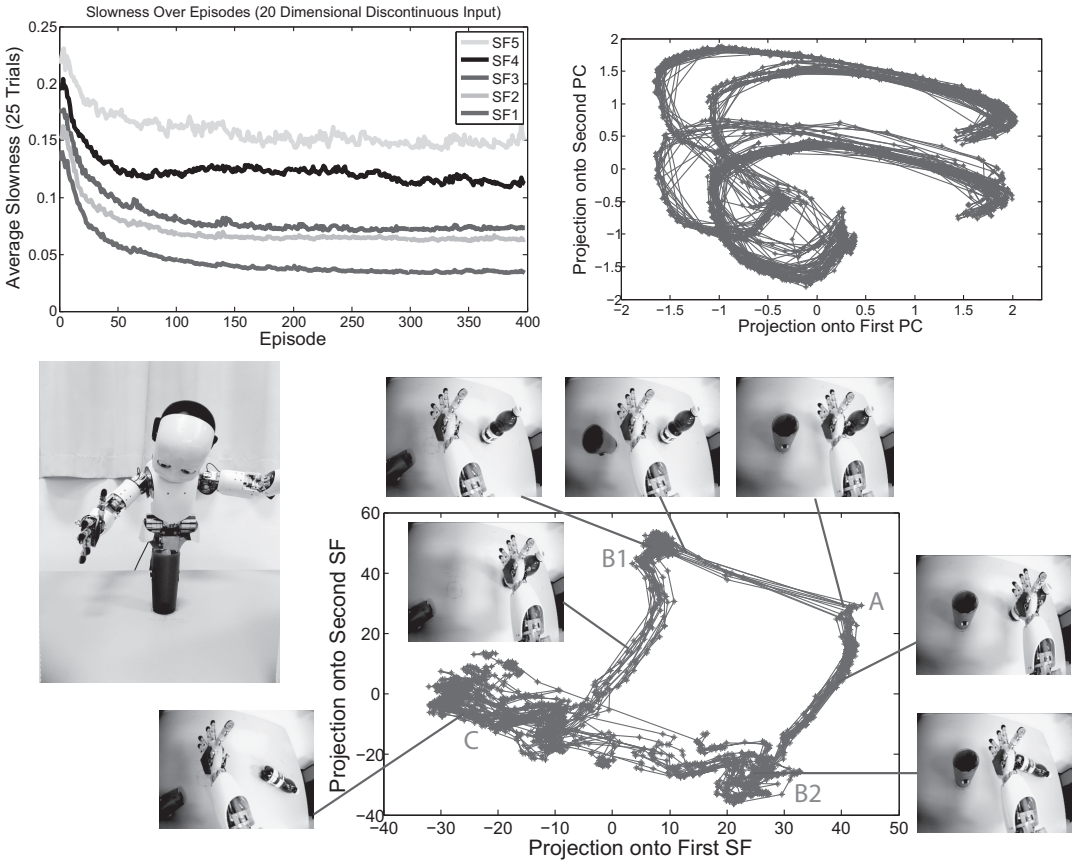


Figure 7: Experimental result of IncSFA on episodes where the iCub knocks down two cups via motor babbling on one joint. Upper left: The average slowness of the five features at each episode. Upper right: after training, several episodes (each episode is an image sequence where the cups are eventually both knocked down) are embedded in the space spanned by the first two PCs. Lower right: the same episodes are embedded in the space spanned by the first two slow features. We show some example images and where they lie in the embedding. The cluster in the upper right (A) represents when both cups are upright. When the robot knocks down the blue cup first, it moves to the cluster in the upper left (B1). If it instead knocks down the brown cup, it moves to the lower right cluster (B2). Once it knocks down both cups, it moves to the lower left area (C).

toppled or upright are the slow events in the scene, IncSFA's encoding is keyed on the object's state (toppled or upright). PCA does not find such an encoding, being much more sensitive to the arm. Such clear object-specific low-dimensional encoding, invariant to the robot's arm position, is useful, greatly facilitating training of a subsequent

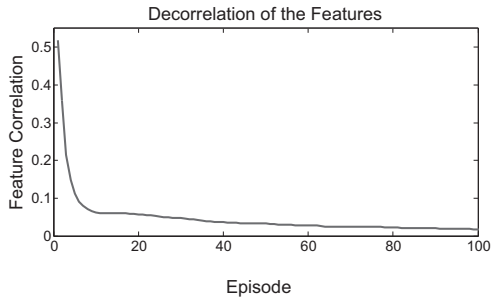


Figure 8: Average slow feature similarity over episodes in the iCub experiment.

regressor or reinforcement learner. A video of the experimental result can be found at `http://www.idsia.ch/~luciw/IncSFAArm/IncSFAArm.html`.

4.6 Hierarchical IncSFA

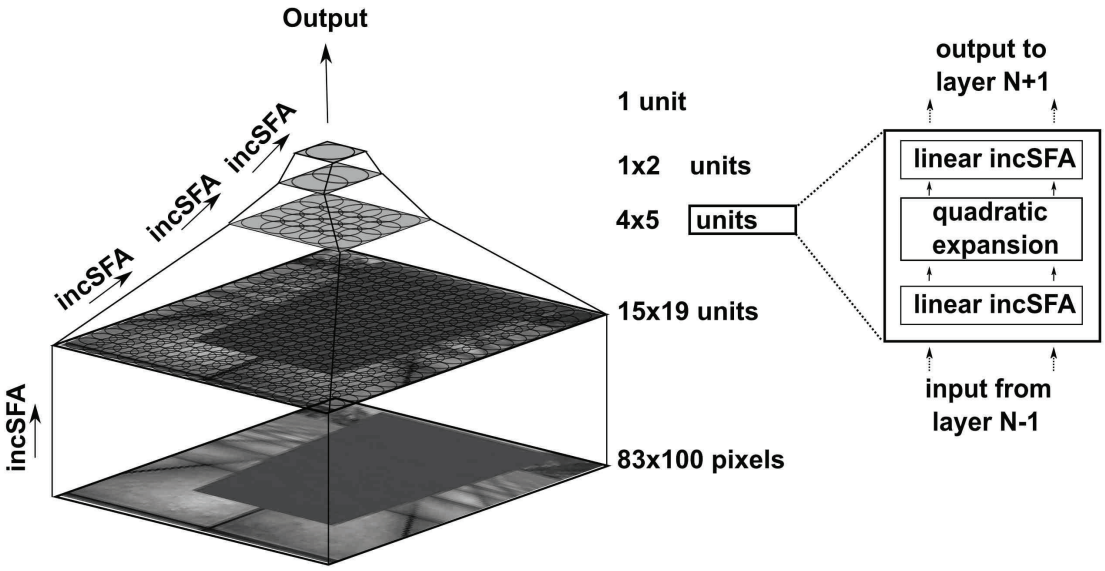


Figure 9: Example Hierarchical-IncSFA Architecture. This also shows the structure of an IncSFA node, which contains a linear IncSFA unit followed by nonlinear expansion followed by another linear IncSFA unit.

Here, for completeness, we test IncSFA within a deep receptive-field based network — at all layers — although the utility of this approach is unclear.

Deep networks composed of multiple stacked BSFA nodes, each sensitive to only a small part of the input (i.e., receptive fields), are typically used for SFA processing

high-dimensional images. The slow features will be linear combinations of the input space components. Since there's no guarantee the useful information is linear in the original sensory space, an expanded space is often used. For example, a quadratic expansion adds all combinations of input components, or a cubic expansion adds all triples. But the degree of expansion required to construct a space where the "interesting information" will be some linear combination, may increase the dimension intractably. What has been done to deal with these cases is to use multilayer, receptive-field based networks (Wiskott and Sejnowski, 2002; Franzius et al., 2007), which reduce the complexity for any SFA module by partitioning spatially on each layer into receptive fields, while having a low-order (e.g., quadratic) expansion within each receptive field. A succession of low-order expansions over multiple layers lead to an overall expansion which is high-order.

Hierarchical networks introduce new parameters (receptive field size, number of layers, etc.) that can be difficult to tune. We have tried to show in the last two experiments that there is another applicable tactic, which is to apply IncSFA monolithically to the (possibly even expanded) high-dimensional input, extracting $K << I$ principle components with CCIPCA and J slow features. But we can also use IncSFA within a deep network architecture.

Figure 9 shows an example deep network, motivated by the human visual system and based on the one specified by Franzius *et al.* (Franzius et al., 2007). The network is made up of a converging hierarchy of layers of IncSFA nodes, with overlapping rectangular receptive fields. Each IncSFA node finds the slowest output features from its input within the subspace of quadratically expanded inputs.

Input images come from a high-dimensional video stream generated by the iCub simulator (Tikhanoff et al., 2008), an OpenGL-based software specifically built for the iCub robot. Our experiment mimics the robot observing a moving interactor agent, which in the simulation takes the form of a rectangular flat board moving back and forth in depth over the range $\{1, 3\}$ (meters) in front of the robot, using a movement paradigm based on that of Franzius. Figure 10(a) shows the experimental setup in the iCub simulator. Figure 10(b) shows a sample image from the dataset. $20,000$ monocular images are captured from the robot's left eye and downsampled to 83×100 pixels (input dimension of $8,300$).

A three-layer IncSFA network is used to encode the images. Each SFA node operates on a spatial receptive field of the layer below. The first layer uses 15×19 nodes, each with 10×10 image patch receptive field and a 5 pixel overlap. Each node on this layer develops 10 slow features. The second layer uses 4×5 nodes, each having a 5×5 receptive field, and developing 5 slow features. The third layer uses two nodes, one sensitive to the top half, the other sensitive to the bottom half (5 slow features). The forth layer uses a single node and a single slow feature. The network is trained layer-wise from bottom to top, with the lower layers frozen once a new layer begins its training. The CCIPCA output of all nodes is clipped to $[-5, 5]$, to avoid any outliers that may arise due to close-to-zero eigenvalues in some of the receptive fields that contain unchanging stimuli. Each IncSFA node is trained individually, that is, there is no weight sharing among nodes.

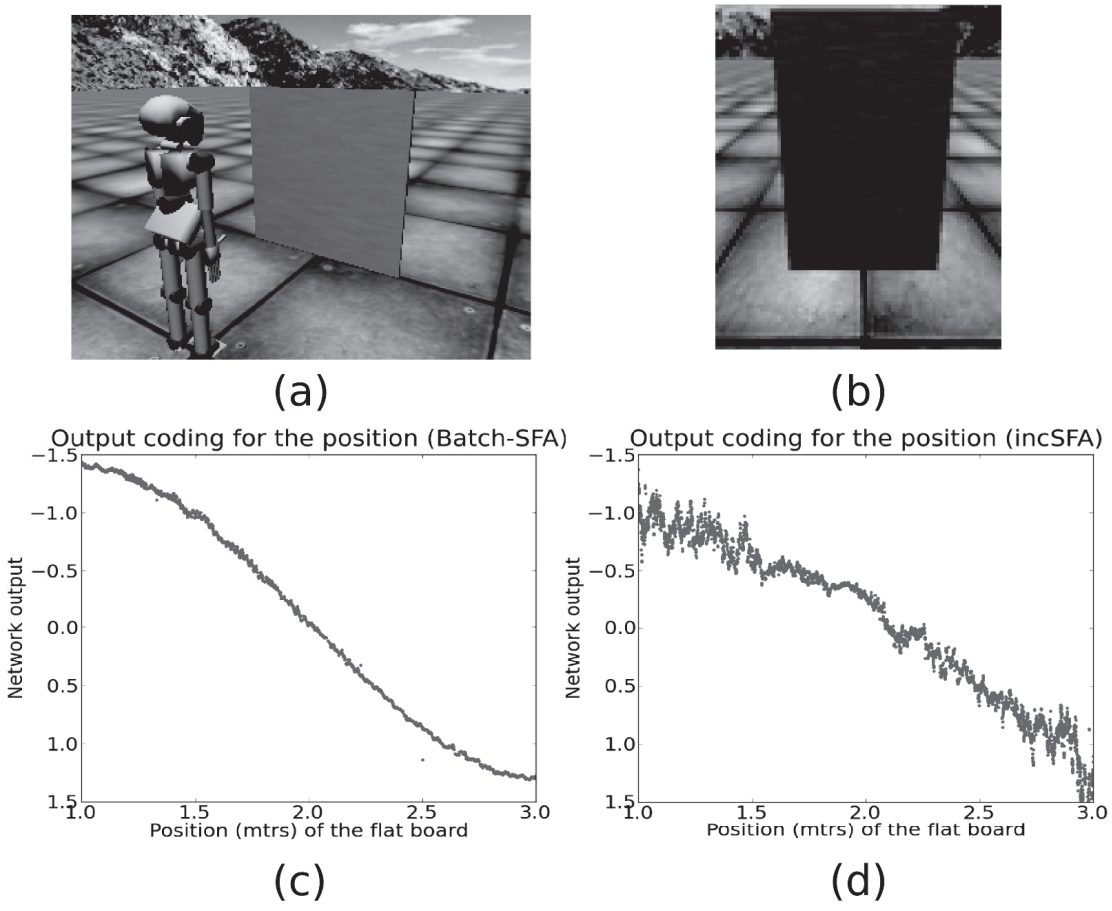


Figure 10: (a) Experimental Setup: iCub Simulator (b) Sample image from the input dataset (c) Batch-SFA output (d) IncSFA output ($\eta_{mca} = 0.005$)

For comparison, a BSFA hierarchical network was also trained on this data. Figures 10 show BSFA and IncSFA outputs. The expected output is of the form of a sinusoid extending over the range of board positions. IncSFA gives a slightly noisy output, probably due to the constant dimensionality reduction value for all units in each layer of the network, selected to maintain a consistent input structure for the subsequent layer; hence some units with eigenvectors corresponding to very small eigenvalues emerge in the first stage, with receptive fields observing comparatively few input changes, thus slightly corrupting the whitening result, and adding small fluctuations to the overall result.

Finally, we evaluate how well the IncSFA feature codes for distance. A supervised quadratic regressor is trained with ground truth labels on 20% of the dataset, and tested on the other 80%, to measure the quality of features for some classifier or reinforcement learner using them. The **RMSE** was found to be equal to 0.043 meters.

5 Supplementary Topics

5.1 On Non-Divergence and Convergence

For CCIPCA: If the standard conditions on learning rate (Papoulis et al., 1965) (including convergence at zero), the first stage components will converge to the true PCs, leading to a "nearly-correct" whitening matrix in reasonable time. So, if input $\mathbf{x}$ is stationary, the slow feature estimates are likely to become quite close to the true slow features in a reasonable amount of updates.

In open-ended learning, convergence is usually not desired. Yet by using a learning rate that is always nonzero, the stability of the algorithm is reduced. This corresponds to the well-known stability-plasticity dilemma (Grossberg, 1980).

For stability and convergence of incremental MCA, the following constraints must be satisfied (Peng et al., 2007),

$$\eta^{MCA}\lambda_1^* < 0.5, \tag{21}$$

$$||\mathbf{w}(0)||^2 \leq \frac{1}{2\eta^{MCA}}, \tag{22}$$

$$\mathbf{w}^T(0)\mathbf{w}^* \neq 0 \tag{23}$$

where $\mathbf{w}(0)$ is the initial feature estimate, $\mathbf{w}^*$ the true eigenvector associated with the smallest eigenvalue, and λ_1^* the largest eigenvalue. In other words, the learning rate must not be too large, and the initial estimate must not be orthogonal to the true component.

It is clear that if whitened signal $\mathbf{z}$ is drawn from a stationary distribution, the MCA convergence proof (Peng et al., 2007) applies. But typically the whitening matrix is being learned simultaneously. In this early stage, while the CCIPCA vectors are learning, care must be taken to ensure that the slow feature estimates will not diverge.

Peng showed that for any initial vector $\mathbf{w}(0)$ within the set $\mathcal{S}$,

$$\mathcal{S} = \left\{ \mathbf{w}(t) | \mathbf{w}(t) \in \mathcal{R}^K \text{ and } \|\mathbf{w}(t)\|^2 \leq \frac{1}{2\eta^{MCA}} \right\}, \tag{24}$$

$\mathbf{w}(t)$ $(\forall t \geq 0)$ will remain in $\mathcal{S}$ throughout the dynamics of the MCA updating. Thus, $\|\mathbf{w}\|$ must be prevented from getting too large until the whitening matrix is close to accurate. With respect to lower-order slow features, there is additional dependence on the sequential addition technique, parameterized by $\gamma(t) = \lambda_1(t) + \epsilon$. This $\gamma(t)$ also needs time to estimate a close value to the first eigenvalue λ_1. Before these estimates become reasonably accurate, the input can knock the vector out of $\mathcal{S}$.

In IncSFA, $\mathbf{w}$ is normalized after each update. If $\|\mathbf{w}(0)\| = 1$ then any learning rate $\eta_{mca} \leq 0.5$ ensures non-divergence.

Even if $\mathbf{w}$ remains in $\mathcal{S}$, the additional constraint $\mathbf{w}^T(0)\mathbf{w}^* \neq 0$ is needed for convergence. But this is an easy condition to meet, as it is unlikely that any $\mathbf{w}(t)$ will be exactly orthogonal to the true feature. In practice, it may be advisable to add a small amount of noise to the MCA update. But we did not find this to be necessary.

5.2 Learning Rate Scheduling

The methods we used to schedule the learning rates η^{PCA} and η^{MCA} are presented in Algorithm 5. There are certainly many other ways to set the learning rates.

Algorithm 5: LEARNINGRATESCHEDULE(θ, t)

//Example Learning Rate Schedule

$//\theta = (t_1, t_2, c, r, \eta_l, \eta_h, T)$

//example: $t_1 = 20, t_2 = 200, c = 3, r = 2000$

$$\mu_t = \begin{cases} 0 & \text{if } t \leq t_1, \\ c(t - t_1)/(t_2 - t_1) & \text{if } t_1 < t \leq t_2, \\ c + (t - t_2)/r & \text{if } t_2 < t. \end{cases}$$

$\eta_t^{PCA} \leftarrow (1 + \mu_t)/t$

//example: $\eta_h = 0.01, \eta_l = 0, T = 2000$

$$\eta_t^{MCA} = \begin{cases} \eta_l + (\eta_h - \eta_l) * \left(\frac{t}{T}\right)^2 & \text{if } t \leq T, \\ \eta_h & \text{if } T < t. \end{cases}$$

return $\{\eta_t^{PCA}, \eta_t^{MCA}\}$

5.1 Pseudo-optimality

For CCIPCA, the learning rate schedule is based around the optimal $\eta_t^{PCA} = \frac{1}{t}$. If we use $1/t$, Eq. 10 will be the most efficient estimator of the principal component. The most efficient estimator on average requires the least samples for learning (among all unbiased estimators). For several common distribution types, e.g., Gaussian, the sample mean is the maximum likelihood estimator of the population mean. And observe that Eq. 10 reformulates the eigenvector estimation problem as a mean estimation problem. Therefore, Eq. 10 and learning rate $1/t$ has a *spatiotemporal optimality*: at *any* t the estimate is expected to be the best as compared to any other unbiased estimator.

Learning rate $1/t$ is only spatiotemporally optimal if every sample from $t = 1, 2, ..., \infty$ is drawn from the same distribution, which will not be the case for the lower-order components, and in general for autonomous agents. We use an amnesic averaging technique, where the influence of old samples on the current estimates diminish over time. We used the three-sectioned amnesic averaging function μ, shown in the algorithm. It uses three stages, defined by points t_1 and t_2. In the first stage, the learning rate is $\frac{1}{t}$. In the second, the learning rate is scaled by c to speed up learning of lower-order components. In the third, it changes with t, eventually converging to $1/r$.

This amnesic average remains an unbiased estimators of the true PCs, and it allows

components to adapt to changing input statistics. But this plasticity introduces an expected error that will not vanish with more samples (Weng and Zhang, 2006). This introduces some expected error into the IncSFA whitening process. Our results show that this is not problematic for many applications, but this typically leads to a slight oscillatory behavior around the true features.

5.2 Preventing divergence via the MCA learning rate

To prevent divergence while CCIPCA is still learning, we used a slowly rising learning rate for MCA, starting from low η_l at $t = 0$ and rising to high η_h at $t = T$, as shown in Algorithm 5. Ideally, T is a point in time when whitening has stabilized.

The upper bound η_b of permissible η_h is related to the first condition in Eq. 21:

$$\eta_h < \eta_b = \frac{1}{2\lambda_1^*}, \tag{25}$$

where λ_1^* is the greatest eigenvalue of the signal. Constant values close to but below the bound will achieve faster convergence.

5.3 Other Methods of Neural Updating in PC and MC Extraction

Neural layers that compute incremental PCA (IPCA) and MCA build on the work of Amari (1977) and Oja (1982). They showed that a linear neural unit using Hebbian updating could incrementally compute the first principal component of a data set (Amari, 1977; Oja, 1982)[5]. Many IPCA algorithms emerged after that. Some well-known ones are Oja and Karhunen's Stochastic Gradient Ascent (SGA) (Oja, 1985), Oja's Subspace algorithm (Oja, 1989), Sanger's Generalized Hebbian Algorithm (GHA) (Sanger, 1989), the Weighted Subspace algorithm (Oja, 1992), and CCIPCA. For more information on comparisons, see (Oja, 1992; Hyvärinen et al., 2001; Weng et al., 2003). CCIPCA (Weng et al., 2003) modified GHA to be "candid" — meaning it is invariant to input vector magnitude, thus learning rate tuning became more intuitive, which increased the practicality of the algorithm for high-dimensional inputs such as in appearance-based computer vision. There is another recent IPCA algorithm that adds

[5]Earlier work of a non-neural network flavor had shown how the first PC, including the eigenvalue could be learned incrementally (Krasulina, 1970).

GSO to CCIPCA Park and Choi (2008), so that the lower-order components should converge quicker, but with higher complexity.

We created IncSFA with experiments on high-dimensional data in mind. Thus, we chose CCIPCA for IncSFA for the following reasons.

1. **Covariance-Free**. Mentioned earlier. Due to high-dimensionality, it is important to keep space complexity down.

2. **Avoid Gram-Schmidt orthonormalization (GSO)** for enforcing orthogonality. Instead, we prefer the residual (Kreyszig, 1988) method. GSO will give more accurate result for lower-order components, but at quadratic (in the number of components) complexity. The residual method is local (linear complexity), but can be less accurate. Again, due to high-dimensionality, we felt it important to avoid quadratic complexity. Further, our experimental results showed that effective slow features could emerge even when the whitening matrix was not perfect.

3. **Both Eigenvalues and Eigenvalues Needed**. We need a method that converges to both eigenvectors and eigenvalues, necessary since whitening requires both.

4. **Intuitive to Tune the Learning Rate**. It is not practical to spend a lot of time tuning learning rates for every different type or set of data.

As for MCA: Xu *et al.* (Xu et al., 1992) were the first to show that a linear neural unit equipped with anti-Hebbian learning could extract minor components. Oja modified SGA's updating method to an anti-Hebbian variant (Oja, 1992), and showed how it could converge to the MC subspace. Studying the nature of the duality between PC and MC subspaces (Wang and Karhunen, 1996; Chen et al., 1998), Chen, Amari and Lin (Chen et al., 2001) (2001) introduced the sequential addition technique. This enabled linear networks to efficiently extract multiple MCs simultaneously. Building upon previous MCA algorithms, Peng (2007) (Peng et al., 2007) derived the conditions and a learning rule for extracting MCs for a constant learning rate. Sequential addition was added to this rule so that multiple MCs could be extracted (Peng and Yi, 2006).

We use a modified version of Peng's MCA updating method, slightly altered to be covariance-free and using GSO (we simply call it CIMCA). Unlike simple MCA algorithms, Peng's MCA is a deterministic discrete time (DDT) method, which requires

setting a constant learning rate to achieve convergence. The method has low computational complexity and is shown to work even for singular or near-singular correlation matrix of the input. This makes it practical and especially feasible for non-stationary data where the correlation coefficient behaves like a random variable.

CIMCA gives us the actual minor components (slow features), not just the subspace they span. It allows for a constant learning rate, which can be quite high, leading to a quick reasonable estimate of the true components, and making learning rate tuning more intuitive. Unlike at the IPCA stage, here GSO is useful and plausible since we expect not to have too many features (minimizing the effect of the quadratic complexity) and we don't care about their magnitude.

It should be noted that there are many different ways of combining an incremental PCA and an incremental MCA. We present our reasons for selecting the methods in IncSFA above. Our motivation should be kept in mind: we intend to apply IncSFA on real-world image sequences and on vision-based robotic platforms, aiming towards autonomous learning, which is open-ended, continuous, etc.

5.4 Links to Biological Systems

BSFA has been shown to derive slow features that operate like biological grid cells from quasi-natural image streams, which are recorded from the camera of a moving agent exploring an enclosure (Franzius et al., 2007). In rats, grid cells are found in entorhinal cortex (EC) (Hafting et al., 2005), which feeds into the hippocampus. Place cells and head-direction cells are found in rat hippocampus (O'Keefe and Dostrovsky, 1971; Taube et al., 1990), while spatial view cells are found in primate hippocampus (Rolls, 1999). Augmenting the BSFA network with an additional competitive learning (CL) layer derives units similar to place, head-direction, and spatial view cells.

Although BSFA results exhibit the above biological link, it is not clear how the full SFA technique might be realized in the brain. IncSFA with its Hebbian and anti-Hebbian updating provides a more biologically plausible implementation of the full SFA algorithm.

5.1 Hebbian Updating in CCIPCA

Hebbian updates of synaptic strengths of some neuron make it more sensitive to expected input activations (Dayan and Abbott, 2001):

$$\mathbf{v} \leftarrow \mathbf{v} + \eta \; g(\mathbf{v}, \mathbf{u}) \; \mathbf{u}, \tag{26}$$

where $\mathbf{u}$ represents pre-synaptic (input) activity, and g post-synaptic activity (a function of similarity between synaptic weights $\mathbf{v}$ and input potentials $\mathbf{u}$). The basic Eq. 26 requires additional care (e.g., normalization of $\mathbf{v}$) to ensure stability during updating. To handle this in one step, learning rate η and retention rate $1 - \eta$ can be used,

$$\mathbf{v} \leftarrow (1 - \eta)\mathbf{v} + \eta \; g(\mathbf{v}, \mathbf{u}) \; \mathbf{u}. \tag{27}$$

where $0 \leq \eta \leq 1$. With this formulation, Eq. 10 is Hebbian, where the post-synaptic activity is the normalized response $g(\mathbf{v}, \mathbf{u}) = \dfrac{\mathbf{u}(t) \cdot \mathbf{v}(t-1)}{\|\mathbf{v}(t-1)\|}$ and the presynaptic activity is the input $\mathbf{u}_i$.

5.2 Anti-Hebbian Updating in CIMCA

The general form of anti-Hebbian updating simply results from flipping the sign in Eq. 26. In IncSFA notation:

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \; g(\mathbf{w}, \dot{\mathbf{z}}) \; \dot{\mathbf{z}}. \tag{28}$$

To see the link between Peng's MCA updating and the anti-Hebbian form, in the case of the first MC, we note Eq. 12 can be rewritten as

$$\begin{aligned}
\mathbf{w}_1 &\leftarrow 1.5\mathbf{w}_1 - \eta \; \left[\mathbf{C}_1 \; \mathbf{w}_1 + [\mathbf{w}_1^T \mathbf{w}_1] \; \mathbf{w}_1\right], && (29)\\
&\leftarrow 1.5\mathbf{w}_1 - \eta \; \left[(\dot{\mathbf{z}} \cdot \mathbf{w}_1) \; \dot{\mathbf{z}} + (\mathbf{w}_1 \cdot \mathbf{w}_1) \; \mathbf{w}_1\right], && (30)\\
&\leftarrow 1.5\mathbf{w}_1 - \eta \; \|\mathbf{w}_1\|^2 \; \mathbf{w}_1 - \eta \; \left((\dot{\mathbf{z}} \cdot \mathbf{w}_1) \; \dot{\mathbf{z}}\right), && (31)\\
&\leftarrow \left(1.5 - \eta \; \|\mathbf{w}_1\|^2\right) \; \mathbf{w}_1 - \eta \; (\dot{\mathbf{z}} \cdot \mathbf{w}_1) \; \dot{\mathbf{z}}, && (32)
\end{aligned}$$

where $(\dot{\mathbf{z}} \cdot \mathbf{w}_1)$ indicates post-synaptic strength, and $\dot{\mathbf{z}}$ pre-synaptic strength.

5.3 Hebbian Learning on Filtered Output

There is an alternative to using anti-Hebbian learning. Sprekeler et al. (Sprekeler et al., 2007) reformulated the slowness objective: instead of minimizing the variance of the time derivative of the output signal, they try to *maximize* the variance of the *low-pass filtered* output signal. They show analytically that the extraction of the single most slowly varying direction from pre-whitened input can be implemented in a linear continuous model with spiking model neurons by means of a modified hebbian learning rule with a specific learning window.

Hebbian learning between a temporally filtered output and input is the basis of several other temporal-stability based learning rules (Földiák, 1991; O'reilly and Johnson, 1994; Wallis and Rolls, 1997). Links between these and slowness learning are provided by Sprekeler *et al.* (Sprekeler et al., 2007). However, even though Sprekeler's method is only for the first feature, it might lead to alternate approach to reach a fully incremental SFA.

5.5 Velocity Estimates of the Input Signal

The velocity estimates (the derivative signal) in the original SFA technique are approximated via a backward difference method $\dot{\mathbf{z}}(t) = \mathbf{z}(t) - \mathbf{z}(t-1)$. This method behaves badly in the presence of input noise compared to other methods (that are computationally expensive) such as higher order difference estimation, cauchy's differentiation formula, or lanczos derivative computation etc. (Groetsch, 1998). However, noise is usually not a severe problem, since it changes at a faster time-scale compared to the slowest components and therefore does not show up in the higher-order slow features. Therefore, we opted the same backward difference method for the IncSFA to keep it computationally simple.

6 Conclusions

This paper describes the novel Incremental Slow Feature Analysis technique, which updates slow features incrementally without computing covariance matrices. IncSFA's main advantages are low computational and space complexities. For many instances,

there is no need to use IncSFA instead of BSFA. But at higher dimensionalities, IncSFA becomes more and more appealing. For some problems with very high-dimensionality and limited memory, IncSFA could be the only option, e.g., an autonomous robot with limited onboard hardware, which could still learn slow features from its visual stream via IncSFA. Experiments showed how IncSFA enables an adaptive SFA, and how it enables SFA to be applied to high-dimensional image streams without using multilayer receptive-field based BSFA architectures. IncSFA's Hebbian and anti-Hebbian updates add biological plausibility to SFA itself. Our future work aims at applying IncSFA to developmental robotics.

Acknowledgments

The experimental paradigm used for the distance-encoding high-dimensional video experiment was first developed by VRK under the supervision of Mathias Franzius, at the Honda Research Institute Europe. We would like to acknowledge Dr. Franzius' contributions in this regard. We thank Alexander Forster and Kail Frank for enabling the experiment with the iCub robot. We would like to thank the anonymous reviewers, whose comments helped improve the paper. Marijn Stollenga and Sohrob Kazerounian also provided useful comments. This work was funded through the 7th framework program of the EU under grants #231722 (IM-Clever project), #270247 (NeuralDynamics project), Swiss National Science Foundation grant CRSIKO-122697 (Sinergia project), and through SNF grant #138219 (Theory and Practice of Reinforcement Learning II).

References

Abut, H., editor (1990). *Vector Quantization*. IEEE Press, Piscataway, NJ.

Amari, S. (1977). Neural theory of association and concept-formation. *Biological Cybernetics*, 26(3):175–185.

Barlow, H. (2001). Redundancy reduction revisited. *Network: Computation in Neural Systems*, 12(3):241–253.

Bergstra, J. and Bengio, Y. (2009). Slow, decorrelated features for pretraining complex

cell-like networks. *Advances in Neural Information Processing Systems 22*, pages 99–107.

Chatterjee, C., Kang, Z., and Roychowdhury, V. (2000). Algorithms for accelerated convergence of adaptive pca. *IEEE Transactions on Neural Networks*, 11(2):338–355.

Chen, T., Amari, S., and Lin, Q. (1998). A unified algorithm for principal and minor components extraction. *Neural Networks*, 11(3):385–390.

Chen, T., Amari, S., and Murata, N. (2001). Sequential extraction of minor components. *Neural Processing Letters*, 13(3):195–201.

Comon, P. (1994). Independent component analysis, A new concept? *Signal Processing*, 36:287–314.

Dayan, P. and Abbott, L. (2001). Theoretical neuroscience: Computational and mathematical modeling of neural systems.

Doersch, C., Lee, T., Huang, G., and Miller, E. Temporal continuity learning for convolutional deep belief networks.

Földiák, P. (1991). Learning invariance from transformation sequences. *Neural Computation*, 3(2):194–200.

Forsythe, G. and Henrici, P. (1958). *The cyclic Jacobi method for computing the principal values of a complex matrix*. Applied Mathematics and Statistics Laboratories, Stanford University.

Franzius, M., Sprekeler, H., and Wiskott, L. (2007). Slowness and sparseness lead to place, head-direction, and spatial-view cells. *PLoS Computational Biology*, 3(8):e166.

Gisslen, L., Luciw, M., Graziano, V., and Schmidhuber, J. (2011). Sequential constant size compressors for reinforcement learning. In *Fourth Conference on Artificial General Intelligence (AGI)*.

Groetsch, C. W. (1998). Lanczos' generalized derivative. *American Mathematical Monthly*, 105(4):320–326.

Grossberg, S. (1980). How does a brain build a cognitive code?. *Psychological Review*, 87(1):1.

Hafting, T., Fyhn, M., Molden, S., Moser, M., and Moser, E. (2005). Microstructure of a spatial map in the entorhinal cortex. *Nature*, 7052:801.

Hinton, G. (1989). Connectionist learning procedures. *Artificial Intelligence*, 40(1-3):185–234.

Hinton, G. E. (2002). Training products of experts by minimizing contrastive divergence. *Neural Comp.*, 14(8):1771–1800.

Hyvärinen, A., Karhunen, J., and Oja, E. (2001). *Independent component analysis*, volume 26. Wiley-interscience.

Jenkins, O. and Matarić, M. (2004). A spatio-temporal extension to isomap nonlinear dimension reduction. In *Proceedings of the twenty-first international conference on Machine learning*, page 56. ACM.

Jolliffe, I. T. (1986). *Principal Component Analysis*. Springer-Verlag, New York.

Klapper-Rybicka, M., Schraudolph, N. N., and Schmidhuber, J. (2001). Unsupervised learning in LSTM recurrent neural networks. In *Lecture Notes on Comp. Sci. 2130, Proc. Intl. Conf. on Artificial Neural Networks (ICANN-2001)*, pages 684–691. Springer: Berlin, Heidelberg.

Kohonen, T. (2001). *Self-Organizing Maps*. Springer-Verlag, Berlin, 3rd edition.

Kompella, V., Luciw, M., and Schmidhuber, J. (2011a). Incremental slow feature analysis. In *International Joint Conference of Artificial Intelligence*.

Kompella, V. R., Pape, L., Masci, J., Frank, M., and Schmidhuber, J. (2011b). Autoincsfa and vision-based developmental learning for humanoid robots. In *IEEE-RAS International Conference on Humanoid Robots*, Bled, Slovenia.

Krasulina, T. (1970). Method of stochastic approximation in the determination of the largest eigenvalue of the mathematical expectation of random matrices. *Automat. Remote Contr*, 2:215–221.

Kreyszig, E. (1988). *Advanced engineering mathematics*. Wiley, New York.

Lee, D. and Seung, H. (1999). Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791.

Lee, H., Largman, Y., Pham, P., and Ng, A. (2010). Unsupervised feature learning for audio classification using convolutional deep belief networks. *Advances in neural information processing systems*, 22:1096–1104.

Legenstein, R., Wilbert, N., and Wiskott, L. (2010). Reinforcement learning on slow features of high-dimensional input streams. *PLoS Computational Biology*, 6(8).

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

Mahadevan, S. and Maggioni, M. (2007). Proto-value functions: A laplacian framework for learning representation and control in markov decision processes. *Journal of Machine Learning Research*, 8(2169-2231):16.

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

Oja, E. (1982). Simplified neuron model as a principal component analyzer. *Journal of mathematical biology*, 15(3):267–273.

Oja, E. (1985). On stochastic approximation of the eigenvectors and eigenvalues of the expectation of a random matrix. *Journal of Mathematical Analysis and Applications*, 106:69–84.

Oja, E. (1989). Neural networks, principal components, and subspaces. *International journal of neural systems*, 1(1):61–68.

Oja, E. (1992). Principal components, minor components, and linear neural networks. *Neural Networks*, 5(6):927–935.

O'Keefe, J. and Dostrovsky, J. (1971). The hippocampus as a spatial map: Preliminary evidence from unit activity in the freely-moving rat. *Brain research*.

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

Papoulis, A., Pillai, S., and Unnikrishna, S. (1965). *Probability, random variables, and stochastic processes*, volume 196. McGraw-hill New York.

Park, M. and Choi, J. (2008). Novel incremental principal component analysis with improved performance. *Structural, Syntactic, and Statistical Pattern Recognition*, pages 592–601.

Peng, D. and Yi, Z. (2006). A new algorithm for sequential minor component analysis. *International Journal of Computational Intelligence Research*, 2(2):207–215.

Peng, D., Yi, Z., and Luo, W. (2007). Convergence analysis of a simple minor component analysis algorithm. *Neural Networks*, 20(7):842–850.

Rolls, E. (1999). Spatial view cells and the representation of place in the primate hippocampus. *Hippocampus*, 9(4):467–480.

Sanger, T. (1989). Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural networks*, 2(6):459–473.

Schmidhuber, J. (1992a). Learning complex, extended sequences using the principle of history compression. *Neural Computation*, 4(2):234–242.

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

Schmidhuber, J. (1992c). Learning unambiguous reduced sequence descriptions. In Moody, J. E., Hanson, S. J., and Lippman, R. P., editors, *Advances in Neural Information Processing Systems 4 (NIPS 4)*, pages 291–298. Morgan Kaufmann.

Schmidhuber, J. (1999). Neural predictors for detecting and removing redundant information. In Cruse, H., Dean, J., and Ritter, H., editors, *Adaptive Behavior and Learning*. Kluwer.

Sprekeler, H. (2011). On the relation of slow feature analysis and laplacian eigenmaps. *Neural computation*, pages 1–16.

Sprekeler, H., Michaelis, C., and Wiskott, L. (2007). Slowness: An objective for spike-timing–dependent plasticity? *PLoS Computational Biology*, 3(6):e112.

Sprekeler, H., Zito, T., and Wiskott, L. (2010). An extension of slow feature analysis for nonlinear blind source separation.

T. Zito, N. Wilbert, L. W. and Berkes, P. (2008). Modular toolkit for data processing (mdp): a python data processing framework. *Frontiers in Neuroinformatics*, 2.

Taube, J., Muller, R., and Ranck, J. (1990). Head-direction cells recorded from the postsubiculum in freely moving rats. i. description and quantitative analysis. *The Journal of Neuroscience*, 10(2):420.

Tikhanoff, V., Cangelosi, A., Fitzpatrick, P., Metta, G., Natale, L., and Nori, F. (2008). An open-source simulator for cognitive robotics research: The prototype of the icub humanoid robot simulator.

Wallis, G. and Rolls, E. (1997). Invariant face and object recognition in the visual system. *Progress in Neurobiology*, 51(2):167–194.

Wang, L. and Karhunen, J. (1996). A unified neural bigradient algorithm for robust pca and mca. *International journal of neural systems*, 7(1):53.

Weng, J. and Zhang, N. (2006). Optimal in-place learning and the lobe component analysis. In *Neural Networks, 2006. IJCNN'06. International Joint Conference on*, pages 3887–3894. IEEE.

Weng, J., Zhang, Y., and Hwang, W. (2003). Candid covariance-free incremental principal component analysis. 25(8):1034–1040.

Wiskott, L. (2003). Estimating driving forces of nonstationary time series with slow feature analysis. *Arxiv preprint cond-mat/0312317*.

Wiskott, L., Berkes, P., Franzius, M., Sprekeler, H., and Wilbert, N. (2011). Slow feature analysis. *Scholarpedia*, 6(4):5282.

Wiskott, L. and Sejnowski, T. (2002). Slow feature analysis: Unsupervised learning of invariances. *Neural Computation*, 14(4):715–770.

Xu, L., Oja, E., and Suen, C. (1992). Modified hebbian learning for curve and surface fitting. *Neural Networks*, 5(3):441–457.

Zhang, Y. and Weng, J. (2001). Convergence analysis of complementary candid incremental principal component analysis. *Michigan State University*.