Incremental Slow Feature Analysis

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

Abstract

The Slow Feature Analysis (SFA) unsupervised learning framework extracts features representing the underlying causes of the changes within a temporally coherent high-dimensional raw sensory input signal. We develop the first online version of SFA, via a combination of incremental Principal Components Analysis and Minor Components Analysis. Unlike standard batch-based SFA, online SFA adapts along with non-stationary environments, which makes it a generally useful unsupervised preprocessor for autonomous learning agents. We compare online SFA to batch SFA in several experiments and show that it indeed learns without a teacher to encode the input stream by informative slow features representing meaningful abstract environmental properties. We extend online SFA to deep networks in hierarchical fashion, and use them to successfully extract abstract object position information from high-dimensional video.

1 Introduction

Without a teacher, Slow Feature Analysis (SFA) [Wiskott and Sejnowski, 2002; Franzius *et al.*, 2007; Legenstein *et al.*, 2010] extracts temporal regularities and invariant representations from rapidly changing raw sensory inputs, in a way similar to how the general technique of *Predictability Maximization* [Schmidhuber and Prelinger, 1993]. [Schmidhuber and Prelinger, 1993] extracts invariant features from different but related data points. However, current SFA techniques are not readily applicable to open-ended developmental learning agents. They estimate covariance matrices from the data via batch processing. We derive the first fully incremental version of SFA, called IncSFA, which does not need to store any input data or expensive covariance matrix estimates[1].

SFA uses principal component analysis (PCA) [Jolliffe, 1986] twice. In the first stage, PCA whitens the signal to decorrelate it with unit variance along each PC direction. In the second stage, PCA on the derivative of the whitened signal yields slow features. Our IncSFA replaces batch PCA by incremental alternates. In the pre-whitening stage we use the state-of-the-art incremental PCA method, Candid Covariance-Free Incremental Principal Component Analysis (CCIPCA) [Weng *et al.*, 2003]. CCIPCA is not feasible for the second stage, where the slow features correspond to the *least* significant components. Minor Components Analysis (MCA) [Oja, 1992; Chen *et al.*, 2001; Peng *et al.*, 2007] incrementally extracts the principal component with the smallest eigenvalue (the slowest feature). We use MCA with sequential addition [Chen *et al.*, 2001] to extract multiple minor components in parallel.

The rest of this paper is organized as follows. Section 2 reviews SFA. Sections 3 and 4 discuss CCIPCA and MCA, respectively. Sec. 5 presents our IncSFA. Sec. 6 contains experiments and results; Sec. 7 concludes.

2 Slow Feature Analysis

SFA is an unsupervised learning technique guided by the *slowness* principle. In many settings, the best functions mapping the input stream to the most *slowly changing* output signals are representative of some fundamental agent-world property, abstracting away irrelevant details picked up by the sensors [Schmidhuber and Prelinger, 1993]. Consider a mobile agent with high-dimensional video input exploring an otherwise static room. The input is caused by the agent's position and orientation, and the emerging slow features compactly encode this information [Franzius *et al.*, 2007].

2.1 Learning Problem

Formally, SFA is concerned with the following optimization problem:

Given an I-dimensional 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* $j \in \{1, ..., J\}$

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

[1] Bergstra & Bengio [Bergstra and Bengio, 2009] discuss a block incremental learning method that uses the slowness principle of kording's work [Kording *et al.*, 2004] via estimating the covariance matrix in mini-batches. Berkes & Wiskott [Berkes and Wiskott, 2005] address the advantages of the SFA framework as compared to the aforementioned approach.

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.

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

2.2 Solution

This learning problem is not straightforward, as it is an optimization of variational calculus [2002]. But it can be greatly simplified through an eigenvector approach. Consider that the g_j are constrained to be linear combinations of a finite set of nonlinear functions $\mathbf{h}$. Now, $y_j(t) = g_j(\mathbf{x}(t)) = \mathbf{w}_j^T\ \mathbf{h}(\mathbf{x}(t))$. Let $\mathbf{z}(t) = \mathbf{h}(\mathbf{x}(t))$. The SFA problem now becomes finding the weight vectors $\mathbf{w}_j$ to minimize

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

subject to the constraints. 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 different weight vectors are orthonormal. To select such an $\mathbf{h}$, a well-known process called whitening (or sphering), which requires the principal components of the input data, will map $\mathbf{x}$ to a $\mathbf{z}$ with zero mean and identity covariance matrix.

Given such a $\mathbf{z}$, the SFA problem becomes linear. In order to solve for the slow features, note Eq. 5 is minimized with the set of J normed eigenvectors of $\langle \dot{\mathbf{z}}\dot{\mathbf{z}}^T \rangle$ with the J smallest eigenvalues. So another set of principal components (of $\dot{\mathbf{z}}$) will provide the desired features.

3 Candid Covariance-Free Incremental Principal Component Analysis

For online whitening of input $\mathbf{x}$, we use Candid Covariance-Free Incremental (CCI) PCA (extending Oja's work [Oja, 1982]). CCIPCA provides the eigenvectors *and* eigenvalues necessary for whitening, and does not keep an estimate of the covariance matrix, which can be computationally prohibitive for high-dimensional input. CCIPCA is guaranteed to converge to the true components [Zhang and Weng, 2001].

A PC is a normed eigenvector $\mathbf{v}_i^*$ of the data covariance matrix, with eigenvalue λ_i^*, the variance of the samples along $\mathbf{v}_i^*$. By definition, an eigenvector and eigenvalue satisfy

$$\mathrm{E}[\mathbf{u}\mathbf{u}^T]\mathbf{v}_i^* = \lambda_i^*\mathbf{v}_i^*, \tag{6}$$

where $\mathbf{u}$ is a zero-mean $\mathbf{x}$. The set of eigenvectors are orthonormal, and are ordered such that $\lambda_1^* \geq \lambda_2^* \geq ... \geq \lambda_K^*$.

The whitening matrix is generated by multiplying the matrix of principal components $\hat{\mathbf{V}} = [\mathbf{v}_1^*, ..\mathbf{v}_K^*]$, by the diagonal matrix $\hat{\mathbf{D}}$, where component $\hat{d}_{i,i} = \frac{1}{\sqrt{\lambda_i^*}}$. After whitening via $\mathbf{z}(t) = \hat{\mathbf{V}}\hat{\mathbf{D}}\mathbf{u}(t)$, then $\mathrm{E}[\mathbf{z}\mathbf{z}^T] = I$. In IncSFA, we use online estimates of $\hat{\mathbf{V}}$ and $\hat{\mathbf{D}}$.

3.1 CCIPCA Updating

For observations $\mathbf{u}_i$, the first PC is the expectation of the normalized response-weighted inputs. Eq 6 can be rewritten as

$$\lambda_i^*\ \mathbf{v}_i^* = \mathrm{E}\left[(\mathbf{u}_i \cdot \mathbf{v}_i^*)\ \mathbf{u}_i\right], \tag{7}$$

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

$$\mathbf{v}_i(t) = (1-\eta)\ \mathbf{v}_i(t-1) + \eta\ \left[\frac{\mathbf{u}_i(t) \cdot \mathbf{v}_i(t-1)}{\|\mathbf{v}_i(t-1)\|}\ \mathbf{u}_i(t)\right]. \tag{8}$$

where η is the learning rate. In other words, both the eigenvector and eigenvalue of the first PC can be found as the sample mean of Eq. 7. Then, the estimate of the eigenvalue is given by $\lambda_i(t) = \|\mathbf{v}_i(t)\|$.

3.2 Lower-Order Principal Components

Any component besides the first not only must satisfy Eq. 6 but must also be constrained to be orthogonal to the other components. CCIPCA uses the residual method, which generates observations in a complementary space for the lower-order eigenvectors. 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 $\mathbf{u}_{i+1}$:

$$\mathbf{u}_{i+1}(t) = \mathbf{u}_i(t) - \left(\mathbf{u}_i^T(t)\frac{\mathbf{v}_i(t)}{\|\mathbf{v}_i(t)\|}\right)\frac{\mathbf{v}_i(t)}{\|\mathbf{v}_i(t)\|}. \tag{9}$$

4 Minor Components Analysis

Incremental PCA is in general unsuitable for the second stage of batch PCA, since the slow features will correspond to the eigenvectors with the smallest eigenvalues. Instead, we use Minor Components Analysis MCA [Oja, 1992; Chen *et al.*, 2001; Peng *et al.*, 2007]. Minor components are also PCs, but the first minor component is the PC direction in which the data has *smallest* variance. Chen *et al.* [2001] pointed out the duality between minor components and PCs. For some positive definite matrix $\mathbf{C}$, the eigenvectors of $\gamma I - \mathbf{C}$, where $\gamma > \lambda_1$, will be of the opposite order. So, switching the sign of incremental PCA (i.e., changing Hebbian to anti-Hebbian updating) is not enough; MCA needs an additional penalty term.

4.1 MCA Updating

To extract the first minor component in each space, we use Peng *et al.*'s low-complexity algorithm [2007] allowing for

a learning rate that does not vanish as time increases (important for open-ended learning). The update rule combines anti-Hebbian learning with a penalty term designed such that convergence can be proven [Peng *et al.*, 2007], provided the constant learning rate is small enough, and the initial estimate is not orthogonal to the true component (or the zero vector):

$$\begin{aligned}\mathbf{w}_i(t) &= 1.5\mathbf{w}_i(t-1) - \eta\,\mathbf{C}_i\,\mathbf{w}_i(t-1) \\ &- \eta\,[\mathbf{w}_i^T(t-1)\mathbf{w}_i(t-1)]\,\mathbf{w}_i(t-1),\end{aligned} \tag{10}$$

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

$$\eta\lambda_1 < 0.5, \quad ||\mathbf{w}(0)||^2 \leq \frac{1}{2\eta} \quad \text{and} \quad \mathbf{w}^T(0)\mathbf{w}^* \neq 0 \tag{11}$$

where $\mathbf{w}(0)$ is the initial feature estimate and $\mathbf{w}^*$ the true eigenvector associated with the smallest eigenvalue.

4.2 Lower-Order Slow Features

For extracting more than one slow feature, the residual method used by incremental PCA will not work due to the aforementioned duality. Instead, we use Chen *et al.*'s sequential addition, which 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 computes the matrix $\mathbf{C}_i,\ \forall i > 1$ as follows:

$$\mathbf{C}_i(t) = \mathbf{C}_{i-1}(t) + \gamma(t) \sum_{j=1}^{i-1} \mathbf{w}_j(t)\mathbf{w}_j^T(t)\mathbf{C}_1(t) \tag{12}$$

where γ 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 for only the first PC. Then, let $\gamma = \lambda_1(t) + \epsilon$, where ϵ is a small positive real number.

5 Incremental Slow Feature Analysis

For each time step $t = 0, 1, \ldots$:

5.1 Algorithm

1. **Sense:** Grab the current raw input as vector $\breve{\mathbf{x}}(t)$.
2. **Non-Linear Expansion:** To deal with non-linearity, optionally generate an expanded signal $\mathbf{x}(t)$ with I components, e.g. for a quadratic expansion:

$$\mathbf{x}(t) = [\breve{x}_1(t), ..., \breve{x}_d(t), \breve{x}_1^2(t), \breve{x}_1(t)\breve{x}_2(t), ..., \breve{x}_d^2(t)] \tag{13}$$

3. **Mean Estimation and Subtraction:** If $t = 0$, set $\bar{\mathbf{x}}(t) = \mathbf{x}(0)$. Otherwise, update mean vector estimate $\bar{\mathbf{x}}(t)$:

$$\bar{\mathbf{x}}(t) = (1-\eta)\,\bar{\mathbf{x}}(t-1) + \eta\,\mathbf{x}(t). \tag{14}$$

and subtract the mean:

$$\mathbf{u}(t) \leftarrow \mathbf{x}(t) - \bar{\mathbf{x}}(t). \tag{15}$$

4. **CCIPCA:** Update estimates of the most significant K principal components of $\mathbf{u}$, where $K \leq I$:
   (a) If $t < K$, initialize $\mathbf{v}_t(t) = \mathbf{u}(t)$.
   (b) Otherwise do the following two steps from $j = 1, 2, ..., K$. Let $\mathbf{u}_1(t) = \mathbf{u}(t)$. Then, execute CCIPCA equations 8 and 9.
5. **Whitening and Dimensionality Reduction:** Let $\mathbf{V}(t)$ contain the normed estimates of the K principal components, and create diagonal matrix $\mathbf{D}(t)$, where $D_{i,i} = 1/\sqrt{\lambda_i(t)}, \forall i \leq K$. Then, $\mathbf{z}(t) = \mathbf{V}(t)\mathbf{D}(t)\mathbf{u}(t)$.
6. **Derivative Signal:** As a forward difference approximation of the derivative, let $\dot{\mathbf{z}}(t) = \mathbf{z}(t) - \mathbf{z}(t-1)$.
7. **Extract First Principal Component:** Use CCIPCA to update the first PC of $\dot{\mathbf{z}}$, to set $\gamma(t)$.
8. **Slow Features:** Update estimates of the least significant J PCs of $\dot{\mathbf{z}}$, where $J \leq K$:
   (a) If $t < J$, initialize $\mathbf{w}_t = \dot{\mathbf{z}}(t)$.
   (b) Otherwise, let $\mathbf{C}_1(t) = \dot{\mathbf{z}}(t)\dot{\mathbf{z}}^T(t)$, and for each $i = 1, ..., J$, execute incremental MCA updates in equations 10 and 12.
9. **Output:** Let $\mathbf{W}(t)$ contain the current estimates of the slow features. Then, $\mathbf{y}(t) = \mathbf{z}^T(t)\mathbf{W}(t)$ is the SFA output.

5.2 Practical Considerations

Dimensionality Reduction is possible after both the first and the second phase. In the latter we can select $J << K$, the number of slow features to compute. In the first, we discard all but the top K components. A method we found to be successful is to set K such that no more than a certain percentage of the previously estimated total data variance (the denominator below) is lost. Let β be the ratio of total variance to keep (e.g., 0.95), and compute the smallest K such that $\frac{\sum_k^K \lambda_k(t)}{\sum_i^I \lambda_k(t-1)} > \beta$.

Learning Rates and Convergence: With a CCIPCA learning rate set to $1/t$, IncSFA will converge to the same features (any number) as batch-based SFA, provided the MCA conditions are satisfied (see 11). In practice, we enable plasticity by letting CCIPCA's learning rate converge to the MCA learning rate, a small constant.

How to set the latter? We are guided by the *slowness measure ($s(x_t)$)* (Wiskott & Sejnowski [2002]) of the input signal, which measures the temporal variation frequency and is related to the greatest eigenvalue of $\dot{z}(t)$ (refer to Section 5). In theory, the upper bound of the MCA learning rate (η_U) varies according to $\eta_U \propto s(x_t)^{-2}$. The closer the rate to this bound, the faster learning. In our experiments, we manually set rates right below this bound.

For open-ended learning, convergence is not desired. But an always nonzero learning rate makes the algorithm less stable (the well-known stability-plasticity dilemma), due to learning rate-caused overshoots/errors. In our experiments this was not a problem though.

6 Experiments and Results

6.1 Simple Signals: Proof of Concept

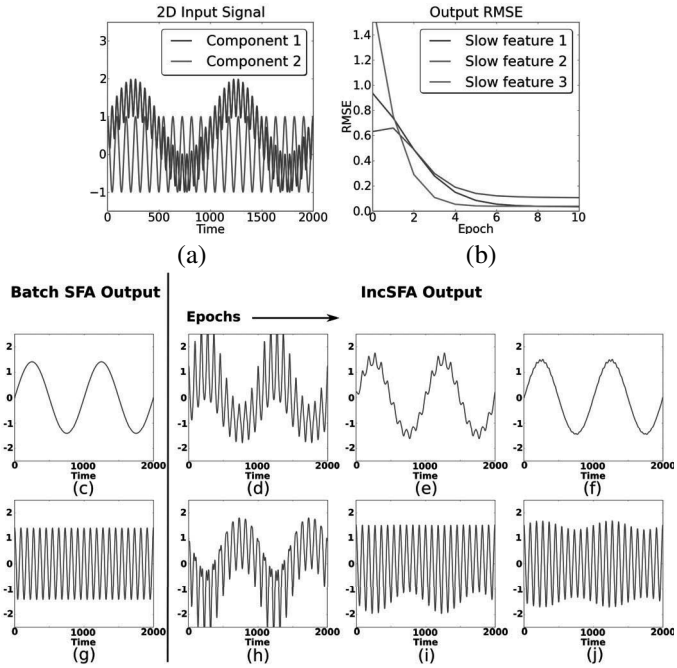

Figure 1: Experiment conducted on a simple non-linear input signal. A learning rate of $\eta = 0.08$ was used. (a) Input Signal (b) Output RMSE plot (c) Batch SFA output of the first slow feature (d)-(f) IncSFA output at t = 2, 5, 10 epochs. (g) Batch SFA output of the second slow feature (h)-(j) IncSFA output at t = 2, 5, 10 epochs.

As a basic proof of concept of IncSFA, we tried a problem introduced in the original SFA paper [Wiskott and Sejnowski, 2002]. The input signal is $\breve{x}_1(t) = \sin(t) + \cos(11\ t)^2$, $\breve{x}_2(t) = \cos(11\ t)$, $t \in [0, 2\pi]$ — both vary quickly over time (see Figure 1(a)). A total of $2,000$ discrete datapoints are used for learning. The slowest feature hidden in the signal is $y_1(t) = \breve{x}_1(t) - \breve{x}_2(t)^2 = \sin(t)$, and the second is $\breve{x}_2(t)^2$ — these are the features extracted by both batch SFA and IncSFA. Figure 1(b) shows the Root Mean Square Error (**RMSE**) of output signals compared to the true output, over multiple epochs of training. Figure 1(c) and (g) shows the feature outputs from batch SFA, and (to the right) the IncSFA outputs at $2,\ 5$, and 10 epochs.

6.2 Logistic Map: Complex Signals and High Dimensionality

To test IncSFA's ability to uncover structure hidden in complex signals and high-dimensional data, we use parts of a chaotic time series derived from a logistic map [T. Zito and Berkes, 2008]:

$$\breve{x}(t+1) = (3.6 + 0.13\,\gamma(t))\breve{x}(t)\,(1 - x(t)), \qquad (16)$$

which has a slowly varying driving force $\gamma(t)$ (to be extracted by IncSFA) hidden in a signal of complicated dynamics. The

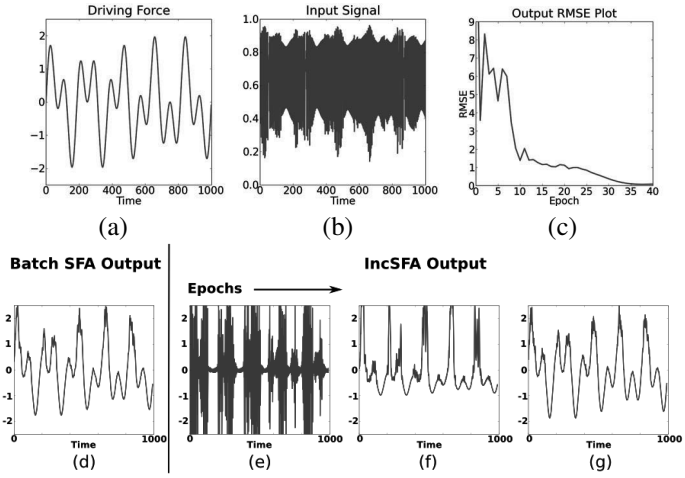

Figure 2: Experiment conducted on a chaotic time series derived from a logistic map. A learning rate of $\eta = 0.004$ was used. (a) Driving Force (b) Input (c) Output RMSE plot (d) Batch SFA output of the slowest feature (e)-(g) IncSFA output at t = 15, 30, 60 epochs.

$\gamma(t)$ used is made up of two frequency components (5 and 11 Hz), given by

$$\gamma(t) = \sin(10\pi t) + \sin(22\pi t). \qquad (17)$$

Figures 2(a) and 2(b) plot the driving force signal $\gamma(t)$ and the generated time series $\breve{x}(t)$, respectively. A total of $1,000$ discrete datapoints are used. To reconstruct the time series, we embed it in a 10 dimensional space using a sliding temporal window of size 10 (we use the *TimeFramesNode* from the MDP toolkit [T. Zito and Berkes, 2008] for this). The signal is then expanded quadratically to generate an input signal with 65 dimensions. Figure 2(c) shows the convergence of the incremental SFA on the batch SFA output. Figure 2(d) shows the batch SFA output, and Figures 2(e)-(g) show the outputs of the incremental SFA at 15, 30 and 60 epochs.

6.3 2D Exploration: Invariant Spatial Coding

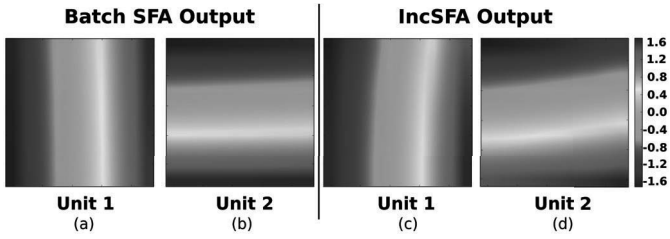

Figure 3: (a) Batch SFA output of the first slow feature and (b) the second slow feature (c) IncSFA output of the first slow feature and (d) the second slow feature after 50,000 samples with a learning rate $\eta = 0.003$. Figures best viewed in color.

Epochs over the same sequence are not needed; IncSFA also works if data is generated from some *movement paradigm* which may not lead to exact sequence repetitions.

We use the paradigm of Franzius *et al.* [Franzius *et al.*, 2007], who fed SFA into ICA to derive computational *place cells* active at specific agent locations, like those in the rat hippocampus. Such invariant representations of the environment are useful for navigational tasks of organisms or robots.

Our simulated agent performs a random walk in a two-dimensional bounded space. Brownian motion is used to generate agent trajectories approximately like those of rats. The agent's position $p(t)$ is updated by a weighted sum of the current velocity and gaussian white noise, with standard deviation v_r. The momentum term m can assume values between zero (massless) and one (infinite mass), so that higher values of m lead to smoother trajectories and more homogeneous sampling of space in less time. Once the agent crosses the spatial boundaries, the current velocity is halved and an alternative random velocity update is generated, until a new valid position is reached. Noise variance $v_r = [3.0, 2.5]^T$, mass $m = 0.75$ and $50,000$ data points are used for generating the training set. We use a separate regular grid test dataset for evaluation. Here is the movement paradigm used:

$currVel \leftarrow p(t) - p(t-1)$;
repeat
  $noise \leftarrow GaussianWhiteNoise2d() * v_r$;
  $p(t+1) \leftarrow p(t) + m * currVel + (1-m) * noise$;
  if $not\ isInsideWalkArea(p(t+1))$:
    $currVel \leftarrow currVel/2$;
until $isInsideWalkArea(p(t+1))$

The movement paradigm [Franzius *et al.*, 2007] yields slow feature outputs in the form of half-sinusoids, shown in Figure 3. These features together code for the agent's x and y position in the environment. The first slow feature (Figure 3(a)) is invariant to the agent's x position, the second (Figure 3(b)) to its y position (y axis horizontal). IncSFA's results (Figures 3(c)-(d)) are close to the ones of the batch version, with an **RMSE** of $[0.0536, 0.0914]^T$.

6.4 High-Dimensional Video: Hierarchical IncSFA

To test the scalability of IncSFA, we feed it with a high-dimensional video stream based on $20,000$ images generated by the iCub simulator [V. Tikhanoff and Nori, 2008], an OpenGL-based software specifically built for the iCub robot. Our experiment mimics the robot observing a moving interactor agent, modeled as 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 similar to the one discussed in Section 6.3. Figure 4(a) shows the experimental setup in the iCub simulator. Figure 4(b) shows a sample image from the dataset. Monocular images were captured from the robot's left eye and downsampled to 83×100 pixels (an input dimension of $8,300$).

Hierarchical Network: We wish the robot to compute features that code for distance. This is a challenging problem, due to the highly nonlinear function mapping pixel intensities to distance. We use a hierarchical model motivated by the human visual system, similar to the one of Franzius *et al.* [2007].

Figure 5 shows the hierarchical architecture, made up of several layers of multiple IncSFA units, acting over overlapping rectangular $receptive fields$ of smaller dimensions. On the lowest layer, the receptive field of each module consists of an image patch of 10×10 pixels. The output from the first layer forms a 15×19 grid with 10 slow features per each module. With overlapping receptive fields of 5×5, this layer's output, with 5 slow features per module, becomes a $4 \times 5 \times 5$ grid. Similarly, third and fourth layers decrease signal dimension to produce one single output. These layers are trained sequentially from bottom to top over the entire dataset.

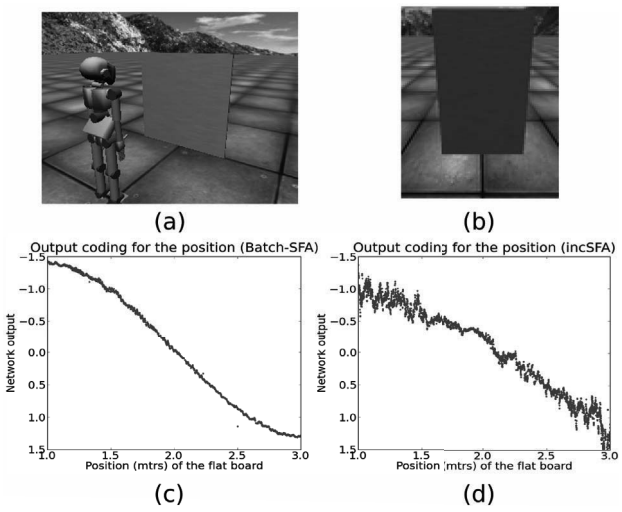

Figure 4: (a) Experimental Setup: iCub Simulator (b) Sample image from the input dataset (c) Batch-SFA output (d) IncSFA output ($\eta = 0.005$)

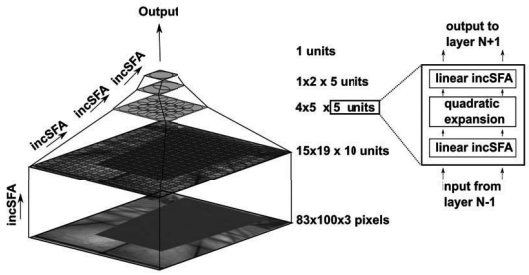

Figure 5: Hierarchical-Network Architecture

In general, the number of successfully extracted slow features is only limited by the performance of CCIPCA as compared to batch PCA. Batch PCA can potentially better extract the eigenvectors associated with the smallest eigenvalues as it can iterate over all the data. However, this information, when passed on to our second phase, is rarely useful (for the input data, small eigenvalue directions can typically be removed).

Figures 4 show batch SFA and IncSFA outputs. The expected output is of the form of a sinusoid (refer to Section 6.3) extending over the range of board positions. IncSFA gives a slightly noisy output, probably due to our usage of a constant dimensionality reduction value for all units in each layer of

the network, selected to maintain a consistent grid structure. Due to this, some units with eigenvectors having very small eigenvalues emerged 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. (Building an adaptive hierarchcial network architecture is one of our future goals.)

A supervised quadratic regressor was 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 (see RMSE plot). IncSFA derived the driving forces from a complex and continuous input video stream in a completely online and unsupervised manner.

7 Conclusions

IncSFA is the first online algorithm for Slow Feature Analysis, constructed via a combination of incremental PCA and MCA. We showed why IncSFA works and how it resists the curse of dimensionality. IncSFA was successfully compared to batch SFA. To deal with the high nonlinearity involved in extracting slow features from images, a hierarchical IncSFA network was built. It performed well on high-dimensional video. IncSFA represents an important step forward, as it opens up the potential of SFA for unsupervised pre-processing of large sets of high-dimensional and/or non-stationary data, such as EEG signals, sensory input streams of autonomous robots, etc.

Acknowledgments

This work was partially funded by the EU project FP7-ICT-IP-231722 (IM-CLeVeR) and SNF Sinergia Project CRSIKO-122697.

References

[Bergstra and Bengio, 2009] James Bergstra and Yoshua Bengio. Slow, decorrelated features for pretraining complex cell-like networks. pages 99–107, 2009.

[Berkes and Wiskott, 2005] Pietro Berkes and Laurenz Wiskott. Slow feature analysis yields a rich repertoire of complex cell properties. *Journal of Vision*, 5(6):579–602, jul 2005.

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

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

[Jolliffe, 1986] I. T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, New York, 1986.

[Kording *et al.*, 2004] Konrad P. Kording, Peter Konig, Konrad P, Christoph Kayser, Christoph Kayser, Wolfgang Einhauser, and Wolfgang Einhauser. How are complex cell properties adapted to the statistics of natural stimuli? *Journal of Neurophysiology*, 91:2004, 2004.

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

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

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

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

[Schmidhuber and Prelinger, 1993] J. Schmidhuber and D. Prelinger. Discovering predictable classifications. *Neural Computation*, 5(4):625–635, 1993.

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

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

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

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

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