Slow Feature Analysis (SFA)
- Applications -

— Lecture Notes —

Laurenz Wiskott
Institut für Neuroinformatik
Ruhr-Universität Bochum, Germany, EU

30 July 2020

— Summary —

Slow feature analysis (SFA) is an algorithm that can be applied to vectorial time series to extract nonlinear and slowly varying features. It does so by maximizing the slowness of the extracted features without being allowed to do averaging. The idea is, that input signals of perceptual systems often vary on a much faster time scale than the objects that are being perceived. In vision, for instance, the pixel values of a video vary on a much faster time scale than the objects moving around in the video. The SFA-algorithm is based on principal component analysis and therefore linear. However, it is usually combined with a nonlinear expansion, which makes it effectively nonlinear.

1 SFA algorithm (→ slides) works in four steps: (1) The input signal is nonlinearly expanded permitting extraction of nonlinear features. (2) The expanded signal is whitened to fulfill the constraints of zero mean, unit variance and decorrelation. (3) The time derivative is taken to be sensitive to the temporal structure. (4) PCA is used to find the directions of smallest variance which means slowest change. → Video 1 (including nonlinear expansion and whitening), Exercises, Solutions

Core text and formulas are set in dark red, one can repeat the lecture notes quickly by just reading these; ♦ marks important formulas or items worth remembering and learning for an exam; ◊ marks less important formulas or items that I would usually also present in a lecture; + marks sections that I would usually skip in a lecture.
More teaching material is available at https://www.ini.rub.de/PEOPLE/wiskott/Teaching/Material/.

2 Applications of SFA in machine learning (→ slides) shows two examples: (1) The extraction of a driving force, i.e. a slowly changing parameter, from a dynamical system. (2) Nonlinear blind source separation, i.e. nonlinearly mixed acoustic signals are isolated from the mixture without knowing anything about the signals nor the mixing. → Videos 2.1, 2.2

Contents

1 SFA algorithm (→ slides)

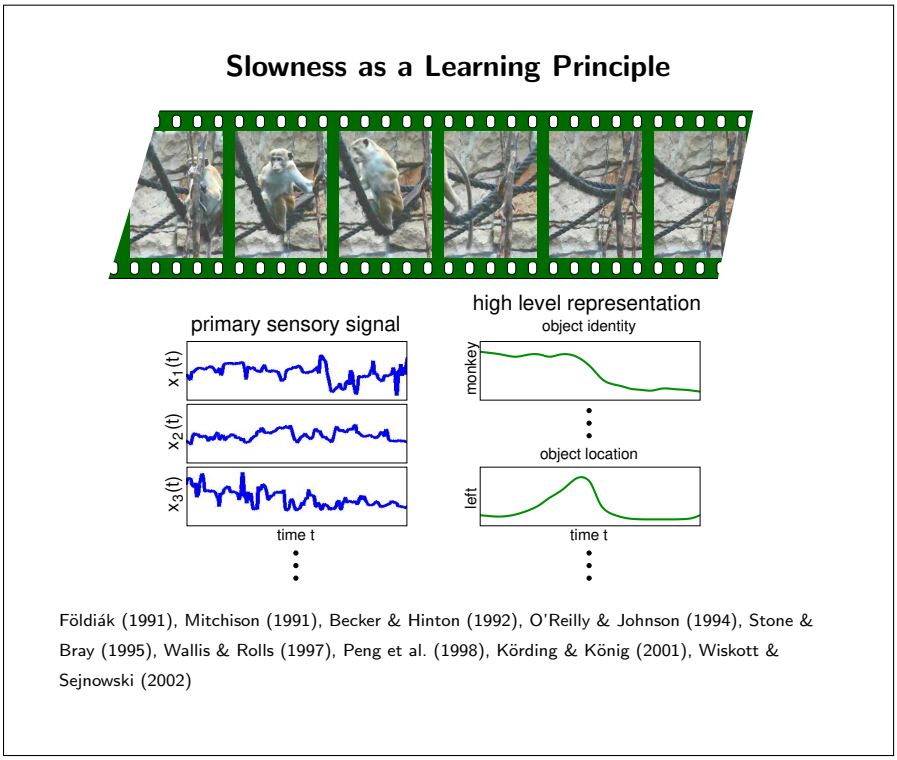
Slow feature analysis was first published in 1998 (Wiskott, 1998). This section is based on (Wiskott and Sejnowski, 2002).

Slow feature analysis is an algorithm that has been developed in context of modeling the primate visual system, but it has also been applied successfully in technical contexts. It is based on the slowness principle, which is introduced here from the view of learning invariances in visual perception.

Slow Feature Analysis

(maxmann, 2016, pixabay, © CC0, URL)

Section title: **Slow Feature Analysis**
Image: (maxmann, 2016, pixabay, © CC0, URL)[1.1]



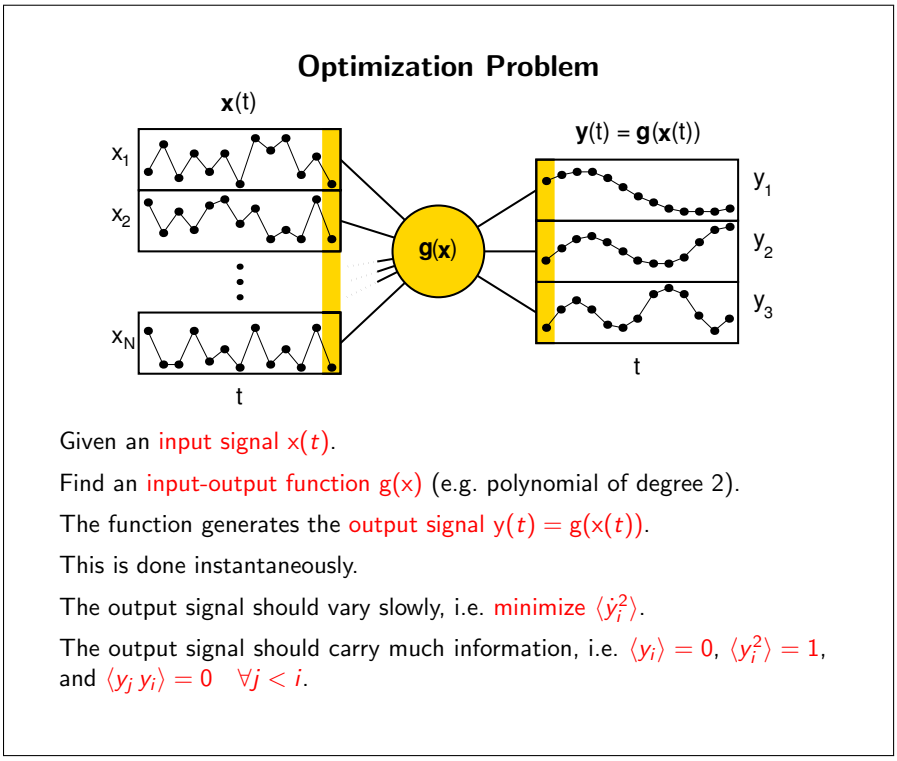
Slowness as a learning principle is based on the observation that different representations of the visual sensory input vary on different time scales. Our visual environment itself is rather stable. It varies on a time scale of seconds.

The primary sensory signal on the hand, e.g. responses of single receptors in our retina or the gray value of a single pixel of a CCD camera, vary on a faster time scale of milliseconds, simply as a consequence of the very small receptive field sizes combined with gaze changes or moving objects. As an example imagine you are looking at a quietly grazing zebra. As your eyes scan the zebra, single receptors rapidly change from black to white and back again because of the stripes of the zebra. But the scenery itself does not change much.

Finally, your internal high-level representation of the environment changes on a similar time scale as the environment itself, namely on a slow time scale. The brain is somehow able to extract the slowly varying high-level representation from the quickly varying primary sensory input. The hypothesis of the slowness learning principle is that the time scale itself provides the cue for this extraction. The idea is that if the system manages to extract slowly varying features from the quickly varying sensory input, then there is a good chance that the features are a good representation of the visual environment.

A number of people have worked along these lines. Slow feature analysis is within this tradition but differs in some significant technical aspects from all previous approaches.

Figure: (Wiskott et al., 2011, Fig. 2, © CC BY 4.0, URL)[1.2]



Slow feature analysis is based on a clearcut optimization problem. The goal is to find input-output functions that extract most slowly varying features from a quickly varying input signal.

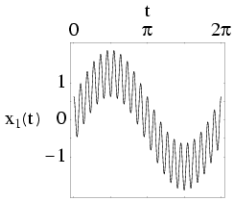
It is important that the functions are instantaneous, i.e. one time slice of the output signal is based on just one time slice of the input signal (marked in yellow). Otherwise low-pass filtering would be a valid but not particularly useful method of extracting slow output signals. Instantaneous functions also make the system fast after training, as is important in visual processing, for instance. It is also possible to take a few input time slices into account, e.g. to make the system sensitive to motion or to process scalar input signals with a fast dynamics on a short time scale. However, low-pass filtering should never be the main method by which slowness is achieved.

Without any constraints, the optimal but not very useful output signal would be constant. We thus impose the constraints of unit variance $\langle y_i^2 \rangle = 1$ and, for mathematical convenience, zero mean $\langle y_i \rangle = 0$. To make different output signal components represent different information, we impose the decorrelation constraint $\langle y_j y_i \rangle = 0$. Without this constraint, all output components would typically be the same. Notice that the constraint is asymmetric, later components have to be uncorrelated to earlier ones but not the other way around. This induces an order. The first component is the slowest possible one, the second component is the next slowest one under the constraint of being uncorrelated to the first, the third component is the next slowest one under the constraint of being uncorrelated to the first two, etc.

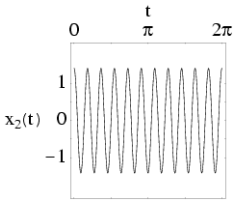
Figure: (Wiskott et al., 2011, Fig. 1, © CC BY 4.0, URL)[1.3]

Simple Example

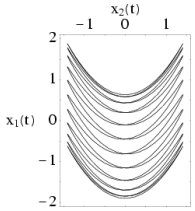
$$x_1(t) = \sin(t) + \cos(11t)^2$$
$$x_2(t) = \cos(11t)$$



component $x_1(t)$



component $x_2(t)$



Input signal $x(t)$

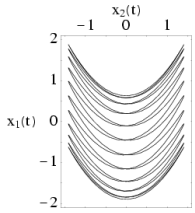
A slow feature of this signal is obviously

$$y(t) = x_1(t) - x_2(t)^2 = \sin(t)\,.$$

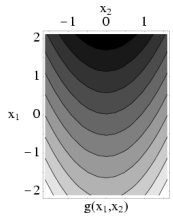
Consider a simple two-dimensional trajectory as an example. $x_1(t)$ and $x_2(t)$ are both quickly varying. Nonlinearly hidden in this signal is $\sin(t)$, which is relatively slow. It can be extracted with a polynomial of degree two, since $x_1(t) - x_2(t)^2 = \sin(t)$. In the trajectory plot (right) one can see the fast back and forth rocking on a parabula, which itself is slowly moving up and down.
Figure: (Wiskott and Sejnowski, 2002, Fig. 2, URL)[1.4]

Simple Example

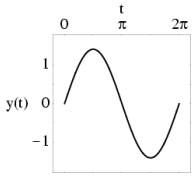
$$\begin{aligned} \text{Minimize} \quad & \langle \dot{y}^2 \rangle \\ \text{with} \quad & y(t) = g(x(t)) \\ \text{under the constraints} \quad & \langle y \rangle = 0 \\ \text{and} \quad & \langle y^2 \rangle = 1\,. \end{aligned}$$



input signal $x(t)$

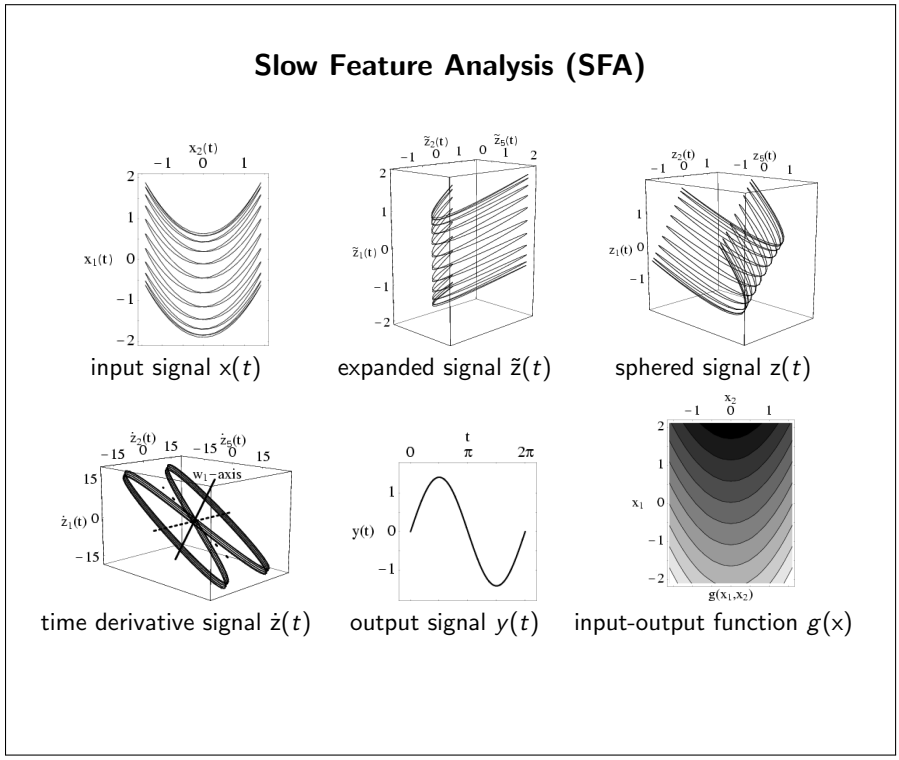


input-output function $g(x)$



output signal $y(t)$

From this input signal we want to extract only a single output component, so that the optimization problem simplifies somewhat. As seen above, the slowest output signal is $y(t) = \sin(t)$ and the function required to extract it is $g(\mathbf{x}) := x_1 - x_2^2$. Its level lines are parabulas and follow the principal curve of the input trajectory.
Figure: (Wiskott and Sejnowski, 2002, Fig. 2, URL)[1.5]



The SFA-algorithm is relatively straight forward. The upper left panel shows the trajectory $x_1(t) = \sin(t) + \cos(11t)^2$ and $x_2(t) = \cos(11t)$. It is quickly rocking back and forth on a parabola, which itself is slowly moving up and down. SFA finds this slow feature as follows:

Step 1: The input signal is nonlinearly expanded into a high-dimensional feature space $\tilde{\mathbf{z}}$ (upper middle panel). We often use polynomials of degree two, which would result in $\tilde{z}_1 := x_1$, $\tilde{z}_2 := x_2$, $\tilde{z}_3 := x_1^2$, $\tilde{z}_4 := x_1 x_2$, and $\tilde{z}_5 := x_2^2$. Within this space the problem can be solved linearly, because any polynomial of degree two can be written as a linear combination of the $\tilde{z}_i$.

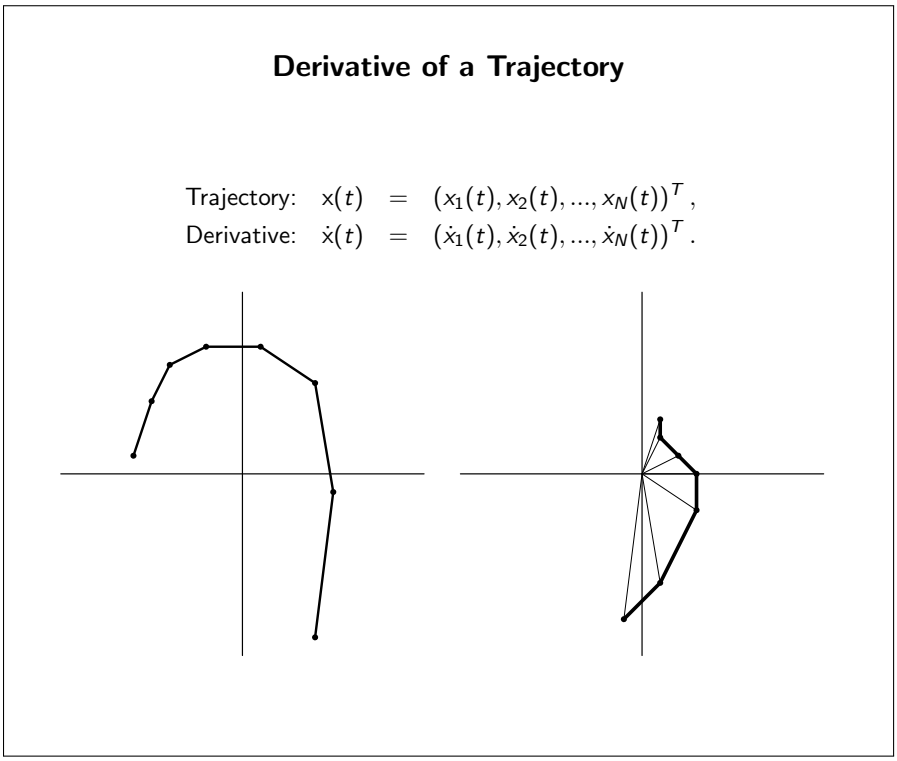
Step 2: The expanded signal is whitened or sphered (upper right panel). This operation first removes the mean and then stretches the signal along its principal axes such that it has unit variance in all directions. This has the great advantage that the constraints are easy to fulfill. If we project the whitened signal onto any unit vector, the projected signal has zero mean and unit variance; if we project the whitened signal onto any set of orthogonal unit vectors, the projected signal components are uncorrelated. Thus, we only have to find the orthogonal unit vectors that produce the slowest signal components. The constraints are then taken care of automatically. In our simple example, we only need one unit vector to project to.

Step 3: To find the direction in which the signal varies most slowly we calculate the derivative of the whitened signal (lower left panel). The variance of the derivative is small in directions of slow variation and large in directions of fast variation.

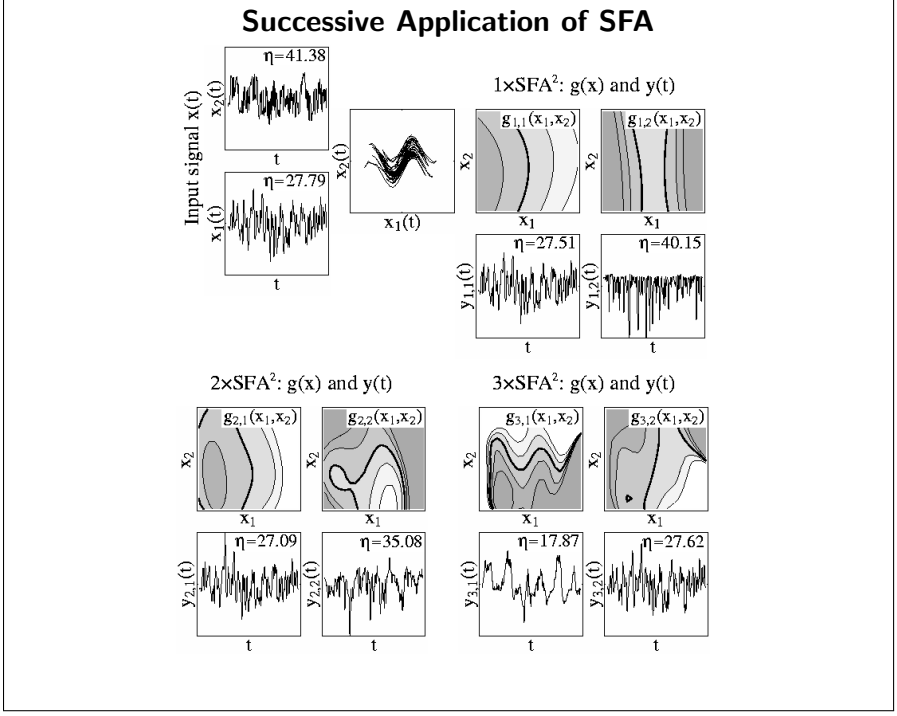
Step 4: We apply principal component analysis. The component with smallest eigenvalue gives us the unit vector that yields the slowest possible output signal component. If we want to extract more slow features, we simply take the principal components with next larger eigenvalues. Thus, once we are here, it is easy to extract many slow output signal components.

If one concatenates the nonlinear expansion, the whitening, and the projection onto the unit vectors, one gets the nonlinear functions $g_i(\mathbf{x})$. The lower right panel shows the function found here for the simple example. Evaluating this function along the input trajectory yields the output signal $y(t)$ (lower middle panel).

Figure: (Wiskott and Sejnowski, 2002, Fig. 2, URL)[1.6]



Formally the derivative of a trajectory is simply the vector of the derivatives of the components of the trajectory. If the trajectory is discretized in time with step size 1, its derivative is simply the sequence of diffence vectors of two successive time points.

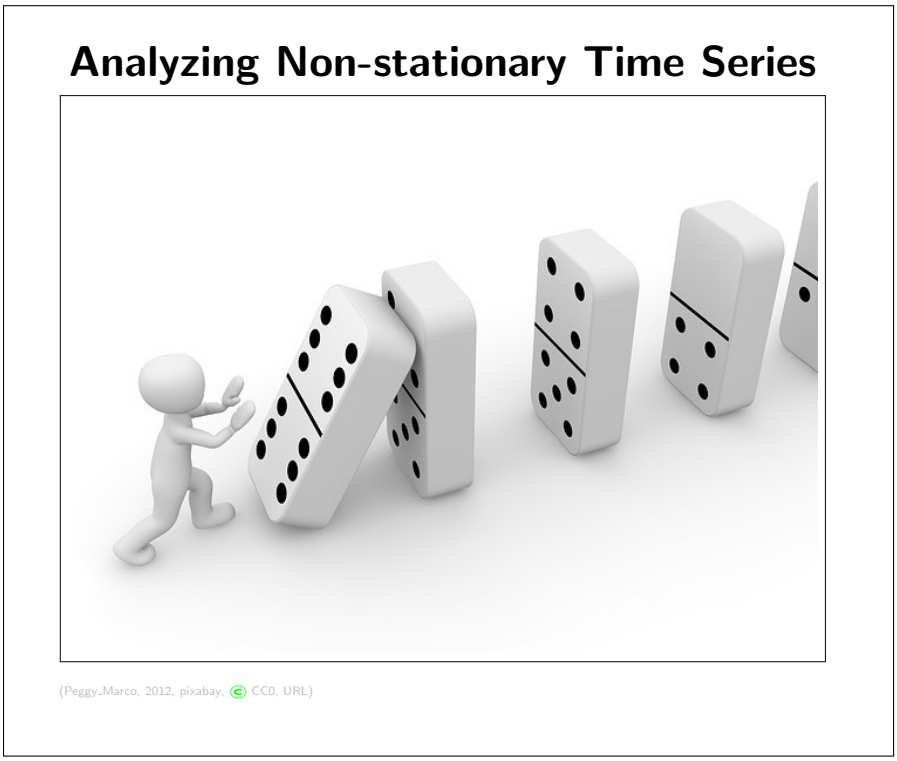


An interesting property of SFA is that it can be applied in a cascade. The input signal of the example shown here has a slow feature in it that cannot be extracted with a polynomial of degree two. Thus SFA^2, i.e. quadratic SFA, alone cannot solve the problem. But if one applies SFA^2 again to the first three output components of the first SFA^2 and then a third time, the slow feature gets extracted, compare the contour plot of $g_{3,1}(x_1, x_2)$ with the trajectory plot of $x_2(t)$ versus $x_1(t)$.
Figure: (Wiskott and Sejnowski, 2002, Fig. 7, URL)[1.7]

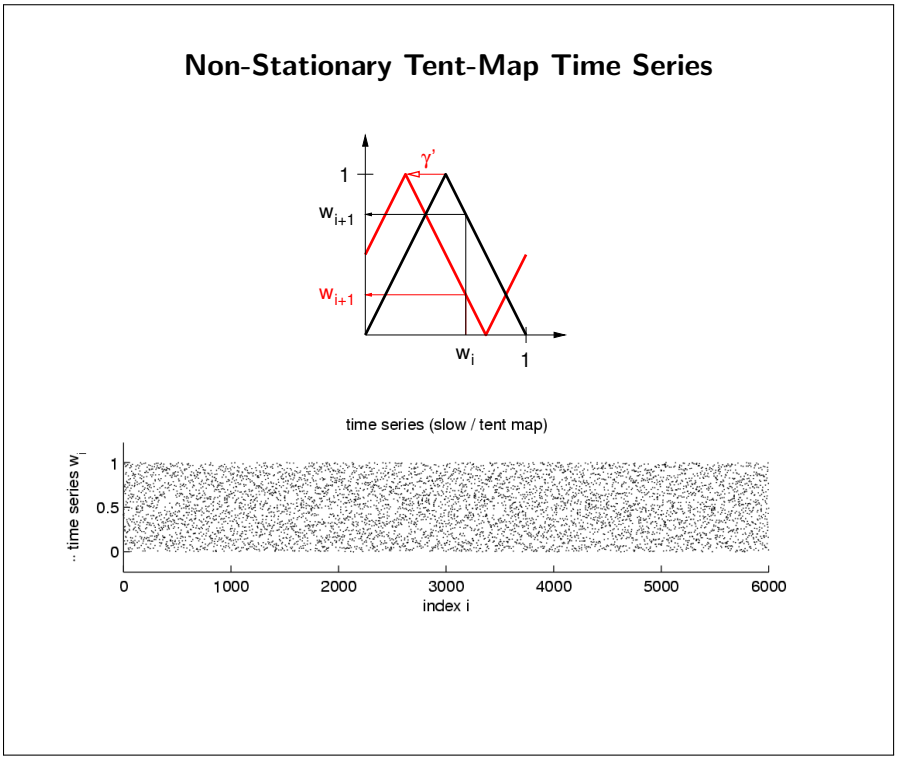
2 Applications of SFA in machine learning (→ slides)

2.1 Extracting driving forces (→ slides)

This section is based on (Wiskott, 2003).

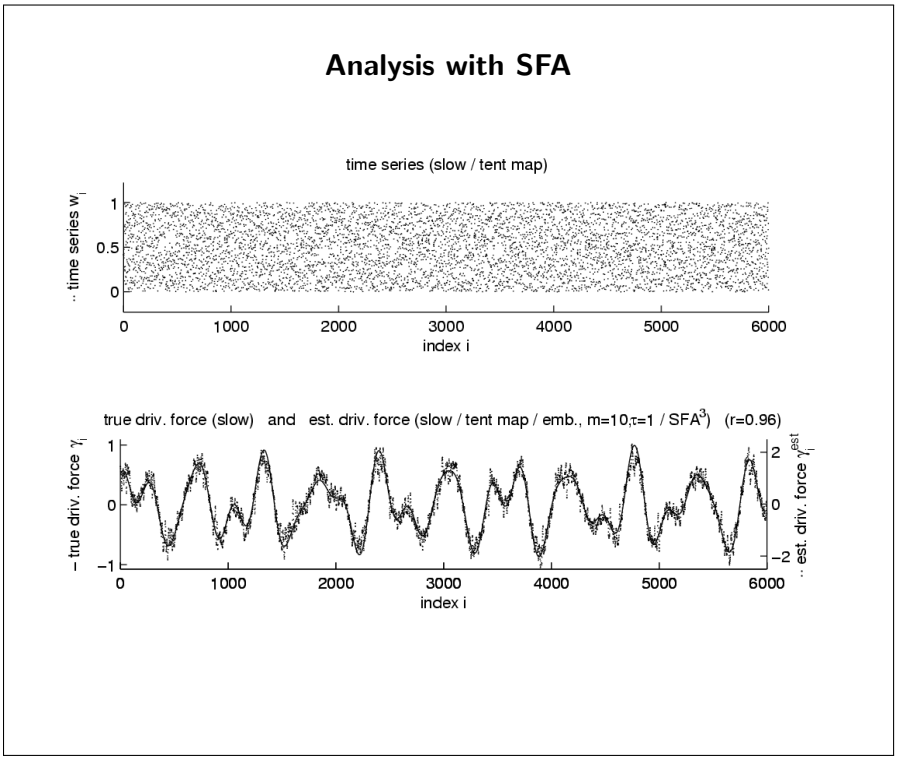


Section title: **Analyzing Non-stationary Time Series**
Image: (Peggy_Marco, 2012, pixabay, © CC0, URL)[2.1]



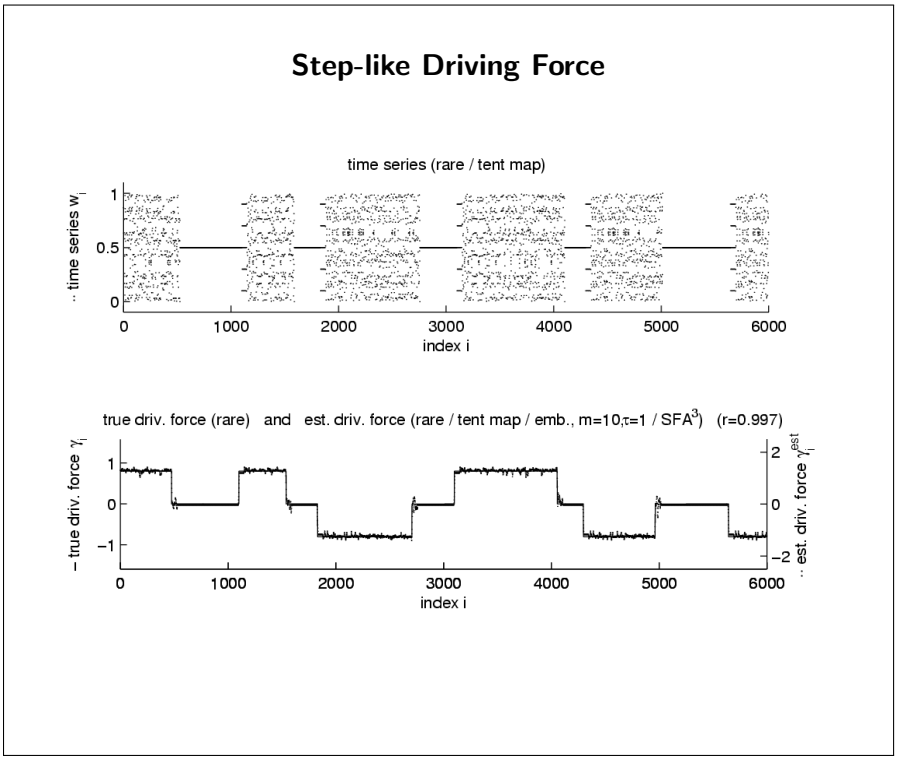
An iterative map is a discrete dynamical system. For the tent map f (shown in black) one starts with an arbitrary value w_0 between 0 and 1 and simply applies f to get the next value $w_1 = f(w_0)$. Repeating this process leads to a time series w_t like the one shown below. The tent map is peculiar in that the resulting time series has no obvious structure and looks like white noise. A similar time series results if one shifts the mapping function f cyclicly within the interval $[0, 1]$ by some value γ' (shown in red). If γ' itself depends on time, it is called a driving force of the system and it changes the system dynamics over time. If $\gamma'(t)$ changes on a slower time scale than the time series itself, SFA should be able to extract it.

Figure (top): (Wiskott et al., 2011, Fig. 11, © CC BY 4.0, URL)[2.2]
Figure (bottom): (Wiskott, 2003, Fig. 1, © CC BY 4.0, URL)[2.3]



The top graph shows a tent-map time series with a driving force $\gamma(t)$ as shown in the bottom graph (solid line). In order to apply SFA, one has to consider several (in this case 10) successive values together as the input vector for SFA, a method called time embedding. In other words SFA sees a sliding window of ten points of the time series and tries to extract some slow feature from it. Applying SFA with polynomials of degree 3 results in the slow feature shown in the bottom graph as points overlayed over the solid curve of the true driving force. Mean and variance, which can principally not be extracted, are normalized to make the curves comparable. The correlation coefficient is $r = 0.96$.

Figure: (Wiskott, 2003, Fig. 1, © CC BY 4.0, URL)[2.4]



SFA cannot only extract continuous features but also features that switch discretely between different values. This graph is the same as above except that the driving force changes in a step-like fashion.

Figure: (Wiskott, 2003, Fig. 3, © CC BY 4.0, URL)[2.5]

2.2 Nonlinear blind source separation (→ slides)

This section is based on (Sprekeler et al., 2014).

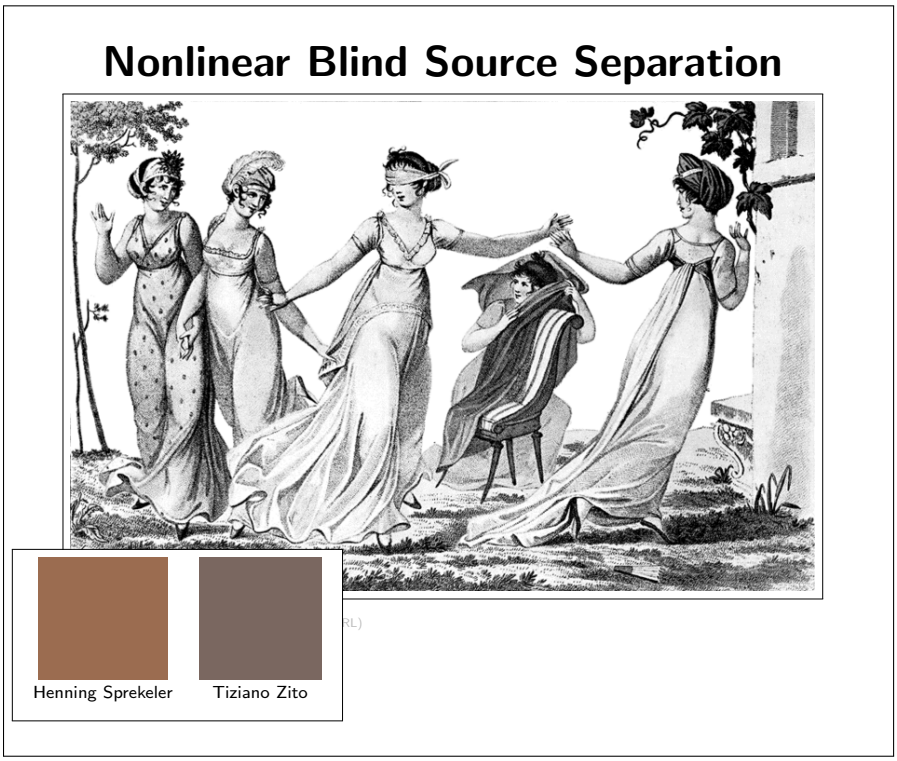
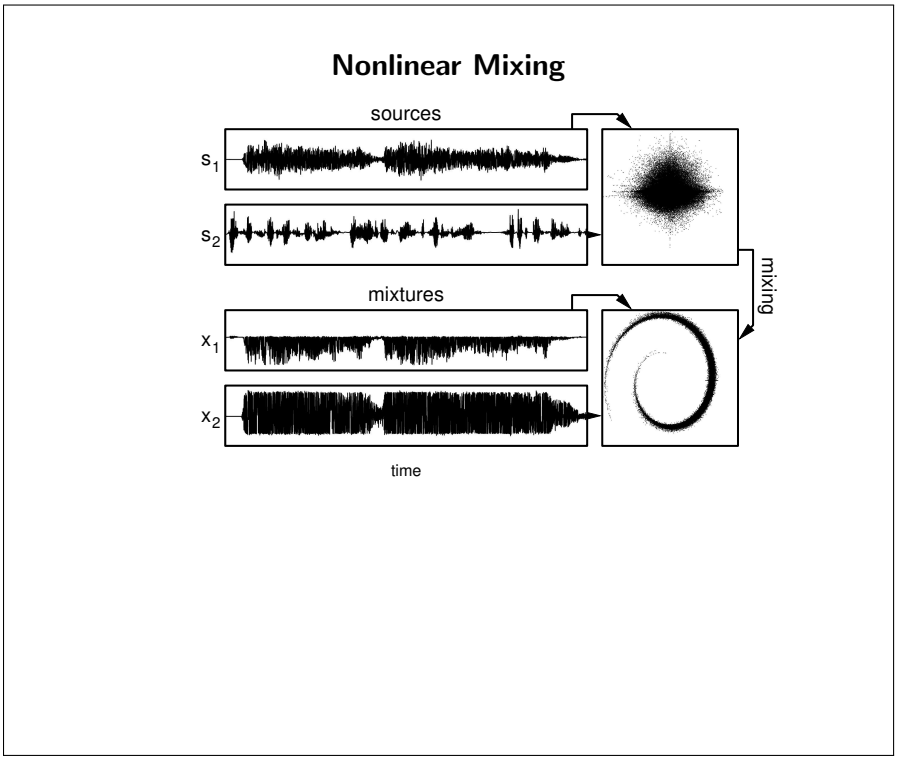


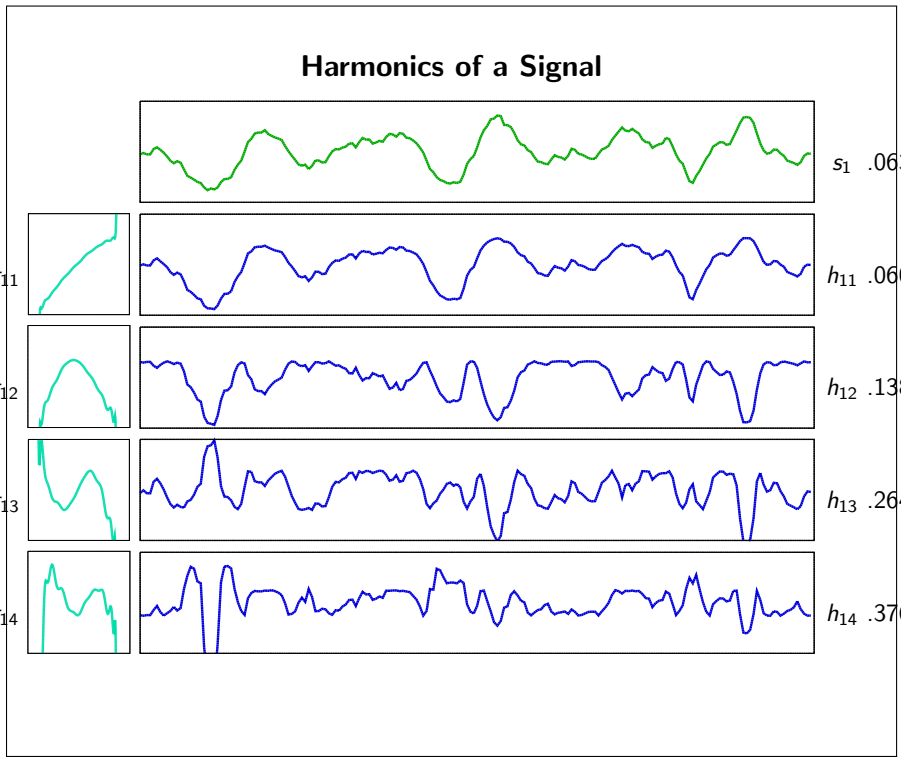
Image: ("Blind mans bluff", 1803, Wikimedia, © CC0, URL)[2.6]



We consider here the problem of nonlinear blind source separation for two sources. The sources s_1 and s_2 are shown at the top left and are plotted together in the 2D scatter plot on the right. The mixing can be viewed as stretching the 2D space and winding it up in a spiral, see scatter plot below. The first source runs along the arms of the spiral; the second source runs perpendicular to it and has a very small amplitude. The resulting mixed signals are shown as x_1 and x_2 on the left. The task of nonlinear blind source separation is to extract the two sources without knowing anything about the mixture or the sources, except that they are statistically independent and smoothly varying, i.e. not white noise.

The theory of SFA formalizes two properties that can explain why SFA might be suitable to perform nonlinear blind source separation. Roughly speaking: (i) Any nonlinearly transformed version of a signal varies faster than the signal itself. (ii) Any mixture of two signals varies faster than the slower of the two signals. (These statements are true only modulo an invertible transformation of the single sources. But that is all you can hope for in any case, because the problem of nonlinear blind source separation is defined only up to an invertible transformation of the single sources.)

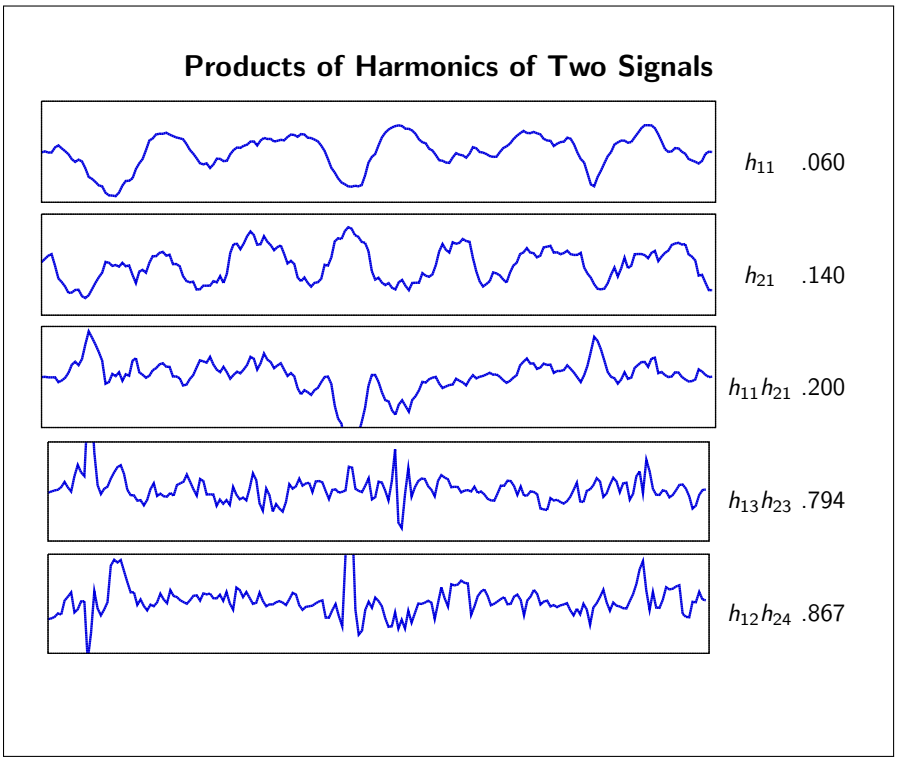
Figure: (Wiskott et al., 2011, Fig. 13, © CC BY 4.0, URL)[2.7]



This graph illustrates the principle that a signal typically gets faster if you transform it. s_1 is the original signal with a Δ-value of 0.063 (the Δ-value measure the 'fastness' of a signal). $g_{11}(s)$ is an optimal invertible transformation that makes the signal slightly slower, see $h_{11} = g_{11}(s_1)$ with a Δ-value of 0.060. The other noninvertible transformations shown all make the signal faster.

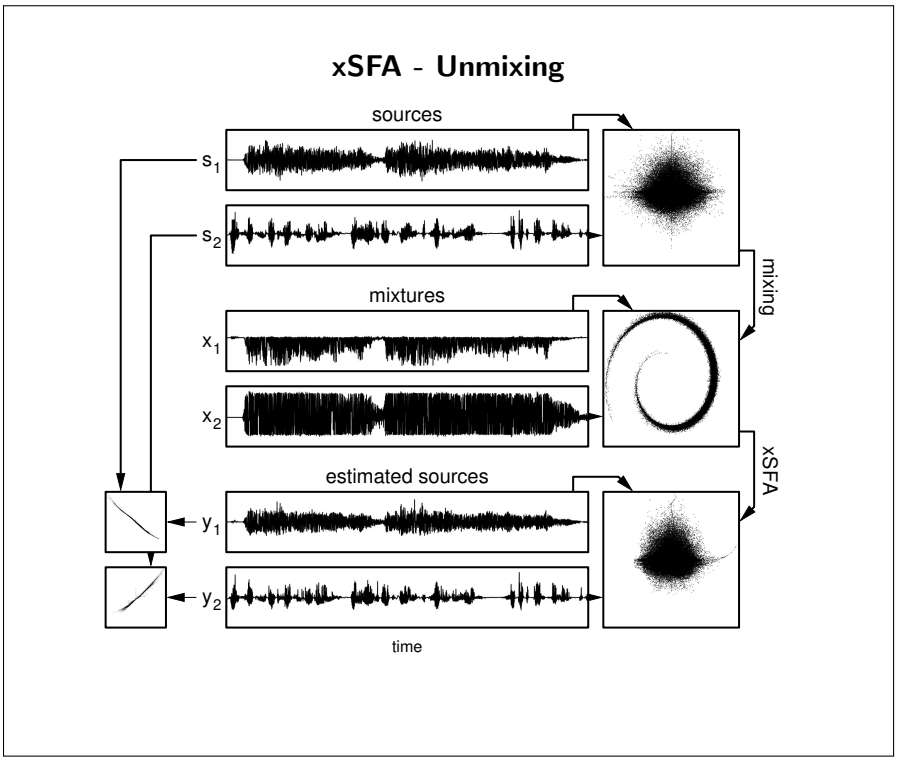
The functions g_{1m} have been found by applying SFA with high polynomials to s_1 directly without time embedding. g_{11} is therefore optimal in yielding a slower version of s_1. The resulting signals h_{1m} are called harmonics and play an important role in the theory of SFA.

Figure: (Wiskott & Escalante, 2010, unpubl., © CC BY 4.0)



This graph illustrates the principle that a mixture of two signals is typically faster than the slower of the two signals. Since any mixture can be expressed as a linear combination of products of harmonics of the two sources (similar to a 2D Taylor expansion), we only consider the first harmonics h_{11} and h_{21} of the two sources and various products of first and higher harmonics. All products are faster, i.e. have a higher Δ-value, than h_{11}.

Figure: (Wiskott & Escalante, 2010, unpubl., © CC BY 4.0)



In xSFA (extended SFA) the mixture is first expanded into a very high-dimensional space. The slowest signal within that space is then extracted with SFA and declared the first source. Next, all harmonics of that first source are projected out of the expanded signal. Then, SFA is applied again and the slowest signal declared the second source. Next, all harmonics and all products of harmonics of first and second source are projected out of the expanded signal. Iterating this scheme can in principle extract an arbitrary number of sources (if they are in the mixture), but noise reduces performance for later sources.
The graphs at the bottom show the extracted sources y_1 and y_2. As the scatter plots on the left between the extracted and the true sources show, the first source has a wrong sign (it can principally not be recovered), but otherwise the extraction is very good.
Figure: (Wiskott et al., 2011, Fig. 13, © CC BY 4.0, URL)[2.8]

References

Sprekeler, H., Zito, T., and Wiskott, L. (2014). An extension of slow feature analysis for nonlinear blind source separation. *Journal of Machine Learning Research*, 15:921–947.

Wiskott, L. (1998). Learning invariance manifolds. In Niklasson, L., Bodén, M., and Ziemke, T., editors, *Proc. 8th Intl. Conf. on Artificial Neural Networks (ICANN'98), Skövde, Sweden*, Perspectives in Neural Computing, pages 555–560, London. Springer.

Wiskott, L. (2003). Estimating driving forces of nonstationary time series with slow feature analysis. e-print arXiv: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.

Notes

[1.1] maxmann, 2016, pixabay, © CC0, https://pixabay.com/en/snail-shell-crawl-mollusk-1330766/

[1.2] Wiskott et al., 2011, Scholarpedia 5(2):1362, Fig. 2, © CC BY 4.0, http://scholarpedia.org/article/Slow_feature_analysis

[1.3] Wiskott et al., 2011, Scholarpedia 5(2):1362, Fig. 1, © CC BY 4.0, http://scholarpedia.org/article/Slow_feature_analysis

[1.4] Wiskott & Sejnowski, 2002, Neur. Comp. 14:715–770, Fig. 2, http://www.ini.rub.de/PEOPLE/wiskott/Reprints/WiskottSejnowski-2002-NeurComp-LearningInvariances.pdf

[1.5] Wiskott & Sejnowski, 2002, Neur. Comp. 14:715–770, Fig. 2, http://www.ini.rub.de/PEOPLE/wiskott/Reprints/WiskottSejnowski-2002-NeurComp-LearningInvariances.pdf

[1.6] Wiskott & Sejnowski, 2002, Neur. Comp. 14:715–770, Fig. 2, http://www.ini.rub.de/PEOPLE/wiskott/Reprints/WiskottSejnowski-2002-NeurComp-LearningInvariances.pdf

[1.7] Wiskott & Sejnowski, 2002, Neur. Comp. 14:715–770, Fig. 7, http://www.ini.rub.de/PEOPLE/wiskott/Reprints/WiskottSejnowski-2002-NeurComp-LearningInvariances.pdf

[2.1] Peggy_Marco, 2012, pixabay, © CC0, https://pixabay.com/en/mikado-domino-stones-pay-steinchen-1013877/

[2.2] Wiskott et al., 2011, Scholarpedia 5(2):1362, Fig. 11, © CC BY 4.0, http://www.scholarpedia.org/article/Slow_feature_analysis

[2.3] Wiskott, 2003, arXiv.org 0312317, Fig. 1, © CC BY 4.0, http://arxiv.org/abs/cond-mat/0312317/

[2.4] Wiskott, 2003, arXiv.org 0312317, Fig. 1, © CC BY 4.0, http://arxiv.org/abs/cond-mat/0312317/

[2.5] Wiskott, 2003, arXiv.org 0312317, Fig. 3, © CC BY 4.0, http://arxiv.org/abs/cond-mat/0312317/

[2.6] "Blind mans bluff", 1803, Wikimedia, © CC0, https://commons.wikimedia.org/wiki/File:Blind_mans_bluff_1803.PNG

[2.7] Wiskott et al., 2011, Scholarpedia 5(2):1362, Fig. 13, © CC BY 4.0, http://www.scholarpedia.org/article/Slow_feature_analysis

[2.8] Wiskott et al., 2011, Scholarpedia 5(2):1362, Fig. 13, © CC BY 4.0, http://www.scholarpedia.org/article/Slow_feature_analysis

Copyrightprotectionlevel: 2/ 2