View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by University of Central Florida (UCF): STARS (Showcase of Text, Archives, Research &...

University of Central Florida
STARS

Electronic Theses and Dissertations, 2004-2019

2004

Sinbad Automation Of Scientific Process: From Hidden Factor Analysis To Theory Synthesis

Olcay Kursun
University of Central Florida

Part of the Computer Sciences Commons, and the Engineering Commons
Find similar works at: https://stars.library.ucf.edu/etd
University of Central Florida Libraries http://library.ucf.edu

This Doctoral Dissertation (Open Access) is brought to you for free and open access by STARS. It has been accepted for inclusion in Electronic Theses and Dissertations, 2004-2019 by an authorized administrator of STARS. For more information, please contact STARS@ucf.edu.

STARS Citation
Kursun, Olcay, "Sinbad Automation Of Scientific Process: From Hidden Factor Analysis To Theory Synthesis" (2004). *Electronic Theses and Dissertations, 2004-2019*. 9.
https://stars.library.ucf.edu/etd/9

University of Central Florida
STARS
Showcase of Text, Archives, Research & Scholarship

SINBAD AUTOMATION OF SCIENTIFIC PROCESS: FROM HIDDEN FACTOR ANALYSIS TO THEORY SYNTHESIS

by

OLCAY KURSUN
B.S. Bogazici University, Turkey, 1998
M.S. Bogazici University, Turkey, 2000
M.S. University of Central Florida, 2001

A dissertation submitted in partial fulfillment of the requirements
for the degree of Philosophy of Science
in the School of Computer Science
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Spring Term
2004

ABSTRACT

Modern science is turning to progressively more complex and data-rich subjects, which challenges the existing methods of data analysis and interpretation. Consequently, there is a pressing need for development of ever more powerful methods of extracting order from complex data and for automation of all steps of the scientific process. *Virtual Scientist* is a set of computational procedures that automate the method of inductive inference to derive a theory from observational data dominated by nonlinear regularities. The procedures utilize SINBAD – a novel computational method of nonlinear factor analysis that is based on the principle of maximization of mutual information among non-overlapping sources (Imax), yielding higher-order features of the data that reveal hidden causal factors controlling the observed phenomena. One major advantage of this approach is that it is not dependent on a particular choice of learning algorithm to use for the computations. The procedures build a theory of the studied subject by finding inferentially useful hidden factors, learning interdependencies among its variables, reconstructing its functional organization, and describing it by a concise graph of inferential relations among its variables. The graph is a quantitative model of the studied subject, capable of performing elaborate deductive inferences and explaining behaviors of the observed variables by behaviors of other such variables and discovered hidden factors. The set of *Virtual Scientist* procedures is a powerful analytical and theory-building tool designed to be used in research of complex scientific problems characterized by multivariate and nonlinear relations.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

Modern science turns to progressively more complex and challenging subjects across many fields – medicine, neuroscience, genomics and related fields, ecology, economics, climatology, cosmology, etc. This expansion of scientific inquiry into until recently inaccessible territories is brought about by ever growing advances in computer and sensor technologies, which enable the collection of large amounts of groundbreaking novel experimental and observational data. On the other hand, the new subjects also address more complex phenomena that reflect causal relations among large numbers of relevant factors with only limited, if any, opportunities for experimental control and manipulation. The growing size and complexity of collected data demand progressively more sophisticated analytical and theory-building methods, methods that can process large amounts of raw data and extract intricate, deeply hidden order (Mjolsness and DeCoste 2001).

In this work, I describe a set of computational procedures that were developed to automate analysis and theory-building process for particularly difficult research problems that: (1) involve complex – multivariate and prominently nonlinear – interrelations among the measured entities; but (2) can be approached only, or mostly, through observation, without benefits of experimental manipulation of conditions; and (3) observations can be made only of spatial, but not temporal, patterns of events.

Figure 1 describes a prototypical example of such a research problem. This example will be used to illustrate practical implementation of the proposed set of theory-building procedures. The dynamical system used in this example is a prototypical mathematical model of a well-known class of physical systems, but its identity will be revealed only at the end of the exercise, so as to avoid using the prior knowledge of the system in interpreting the observational data. In this exercise, designed to demonstrate the operation of the theory-building procedures, the studied system's state is evaluated by taking seven different measurements, referred to as the "observed variables" $V_1 \ldots V_7$. Some variables are binary, others are continuous. The seven measurements taken simultaneously at any given moment constitute a single "observation"; and the theory is to be built from large numbers of such observations. The observations are not continuous, but are collected randomly without any temporal order.

My approach to theory building is based on consideration of the importance of hidden causal factors in dynamical systems, i.e., subjects of study that can be characterized by variables and quantitative relations among those variables (Clark and Thornton 1997; Favorov and Ryder 2004; Ryder 2004). That is, the behaviors of the observed entities, or *variables*, might be controlled by some unknown factors (i.e., they are not among the variables that are observed in the study).

Such hidden factors can vary greatly in the extent of their impact on the studied dynamical system, from factors that impact behaviors of just one or a few observed variables to factors that impact behaviors of most or even all of the observed variables. Hidden factors that are reflected in the behaviors of sufficiently large numbers of the observed variables can, in principle, be extracted from them through some computation. And they should be extracted: their reflectance in behaviors of multiple observed variables implies that these factors play prominent and central roles in the functioning of the studied dynamical system. This makes the knowledge of such hidden, but extractable factors crucial to theory building.

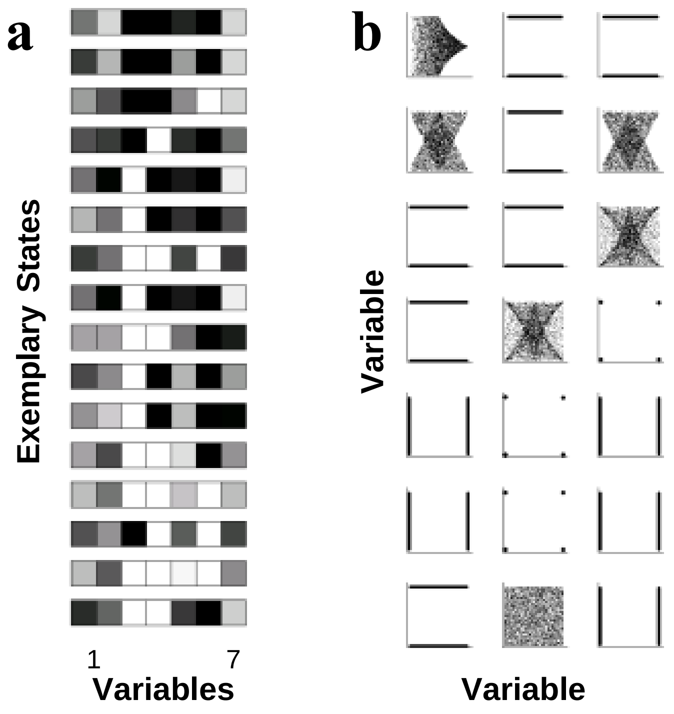

Figure 1: A research project to build a theory solely from snapshot observations of an unknown dynamical system. (a) 16 exemplary observations of the system's state. The value of each observed variable is grayscale-coded (from white = 0 to black = 1). **(b)** Plots of pair-wise relations among the observed variables. Each variable is plotted as a function of every other variable. These plots reveal no clear order among the variables, suggesting that the variables might be only weakly interdependent, or the order might be hidden in relations that are multivariate and possibly nonlinear.

Thus, the theory-building process should first analyze the behaviors of the observed variables to infer the presence of as many hidden factors as possible and learn how to compute them from the observed variables. In the next step, the theory-building process should learn orderly relations among all the observed variables and inferred factors. What will emerge out of this process is a *theory* – a detailed quantitative tracing of connections among the components of the studied dynamical system. This is a traditional approach to theory building (with hidden-factor learning recognized as conceptualization); its penetration of a subject depends on the ability of the employed analytical methods to discover hidden causal factors. Recently we have developed SINBAD (Set of INteracting BAckpropagating Dendrites) – a novel computational method of nonlinear factor analysis designed specifically for finding deeply hidden factors (Ryder and Favorov 2001; Kursun and Favorov 2002; Favorov et al. 2003; Kursun and Favorov 2003, 2004; Favorov and Ryder 2004). This method enabled me to design a set of computational procedures for building theories of particularly difficult research subjects characterized by multivariate and nonlinear relations. Reflecting on the fact that these procedures perform a quintessential work of a scientist, I named this set a "Virtual Scientist" (Kursun and Favorov 2004).

METHODS FOR FINDING HIDDEN VARIABLES

Hidden variables are often referred as high-order features, latent variables, or factors. Extracting hidden variables has attracted many researchers for a variety of reasons other than theory-building and modeling, for example, it has been linked to modeling perceptual processing stages in brain and also has had uses in data visualization. Since it involves no target values, for most varieties of the unsupervised learning algorithms used, the targets are taken to be the same as the inputs. In other words, many types of procedures for searching for hidden variables perform the same task as an auto-associative network compressing the information from the inputs. For example, Hebbian learning is a very popular variety of unsupervised learning, which minimizes the same error function as an auto-associative network with a linear hidden layer, trained by least squares, and is therefore a form of dimensionality reduction.

Although, this vast amount of research interest has been put into finding most compact representations of the data, the question of *"what to do with the data"* has not been properly addressed. However, the purpose of collecting information is *not* to store it in a compact form, but to use it in developing a complete and concise understanding of the orderly relations in it. Therefore, the approach for finding hidden variables should be shaped up according to what will be the ultimate use of these variables. I will show that SINBAD method should be employed if the goal is to find inferentially powerful hidden variables rather than highly-compressing ones.

For a quick demonstrative comparison of different approaches on their abilities to extract hidden factors, we created an artificial problem with four observed variables. The variables are, in fact, pixels from images of the same size (respective x-y locations). There are, in fact, three hidden variables whose nonlinear integration gave rise to these four observed variables: pixels from images 1, 4, and 5 are nonlinearly mixed to produce pixels of the images 2 and 3. That is, each pixel in image 2 is a nonlinear function of identically located pixels in images 1 and 5 (scaled between 0 and 1): $IM2_{xy} = IM1_{xy} + IM5_{xy} - 2 \cdot IM1_{xy} \cdot IM5_{xy}$. The same holds for image 3 – each pixel in image 3 is the analogous function of identically located pixels in images 4 and, again, 5. Thus, image 5 is a hidden factor determining the contents of images 2 and 3. Images 1 and 4 are random gray scale images. Image 5 is chosen to be a gray scale image of a fine natural scene of some bushes and grass, which is to be hidden but coded nonlinearly into the patterns of the other variables. The task is to discover the hidden factor image 5, using the four variables (images).

SINBAD Method for Finding Hidden Factors

SINBAD belongs to the class of unsupervised learning algorithms that are based on the principle of maximization of mutual information among disjoint sources of information, developed by Becker as *Imax* (1995, 1996, 1999; Becker and Hinton 1992). According to Becker and Hinton (1992), hidden factors can be discovered through a search for different, but nevertheless highly correlated functions of any kind over non-overlapping subsets of the available variables. Such *correlated functions* must have a reason for their statistical interdependence, a causal source in the domain of the data. Therefore, the correlated functions over different subsets of variables express a previously unrecognized feature (a hidden factor) that is responsible for the correlation (Becker and Hinton 1992; Phillips and Singer 1997; Ryder and Favorov 2001; Favorov and

Ryder 2004). For examples of applications of this approach, such as discovery of surfaces in stereograms, recognition of moving objects, speaker-independent vowel recognition, see Becker and Hinton (1992), Becker (1995, 1996, 1999), Stone (1996).

Becker's Imax method works by maximizing Shannon's mutual information measure among the outputs of learning modules receiving different subsets of input variables. Unfortunately, this method is computationally complex and requires making various restrictive assumptions about the output distributions of the modules to get a tractable expression for the mutual information between two continuous signals (Becker 1996). SINBAD is a simpler approach that works by minimizing the mean-square-error among the outputs of the learning modules. SINBAD method avoids trivial minimization of this error by forcing the output signals to have nonzero variance (otherwise, if all modules give the same constant output, the error would be trivially minimized).

SINBAD cell is a powerful discoverer of hidden factors. For a demonstration, consider the dataset of the four images. As explained earlier), Images 1 and 2 share mutual information with images 3 and 4 in the form of the hidden image 5. To discover this hidden factor, SINBAD cell is trained on the values of identically located pixels in images 1, 2, 3, 4. Pixels from images 1 and 2 are given to Dendrite 1, pixels from images 3 and 4 are given to Dendrite 2. As shown in Figure 2, SINBAD cell gradually learns – without any guidance – to output the value of the identically located pixel in the hidden image 5 (compare IM5 with the image below the cell). In conclusion, although image 5 was so well hidden in images 2 and 3 as to be invisible there, SINBAD cell nevertheless was able to easily detect its presence and extract its content. In contrast, linear factor analysis methods, including PCA and ICA (Hyvarinen et al. 2001), will not work here because of nonlinearity of the underlying relationship. And even nonlinear factor extraction methods – Nonlinear PCA (Kramer 1991) and Nonlinear ICA (Lappalainen and Honkela 2000; Valpola et al. 2001) – could not accomplish this task (see Figure 4), reasons for which will be discussed later.

The SINBAD architecture illustrated in Figure 2, contains two learning modules, each in a form of a backprop net (an error backpropagation network of Rumelhart et al. 1986). However, it is straight-forward to modify the architecture to include more than two learning modules or to use some other learning modules such as support vector machines instead of backprop-nets. The backprop nets receive different, non-overlapping sets of input channels, while their outputs are added together to produce a final output. This final output is used as a teaching signal for each of the backprop nets, which means that the nets are set up to teach each other to produce maximally *correlated* outputs in response to their *different* inputs. As a result, the nets tune to the causal source (a hidden factor) responsible for the correlation.

Imax approach is an important elaboration of information theoretical approaches as it defines a measure of value of the content of the data. In other words, Imax-based methods uses redundancies to decide what is interesting in the data (and more interesting if more independent modules can maximize their mutual information).

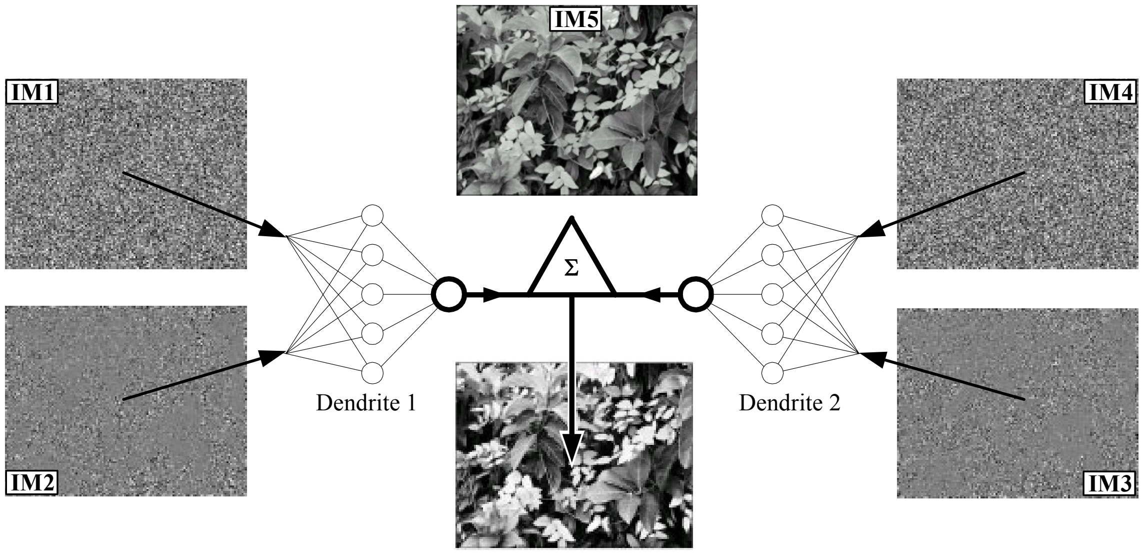

Figure 2: The SINBAD cell with two dendrites. In this particular design, each dendrite is an error backpropagation network with one output unit and a single layer of hidden units.

SINBAD was originally developed as a model of a single neuron in the cerebral cortex (Ryder and Favorov 2001; Favorov et al. 2003; Favorov and Ryder 2004). According to SINBAD hypothesis, the basic function of a cortical neuron is to discover and represent one of the hidden factors in its sensory environment. This task is proposed to be accomplished by endowing each of several dendrites that originate from a neuron's body with functional capabilities comparable to those of a backprop net. Although the neurobiological origins of SINBAD algorithm are not important for the present subject, in this work I continue to call the entire setup a "SINBAD cell" and each backprop net a "dendrite".

The detailed formulation of the SINBAD cell can be found in (Ryder and Favorov 2001). As a brief description of its implementation in this section, the activity of a hidden unit *h* in dendrite *d* is computed as a sigmoid function of the activities of its input sources:

$$H_{d,h} = \tanh(\sum_i w_{d,i,h} \cdot A_{d,i}), \qquad (1)$$

where $A_{d,i}$ is the activity of input source d,i and $w_{d,i,h}$ is the weight of its connection onto the hidden unit h of dendrite d. The activity of the output unit, i.e. the output of dendrite d, is:

$$D_d = \sum_h w_{d,h} \cdot H_{d,h}, \qquad (2)$$

where $w_{d,h}$ is the weight of the connection from the hidden unit d,h to the output unit. The outputs of the two dendrites are summated to produce the cell's output:

$$A = D_1 + D_2. \qquad (3)$$

The cell's output A is the principal contributor to the training signal T; it is used to adjust the weights of connections on the two dendrites. Additional factors contributing to the training signal are: (1) the average output activity of the cell, $\bar{A}$, driving the cell to have $\bar{A} = 0$; and (2) deviation of the current output activity from the average, $A - \bar{A}$, designed to expand the dynamic range of output values. Thus,

$$T = A - \alpha \cdot \overline{A} + \beta \cdot (A - \overline{A}), \qquad (4)$$

where α and β are scaling coefficients. The coefficient β is determined by the variability of the output activity: the smaller the variability, the greater the value of β. It is computed as:

$$\beta = \left[\beta_{\max} - \gamma \cdot \overline{|A - \overline{A}|}\right]^+, \qquad (5)$$

where $\beta_{\max}$ and γ are controlling parameters, and $[\cdot]^+$ indicates that if the quantity is negative, the value is to be taken as zero. The connections of the hidden units are adjusted according to the error backpropagation algorithm of Rumelhart et al. (1986). Specifically, the error signals δ_d is first computed for the two dendrites as:

$$\delta_d = T - 2 \cdot D_d . \tag{6}$$

For the hidden units, δ_d is backpropagated as:

$$\delta_{d,h} = \delta_d \cdot w_{d,h} \cdot (1 - H_{d,h}^2). \tag{7}$$

Connection weights are adjusted by:

$$\Delta w_{d,i,h} = \mu_i \cdot A_{d,i} \cdot \delta_{d,h} \quad \text{and} \quad \Delta w_{d,h} = \mu_h \cdot H_{d,h} \cdot \delta_d , \tag{8}$$

where μ_i and μ_h are learning rate constants for the input and hidden unit connections, respectively.

Other Methods for Finding Hidden Variables

Linear Methods

Linear factor analysis is limited to identifying the underlying sources that are mixed linearly to produce the observations. However, generally, the sources are convolved nonlinearly. Two most frequently applied linear factor analysis methods are principal component analysis (PCA) and independent component analysis (ICA). ICA is a linear factor analysis method that is recently developed for the case where the sources are nongaussian; whereas, PCA assumes gaussian sources. An important use of linear methods is that nonlinear hidden variables can be searched after the linear ones are extracted first, mainly for reducing the nonlinear search time by linear dimensionality reduction. However, this approach can easily fail due to the creation of artificial linear dimensions.

<u>Principle Components Analysis:</u>

Principal component analysis and the closely related Karhunen-Loeve transform are classic techniques in linear factor analysis. The goal of PCA is to find a smaller set of variables with no linear redundancy that is as good a representation as desired. The measure of redundancy used in PCA is the linear correlations among the found factors. Because of that, PCA is based on second-order statistics only. On the other hand, the measure of redundancy used in ICA is the concept of independence, which is much more general.

Computationally, closed-form and on-line learning algorithms are present for PCA. The on-line learning for PCA is a special case of nonlinear PCA discussed later. The closed form solution for PCA is based on first computing the covariance matrix and then solving for the eigenvectors of the covariance matrix. The eigenvectors with corresponding non-zero eigenvalues are the linear dimensions that are found to be more compact in explaining the given data.

The dataset of four images is passed through the principal component analysis. Although we know there are only three true independent underlying factors, PCA yields four non-zero eigenvalues. That is because PCA is only a linear method. Moreover, as expected, the found sources had no correlation to the original sources.

Independent Components Analysis:

ICA tries to find interesting linear dimensions. By Central Limit Theorem, mixtures of random variables tend to be more Gaussian than the original ones. Therefore, identifying random variables that are the least Gaussian yields single hidden variables. The classical measure of nongaussianity is kurtosis or the fourth-order cumulant. The kurtosis of a random variable *y* is classically defined by

$$\mathrm{kurt}(y) = \mathbf{E}\{y^4\} - 3(\mathbf{E}\{y^2\})^2 \qquad (9)$$

Kurtosis is zero for a gaussian random variable. For most (but not all) nongaussian random variables, kurtosis is nonzero. Kurtosis can be both positive or negative. Random variables that have a negative kurtosis are called subgaussian, and those with positive kurtosis are called supergaussian. Supergaussian random variables have typically a "spiky" probability distribution function (pdf) with heavy tails, i.e. the pdf is relatively large at zero and at large values of the variable, while being small for intermediate values. Subgaussian random variables, on the other hand, have typically a "flat" pdf, which is rather constant near zero, and very small for larger values of the variable. A typical example is the uniform distribution.

Typically nongaussianity is measured by the absolute value of kurtosis. The square of kurtosis can also be used. These measures are zero for a Gaussian variable, and greater than zero for most nongaussian random variables. In practice we start from some weight vector **w**, compute the direction in which the kurtosis of $\mathbf{y} = \mathbf{w}^T \mathbf{x}$ is growing most strongly (if kurtosis is positive) or decreasing most strongly (if kurtosis is negative) based on the available sample of mixture vector **x**, and use a gradient method for finding a new vector **w**.

Instead of having different update rules for minimizing or maximizing kurtosis, we can estimate the kurtosis of the current projection online and continue with the sign of the online kurtosis measure K:

$K = m^4(t) - 3(m^2(t))^2$, (10)

where, $m^n(t+1) = [(1-\tau)\, m^n(t)] + [\tau\, (A_t)^n]$, (11)

where, m^n is the online estimate of the nth moment of the target cell activity A_t. Finally, the update rule for the weight is given by:

$\mathbf{w}_{\text{new}} = \mathbf{w}_{\text{old}} + R_L\, [\text{sign}(K)\, A_s\, A_t^3 + A_s\, A_t\, (1-||w||^4)]$ (12)

where, R_L stands for the rate of learning (rate of maturation for the weights).

The rate of learning should be chosen small and should be annealed to zero for convergence. However, as expected again, due to its linearity, when applied to our illustrative example of images, linear independent component analysis failed to extract the hidden factor image 5.

Nonlinear Methods

Not surprisingly, extracting the nonlinear factors is a much more difficult task than extracting the linear ones. In fact, the non-uniqueness of nonlinear factors has been proven, which can simply be summarized as many of the nonlinear mixtures of the identified nonlinear factors might still be independent. In fact, this is where our approach claims that, under certain conditions, only some of the independent nonlinear components are true factors as they can be measured to be more efficient and concise in explaining the regularities among the observed variables.

Several techniques have been developed in the nonlinear factor analysis field with different assumptions. Two of the most remarkable types of these techniques that we will compare to are the nonlinear extensions of the linear methods PCA and ICA, known as nonlinear principal components analysis and nonlinear independent components analysis, respectively.

In this section, we will review five-layer auto-associative neural network, which belongs to nonlinear principal components analysis category; Bayesian ensemble learning, which belongs to nonlinear independent components analysis category.

<u>Five-Layer Auto-Associative Neural Network:</u>

For auto-associative neural networks, output of the network should be equal to the input to the network. The only restriction here is that we have a bottleneck layer (Figure 3). Thus, the network should find some nonlinear directions that more efficiently code the given data.

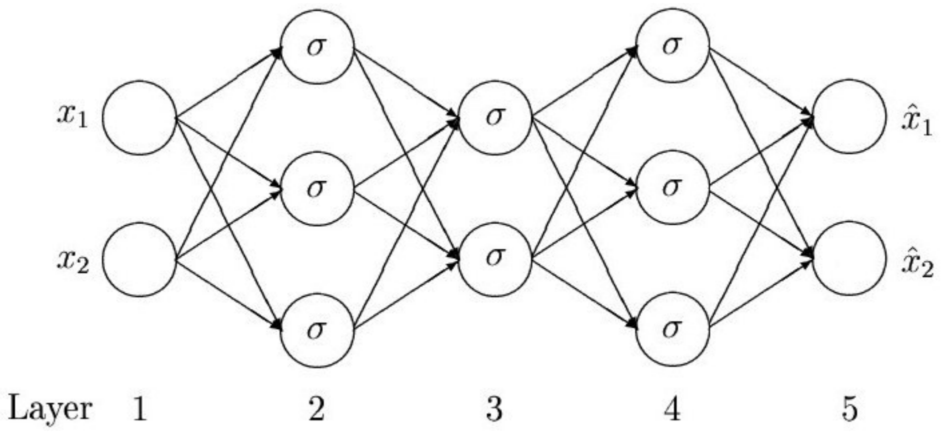

Figure 3: Five-layer auto associative neural network, where σ represents the nonlinear activation functions of the units. The other units have linear activation functions. The units in the central layer can tune to any function of the input variables. In general, they tune to functions of all input variables. Such functions ought not to be local nor correspond to causal sources in the observed system as these causal sources are not complex mixtures in their nature.

If the central layer is given less number of units than the input layer, which is called a bottleneck layer, this architecture is forced to find a compact and hidden representation that captures all the information. First layer to third layer is where the compression takes place and third layer to fifth layer accomplishes the decompression. In this architecture, there is a one-to-one correspondence between the input and output layers, i.e., there are equal number of nodes in input layer and output layer and the training signal for the output channel j is equal to the activity of input channel j. The error function to be minimized for the training sample j becomes:

$$E^j = \sum_i (X_i^j - \hat{X}_i^j)^2 \tag{13}$$

where i ranges over all input/output channels.

The connection weights are updated according to the standard back-propagation algorithm (Rumelhart et al. 1986). It has been shown that this technique is equivalent to PCA if the nonlinear activations are replaced by linear activations. If we keep the nonlinear activations, this architecture extracts nonlinear factors that can more compactly represent the input. However, these factors are, in general, nonlinear mixture of most of the input variables, which are neither local, nor orderly. Thus, when applied to the dataset of four images, none of the found nonlinear factors corresponded to the hidden image 5.

<u>Bayesian Ensemble Learning:</u>

Bliss is a research project funded by the European Commision in the scope of the Information Societies Technology (IST) programme, whose one of the main objectives is to achieve significant advances in nonlinear factor analysis. Their web-address is http://www.bliss-project.org/. We followed the links to nonlinear factor analysis and downloaded a

Matlab toolbox for a recently developed nonlinear factor analysis technique at http://www.cis.hut.fi/projects/ica/bayes/ for our comparative simulations. In this section, we summarize the basics of ensemble learning developed by the top-researchers in factor analysis field working for Bliss-project.

This nonlinear independent component analysis algorithm is based on generative learning, which means it looks for a compact model which allows a shorter description of the observed data (nonlinear dimensions) in the hope of discovering some of the underlying causes of the observations.

The algorithm uses multi-layer perceptron (MLP) network to model the nonlinear mapping from sources to observations and ensemble learning to estimate the posterior distributions of the unknown variables of the model, consisting of the parameters of the MLP network, source signals, noise levels, etc.

The learning algorithm is a gradient based second order method. It is able to efficiently prune away superfluous parts of the network, which is linked to the robustness of the learning algorithm against overfitting. It is necessary when fitting a flexible nonlinear model such as an MLP network to observations.

The distribution of the sources is modeled by a mixture of Gaussians. The procedure can be summarized as first using nonlinear PCA to estimate a nonlinear subspace and then using nonlinear ICA to refine the model. This is analogous to the linear case where linear PCA is often used for estimating a linear subspace for the linear ICA.

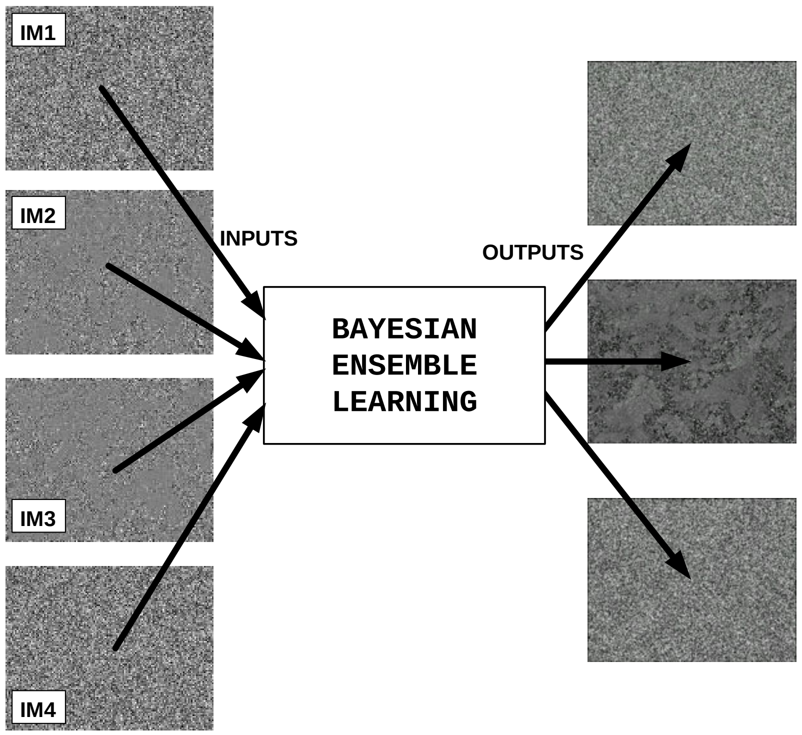

Figure 4: Discovery of nonlinear independent components by Bayesian Ensemble Learning algorithm for nonlinear factor analysis (Lappalainen and Honkela 2000; Valpola et al. 2001). This algorithm was applied to the set of 4 images shown in Figure 2 and run to extract three independent components (since, as explained in Figure 2 legend, that is the true number of independent variables characterizing these images). Note that neither of the extracted independent components, shown as a set of three images on the right, resembles the true hidden variable, i.e., Image 5 in Figure 2.

Although it is possible to measure the complexity of the mapping and the sources in generative approaches, no algorithms which would do this for nonlinear ICA have been proposed apart from this algorithm. Self-organising maps (SOM) and generative topographic mapping (GTM) have been used for nonlinear ICA. The number of parameters grows exponentially as a function of sources both in SOM and GTM, which makes these mappings unsuitable for larger problems (Valpola et al. 2001).

However, ensemble learning becomes complicated if there are multiple computational paths from a latent variable to an input variable (Valpola et al. 2001). Thus, one more time, we are not able to extract the hidden variable IM5 in our illustrative example of images. Found sources are shown in Figure 4.

FINDING MINIMAL SETS OF RELATED VARIABLES

For a SINBAD cell to be able to find a hidden factor, its dendrites must be given *different but related* groups of input variables; i.e., groups with no input variable(s) in common, but with *implicit* information about the same hidden factor. In some cases such related groups might be obvious or easy to guess. For example, in binocular vision, receptors in the two eyes obviously make up two different but related groups of sensory variables with mutual information about the third visual dimension (Becker and Hinton 1992). In general, however, such knowledge is not available and the related groups of variables can only be found by trial and error. In the absence of any heuristic, such a search is exponential – if we have a total of *N* observed variables, then we can separate them into $O(2^N)$ possible pairs of groups. Each such pair will have to be tested for whether a hidden factor can be extracted from it by a SINBAD cell. Overwhelming majority of such blindly tried partitions of the observed variables into pairs of groups will produce unrelated or insufficiently related groups, which will fail to yield any hidden factors (i.e., SINBAD cell's dendrites will fail to learn to produce correlated outputs).

Because it is exponential, an exhaustive search among even small (e.g., 20) numbers of variables is prohibitively expensive and must be minimized as much as possible. My approach to such minimization is based on choosing one of the variables as a "target" variable and then identifying the other variables that together can be used to make the most accurate prediction of that target variable, thus indicating a minimal but complete set of closely related variables (locality in the inferential space). In other words, suppose that among the entire set of *N* observed variables, $V_1 \ldots V_N$, a variable V_t (a "target" variable) can be computed, with minimal error ε, from a subset of variables $V_a \ldots V_k$ and a hidden factor H: $V_t = f(V_a \ldots V_k, H) + \varepsilon$. Suppose that this relationship is fully or at least mostly reversible, allowing H to be computed, with some minor error ε^*, from V_t and $V_a \ldots V_k$: $H = f^*(V_t, V_a \ldots V_k) + \varepsilon^*$. Suppose also that H can be computed from the observed variables $V_l \ldots V_p$: $H = h(V_l \ldots V_p)$.

This means that if we set up a backprop net with inputs from all the observed variables $V_1 \ldots V_N$, except V_t, we might be able to train that net to output target variable V_t with an error close to ε (since the input variables include $V_a \ldots V_k$ and $V_l \ldots V_p$). This also means that the number of inputs to the backprop net can be reduced without any loss of its performance, *as long as the discarded variables do not include $V_a \ldots V_k$, $V_l \ldots V_p$.*

In my terminology, the set of the observed variables $V_a \ldots V_k$, $V_l \ldots V_p$ is the "Predictive Set" of the "target" variable V_t. A target variable together with its Predictive Set make up a "Minimal Set of Related Variables."

The Minimal Set defined for target variable V_t contains all the observed variables needed for finding hidden factor H, and no other variables: it is the smallest possible set for finding H. If we test all possible two-group partitions of this set, one of the partitions will be $\{V_t, V_a \ldots V_k\}$ vs. $\{V_l \ldots V_p\}$. With one dendrite of a SINBAD cell receiving V_t, $V_a \ldots V_k$ and the other dendrite receiving $V_l \ldots V_p$, the two dendrites will learn to produce correlated outputs by computing

$$f^*(V_t, V_a \ldots V_k) \approx h(V_l \ldots V_p) = H. \qquad (14)$$

The great benefit of confining an exhaustive search of variable partitions to a Minimal Set, rather than searching over all the observed variables, is a drastic reduction of the number of partitions that will require testing for hidden factors. Moreover, conceptually, finding the minimal sets of related variables is necessary because the unsupervised search for the mutual information among the non-overlapping subsets would potentially lead to complex combinations of hidden variables if the subsets were allowed to be arbitrarily large. Such a search might, then, produce useless functions of the observed variables.

For example, consider pixels in a visual field as the observed variables. If the subsets were allowed to be arbitrarily large, we could divide this visual field into two halves and look for mutual information between them. However, this search would not end up with any local image features such as lines, curves, textures which are the most prominent local features of natural images (Kursun and Favorov 2002, 2003; Olshausen and Field 1996) and the subject of the earliest stages of visual processing in the primary visual cortex (DiCarlo and Johnson 2000).

Instead, it might end up deriving variables that are too trivial. An example of such trivial hidden variables discovered is the average illumination. Given such big subsets to train against each other, we might also end up with very complex, very high-level functions between these subsets, but we would not expect this search to end up with a hidden variable that represents, say a *tree*, in the visual field. On the other hand, first discovering lines and curves and then building on it, would allow us to be able to eventually discover deeper order regularities.

The need to extract hidden variables from local-to-global is another important reason for finding minimal sets. The local hidden variables will have multiple uses (manifestations) and unless they are made explicit they will have to be implicitly computed in more complex inferential links. Consider that, as a simple illustrative example, variable x is one of the observed variables and many inferential links among the observed variables use some functions of x^2, such as $sin(x^2)$ and $cos(x^2)$. It would be more cost-efficient to make x^2 an explicit variable before

$sin(x^2)$ and $cos(x^2)$ to reduce the time and space complexity of the computations (Kursun and Favorov 2004). Relating to visual data, it would be easier to explain complex object in terms of building blocks such as rectangles, circles and so on; and in turn to explain these building blocks in terms of lines and curves.

To summarize, for the SINBAD search for hidden factors to be thorough while accomplishable in reasonable time, Minimal Sets of related variables should be identified first. Then each Minimal Set should be partitioned in all possible ways into pairs of subsets. Each such pair of subsets should be tested, using SINBAD cell method, for whether it will yield a hidden factor.

To identify Minimal Sets, we can use each observed variable in turn as a target variable. For each Minimal Set, the first step is to determine how accurately its target variable can be predicted from all the available observed variables. The next step is to determine which of the variables contribute to this prediction and which ones can be dropped without loss of prediction accuracy. A simple way to accomplish the first step is to set up a backprop net with inputs from all but the target observed variables and train it on the target variable. However, this approach can under-perform or even fail if the relationship between the target variable and the other variables is orderly but not unique. For example, the available observed variables might only have information about how much the target variable deviates from its mean, but not in which direction (e.g., an ability to predict a line segment in an image from its context, but not whether it is darker or lighter than background). In such cases it will be necessary to identify the "regular" component of the target variable; i.e., such a derivative of the target variable that preserves maximal information about the variable, *and* that is also maximally predictable from the other available variables. Such a regular component of a variable might be the variable itself or another *unary* function of it.

The task of learning the regular component of the target variable and determining the accuracy of its prediction from the observed variables can be accomplished by using a SINBAD cell (Figure 5). For this task one dendrite is given all but the target observed variables and the other dendrite is given only the target variable. After a training period the second dendrite will learn to output the regular component of the target variable, while the first dendrite will learn to output the closest possible approximation of that component. The accuracy of this approximation can be measured by coefficient of determination (i.e., the squared coefficient of correlation of the outputs of the two dendrites).

To determine which variables are really used by the first dendrite in predicting the regular component of the target variable, a version of *sequential backward elimination* technique (Bishop 1995) is used. Other network pruning techniques, such as weight elimination (Hanson and Pratt 1989; Lang and Hinton 1989), optimal brain damage (LeCun et al. 1990), or optimal brain surgeon (Hassibi and Stork 1993) are possible alternatives among famous methods for feature selection. These methods prune the unnecessary connections after detecting them by using a measure of saliency of the weights.

Weight elimination introduces a weight decay term as a form of regularization that favors very small or very large weights. At the end of the training, the saliency of a weight is taken to be the magnitude (strength) of it.

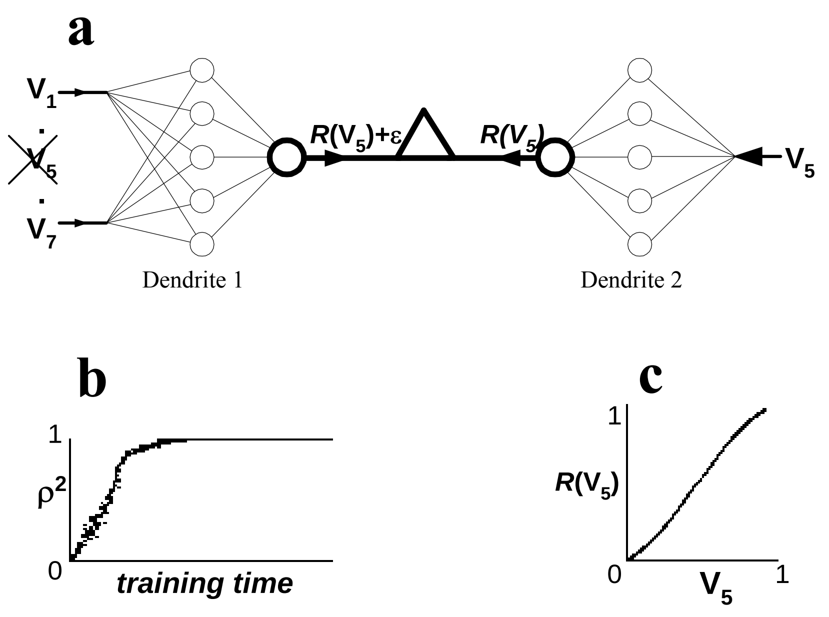

Figure 5: Learning the regular component of a target variable and the accuracy of its prediction from the observed variables. Variable V_5 from the research project described in Figure 1 is used as an example. **(a)** SINBAD cell with dendrite 2 receiving V_5 and dendrite 1 receiving all the other six observed variables. $R(V_5)$ – the regular component of V_5. After a learning period, dendrite 2 will output $R(V_5)$ and dendrite 1 will output $R(V_5)$ + a minimal error. **(b)** Time-course of learning. ρ – correlation coefficient between outputs of dendrites 1 and 2. ρ^2, coefficient of determination, is plotted as a function of training time (i.e., the number of training trials). Note that the two dendrites learned to match each other's output almost perfectly ($\rho^2 = 0.999$). **(c)** The regular component of V_5. The plot shows that the regular component of V_5 is the variable itself.

Optimal brain damage, instead of computationally demanding direct evaluation of the change in the error function when a weight is set to zero (deleted), uses an estimate of the saliency of a weight by using the diagonal terms of the Hessian matrix. However, the assumption that the Hessian matrix for a network is diagonal is, generally, a poor one. Optimal brain surgeon technique does not make such an assumption and the saliency of a weight is computed by using the inverse Hessian matrix of the network, which is costlier in terms of computation.

Leave alone its simplicity and intuitiveness, there are three main reasons I chose to use sequential backward elimination in contrast to the other methods for finding minimal sets. First and most importantly, these techniques are, ultimately, based on some approximations of the saliency of the weights, whereas, the most natural measure of the saliency of an input variable to a learning module regardless of what learning algorithm it uses (neuronal or not) is the increase in the error of the estimate of the training signal as a result of deleting that input variable, which is exactly what is measured by sequential backward elimination. Secondly, I wanted the Virtual Scientist procedures to be independent of the learning algorithm used in SINBAD implementation. Thus, the backward elimination algorithm appeared to be the only suitable choice as the others were neuronal algorithms. Third and lastly, other network pruning techniques require additional parameter optimization. However, based on a particular choice of learning algorithm used in SINBAD implementation, any suitable network pruning technique can be used for finding the minimal set of related variables.

The version of backward elimination I employed works as follows: starting with the first dendrite connected to all but the target observed variables, we remove those variables one at a time and each time re-train (not necessarily fully) the dendrite. At this stage, learning in the second dendrite must be stopped, so that the training signal for the first dendrite will remain to be the regular component of the target variable, already found by the second dendrite. In effect, the SINBAD cell is reduced here to a single backprop net trained on the regular component. If correlation between the two dendrites declines as a result of the removal of an input variable from the first dendrite, that means that the removed variable was useful for predicting the regular component of the target variable and should be restored. If dendritic correlation does not decline, it means that the removed variable was not relevant to the computation and should not be restored (Figure 6).

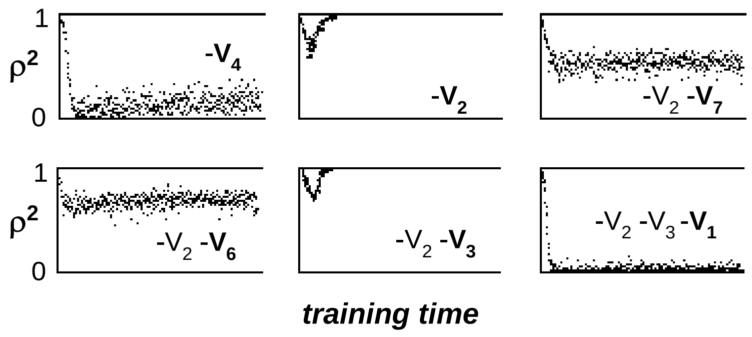

Figure 6: Finding a Minimal Set of related variables, with V_5 as a target variable. Dendrite 1 of the SINBAD cell in Figure 5 is trained on $R(V_5)$. Each plot shows the time-course of dendrite 1 learning after removal of some of the input variables. For a benchmark, the horizontal line at the top of each plot shows the magnitude of the coefficient of determination (ρ^2) between $R(V_5)$ and the output of dendrite 1 when it had all six input variables, $V_1 – V_4$, V_6, V_7.

In the panels from left to right in Figure 6, variable V_4 was removed from dendrite 1 first. Correlation between $R(V_5)$ and dendrite 1 output ("dendritic correlation") dropped almost to 0 and never recovered, indicating that V_4 is crucial for predicting $R(V_5)$ and should be kept as an input to dendrite 1. V_2 was removed next (next panel). Dendritic correlation declined only transiently and made a quick and complete recovery to the original level, indicating that V_2 is not needed for predicting $R(V_5)$. Next, V_2 and V_7 were removed, and since dendritic correlation dropped permanently, V_7 was judged to be necessary for $R(V_5)$ prediction. The same conclusion was reached for V_6. However, removing V_2 and V_3 did not reduce dendritic correlation, indicating that V_3 can be discarded. Finally, V_1 was found to be needed. Based on this series of tests, $R(V_5)$ can be best predicted from the observed variables V_1, V_4, V_6, V_7. Therefore, this set of variables is a predictive set of V_5 and together they make up a Minimal Set $\{V_1\ V_4\ V_5\ V_6\ V_7\}$.

To be more precise, when we test whether a variable contributes to the prediction of a target variable, we compare the accuracy of the prediction with and without that variable. If the difference is less than a predetermined *threshold*, then we conclude that the variable is not needed and remove it from the dendrite. The variables that remain still connected to the first dendrite after testing all of them constitute the *predictive set* of the target variable. The threshold, T, determines which and how many variables are chosen for the predictive set of the target variable. Let P_T denote the predictive set of the target variable found by the sequential backward selection algorithm with threshold T, where $|P_T|$ is the size (cardinality) of the predictive set, and ρ^2_T is the coefficient of determination between the regular component of the target variable and its prediction by the set P_T. We want to use the threshold that yields the predictive set of the smallest size, but with the maximal prediction of the target variable. Formally, we are looking for a T value such that $|P_{T+\xi}| \leq |P_T| << |P_{T-\xi}|$, but $\rho^2_{T+\xi} << \rho^2_T \approx \rho^2_{T-\xi} \approx \rho^2_0$, where ξ is a small number and ρ^2_0 is the coefficient of determination using all variables.

A given target variable and its predictive set constitute one Minimal Set of related variables. We identify multiple such Minimal Sets, using every observed variable as a target variable (Table 1). It is a common occurrence that Minimal Sets identified using different variables as targets will have identical compositions. For example, in Table 1, sets 1, 4, and 5 are identical, and so are sets 2 and 3, and sets 6 and 7. As a result of such rediscoveries of the same Minimal Sets, the number of distinct Minimal Sets identified in a given study is likely to be smaller than the number of the observed variables.

Table 1

List of Minimal Sets of related variables extracted from the observed variables $V_1 - V_7$.

Set #	**Target Variable**	**Minimal Predictive Set**	ρ^2
1	V1	V4 V5 V6 V7	.998
2	V2	V3 V4 V5 V6 V7	.921
3	V3	V2 V4 V5 V6 V7	.941
4	V4	V1 V5 V6 V7	.994
5	V5	V1 V4 V6 V7	.999
6	V6	V1 V2 V3 V7	.995
7	V7	V1 V2 V3 V6	.999

PARTITIONING MINIMAL SETS TO FIND HIDDEN VARIABLES

Once the Minimal Sets of related variables are identified, each of them should be partitioned in all possible ways and each partition should be tested for hidden variables. As stated above, if in a given predictive set, the role of a subset of variables is to contribute information about a hidden factor relevant to the prediction, then a SINBAD cell will learn this factor if one of its dendrites is given this subset of variables and the other dendrite is given the rest of the predictive set together with the target variable (Equation 14). Any other partitioning of the predictive set across the two dendrites will only reduce their mutual information, and will therefore reduce correlation of the dendrites' outputs (Figure 7).

A SINBAD cell was trained on each of the partitions shown in Figure 7, with time-course of learning shown in one of the panels. For example, in the top-left panel, dendrite 1 was given variables V_1, V_4, V_6 and dendrite 2 was given V_5 and V_7. The plot of dendritic correlation (ρ^2, coefficient of determination of the two dendrites' outputs) shows that the dendrites were unable to correlate their outputs as well as in some other panels. The best correlation was achieved in two panels, for partitions V_1 V_6 V_7 vs. V_4 V_5 and V_6 V_7 vs. V_1 V_4 V_5. For these partitions, $\rho^2 = 0.999$ – the same level as was achieved by the entire predictive set of V_5 against V_5 (see Figure 6). All other partitions produced clearly much lower dendritic correlations, indicating that they are unnatural. Therefore we take the two best partitions as apparently revealing two candidate hidden factors. These factors are new variables derived from the observed variables. They expand the list of variables characterizing the studied dynamical system; we label them as V_8 and V_9: $V_8 = f_8(V_1, V_6, V_7)$ and $V_9 = f_9(V_6, V_7)$. These derived variables are computed by the dendrite that does not receive the target variable (see Equation 14); the dendrite that receives the target variable computes only an approximation of the derived variable.

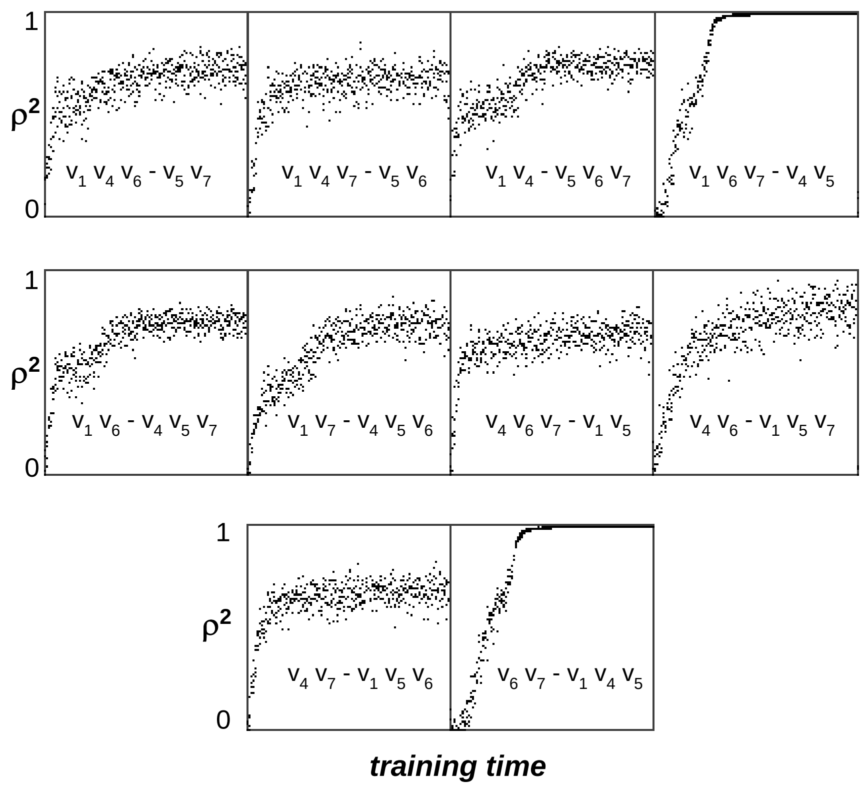

Figure 7: Extracting hidden factors from a Minimal Set of related variables. The analysis is performed on the Minimal Set $\{V_1\ V_4\ V_5\ V_6\ V_7\}$, identified in Figure 6 using V_5 as the target variable. This set of five variables can be partitioned in 10 different ways (with at least two variables on each side of a partition).

SINBAD testing of Minimal Set partitions identifies *candidates* for hidden factors. To make certain that these candidates are true hidden factors, they should be tested for their ability to substitute fully for the observed variables from which they were derived (Figure 8). If a candidate cannot predict the target variable as well as the variables from which it is computed, then this candidate should be discarded. As seen in Figure 8, hidden variables can greatly improve the speed of learning inferential relations, which makes them highly desirable to have in addition to the observed variables.

Hidden variables boost the rate of learning convergence by reducing the number of samples required for training, because they reduce the number of variables (dimensions) that are used in the prediction. Thus, hidden variables offer a means of dealing with the phenomenon known as "the curse of dimensionality"; i.e., the fact that characterizing a complex relation with many input variables requires exponentially greater numbers of training samples in the high-dimensional data space.

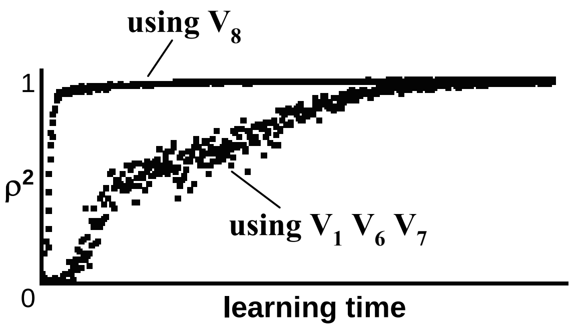

Figure 8: Testing a candidate hidden factor.

According to Figure 6, $V_5 = f_5(V_1, V_4, V_6, V_7)$. In Figure 7, this Minimal Set was successfully partitioned into $\{V_1\ V_6\ V_7\}$ vs. $\{V_4\ V_5\}$, suggesting that $V_5 = f(V_4, V_8)$, where V_8 is taken as the output of the SINBAD dendrite in Figure 7 with inputs from V_1, V_6, V_7. In other words, $V_8 = f_8(V_1, V_6, V_7)$ might be a true hidden factor. To verify this suggestion, a backprop net (dendrite 1 used in Figure 5) is trained on V_5, receiving its input either from V_1, V_4, V_6, V_7 or, alternatively, from V_4, V_8. The plot in Figure 8 shows the time-courses of learning under these two conditions, reaching maximal $\rho^2 = 0.999$. The plot reveals not only that V_5 prediction from V_8 is just as good as from V_1, V_6, V_7, but also that the net learns much faster with V_8, indicating that V_8 makes learning task much easier. Thus we conclude V_8 is a true hidden factor. Learning the inferential relation using V_8 is much easier than using V_1, V_6, and V_7; as in the latter case, the network is learning to solve a bigger problem that involves some type of implicit computation of V8 for the computation of V5. Also note that, unlike this simple illustration here, for more complex relations bigger networks will fail to produce as good approximations as the compact ones – higher input dimensionality will surrender to curse of dimensionality.

Hidden factors with effects on multiple observed variables are likely to be discovered again and again while searching different Minimal Sets of related variables. For this reason, hidden factors derived from different Minimal Sets should be compared with each other (for example, compute cross-correlations among all the discovered factors) to identify any clones. Overall, a single Minimal Set can yield multiple hidden factors and different Minimal Sets can yield the same hidden factor (Table 2).

Table 2

Hidden factors derived from all the Minimal Sets listed in Table 1.

Minimal Set	**Successful Partitions**	**Hidden Factor**
V1 V4 V5 V6 V7	V4 V5 – V1 V6 V7 V6 V7 – V1 V4 V5	V8 V9
V2 V3 V4 V5 V6 V7	V4 V5 – V2 V3 V6 V7 V6 V7 – V2 V3 V4 V5 V2 V3 – V4 V5 V6 V7	V8 V9 V10
V1 V2 V3 V6 V7	V6 V7 – V1 V2 V3 V2 V3 – V1 V6 V7	V9 V10

LEARNING ORDERLY RELATIONS

Hidden factors extracted from all the Minimal Sets extend, as derived variables, the list of variables with which we can characterize the studied dynamical system. The next task of theory building is to identify, quantify and express in a comprehensive way, the interdependencies among all the available – observed and derived – variables. To accomplish this goal, we need to identify the most direct relations among variables; these are the relations that involve minimal numbers of variables. Suppose variable V can be computed from variables X, Y, and Z; this relation would not be a most direct relation if variable V could be computed from X and W, where W is a hidden variable discovered as a function of Y and Z. That is to say that the *natural computation* of variable V requires having Y and Z integrated to compute W to be used in the computation of V. Direct relations do not contain these types of redundancies in them. Thus, such direct relations are the most practical ones with minimal representation for storage and minimal computation when used for inference.

Once such relations are known in sufficient numbers to form a more or less complete web, then interdependencies among all the available variables can be traced through chains of these direct relations.

In the process of identifying the Minimal Sets and the hidden factors, we have already learned a large number of relations, which computed one variable from a set of other variables. Together, they form the initial set of relations (Table 3). These relations give us a starting point for finding the most direct relations; i.e., relations involving minimal numbers of variables.

Table 3

List (compiled from all the relations listed in Tables 1 and 2) of all the ways by which each variable was learned, in the course of finding Minimal Sets and hidden factors, to be computed from other variables.

Relation #	**Computed Variable**	**Predictive Set**	ρ^2
1	1	4, 5, 6, 7	.998
2	2	3, 4, 5, 6, 7	.921
3	3	2, 4, 5, 6, 7	.941
4	4	1, 5, 6, 7	.994
5	5	1, 4, 6, 7	.999
6	6	1, 2, 3, 7	.995
7	7	1, 2, 3, 6	.999
8	8	4, 5	1.00
9	8	1, 6, 7	.999
10	8	2, 3, 6, 7	.998
11	9	6, 7	1.00
12	9	1, 4, 5	.999
13	9	1, 2, 3	.999
14	9	2, 3, 4, 5	.997
15	10	2, 3	1.00
16	10	4, 5, 6, 7	.998
17	10	1, 6, 7	.999

The basic idea is to substitute a group of variables in a known relation with a single variable. For example, we notice, in Table 3, that in relation #4, V_4 is computed from V_1, V_5, V_6, V_7, but that V_1, V_6, V_7 are also used to compute V_8 (relation #9). This observation raises the possibility that the role of V_1, V_6, V_7 in relation #4 is to compute V_8, suggesting that relation #4 can be simplified to that of $V_4 = f(V_5, V_8) + \varepsilon$. An alternative possibility, however, is suggested by relation #17: in this relation V_1, V_6, V_7 are used to compute V_{10}, not V_8. Therefore, a possible role of V_1, V_6, V_7 in relation #4 might instead be to compute V_{10}, allowing relation #4 to be simplified to that of $V_4 = f(V_5, V_{10}) + \varepsilon$. We can test these alternative hypotheses by training a backprop net on inputs either from V_5, V_8 or from V_5, V_{10}, with V_4 as the training signal. If the backprop net learns to perform on one of these inputs just as well as when learning on V_1, V_5, V_6, V_7, then we succeeded in simplifying relation #4 and finding a new and *more direct* relation. We will add this new relation to our list of known relations and look for other such opportunities.

The following algorithm automates the process of generating such substitutions and finding all the most direct relations. To give a definition of this recursive algorithm, Set I is a set of inferential relations of the type $S_i \rightarrow V_x$, where the state of variable V_x is inferred with a maximal degree of accuracy by performing an optimized computation (carried out by a backprop net) on the states of a set of variables S_i. At the start, Set I comprises all the relations learned by SINBAD dendrites in the course of finding Minimal Sets and hidden factors. This set is expanded by applying Rule 1 to it.

Rule 1: if I contains two relations $S_i \rightarrow V_x$ and $S_j \rightarrow V_y$ such that S_j is a subset of S_i, and if the prediction accuracy of V_x by $[\{V_y\} \cup (S_i - S_j)] \rightarrow V_x$ is not significantly worse than prediction accuracy of V_x by $S_i \rightarrow V_x$, then relation $[\{V_y\} \cup (S_i - S_j)] \rightarrow V_x$ is added to Set I.

Rule 1 is applied iteratively, expanding the size of Set I. The search for new relations continues until no new pair of relations satisfying the rule's conditions can be found anymore in I.

Next, among all the relations in I, those that were not simplified by Rule 1 at any time during the search are taken to be *the most direct relations*. They form a new set, D. A new rule, Rule 2, is applied iteratively to relations in Set D. This rule does not simplify relations; instead, it defines new relations by trading one variable in the already present relation for a different variable.

Rule 2: if D contains two relations $[\{V_x\} \cup S_i] \rightarrow V_y$ and $[\{V_y\} \cup S_j] \rightarrow V_z$, and if S_i is a subset of S_j or is the same as S_j, then relation $[\{V_x\} \cup S_j] \rightarrow V_z$ is added to Set D.

In the last part of the algorithm, when after its iterative applications, Rule 2 fails to find any new pairs of relations satisfying the rule's conditions, new relations are defined by reversing every relation in Set D and testing them for their predictive powers. The reason is that if we know a relation in which V_a and V_b successfully predict V_c, then we can expect that V_a can also be predicted, more or less accurately, from V_b and V_c, and so can V_b from V_a and V_c. The predictive accuracies of the reversed relations are evaluated by training backprop nets to implement them.

The product of this algorithm is a set of the most direct relations among the observed variables and the discovered factors, each relation executable by a backprop net. This is a very efficient searching algorithm. Applied, for example, to the set of 17 original relations in Table 3, it identified and tested on backprop nets 36 potential relations, 32 of which were accepted. The algorithm found 21 most direct relations, listed in Table 4. In comparison, if we searched for the most direct relations by testing, using backprop net training, all the possible relations among the 10 available variables, we would have to perform 5020 such tests.

Table 4

The most direct inferential relations among the seven observed and the three derived (hidden) variables, listing the sets of variables that are predictive of each of the 10 variables.

Relation #	**Computed Variable**	**Most Direct Predictive Set**
1	1	8, 9
2	1	8, 10
3	1	9, 10
4	2	3, 10
5	3	2, 10
6	4	5, 8
7	5	4, 8
8	6	7, 9
9	7	6, 9
10	8	4, 5
11	8	1, 9
12	8	9, 10
13	8	1, 10
14	9	6, 7
15	9	1, 8
16	9	1, 10
17	9	8, 10
18	10	2, 3
19	10	1, 9
20	10	8, 9
21	10	1, 8

FORMULATING A THEORY OF THE STUDIED SUBJECT

The set of direct relations that Virtual Scientist procedures extract from the observed variables describe most explicitly the order discernable in the observed dynamical system (given that the only source of information about the system are the observed variables). For a full appreciation of the interdependencies among the variables, this set of relations should be considered as a unified graph, rather than as a disjointed list of separate relations. Different relations are linked by variables they have in common (such as, for example, when the same variable is predicted by one relation and is used for prediction by another relation). As a result of such overlaps, all the different direct relations are linked together into a single functional entity, a *web of inferential relations* (Figure 9).

In Figure 9, all 21 direct relations listed in Table 4 are shown by lines connecting the variables. Note that the lines do not connect variables directly, but go through *hubs*, which tie several lines together. The presence of a hub indicates that a given variable does not have predictive significance for another variable just by itself, but in conjunction with one or more other variables joining it in the hub. Thus, each hub identifies a set of variables engaged together in a multivariate, typically nonlinear relation.

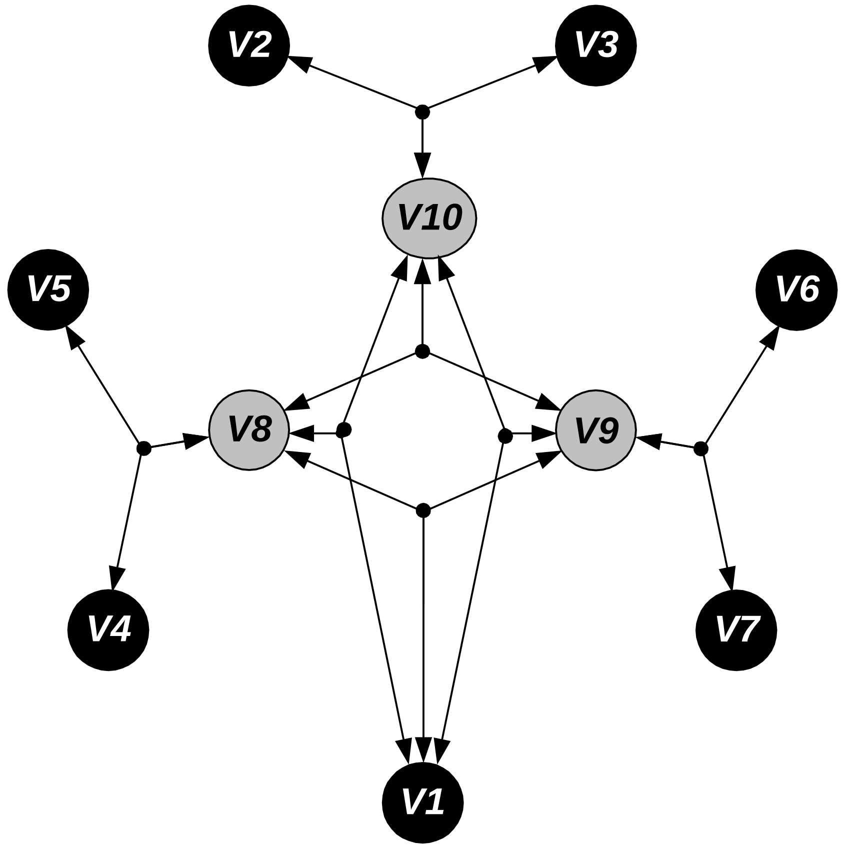

Figure 9: The web of connections among the observed variables and hidden factors discovered by Virtual Scientist in the studied dynamical system. The plot shows the seven observed variables (solid circles) and the three derived variables (shaded circles).

An arrow emerging from a hub and pointing at a variable indicates that the state of that variable can be predicted with a significant degree of accuracy (in this plot, all have $\rho^2 > 0.9$) from a combination of the states of the other variables linked by the hub. Note also that multiple arrows converging on a variable indicate alternative sources of prediction and should *not* be viewed as additive in their effects. Several of the variables in the plot are engaged in more than one relationship and, therefore, can alternatively be predicted from more than one set of variables.

Viewed as a whole, this diagram of variables linked together by the web of connections provides a graphic representation of the order discovered by Virtual Scientist procedures in the studied dynamical system. By virtue of its power to unravel and explain behaviors of the variables in terms of other variables, this web of inferential relations offers a concise and comprehensive *theory* of the studied subject.

The web of inferential relations reveals all the pathways by which even remote interactions among distant variables can be traced through chains of variables connecting them. Furthermore, the usefulness of the web of relations extends beyond mapping functional interconnections among the variables. Each direct relation in the web has been learned – in the process of its discovery – by a separate backprop net and can, therefore, be used to make predictions. This type of insightful inference is very crucial especially in case of missing information. Apart from Virtual Scientist, most learning algorithms are based on mapping of a set of input channels to some output. These algorithms do not evaluate the significance of the input channels and which ones are truly necessary for a specific mapping task. There are feature selection algorithms proposed for preprocessing tasks. However, these approaches are of limited contribution for building inferential model of the observed system as their use are limited to an initial, one-time-only feature selection.

Therefore, given partial input, other approaches will fail to make valid inferences because their mapping, in general, requires that all variables to be supplied into the black-box mapper. The reason I use the term "black-box" is that looking from outside we will have no idea how this mapping is done and how does the computation merge (integrate) different variables for the ultimate computation. Another disadvantage of the black-box approach is that given partial input may be sufficient or insufficient for the computations but a black-box approach will fail to recognize that they failed. However, using the method presented in this work, we can determine, following the inferential chains, which target variables we can correctly infer. In this respect, Virtual Scientist can be thought as a self-aware system – it knows that it is designed for making the most reliable inferences.

In addition, with a direct relation involving, typically, only a small number of variables, such a low-dimensional relation can be visualized by plotting the involved variables against each other, so that the underlying mathematical form of this relation can then be appreciated. Finally, standard least-square approximation methods can be used to fit the data distribution with a suitable approximating function, thus expressing the relation mathematically. Fitting approximating functions to all the direct relations in the web, it might be possible to arrive at a mathematical description of the studied dynamical system in a form of a system of equations.

Whether expressed by formulae or by backprop nets, the direct relations together make up a quantitative model of the studied subject, which is the kind of *theory* that Virtual Scientist procedures extract from the observational data. Among the central contributions of such a theory are (i) derivation of explanatorily useful concepts of influential hidden factors, (ii) explanation of the behaviors of the observed variables from behaviors of other such variables and the conceptualized hidden factors, and (iii) elucidation of the functional organization of the studied dynamical system.

The web of inferential relations shown in Figure 9 is an example of such a theory, developed by Virtual Scientist procedures from the observational data described in Figure 1. These data were generated by a mathematical model of a well-known dynamical system, the

Kitchen Sink. Sinks encountered at various times and in different kitchens can be viewed as a single dynamical device – the Sink – that varies its configuration (i.e., how pipes and valves are arranged) in different kitchens and varies its state (i.e., how much water is flowing and its temperature) at different times. To simplify matters, the mathematical model describes sinks that all receive water from two pipes and are controlled by the following five variables: *HC* indicates which of the two pipes carries hot/cold water (0 – the left pipe is hot and the right pipe is cold, 1 – vice versa), DIR_L and DIR_R are the directions in which the left and the right knobs should be turned to open the pipes (0 – clockwise, 1 – counterclockwise), and KP_L and KP_R are the radial positions of the two knobs. Sinks in different kitchens can have different DIR_L, DIR_R, and *HC*; KP_L and KP_R can vary in the same sink.

These variables determine the flows of water through the two pipes:

$$F_L = (1 - DIR_L - KP_L + 2 \cdot DIR_L \cdot KP_L)/2 \quad \text{and} \qquad (15)$$

$$F_R = (1 - DIR_R - KP_R + 2 \cdot DIR_R \cdot KP_R)/2 \,. \qquad (16)$$

In turn, the flows of water through the two pipes determine the total water outflow from the faucet and its temperature:

$$F_T = F_L + F_R \quad \text{and} \qquad (17)$$

$$t^\circ = ((1 - HC) \cdot F_L + HC \cdot F_R)/(F_L + F_R) \,. \qquad (18)$$

Collection of observational data was envisioned to involve observing a random sequence to “snapshots” of sinks in various configurations of pipes and valves, and with various knob positions and the resulting water outputs. The observed variables $V_1 \ldots V_7$, analyzed by Virtual Scientist procedures, had the following identities:

$V_1 = F_T$, $V_2 = t°$, $V_3 = HC$, $V_4 = DIR_L$, $V_5 = KP_L$, $V_6 = DIR_R$, $V_7 = KP_R$.

Importantly, two sink variables of central significance for sink functional organization were not observed, turning them into *hidden factors*. These variables are F_L and F_R, the flows of water in the left and right pipes, respectively. Although hidden, these factors were, nevertheless, discovered by SINBAD cells and labeled as V_8 ($=F_L$) and V_9 ($=F_R$). Interestingly, SINBAD cells also discovered an additional, unanticipated, hidden factor, V_{10}. My analysis of its behavior reveals that V_{10} is a ratio: $V_{10} = F_L/F_T$. Upon some consideration, V_{10} is, in fact, an important factor: it determines, together with *HC*, the water temperature, *t°* (see Equation 18).

Following the discoveries of hidden factors F_L, F_R, and V_{10}, Virtual Scientist procedures learned how F_L and F_R are determined by knob positions ($V_5 = KP_L$, $V_7 = KP_R$) and directions to open the valves ($V_4 = DIR_L$, $V_6 = DIR_R$), and how F_L and F_R, in turn, determine the total water outflow F_T and its temperature *t°* (see Figure 9). Overall, the web of inferential relations in Figure 9, generated by the Virtual Scientist, does reflect accurately and efficiently the functional organization of kitchen sinks (actually, to be precise, the organization of the designated mathematical model of sinks). The web is an inferential model of sinks and, as such, it can be used to predict, for example, how knobs should be positioned in a given sink in order to produce desired water flow and temperature, or to deduce the knob positions from the known faucet output, or to perform any other prediction that can be performed using Equations 15-18.

In another demonstration of the close correspondence between the true mathematical description of the studied dynamical system (i.e., Equations 15-18) and the Virtual Scientist model of it, the *hubs* in Figure 9 representation of the direct relations actually correspond to the equations that govern the system. Specifically, the hub linking V_4, V_5, V_8 represents Equation 15, the hub linking V_6, V_7, V_9 represents Equation 16, the hub linking V_1, V_8, V_9 represents Equation 17, while two hubs linking V_2, V_3, V_8, V_9 represent Equation 18.

Thus, in conclusion, Virtual Scientist procedures were fully successful in reconstructing from the observed data the prominently nonlinear mathematical model that generated those data.

SVMS FOR VIRTUAL SCIENTIST PROCEDURES

Support Vector Machines (SVMs) (Vapnik 1995, 1998) have been shown to be very powerful tools for both linear and nonlinear classification and regression problems in the field of machine learning. Both the attractiveness of their underlying theory, due to its intuitiveness and simplicity, and their success in solving difficult pattern recognition problems have resulted in a fast growing interest in SVMs. Among the problems SVMs have performed well, text-categorization, face detection and recognition, and gene selection may be listed.

SVMs' origin traces back to 1909 when James Mercer developed the theory of Kernel, which is one of the fundamental concepts of SVMs. Kernel Hilbert space, which is a subfield of Hilbert Space theory was developed by Aronszajn in 1940s. In 1964, Aizerman, Braverman and Rozoner introduced the concept of interpreting kernels as inner products in a feature space, using Mercer's theorem. In 1979, Vladimir Vapnik et al finally developed the theoretical foundation for SVMs based on previous works. SVMs are now as widely applied and studied as neural networks. There are several major advantages of SVMs over neural networks. These advantages are of significant interest for most efficient implementations of the Virtual Scientist procedures that are originally put into practice using neural networks for the SINBAD cell. As the advantages of SVMs over neural networks have become well established; I would, naturally, like to take advantage of SVMs as learning modules in SINBAD architecture.

Advantages of SVMs over Neural Networks

In this section, I will provide a brief look at the underlying principles of SVM learning to appreciate its capabilities and to motivate its use in Virtual Scientist procedures. The most striking property of SVM learning that has been widely drawn attention to in the literature is that they have better generalization ability, and thus, less likelihood of over-fitting. It has been also shown that SVMs are robust to high input dimensionality and they require less training samples. Moreover, SVM learning involves smaller number of hyper-parameters that needs to be experimented for a given learning task; in this regard, there are techniques proposed for automated hyper-parameter optimization for SVMs.

For the sake of simplicity, I will talk about SVMs for classification problems in supervised learning context. The principles can be easily generalized to the case of regression with SVMs. In a classification problem, we are given a set of l training examples $\{x_i,y_i\}$, where x_i is the input vector and y_i is the output (class code), defined by the classification function $f(x_i)=y_i$. The task of learning is to find a "good" hypothesis function h(x) that approximates f(x). In classification problems, it can be assumed that y only takes values of ±1. Since we cannot try all possible hypothesis functions, it is easier to restrict h(x) to lie inside a hypothesis space H, such as polynomial functions. Even then, however, it is possible to find several hypotheses that classify the training set correctly. To avoid over-fitting it is necessary to choose the simplest one – a principle known as Occam's razor. Based on this principle, Vapnik-Chervonenkis statistical learning theory (VC) (Vapnik 1995, 1998) shows that it is essential to restrict the class of functions that f is chosen such that its capacity is suitable for the amount of available training data. Without such a restriction, a function that performs well on the training set need not generalize well to unseen examples (test set). Hence, only minimizing the training error (empirical risk) does not imply a small test error (risk). Neural networks, for example, are based on empirical risk minimization, thus, well known to over-fit the training data and fail on test data as a result.

In order to minimize risk, it is necessary to find the optimal combination of both the empirical risk and the capacity of the function class. This principle is known as structural risk minimization. The best-known capacity concept of VC theory is the VC dimension, defined as the largest number of points that can be separated in all possible ways using functions of the given class. In other words, VC dimension is equal to the maximal number d of training examples that can be split into two sets in all 2^d ways using functions from the hypothesis space.

In order to restrict the expressiveness of the hypothesis space, the SVM searches for the simplest solution that classifies the data correctly. This is equivalent to maximising the distance, normal to the hyperplane, between the convex hulls of the two classes; this distance is called the margin. Note that among all hyperplanes separating the data, there exists a unique one yielding the maximum margin of separation between the classes; and the capacity of the hypothesis decreases with increasing margin.

A linear classifier may not be the most suitable hypothesis for the two classes. The SVM can be used to learn non-linear decision functions by first mapping the data to some higher dimensional *feature space* and constructing a separating hyperplane in this space. Thus, SVM approach to learning is based on taking advantage of pair-wise similarity of training samples that is measured by a kernel function. As a result, high input dimensionality is not a limiting factor for SVM learning. On the other hand, for neural networks, to accommodate a large input vector the size of the input and the hidden layer should be extended, which results in a big and powerful network that is able to over-fit. Even though, keeping the number of hidden units minimal and applying weight decay techniques have been proposed, the bigger the network is the search becomes much more difficult and involves spurious local minima.

SVMs, on the other hand, avoid over-fitting by choosing a specific hyperplane among the many that can separate the data in the feature space. This hyperplane is called the *maximum margin hyperplane*, which maximizes the minimum distance from the hyperplane to the closest training point. The maximum margin hyperplane can be represented as a linear combination of training points. Support vector machines can locate a separating hyperplane in the feature space and classify points in that space without ever representing the space explicitly, simply by defining a function, called a *kernel function*, which avoids the computational burden of explicitly representing the feature vectors. The optimisation is now a convex quadratic programming (QP) problem

$$\begin{aligned}&\underset{\mathbf{w},b}{\text{Minimize}}\ \Phi(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2\\ &\text{subject to } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, i = 1,\ldots,l.\end{aligned} \qquad (19)$$

This problem has a global optimum; thus the problem of many local optima in the case of training e.g. a neural network is avoided. This has the advantage that parameters in a QP solver will affect only the training time, and not the quality of the solution.

Thus, the SVM learns the optimal separating hyperplane in some feature space, subject to ignoring certain points which become training misclassifications. The learnt hyperplane is an expansion on a subset of the training data known as the support vectors. By use of an appropriate kernel function the SVM can learn a wide range of classifiers including a large set of RBF networks and neural networks. The flexibility of the kernels does not lead to over-fitting since the space of hyperplanes separating the data with large margin has much lower capacity than the space of all implementable hyperplanes.

The selection of an appropriate kernel function is important, since the kernel function defines the feature space in which the training set examples will be classified. As long as the kernel function is legitimate, an SVM will operate correctly even if the designer does not know exactly what features of the training data are being used in the kernel-induced feature space. The definition of a legitimate kernel function is given by Mercer's theorem (Vapnik, 1998): the function must be continuous and positive definite. Human experts often find it easier to specify a kernel function than to specify explicitly the training set features that should be used by the classifier. The kernel expresses prior knowledge about the phenomenon being modeled, encoded as a similarity measure between two vectors.

In experimental sciences, it is very common case that the number of training samples is small, whereas the number of variables measured is numerous. Therefore, for many neural network based approaches, including the original SINBAD design utilizing the backprop nets, it would be inescapable to over-fit complex functions on the training data. In summary, based on the structural risk minimization principle from statistical learning theory, SVMs provide maybe the most intuitive way of learning from examples, simply corresponding to a linear method in a high dimensional feature space that is nonlinearly related to the input space through the use of kernels. SVMs, thus, scale well to very many inputs and avoid the computations in the high dimensional nonlinear feature spaces and keep all the computations performed in the input space. SVMs perform very well, without over-fitting, on a set of limited number of training samples. However, it should be kept in mind that it has been proven that neural networks can approximate any arbitrarily complex function and therefore they may be helpful when the SVM learning does not seem to work under certain conditions.

Novelty of SVMs in Unsupervised Learning

While some researchers show the practicality of this promising learning tool in application to real-life problems, many others have put efforts in different versions of SVMs for better and/or faster generalization ability. There have been quite a number of versions of SVMs and some issues regarding to online versus batch learning, variable selection, and tuning hyperparameters are explored. However, most of these efforts relate to supervised learning (Vapnik 1995). There are a few interesting approaches for unsupervised (nonlinear) factor analysis with SVMs, such as kernel-PCA. However, these methods are closely linked with comparable neural network based approaches, for example, kernel-PCA performs nonlinear principle components of Kramer (1991). Thus, it would not be far off to say none is proposed for SVMs use in unsupervised learning.

On the contrary, neural networks' intuitive applicability to unsupervised learning by auto-association has lead many researchers in the field of (nonlinear) factor analysis to search for the ways of forcing the hidden units to tune to true hidden (causal) factors. Using weight decay during learning or applying independent component analysis (ICA) on the functions learned by the hidden units can be given as examples of methods applied in auto-associative neural networks in the hope of discovering interesting factors.

As a severe consequence of the fact that, for quite long time, the task of unsupervised learning has been considered to be transforming high dimensional data into a low dimensional nonlinear space such that empirical risk is minimized, many traditional approaches in unsupervised learning are built on data compression. However, as discussed earlier, the compactness of a representation does not guarantee that the descriptors will turn out to correspond to true, *inferentially useful* hidden factors.

Even though, SVMs have come out with great advantages over neural network models, unfortunately, they do not provide as obvious means of compression as, for example, many varieties of bottleneck-layered neural networks do. Hence, we do not see SVMs being used for unsupervised learning. It is evident that in the field of machine learning and factor analysis, the approach to unsupervised learning is merely about data compression. Fortunately, a fundamentally different approach to unsupervised learning – based on the principle of maximization of mutual information (Imax) among non-overlapping sources of information – can be readily implemented using SVMs.

Using this approach, just as before with error backpropagation networks in SINBAD implementation, hidden factors can be discovered through a search for different, but nevertheless highly correlated functions of any kind over disjoint subsets of the observed variables. Such correlated functions must have a reason for their statistical interdependence, a causal source in the domain of the data. Thus, a pair of SVMs, set up to use each other's output as their own training signal, can teach each other higher-order features of the data that reveal hidden causal factors controlling the observed phenomena. Hidden variables discovered by this method form a graph that reveals the functional organization of the studied subject.

Virtual Scientist Procedures with SVMs

I used a simple implementation of SVM regressor (approximator). This SVM regressor works in batch mode. Other versions of SVMs for regression, as well as any other supervised learning algorithms, can be used instead. Given below is the pseudo-code algorithm for a single SINBAD cell using two SVMs by simply calling "Train SVM" to run the regressor. Auto-scale is the technical name of normalization to zero-mean and unit standard deviation. Correlation coefficient, ρ, between two signals x_i and y_i is a well-known measure of how highly two signals correlate, which is computed as follows:

$$\rho = \frac{N \cdot \sum_i (x_i \cdot y_i) - \left(\sum_i x_i \cdot \sum_i y_i \right)}{\sqrt{\left(N \cdot \sum_i x_i^2 - \left(\sum_i x_i \right)^2 \right) \cdot \left(N \cdot \sum_i y_i^2 - \left(\sum_i y_i \right)^2 \right)}}, \qquad (20)$$

where N stands for the number of observations.

Pseudo-code Algorithm of SINBAD cell training:

```
Randomly initialize Output-1($x_i$)
Randomly initialize Output-2($x_i$)
Repeat
    Auto-scale Output-1
    Auto-scale Output-2
    Train SVM-1 to regress Output-2
    Train SVM-2 to regress Output-1
    Re-compute Output-1 from SVM-1
    Re-compute Output-2 from SVM-2
    Compute correlation coefficient $\rho_d$ between Output-1 and Output-2
Until $\rho_d$ does not improve (convergence)
```

End SINBAD cell training

The update equations of the original version of the backprop-version of SINBAD were very complex including some artificial factors, α, β, mainly to force the cell to expand the dynamic range of the learnt features. This force had some side effects such as forcing the cell to be always active (highly activated), which is not realistic as the preferred hidden variable may simply have near zero values at times. Moreover, compared to SVM-SINBAD, Backprop-SINBAD has more hyper-parameters to optimize such as initial weights, learning rate, number of hidden units etc.

Pseudo-code Algorithm of Hidden Variable Discovery:

```
Choose a target variable T among the observed variables
SVM-1 is given all observed variables except T
SVM-2 is given only T
Train SINBAD cell using SVM-1 against SVM-2
Compute Output-1 from SVM-1
Compute Output-2 from SVM-2
Measure the coefficient of correlation ($\rho$) between Output-1 and Output-2
For each variable V connected to SVM-1
    Remove V from SVM-1
    Train SVM-1 to regress Output-2
    Re-compute Output-1 from SVM-1
    Measure $\rho_{new}$ between Output-1 and Output-2
    If $\rho_{new} << \rho$ Then Connect V back to SVM-1
End For
Minimal Set $\leftarrow$ {variables connected to SVM-1} $\cup$ {T}
For each bi-partition of Minimal Set
    Connect variables in one partition to SVM-1
    Connect variables in the other partition to SVM-2
    Train SINBAD cell using SVM-1 against SVM-2
    Measure $\rho_{candidate}$
    If $\rho_{candidate} \approx \rho$ Then Functions computed by SVMs are hidden variables
End For
```
End Hidden Variable Discovery

For a first comparison of the capabilities of the two SINBAD implementations (SVM-SINBAD and Backprop-SINBAD), both versions are applied to the dataset of four images presented earlier, this time using a different hidden image for the sake of variety – i.e. not to repeat the same image display over and over again.

For finding minimal sets, SVMs' use can be very effective as they are robust to high input dimensionality. That is because finding minimal set of related variables involves predicting a target variable from all the other variables. It is very likely for the neural networks to stuck in a local minima or over-fit to the training data, especially given small number of training examples. For an illustration, consider some truly random variables (called r1, r2, r3, etc...) are among the observed variables in our demonstrative dataset of four images. In reality, these random variables can as well be some meaningful variables that are irrelevant to the target variable we would like to predict. In their effect, these irrelevant variables make the learning task harder as shown in Figure 10, where the number of samples required for accurate prediction ($\rho^2 > 0.9$) of the target variable IM1 is shown for both Backprop-SINBAD and SVM-SINBAD and plotted as a function of the number of irrelevant variables.

To further explicate the capabilities of the learning modules, I limited the number of training samples to 100. When SVM-SINBAD is given {IM1, IM2} versus {IM3, IM4} (Figure 11) connected to its learning modules SVM-1 and SVM-2, respectively, it converges very quickly within 10 updates (10 complete passes over the training set) and learns the underlying image 5 perfectly from (compare Figures 12 and 13). Whereas, in Backprop-SINBAD, backprop-1 and backprop-2 take more than 500 updates to accomplish only a close approximation to image 5 (Figure 14).

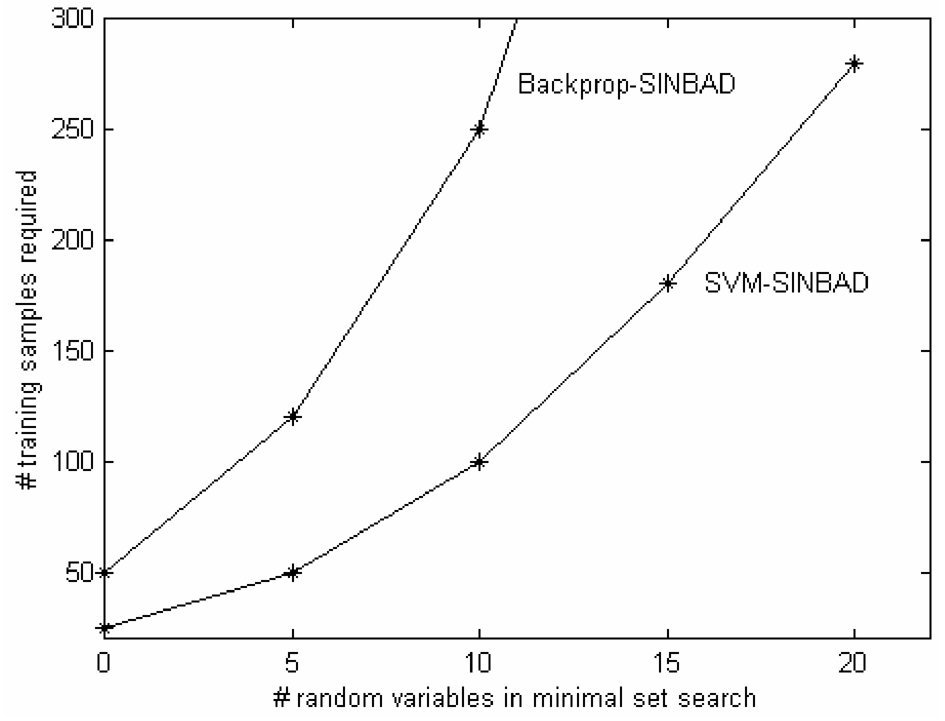

Figure 10: The number of training samples required for predicting IM1 with $\rho^2>0.9$ from other observed variables as a function of the number of irrelevant (random) observed variables.

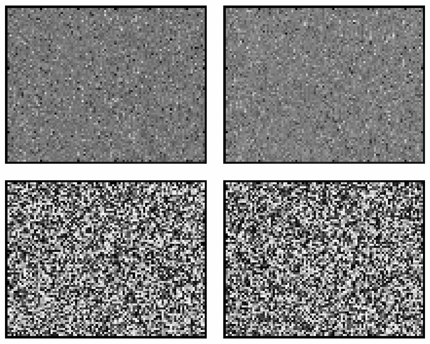

Figure 11: The four images which characterize a hidden image shown in Figure 12.

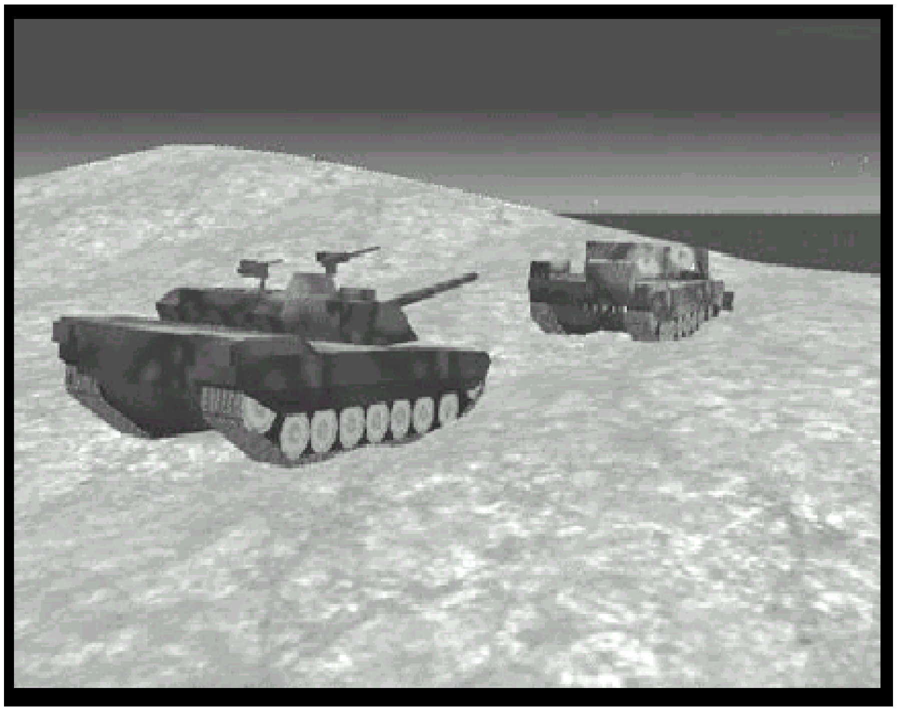

Figure 12: The hidden image: image 5.

Figure 15 shows the mean-squared-error (MSE) of both approaches as a function of training updates. Note that Backprop-SVM could not perfectly converge to the hidden image 5, which is clearly seen in its output image. On the other hand, SVM-SINBAD learned the hidden variable perfectly, see Figure 16 for a comparison of these approximations.

Even though a parallel algorithm can take the same time-complexity to accomplish minimal set search exhaustively, the proposed algorithm is heuristic, saving space complexity and total amount of computation needed. However, it is inescapable for a heuristic algorithm to make errors. Thus, it is probable that some unrelated variables will be let in the minimal sets. These variables will be present in the partitions to be tested for hidden variables. However, presence of unrelated variables in partitions will only confuse the learning modules. Recall that neural networks are more prone to get stuck in a local minimum in high dimensional input space.

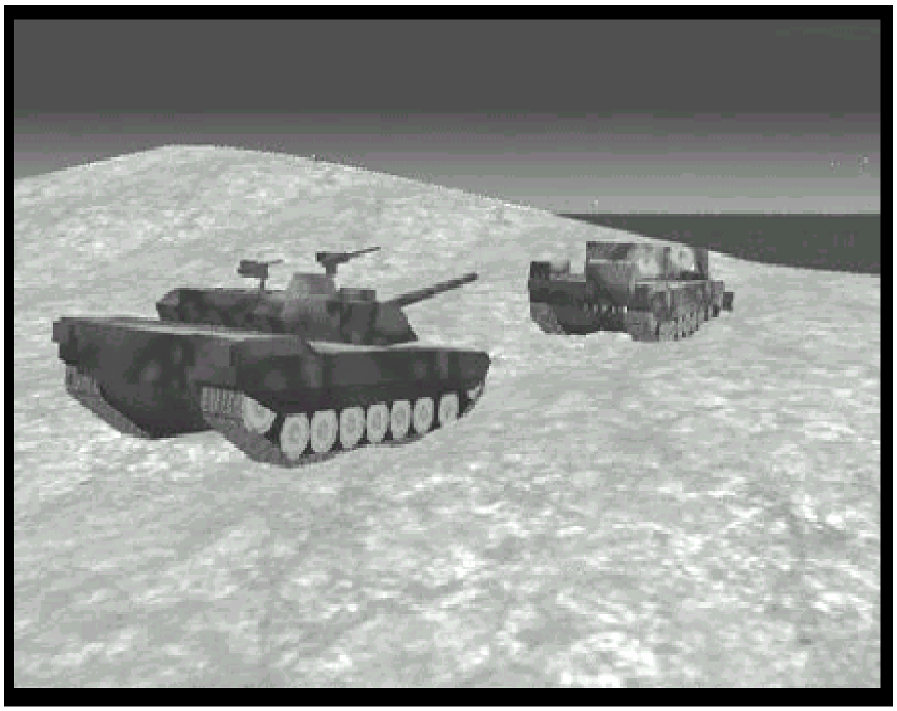
Figure 13: The approximation of SVM-SINBAD to the hidden image 5.

For an illustration, consider some of the random variables (r1, r2, r3, etc...) are, accidentally, in our minimal set {IM1, IM2, IM3, IM4} to be partitioned. For the sake of simplicity, I will consider partitions with equal number of random variables in both partitions, such as {IM1, IM2, r1} versus {IM3, IM4, r2}. Applied on the various training set of 100 examples (pixels) from four images, if these partitions are given to Backprop-SINBAD, it fails to find the hidden image 5. However, SVM-SINBAD could handle up to six random variables in each partition.

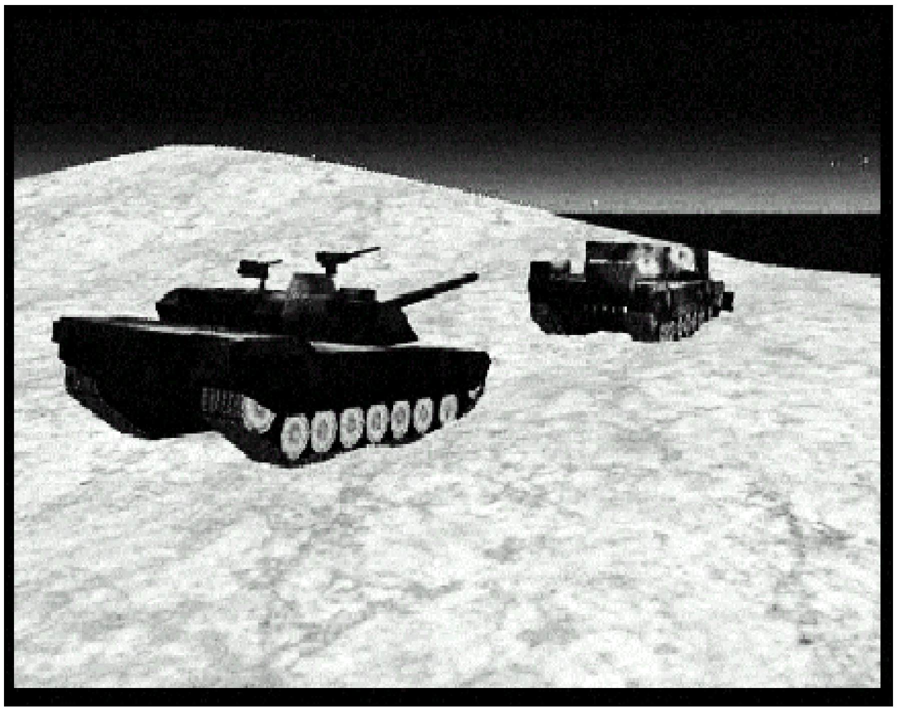
Figure 14: Backprop-SINBAD discovers only a close approximation to the hidden image 5.

Finally, on the kitchen sink example presented in previous sections, SVM-SINBAD performed very well and the relations learnt with this new implementation exactly matched the relations learnt with Backprop-SINBAD. However, as expected, the accuracies of the learnt relations show that SVMs outperform backprops, once again, for SINBAD learning (Table 5).

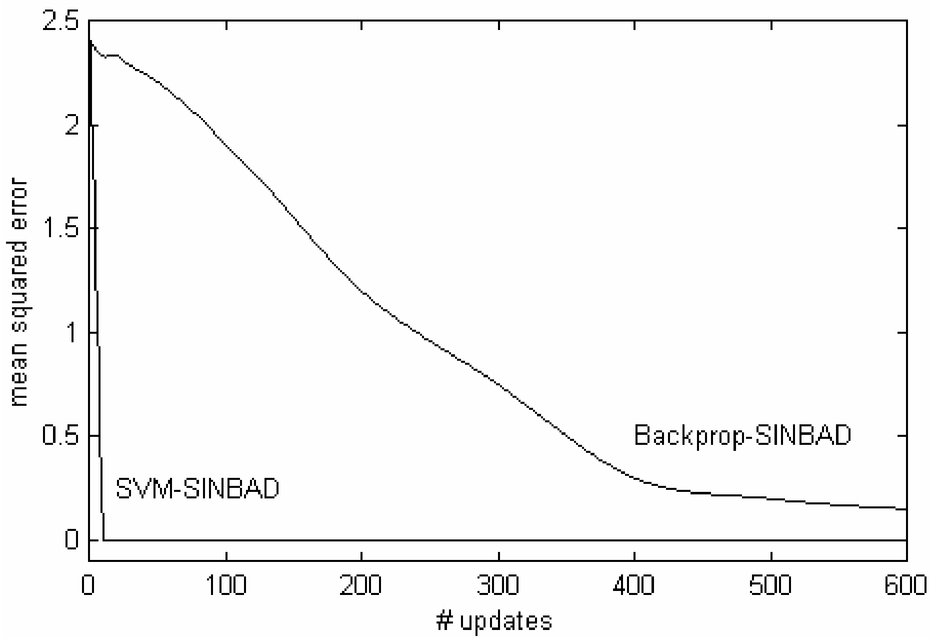

Figure 15: The mean-squared-error of SVM-SINBAD and Backprop-SINBAD on the test image as a function of training updates.

Comparison to my previous work demonstrated in previous sections, in which this approach was implemented with backprop nets, shows that unsupervised learning with SVMs is, in fact, more robust to high input dimensionality and requires much smaller number of training samples and iterations. This improvement is promising for practical applications of Virtual Scientist to real-world problems especially experimental sciences such as bioinformatics, where the number of training samples is small but the number of observed variables is plentiful.

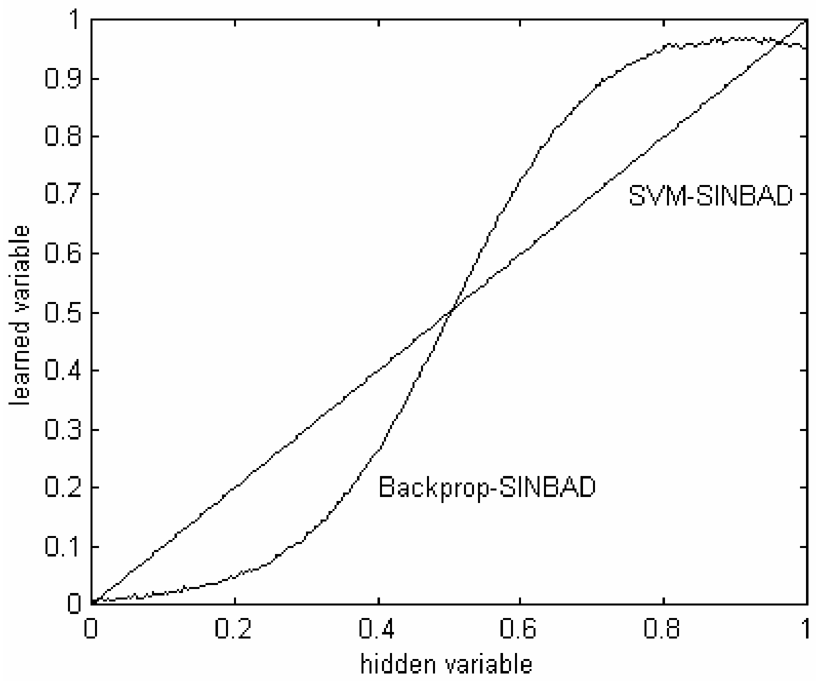

Figure 16: Approximations of SVM-SINBAD and Backprop-SINBAD to the hidden image 5.

Table 5

Comparison of SINBAD implementations on the most direct inferential relations among the seven observed variables and the three derived variables. Different implementations of SINBAD, in this specific problem, did not change the learnt relations but their accuracies differ. The relations are extracted more accurately with SVM-SINBAD compared to the original SINBAD implemented with backprop nets (Backprop-SINBAD).

Computed Variable	**Backprop-SINBAD**		**SVM-SINBAD**	
	Predictive Set	ρ^2	**Predictive Set**	ρ^2
V1	V8 V9	.998	V8 V9	.998
V2	V3 V10	.966	V3 V10	.975
V3	V2 V10	.868	V2 V10	.947
V4	V5 V8	.947	V5 V8	.988
V5	V4 V8	.997	V4 V8	.998
V6	V7 V9	.949	V7 V9	.987
V7	V6 V9	.997	V6 V9	.998
average	*2.00 inputs*	*.960*	*2.00 inputs*	*.984*

VIRTUAL SCIENTIST APPLICATION TO METABOLOMICS

What is Metabolomics?

Virtual Scientist has been developed for use on scientific problems that involve systems of nonlinear multivariable relations, and in which data are collected in snapshot observations of the state of the studied subject. This kind of problems is especially characteristic of genomics and related fields. One of these related fields is Metabolomics (Nicholson et al. 2002). It concerns metabolic – chemical – reactions taking place in the cells of the various tissues and organs of a living organism. These reactions involve *metabolites* (small molecular size chemical compounds), their concentrations, and rates of conversion of metabolites into other metabolites:

$$M_a + M_b \underset{r_{-1}}{\overset{r_1}{\Leftrightarrow}} M_c + M_d,$$

where M_a, M_b, M_c, M_d are different metabolites and r_1 and r_{-1} are rates of forward and backward chemical reactions, typically controlled by specific protein enzymes.

One of the basic experimental techniques used in Metabolomics is Nuclear Magnetic Resonance, or NMR (Reo 2002, Griffin 2003). NMR can be applied to biofluids extracted from specific cells, tissues, organs, or it can be applied to blood plasma or urine, with the latter being particularly popular because of noninvasiveness of sample collection. NMR measures concentrations of various metabolites in the studied biofluids, which reflect chemical reactions taking place in the body.

Thus, a single NMR measurement of a biofluids sample produces information about hundreds metabolites present in the sample. NMR spectral data are used for diagnostic purposes – to detect various abnormalities in the functioning of the organism due to toxins, infections, or other causes. NMR spectral data are also used in elucidation of metabolic pathways, interactions, processes of basic scientific interest for understanding physiological functioning of the body under different conditions.

Current Analytical Methods

NMR spectra are conventionally analyzed using methods of principal component analysis (PCA), linear discriminant analysis (LDA), partial least squares-discriminant analysis (PLS-DA), soft independent modeling of class analogies (SIMCA), among others (Otto 1999). The standard approach is to do PCA first to reduce dimensionality of the data. Graphic display of the first Principal Component plotted against the second Principal Component can reveal a tendency of data to cluster in two or more different regions of the plot, indicative of the presence of multiple metabolically distinct classes ("phenotypes") among the studied subjects. If NMR analysis is performed to compare two or more different groups of subjects, or phenotypes, such as for example normal subjects vs. those with a particular disease, then supervised learning technique of LDA is used to identify the best linear discrimination of phenotypes on the basis of their NMR spectra.

Analysis of loading parameters, a measure of involvement of metabolites in the PCA dimensions, is used to identify those metabolites that are most involved in a given phenotype. These metabolites are *biomarkers*; the knowledge of their identities is highly useful for understanding the underlying nature of a given phenotype (e.g., the causes or consequences of a given pathology) and for diagnosis purposes.

While the currently used analytical approaches have yielded many important findings, their fundamental limitation is that they are *linear* and therefore incapable of taking advantage of richer, nonlinear regularities in the NMR data. That is, these methods do not utilize NMR method effectively; they do not make full use of information carried by NMR spectra. To extract and make use of such information requires *nonlinear* analytical techniques.

Analysis of NMR Spectra by *Virtual Scientist*

Virtual Scientist is well suited for extracting deep information from NMR spectra and finding connections among metabolites in normal and pathological states. To illustrate how *Virtual Scientist* can be applied to NMR-based Metabolomics, we will make use of data collected in an experimental study by Dr. Jeffrey Macdonald at University of North Carolina in Chapel Hill.

To give a brief description of the data and its preprocessing, urine samples were collected from normal healthy mice (controls with liver score = 0; a total of 17 samples in this category), as well as from mice with varying stages of liver cancer (liver score ranging from 1 to 3 where a score of 0 means no cancer and liver score = 3 is max; a total of 8 samples in this category). The urine samples were analyzed on a 600 MHz INOVA NMR spectrometer. The raw measurements were converted into frequency domain using Fourier transform, thus producing an NMR spectrum for each subject. An example of a healthy phenotype spectrum and a cancerous phenotype spectrum are shown in Figure 17.

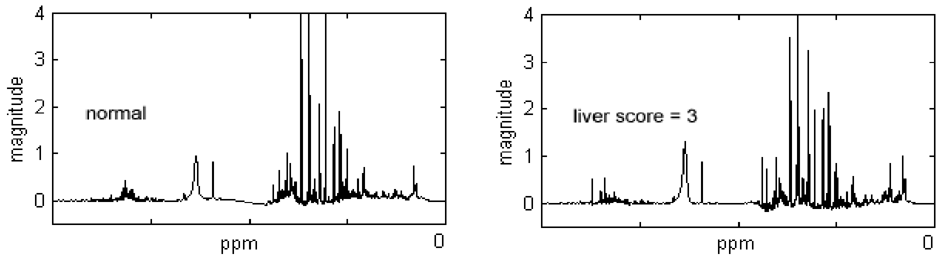

Figure 17: Exemplary NMR spectra.

To compensate for translational jitter across different NMR readings, all spectra were aligned on the rightmost spike, generated by TSP chemical compound. To remove such global factors of no physiological significance as NMR spectrometer calibration variations and urine concentration variations across animals, the spectra were normalized by their total integrated area.

The spectra were found to contain 221 spikes of magnitudes significantly above the background (Figure 18). Each such spike reflects the presence in the urine of one of the metabolites. Metabolic identities of some of the spikes are known, others are not. The magnitude of a spike is proportional to the metabolite's concentration in the urine sample.

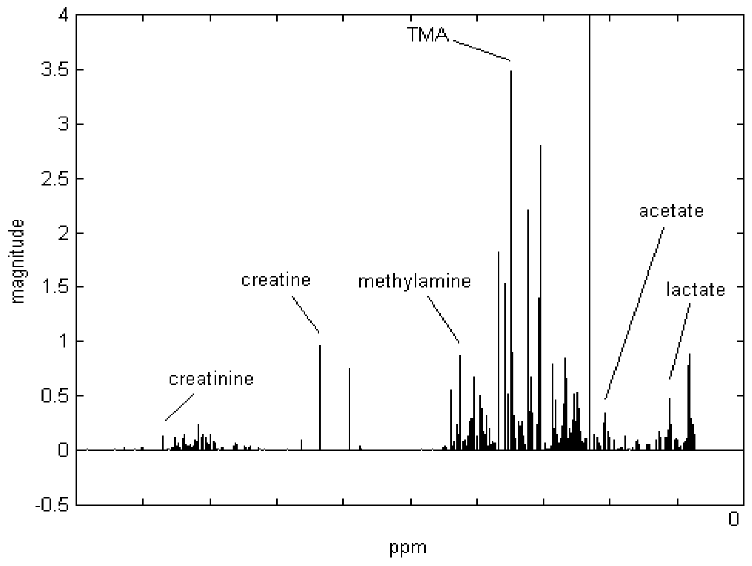

Figure 18: Plot of 221 spikes from one of the NMR spectra, with names of some of spikes marked on the plot.

The standard analysis of NMR spectra involves PCA for dimension reduction (Figure 19). Figure 19 plots the first principal component (PC_1) versus the second principal component (PC_2) for 25 urine samples from mice with varying stages of cancer, revealing no tendency for clustering that would reflect the presence or the stage of cancer in the subject.

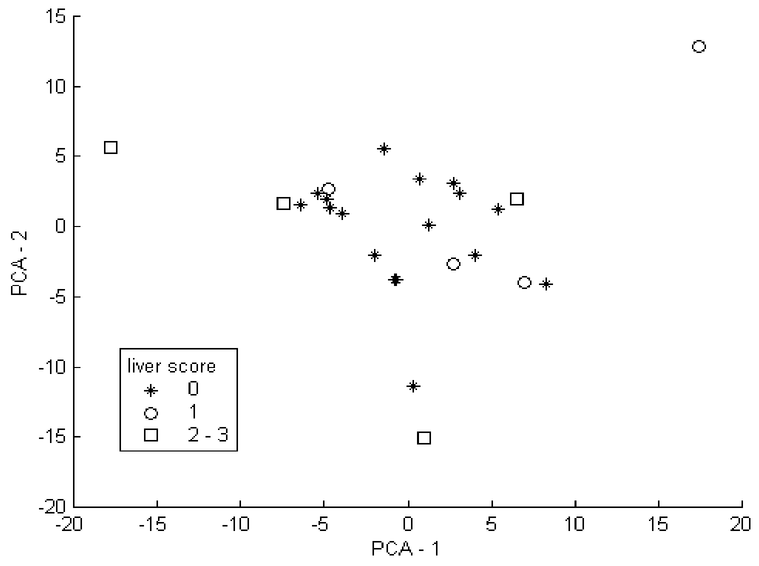

Figure 19: Plot of PC-1 vs. PC-2. Different phenotypes are shown with different symbols.

Next, we can perform linear regression analysis on the data using a Support Vector Machine, SVM, with a linear kernel. Using the leave-one-out approach to testing, the SVM was trained on 24 out of 25 data samples and then tested on the left-out sample. The SVM was trained 25 times, each time leaving a different sample out. Figure 20 plots SVM predictions on the test samples as a function of the actual liver scores. It reveals that linear regression fails to detect the relationship between the presence/stage of liver cancer and the urine NMR spectrum (the coefficient of determination – i.e. the correlation coefficient squared, expressing how much of SVM output variation can be accounted for by the liver score – was an insignificant 20%).

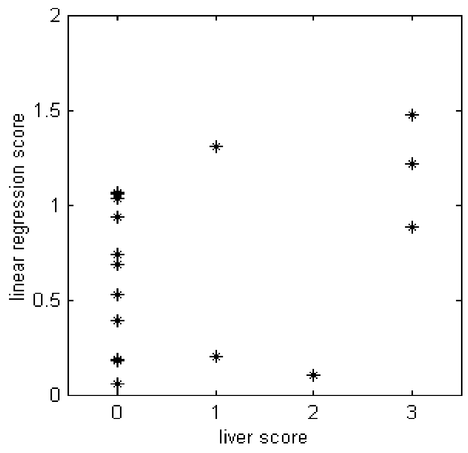

Figure 20: Linear regression of liver score.

Next, *nonlinear* regression analysis was performed using an SVM with a nonlinear, second-degree polynomial kernel. Figure 21 plots predictions of this SVM against the actual liver scores in the leave-one-out testing. This plot shows that nonlinear regression is much more powerful – its coefficient of determination was as high as 70%.

The nonlinear SVM computed its predictions from the magnitudes of only 8 spikes in the NMR spectrum. Locations of these spikes are indicated in Figure 22 by vertical tick marks, identifying these metabolites as potentially useful biomarkers (when considered together) of liver cancer.

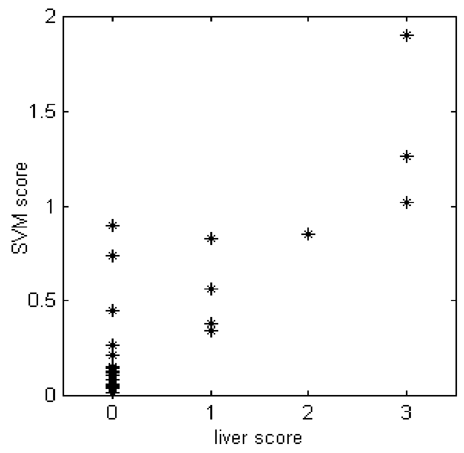

Figure 21: Nonlinear regression of liver score.

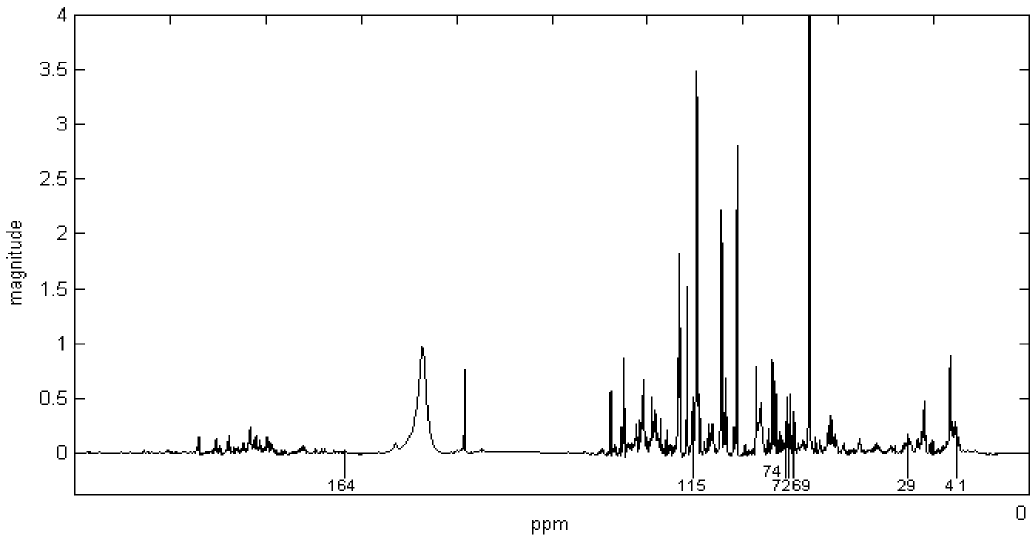

Figure 22: An exemplary minimal set of spikes required for regression of liver score by SVM.

As a conclusion, it should be pointed out that even the nonlinear SVM was not fully successful in predicting liver cancer from urine spectra. One reason is a very limited data sample – only 25 spectra were used, 17 of which were from healthy animals. Another possibility is that the relationship between the urine NMR spectrum and liver cancer might be too complex for an SVM to be able to learn it fully. If that is the case, then higher-order features of urine spectra (and not simply the spectral bins) should be explored as inputs to the SVM, because they might reduce the complexity of the sought relationship. Such higher-order features can be discovered by applying SINBAD to the data, treating each spike in the NMR spectrum as one of the observed variables. The cancer score (0 – 3) is also treated as one of the observed variables.

To give a quick example of such higher-order features, or hidden factors, we can make use of the set of 8 spikes, which were used by the SVM above to predict the cancer score from NMR spectra. In the *Virtual Scientist* terminology adopted in the first chapter, the cancer score is a Target Variable, whereas the 8 spikes make its Predictive Set. Together, the cancer score and the 8 spikes constitute a Minimal Set of Related Variables. To extract a Hidden Factor from this Minimal Set, we can test all possible partitions of this set on a SINBAD cell. One of the successful partitions is: spikes 1 and 4 vs. cancer score and the other six spikes in the minimal set (spikes 29, 69, 72, 74, 115, and 164). With this partition the two dendrites of the SINBAD cell were able to maximally correlate their outputs. So, what is the hidden factor discovered and computed by the dendrites? Its mathematical form can be appreciated by plotting the output of dendrite 1 as a function of spikes 1 and 4 (Figure 23). This higher-order feature of NMR spectra – although at this point we do not know its physiological identity – identifies apparently an important factor involved in liver pathology.

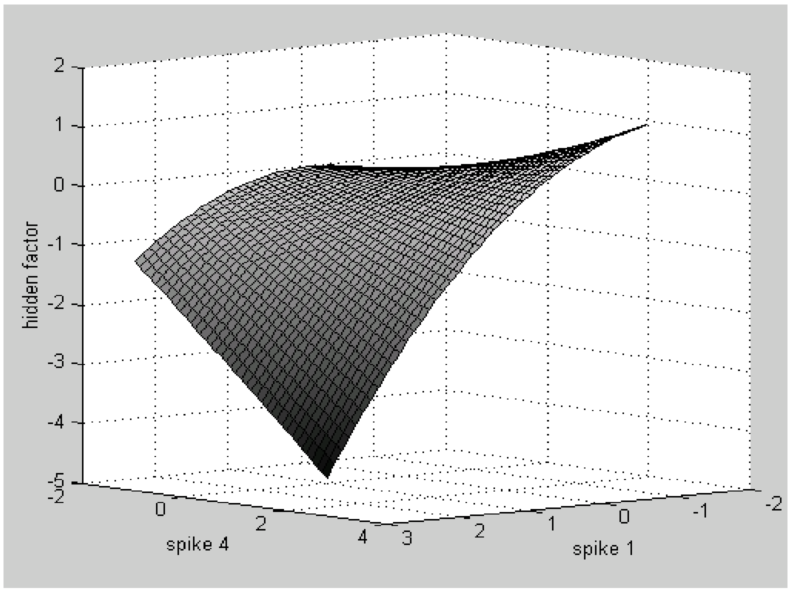

Figure 23: A 3-D plot of the hidden factor discovered by Virtual Scientist procedures.

A more systematic approach to discovery of higher-order features in NMR spectra of urine is that of *Virtual Scientist*, as described in previous chapters. To review this approach briefly, in its specific application to urine NMR-based data, minimal sets should be found for each spike by using each spike in turn as target. A given target spike and its predictive set constitute one Minimal Set of related spikes. Even though the number of spikes is high, it is very likely that many minimal sets will be identical as described earlier. As a result of such rediscoveries of the same Minimal Sets, the number of distinct Minimal Sets identified is likely to be smaller than the number of the observed variables.

Once the Minimal Sets of related spikes are identified, each of them should be partitioned in all possible ways and each partition should be tested for biomarkers (hidden factors). Hidden factors discovered by SINBAD cells in the NMR spectral data will extend, as derived variables, the list of variables (i.e., bins in NMR spectra) with which we can characterize the metabolic state of an organism. This extended set of variables will be used as inputs to SVMs, training them again to recognize the different phenotypes present in the study, and using the same dimensionality-reduction approaches described above.

The next task is to identify, quantify and express in a comprehensive way, the interdependencies among all the available – observed and derived – variables (spikes, biomarkers, diseases). To accomplish this goal, we need to identify the most direct relations among variables; these are the relations that involve minimal numbers of variables. Once such relations are known in sufficient numbers to form a more or less complete web, a single functional entity to represent learned *inferential relations*, then interdependencies among all the available variables can be traced through chains of these direct relations.

The use of higher-order features of NMR spectra as inputs to SVMs is expected to substantially enhance the ability of SVMs to discriminate among the different phenotypes. A yet another round of enhancements is possible by discovering hidden factors among the first layer of the already discovered factors (that is, by training SINBAD cells on the first layer of discovered factors). Once diagnostically useful hidden factors are identified as valuable biomarkers, their computational nature will be made explicit (i.e., which metabolites contribute to a given factor and what is the mathematical expression of the function being computed from them) with an intention of elucidating their underlying metabolic nature.

Among the central contributions of such an analysis, that can be performed by a thorough application of Virtual Scientist procedures, are (i) derivation of explanatorily powerful biomarkers for describing the state of the metabolism, (ii) explanation of the behaviors of the metabolites based on states of other metabolites and the biomarkers, and (iii) elucidation of the functional organization of the metabolism.

To summarize, metabolism is a system of a large number of interconverting chemicals, controlled through enzyme-catalyzed biochemical reactions. It is a complex web of molecular interactions that requires a thorough analysis using state of the art factor analysis and theory building techniques. The questions we believe that we can answer through a comprehensive metabonomics study using Virtual Scientist procedures are: topological organization of individual chemical reactions into metabolic network, the principles that govern the functional use of different reactions under different physiological conditions, such as exercise, rest, stress, various diseases, etc., and interplay between the underlying topology of the metabolic network and its functional organization and operation.

CONCLUSIONS

The set of Virtual Scientist procedures described in this work automates the method of inductive inference to produce a theory of a studied subject that can explain and relate the observed phenomena. The theory emerges in a form of a quantitative model of the subject capable of performing elaborate deductive inferences.

Developing an understanding of a new research subject commonly involves creation of new, subject-specific *concepts* that reveal deeper relations, interdependencies among the subject's observed phenomena. One central function of concepts is to express hidden causal factors, which – once recognized – serve to reduce complexity of the observed relations. As an example, in the kitchen sink study described above, flows of water in the two pipes, F_L and F_R, were two centrally important, but hidden factors. Despite lack of knowledge of F_L and F_R, relations among the observed variables could still be defined, because F_L and F_R were implicit in the states of the observed variables. However, such relations would be more complex. For example, compare the following relation, which uses only the observed variables to express $t°$, with Equation 18, which uses hidden factors F_L and F_R:

$$t^\circ = \frac{(1-HC)(1-DIR_L-KP_L+2\cdot DIR_L\cdot KP_L)+HC\cdot(1-DIR_R-KP_R+2\cdot DIR_R\cdot KP_R)}{2-DIR_L-KP_L+2\cdot DIR_L\cdot KP_L-DIR_R-KP_R+2\cdot DIR_R\cdot KP_R}\quad . \quad (21)$$

The more complex the inferential relations, the more difficult it will be to learn them; at some degree of complexity, learning will become impossible (Clark and Thornton 1997; Favorov and Ryder 2004). Thus, when dealing with complex subjects, generation of hidden factor concepts is absolutely necessary, if one were to learn interdependencies that otherwise would be too complex to be picked up directly from observations.

This insight is not a new one. Many different methods have been developed aimed at discovery of hidden factors, such as linear methods of correlation analysis, principal component analysis (PCA), belief networks and graphical models, independent component analysis (ICA), or nonlinear methods such as IMAX, nonlinear PCA, and nonlinear ICA (see, for example, Boyen et al. 1999, Hyvarinen et al. 2001, Ilin and Valpola 2003, Jutten and Karhunen 2003, Jordan 2004). Most of the methods based on graphical models use a costly structural search over all possible models or assume a fixed number of hidden variables with particular topologies etc; thus, finding hidden nodes in belief networks are known to be a notoriously difficult task (Boyen et al. 1999).

Linear factor analysis methods (Hyvarinen et al. 2001) are well advanced, but they are only of limited use, as most of the real-world problems tend to be nonlinear. Among nonlinear methods, our IMAX-based (Becker and Hinton 1992) SINBAD method is fundamentally different from most other approaches, which are based on the idea of *data compression*. Data compression based approaches transform raw input information into compact representations that utilize high-level data descriptors; however, the compactness of a representation does not guarantee that the descriptors will turn out to correspond to real, *inferentially useful* hidden factors. Rather, the search for descriptors by compression of the observed data is most likely to produce artificial descriptors that do not reflect the true causal factors operating in the system (see Figure 4 for an illustration). Unlike SINBAD-generated variables, such artificial descriptors are not designed to simplify relations among the observed variables and, as a result, they are not well suited for learning relations and for extracting information, making predictions from partial knowledge of the system's state.

To illustrate, I compared the inferential performance of the hidden factors V_8, V_9, V_{10} with the inferential performance of five variables derived by the Bayesian Ensemble Learning algorithm for nonlinear factor analysis (Lappalainen and Honkela 2000; Valpola et al. 2001). As shown in Table 6, even though Ensemble Learning discovered 5 variables (labeled E1 to E5) that can encode the information content of the seven observed variables, many of these variables are needed together for most of the inferences in the observed system and their inferential performance is inferior to that of SINBAD-generated variables. The reason for this difference is that the method loaded the derived variables with maximal information content (backbone of compression-based approaches) instead of picking single hidden factors.

In contrast to data-compression approaches, SINBAD belongs to the class of unsupervised learning algorithms that are based on the IMAX principle of identifying the sources of mutual information among disjoint sources of information, which yields functionally important features in the data reflecting the underlying causal structure of the system. The current implementation of my approach scales well (quadratic in time and linear in space) to larger systems; and parallel implementations might be able to perform even in linear time. The current version of the SINBAD method makes use of error-backpropagation neural network architecture to implement its learning modules. However, there are no restrictions on the type of learning modules that can be used (neural network or otherwise) as will be elaborated in later sections.

Table 6

The most direct inferential relations among the seven observed variables and either the three hidden variables derived by SINBAD method or, in comparison, the five variables derived by Bayesian Ensemble Learning.

Computed Variable	**SINBAD Method**		**Ensemble Learning**	
	Predictive Set	ρ^2	**Predictive Set**	ρ^2
V1	V8 V9	.998	E1 E3 E4 E5	.811
V2	V3 V10	.975	E1 E2 E3 E4	.919
V3	V2 V10	.947	V5E1E3E4E5	.752
V4	V5 V8	.988	E1 E3	.999
V5	V4 V8	.998	E2 E3 E4 E5	.957
V6	V7 V9	.987	E1 E3	.999
V7	V6 V9	.998	E2 E3 E4 E5	.988
average	*2.00 inputs*	*.984*	*3.57 inputs*	*.918*

SINBAD discovery of influential hidden factors is one of the cornerstones of the Virtual Scientist approach. Another cornerstone of the Virtual Scientist approach is its focus on learning those relations among the observed and derived variables that predict the state of *one* variable from the states of other variables. The aim is to learn as many ways to infer each variable from the others as can be found. By pursuing this simpleminded strategy of expressing each variable in many different ways in terms of other variables, which in turn are expressed in terms of yet other variables, etc., Virtual Scientist automatically acquires a rich web of inferential relations. In the ideal case, this web (which is grounded in the orderly structure of the studied dynamical system) would link each variable either directly or via intermediaries to every other variable.

In this web, all the discovered inferential relations are tied together into a single functional entity – *an inferential model* – revealing the causal organization of the studied subject (Ryder 2004). This type of insightful inference is very crucial especially in case of missing information. Apart from my approach, most learning algorithms are based on mapping of a set of input channels to some output. These algorithms do not evaluate the significance of the input channels and which ones are truly necessary for a specific mapping task. There are feature selection algorithms proposed for preprocessing tasks; however, these approaches are of limited contribution for building inferential model of the observed system as their use are limited to an initial, one-time-only feature selection. Therefore, given partial input, other approaches will fail to make valid inferences because their mapping requires that all variables to be supplied into the black-box input/output mapper. The reason I use the term "*black-box*" is that we have no idea how this mapping is done and how does the computation merge (integrate) different variables for the computations. Another disadvantage of the black-box approach is that given partial input may be sufficient or insufficient for the computations; but a black-box approach will fail to recognize that they failed. However, using the method presented in this work, we can determine, following the inferential chains, which target variables we can correctly infer. In this respect, SINBAD is a self-aware system – it knows that it is designed for making the most reliable inferences.

A theory constructed by Virtual Scientist can be developed further. Knowledge of the presence of a particular hidden factor, its connections to the observed variables, knowledge of its behavior under various conditions can all be used as clues for discovering its physical identity (as Mendel's inference of the existence of "hereditary factors," for example, led eventually to discovery of genes). Once the physical identities of factors are established, the ways might be devised to measure them directly (thus changing them into observed variables). Direct monitoring of such factors will result in an improved, more efficient and informative data collection, which might lead to discoveries of more deeply hidden factors and development of more insightful updates of the theory. On another tack, the background knowledge of the studied subject, the nature of sensors and their particular locations in the studied system (i.e., information about the subject that is not carried by the observed variables) can also be used to link the Virtual Scientist theory with the larger body of knowledge, fitting it in the context of the overall science.

In conclusion, I believe that the set of Virtual Scientist procedures offers a powerful analytical tool for use in research of complex scientific subjects rich in multivariate and nonlinear relations.

FUTURE WORK

As modern science turns to progressively more complex and challenging subjects across many fields, the growing size and complexity of collected data demand progressively more sophisticated analytical and theory-building methods, methods that can process large amounts of raw data and extract intricate, deeply hidden order. The set of computational procedures that I developed and called *Virtual Scientist* can automate analysis and theory-building process for these particularly difficult research problems modern science faces.

My approach to unsupervised learning is based on explaining interrelations among observed variables through chains of discovered hidden variables. Therefore, it has links to reasoning algorithms in regard to deciding how to trace the web (the represented knowledge of the underlying observed system) for the best possible computational paths for a given particular problem. This issue will be addressed in near future. I have already experimented with Virtual Scientist on natural images, face detection and recognition, NMR-based metabolomics, stock-market, and some artificial problems and it has shown great success, which will be the subjects of my subsequent publications.

The current version of the Virtual Scientist approach described here does not take temporal information into account, as it was intended for problems in which temporal information is not available. The Virtual Scientist strategy, however, is flexible and can readily be adopted for exploration of temporal order as well. Such an extension of Virtual Scientist procedures can be experimented on speech recognition and video processing.

LIST OF REFERENCES

Becker S (1999) Implicit learning in 3D object recognition: the importance of temporal context. *Neural Computation* 11: 347-374

Becker S (1996) Mutual information maximization: models of cortical self-organization. *Network* 7(1): 7-31

Becker S (1995) JPMAX: learning to recognize moving objects as a model-fitting problem. In: *Advances in neural information processing systems* 7, Morgan Kaufmann Publishers, San Mateo, CA, pp 933-940

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

Bishop CM (1995) Neural Networks for Pattern Recognition. Oxford University Press, Oxford

Boyen X, Friedman N, Koller D (1999) Discovering the hidden structure of complex dynamic systems. Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI 99), Stockholm, Sweden, pp. 91-100

Clark A, Thornton C (1997) Trading places: computation, representation, and the limits of uninformed learning. *Behavioral and Brain Sciences* 20: 57-90

DiCarlo JJ, Johnson KO (2000) Spatial and temporal structure of receptive fields in primate somatosensory area 3b: effects of stimulus scanning direction and orientation, *Journal of Neuroscience*, 20: 495-510

Favorov OV, Ryder D (2004) SINBAD: A neocortical mechanism for discovering environmental variables and regularities hidden in sensory input. *Biological Cybernetics* (in press)

Favorov OV, Ryder D, Hester JT, Kelly DG, Tommerdahl M (2003) The cortical pyramidal cell as a set of interacting error backpropagating networks: a mechanism for discovering nature's order. In: Hecht-Nielsen R and McKenna T (eds) *Computational Models for Neuroscience*, Springer Verlag, London, pp.25-64

Griffin JL (2003) Metabonomics: NMR spectroscopy and pattern recognition analysis of body fluids and tissues for characterization of xenobiotic toxicity and disease diagnosis. Curr. Opin. Chem. Biol. 7: 648-654

Hanson SJ, Pratt LY (1989) Comparing biases for minimal network construction with backpropagation. In: Touretzky DS (ed), *Advances in Neural Information Processing Systems*, 1: 177-185. San Mateo, CA: Morgan Kaufmann

Hassibi B, Stork DG (1993) Second order derivatives for network pruning: optimal brain surgeon. In: Hanson SJ, Cowan JD, and Giles CL (eds) *Advances in Neural Information Processing Systems* 5:164-171. San Mateo, CA: Morgan Kaufmann

Hyvarinen A, Karhunen J, Oja E (2001) Independent Component Analysis. John Wiley&Sons, Toronto

Ilin A, Valpola H (2003) On the effect of the form of the posterior approximation in variational learning of ICA models. In Proc. of the 4th Int. Symp. on Independent Component Analysis and Blind Signal Separation (ICA2003), Nara, Japan, pp. 915-920

Jordan MI (2004) Graphical models. *Statistical Science* (to appear)

Jutten C, Karhunen J (2003) Advances in nonlinear blind source separation. In Proc. of the 4th Int. Symp. on Independent Component Analysis and Blind Signal Separation (ICA2003), pp. 245-256

Kramer MA (1991) Nonlinear principal component analysis using autoassociative neural networks. *Aiche Journal* 37(2): 233-243

Kursun O, Favorov O (2004) SINBAD automation of scientific discovery: from factor analysis to theory synthesis. Natural Computing (to appear in June 2004 issue)

Kursun O, Favorov OV (2003) Single-frame super-resolution by inference from learned features. *Istanbul University Journal of Electrical & Electronics Engineering* 3(1): 673-681

Kursun O, Favorov OV (2002) Single-frame super-resolution by a cortex based mechanism using high level visual features in natural images. IEEE Workshop on Applications of Computer Vision, Orlando, FL

Lang KJ, Hinton GE (1989) Dimensionality reduction and prior knowledge in e-set recognition. NIPS 1989: 178-185

Lappalainen H, Honkela A (2000) Bayesian nonlinear independent component analysis by multi-layer perceptrons. In: Girolami M (ed) *Advances in Independent Component Analysis*, pp 93-121. Springer-Verlag. MATLAB toolbox at http://www.cis.hut.fi/projects/bayes/software/

LeCun Y, Denker JS, Solla SA (1990) Optimal brain damage. In: Touretzky DS (ed) *Advances in Neural Information Processing Systems* 2: 598-605. Morgan Kaufmann, San Mateo, CA

Mjolsness E, DeCoste D (2001) Machine learning for science: state of the art and future prospects. *Science* 293: 2051-2055

Nicholson JK, Connelly J, Lindon JC, Holmes E (2002) Metabonomics: a platform for studying drug toxicity and gene function. Nat. Rev. Drug Discovery 1: 153-161

Olshausen BA, Field DJ (1996) Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381: 607-609

Otto M (1999) Chemometrics. Statistics and Computer Application in Analytical Chemistry. Wiley-VCH, Weinheim, Germany

Phillips WA, Singer W (1997) In search of common foundations for cortical computation. *Behavioral and Brain Sciences* 20: 657-722

Reo NV (2002) NMR-based Metabolomics. Drug Chem. Toxicology 25: 375-382

Rumelhart DE, Hinton GE, Williams RJ (1986) Learning internal representations by error propagation. In: Rumelhart DE, Mcclelland JL, PDP Research Group (eds) *Parallel Distributed Processing: Explorations in the Microstructure Of Cognition*, MIT Press, Cambridge, Mass, 1: 318-362

Ryder D (2004) SINBAD Neurosemantics: a theory of mental representation. *Mind & Language* (in press)

Ryder D, Favorov OV (2001) The new associationism: a neural explanation for the predictive powers of cerebral cortex. *Brain and Mind* 2: 161-194

Stone J (1996) Learning perceptually salient visual parameters using spatiotemporal smoothness constraints. *Neural Computation* 8: 1463-1492

Valpola H, Raiko T, Karhunen J (2001) Building blocks for hierarchical latent variable models. *Proc. Int. Conf. on Independent Component Analysis and Signal Separation*, San Diego

Vapnik V (1995) The nature of statistical learning theory. Springer Verlag, New York

Vapnik V (1998) Statistical Learning Theory. Wiley, New York