HiperLearn: A High Performance Channel Learning Architecture

Gösta Granlund, Per-Erik Forssén, Björn Johansson

May 22, 2003

Abstract—
A learning system architecture using a *channel information representation* has been developed. This representation implies that signals are *monopolar* and *local*. Monopolar means that data only utilizes one polarity, e.g. positive values, allowing zero to represent *no information*. Locality derives from a partially overlapping mapping of signals into a higher-dimensional space. This implies that a signal is non-zero only within a limited range of the input domain. This combination of monopolarity and locality leads to an efficient *sparse* representation.

Locality of features allows the generation of highly non-linear functions using a linear mapping. The averaging properties of channel functions allow representation of discontinuities, while implementing an interpolating mapping in other regions. Mapping from an input channel set onto an output channel set, allows the use of confidence statements in data, leading to a low sensitivity to noise in features.

The optimization uses a modified projected Landweber method. The sparse monopolar representation together with locality, using individual learning rates, allows a fast optimization, as the system exhibits linear complexity. The monopolarity implies a regularization, providing a better generalization.

Experiments on functionality and noise sensitivity are presented.

Keywords— **Channel representation, monopolar, association, sparse, confidence, learning, learning rate, optimization, neural networks, RBF, SVM, local features, classification, linkage matrix, non-linear mapping, wavelets, function approximation, computer vision, signal processing.**

I. Introduction

OVER the years, the complexity of systems for information processing has increased dramatically. In many fields such as vision, robotics, speech and control it has proved increasingly difficult to specify complex procedures, where each subprocedure may only be valid within limited windows of context, or it depends upon contextual parameters. It would be of great benefit if such procedures could be designed through learning of the complex and usually non-linear relationships.

There has over the years been an extensive research on mechanisms which would allow such an acquisition of information, e.g. under the heading of neural networks [1]. However, the number of powerful applications using learning has been limited, due to the still limited capacity of today's learning structures.

A large class of problems, which would benefit from the use of learning methods, is characterized by the following properties:

The authors are with the Computer Vision Laboratory, Department of Electrical Engineering, SE-581 83 Linköping, Sweden (E-mail: {gosta,perfo,bjorn}@isy.liu.se)

- A large number of input and output variables
- A highly non-linear mapping, which in parts is locally continuous and in other parts transiential switching
- A mapping which requires the interpolation between a large number of models

These characteristics are true for many problems in control, robotics, signal processing and vision. An improvement of methodology for this class of problems would have a large impact upon the number of feasible applications. The architecture proposed in this paper is intended to provide efficient learning for problems within this class.

II. Notations

Italic letters (x,X) denote scalars, lowercase letters in boldface ($\mathbf{x}$) denote vectors, and uppercase letters in boldface ($\mathbf{X}$) are used for matrices. Further, $\langle \cdot , \cdot \rangle$ denotes the standard Euclidean inner product, and $\|\cdot\|$ the induced norm. Weighted inner products are given by $\langle \cdot, \cdot \rangle_{\mathbf{W}} = \langle \cdot, \mathbf{W} \cdot \rangle$, and $\|\cdot\|_{\mathbf{W}}$ denotes the induced weighted norm. The norm of a matrix is assumed to be the Frobenius norm, $\|\mathbf{A}\|^2 = \mathrm{trace}(\mathbf{A}^T\mathbf{A})$. Moreover, $\mathbf{D} = \mathrm{diag}(\mathbf{v})$ denotes a diagonal matrix with $D_{ii} = v_i$, $\mathbf{v} = \max(\mathbf{x}, \mathbf{y})$ denote element-wise maximum, i.e. $v_i = \max(x_i, y_i)$, and $\mathbf{x} > \mathbf{0}$ and $\mathbf{x} \geq \mathbf{0}$ denotes positive and non-negative vectors respectively.

We will use both row vectors and column vectors in this paper. To avoid confusion, we introduce additional notations. A vector without index ($\mathbf{x}$) denotes a column vector. Element i in a column vector $\mathbf{x}$ is denoted in superscript italics, x^i. A superscript vector ($\mathbf{x}^i$) denotes a row vector. A subscript vector ($\mathbf{x}_i$) denotes a column vector. As a consequence, an element in a matrix may be written with one subscript and one superscript ($X_{ij} = X^i_j$).

Additional notations are introduced when needed.

III. Architecture Overview

In the proposed architecture, the choice of information representation is of fundamental importance. The architecture uses a monopolar channel information representation. The channel representation implies a mapping of signals into a higher-dimensional space, in such a way that it introduces locality in the information representation with respect to all dimensions; geometric space as well as property space.

The locality obtained in this channel representation gives two advantages:

- Nonlinear functions and combinations can be implemented using linear mappings

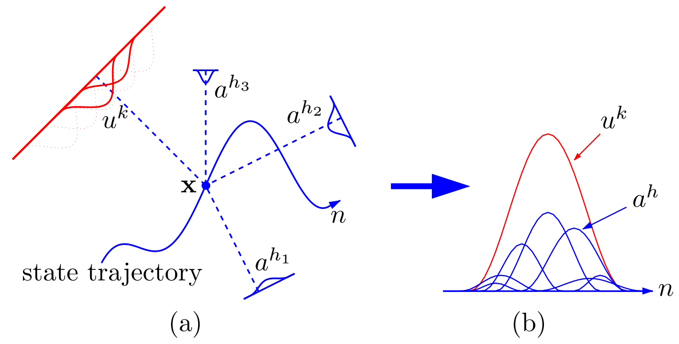

Fig. 1. Architecture overview. (a) The system is moving along a state space trajectory. Response channels, u^k, and feature channels, a^h, measure different (local) aspects of the state vector $\mathbf{x}$. (b) The response channels and the feature channels make up localized functions along the trajectory. A certain response channel is associated with some of the feature channels.

- Optimization in learning converges much faster

Figure 1 gives an intuitive illustration of how signals are represented as local fragments, which can be freely assembled to form an output. The system is moving along a state space trajectory. The state vector $\mathbf{x}$ consists of both internal and external system parameters. The response space is typically a subset of those parameters, e.g. orientation of an object, position of a camera sensor in navigation, or actions of a robot. Response channels and feature channels measure local aspects of the state space. The response channels and feature channels make up reponse channel vectors $\mathbf{u}$ and feature vectors $\mathbf{a}$ respectively.

The processing mode of the architecture is association where the mapping of features a^h onto desired responses u^k is learned from a representative training set of observation pairs $\{\mathbf{a}_n, \mathbf{u}_n\}_{n=1}^N$, see figure 1(b). The feature vector $\mathbf{a}$ may contain some hundred thousand components, while the output vector $\mathbf{u}$ may contain some thousand components.

The monopolar property implies that data only utilizes one polarity, e.g. only positive values, in addition to zero. This allows zero to represent not just another value, such as temperature zero as opposed to other values of the temperature, but to represent no information.

For most features of interest, only limited parts of the domain will have non-zero contributions. This provides the basis for a sparse representation, which gives improved efficiency in storage and better performance in processing.

The model of the system is in the standard version a linear mapping from a feature vector $\mathbf{a}$ to a response vector $\mathbf{u}$ over a linkage matrix $\mathbf{C}$,

$$\mathbf{u} = \mathbf{C}\mathbf{a}\,. \tag{1}$$

In some training process, a set with N samples of output vectors $\mathbf{u}$ and corresponding feature vectors $\mathbf{a}$ are obtained. These form a response matrix $\mathbf{U} = \begin{pmatrix}\mathbf{u}_1 & \dots & \mathbf{u}_N\end{pmatrix}$ and a feature matrix $\mathbf{A} = \begin{pmatrix}\mathbf{a}_1 & \dots & \mathbf{a}_N\end{pmatrix}$. The training implies finding a solution matrix $\mathbf{C}$ to

$$\mathbf{U} = \mathbf{C}\mathbf{A}\,. \tag{2}$$

The linkage matrix is computed as a solution to a least squares problem with a monopolar constraint $\mathbf{C} \geq \mathbf{0}$. This constraint has a regularizing effect, and in addition it gives a sparse linkage matrix. The monopolar representation together with locality, allows a fast optimization, as it allows a parallel optimization of a large number of loosely coupled system states.

We will compare the standard version (1) to models where the mapping is made directly to the response subset of the state parameters, i.e. typically what would be used in regular kernel machines. We will in these cases use a modified model with various normalizations of $\mathbf{a}$.

IV. Channel representation of signals

The *channel representation* [2], [3], [4] is a high-dimensional representation of a signal, in such a way that confidence or certainty can be represented and used in computations. This is achieved using a *position encoding* of the signal value. By applying a set of non-negative kernel functions $\{\mathrm{B}^k(x)\}_1^K$ to a signal $x(t)$, and weighting the result with the corresponding confidence $r(t)$ we obtain a vector

$$\mathbf{u} = \begin{pmatrix} r\mathrm{B}^1(x) & r\mathrm{B}^2(x) & \dots & r\mathrm{B}^K(x)\end{pmatrix}^T. \tag{3}$$

This operation defines the *channel encoding* of the signal/confidence pair (x, r), and the resultant vector $\mathbf{u}$ constitutes a channel representation of the signal/confidence, provided that the channel encoding is *injective*; i.e. there exists a corresponding decoding that reconstructs the signal, and its confidence from the channels. We will in this context forego the discussion about how confidence of signals can be established, but reference is made to [5].

Examples of suitable kernels for channel representations include Gaussians[6], B-splines[7], and other localized windowing functions with a shape similar to the kernel in figure 2.

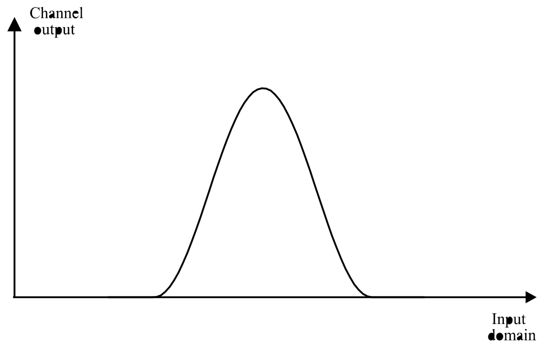

Fig. 2. A kernel function that generates a channel from a signal.

A. The $\cos^2$ kernel

In this paper, channel properties will be exemplified using the following family of kernel functions

$$\mathrm{B}^k(x) = \begin{cases} \cos^2(\omega d(x,k)) & \text{if} \quad \omega d(x,k) \leq \frac{\pi}{2} \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

Variable k is the *kernel center*, ω is the *kernel width*, and $d(x,d)$ is a distance function. For variables in linear spaces the Euclidean distance is used,

$$d(x,k) = |x-k|\,, \tag{5}$$

and for periodic spaces with period K a modular[1] Euclidean distance is used,

$$d_K(x,k) = \min(\mathrm{mod}(x-k,K), \mathrm{mod}(k-x,K)). \tag{6}$$

The measure of an angle is a typical example of a variable in a periodic space. Rather than separate signal x, and confidence r values, we now have a set of channel values $\{u^k = r\mathrm{B}^k(x)\}_1^K$. The total interval of a signal x can be seen as cut up into a number of local but partially overlapping intervals, $d(x,k) \leq \frac{\pi}{2\omega}$.

In order to simplify the notation, we have defined the numbers k as consecutive integers, directly corresponding to the indices of consecutive kernel functions. We are obviously free to scale and translate the actual signal value in any desired way, before we apply the set of kernel functions. For instance, a signal value ξ can be scaled and translated in the desired way,

$$x = scale \cdot (\xi - translation), \tag{7}$$

to fit the interval spanned by the set of kernel functions $\{\mathrm{B}^k(x)\}_1^K$. Non-linear mappings $x = \mathrm{f}(\xi)$ are of course also possible, but they should be monotonous for the representation to be non-ambiguous.

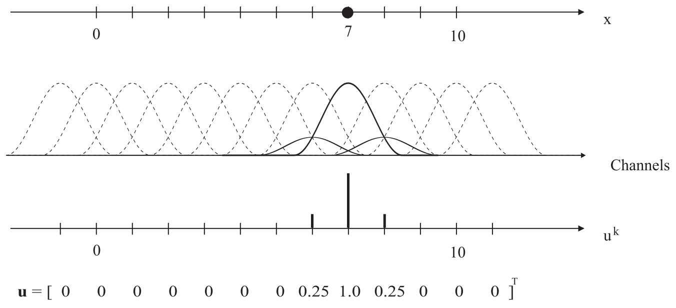

Fig. 3. Channel representation of the signal value $x = 7$.

Figure 3 shows an example of a channel representation of a single signal value. It is represented using a one-dimensional set of $K = 13$ sequentially ordered kernels with a width of $\omega = \pi/3$. The kernels are indicated by the dashed lines. The set of channels is activated by the signal value $x = 7$, as indicated by a dot. The values of the activated channels are indicated by the heights of the solid curves. The coefficients in the resultant channel vector $\mathbf{u}$ for $r = 1$ are shown in the bottom of figure 3. Note that the channel set in figure 3 is designed to represent signals in the interval $-0.5 \leq x \leq 10.5$.

[1]using the modulo operation $\mathrm{mod}(x,K) = x - \lfloor x/K \rfloor K$

B. Sparse data representation

The representation of a signal as 13 numbers instead of one may seem like a waste of memory. There are however two redeeming features:

- Due to the localized support of the kernel functions used, see (4), we are guaranteed that only a small fraction of the channels are non-zero at a time, i.e. the representation is *sparse*. For instance $\omega = \pi/3$ gives at most three non-zero channels. For sparse representations there exist a number of methods to reduce the memory storage requirements, and to reduce the computation time significantly.
- The redundancy in the nonzero values allows a description of confidence in addition to signal value.

C. Representation of multiple values

The channel representation has a distinct advantage to conventional signal representations in that it allows several signal values to be represented simultaneously. This is useful in cases where multiple alternatives of a variable shall be provided to a subsequent processing step.

As an example, the channel set from the previous section can simultaneously represent the signal values $x = 1$ and $x = 7$, as shown in figure 4.

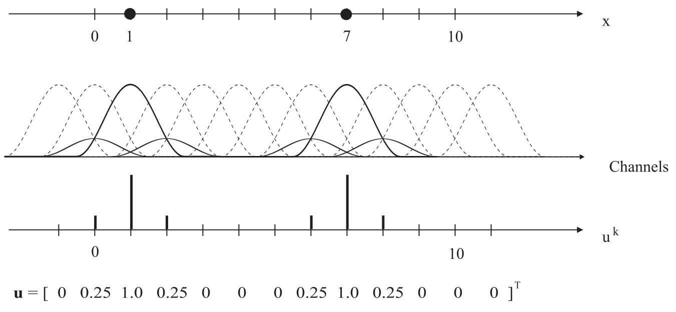

Fig. 4. Channel representation of $x = 1$ and $x = 7$.

In this case both signal values have the same confidence, but in a typical application they may differ. It is apparent that as the difference between the two signal values decreases, there will be interference between the contributions. This indicates a need to worry about proper channel resolution, like for any sampling process.

D. Properties of the $\cos^2$ kernel

A major motivation for the $\cos^2$-kernel in (4) is that it has a localized support, which ensures sparsity. Another motivation is that for values of $\omega = \pi/N$ where $N \in \{3,4,...\}$ we have

$$\sum_k \mathrm{B}^k(x) = \frac{\pi}{2\omega} \qquad \text{and} \qquad \sum_k (\mathrm{B}^k(x))^2 = \frac{3\pi}{8\omega}. \tag{8}$$

This implies that the sum, and the vector norm of a channel value vector generated from a single signal/confidence pair is *invariant* to the value of the signal x, as long as x is within the definition interval of the channel set (for a proof, see [4]). The constant norm implies that the kernels locally constitute a *tight frame* [8], a property that ensures uniform distribution of signal energy in the channel space, and makes a decoding operation easy to find.

E. Decoding a channel representation

An important property of the channel representation is the possibility to retrieve the signal/confidence pair corresponding to a given channel vector. The problem of decoding signal and confidence values from a set of channel function values, superficially resembles the reconstruction of a continuous function from a set of frame coefficients. However, in the present situation of position encoding, there is a significant difference: *we are not interested in reconstructing the exact shape of a function, we merely want to find all peak locations and their heights.*

As mentioned earlier, the channel representation allows several values to be represented simultaneously. As the difference between these values decreases, there will be an interference between channel statements. This turns out to be an advantage, as it automatically allows an averaging over adjacent values, while sufficiently separated values maintain their identity [9].

In order to decode several signal values from a channel vector, we have to make a *local decoding*, i.e. a decoding that assumes that the signal value lies in a specific limited interval (see figure 5).

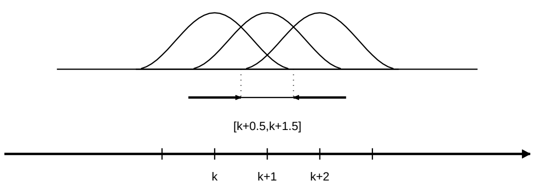

Fig. 5. Interval for local decoding ($\omega = \pi/3$).

For the $\cos^2$ kernel, and the local tight frame situation (8), it is suitable to use decoding intervals of the form $[k + N/2 - 1, k + N/2]$, see figure 5. Decoding a channel vector thus involves examining all such intervals for signal/confidence pairs.

The local decoding is computed using a method illustrated in figure 6. The channel values, u^k, are now seen as samples from a kernel function translated to have its peak at the represented signal value $\hat{x}$.

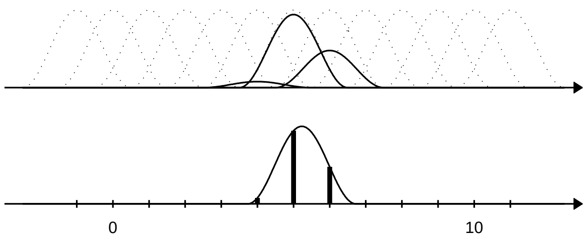

Fig. 6. Example of channel values ($\omega = \pi/3$, and $\hat{x} = 5.23$).

We denote the index of the first channel in the decoding interval by l (in the figure we have $l = 4$), and use local intervals of channels $\{u^l, u^{l+1}, \ldots, u^{l+N-1}\}$. For the local tight frame situation (8), we can now compute $\hat{x}$ as a local weighted summation of complex exponentials[2] [4]:

$$\hat{x} = \text{dec}(\mathbf{u}) = l + \frac{1}{2\omega}\arg\left[\sum_{k=l}^{l+N-1} u^k e^{\mathbf{i}2\omega(k-l)}\right]. \quad (9)$$

This decoding assumes that the channel values u^k have been generated using (4). For response channels estimated using (1) this is not necessarily true even though we may have supplied such responses during training.

The most straightforward way to decode the confidence is to use a scaled sum of the channel values used to estimate the corresponding signal value,

$$\hat{r} = \frac{2\omega}{\pi}\sum_{k=l}^{l+N-1} u^k. \quad (10)$$

A study of the robustness of this signal/confidence decoding, as well as the behavior in case of interfering signal/confidence pairs, can be found in [6].

We conclude this discussion by emphasizing that the relation between neighboring channel values tells us the signal value, and the channel magnitudes tell us the confidence of this statement. In signal processing it is important to attach a measure of confidence to signal values [5]. The channel representation can be seen as a unified representation of signal *and* confidence.

F. Comparison with other kernel methods

Kernel functions are used in the design of kernel machines for classification and regression, such as Radial Basis Function (RBF) networks and Support Vector Machines (SVM), see e.g. [1], [10], [11], [12]. A desired property in those systems is often that the kernel function is positive definite. *Mercers theorem* states that the kernel function is equivalent to an inner product in a high-dimensional space. This is called the *kernel trick*. The kernels are typically chosen in such a way that there is a kernel centered around each training data, and the kernel trick can be used to design sparse networks that only use a few of the kernels, see e.g. [12]. The kernels in the present paper are not assumed to be positive definite, only to be monopolar and to have a local spatial support.

Another method is to choose kernel parameters, e.g. center and width, independently of the training set. Note that a proper regularization is required if the number of kernels is higher than the number of samples. This is the approach used in this paper. The kernel parameters are sometimes included in the optimization, see e.g. generalized RBF networks [1]. However, this will give a non-linear optimization problem, which is more difficult to solve than a linear problem. We assume in this paper that the kernel parameters are given by other systems requirements, such as sensor resolution, and are suitably chosen from the start.

[2]Note the similarity to a local Fourier reconstruction of a Dirac delta distribution.

V. Representation of system output states

For a system acting in a continuous environment, we can define a *state space* $\mathcal{X} \subset \mathbb{R}^M$. A state vector, $\mathbf{x} \in \mathcal{X}$, completely characterizes the current situation for the system, and $\mathcal{X}$ is thus the set of all situations possible for the system. The state space has two parts termed *internal* and *external*. Internal states describe the system itself, such as its position and its orientation. External states describe a subset of the total states of the environment, which are to be incorporated in the system's knowledge, such as the position, orientation and size of a certain object.

The estimation of external states requires a coupling to internal states, which can act as a known reference in the learning process. Generally it is desirable to estimate either a state, or a *change of state* or *response*. The detailed implementation of this, however, goes beyond the scope of this presentation, and we will for simplicity only assume that the desired response variables are components of the state vector $\mathbf{x}$.

We assume that the system is somehow induced to change its state, such that it covers the state space of interest for the learning process. The system state change must take place in a continuous way, due to the inertia caused by limited power for displacement of a certain mass (See [13] for a more extensive discussion).

The state variables are defined at sample points $n = 1, 2, \ldots, N$. We will denote the state at sample point n by a vector $\mathbf{x}_n = \begin{pmatrix} x_n^1 & x_n^2 & \ldots & x_n^M \end{pmatrix}^T$. The sample points n may e.g. correspond to different time instances, or simply ordered points along a system trajectory. We can express such a system state trajectory as a matrix $\mathbf{X} = \begin{pmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \ldots & \mathbf{x}_N \end{pmatrix}$.

A. Channel representation of the state space

The normal form of output for the structure is in channel representation. It is advantageous to represent the scalar state variables in a regular channel vector form, as this allows multiple outputs when the input is ambiguous, see IV-C. The channel representation also forms the basis for learning of discontinuous phenomena, and is an appropriate representation for continued processing at a next level.

A *response channel* vector $\mathbf{u}^m$ is a channel representation of one of the components x^m of the state vector $\mathbf{x}$, see (3). The vector $\mathbf{u}^m$ is thus a non-ambiguous representation of position in a *response state space* $\mathcal{R}^m = \{x^m : \mathbf{x} \in \mathcal{X}\}$.

With this definition, a response channel will be non-zero only in a very limited region of the state space. The value of a channel can be viewed as a confidence about the distance of the current state to a prototype state. When a specific channel is non-zero it is said to be *active*, and the subspace where a specific channel is active is called the *active domain* of that channel. As the active domain is always much smaller than the inactive domain, an inactive channel will convey almost no information about position in state space. The small active domain is also what makes a sparse representation effective.

The response channel vectors $\mathbf{u}_n^m$ can be put into response channel matrices $\mathbf{U}^m = \begin{pmatrix} \mathbf{u}_1^m & \mathbf{u}_2^m & \ldots & \mathbf{u}_N^m \end{pmatrix}$. All such response channel matrices are stacked row-wise to form the response channel matrix $\mathbf{U}$. While $\mathbf{U}$ will have a much larger number of rows than the original state matrix $\mathbf{X}$ due to the increase of dimensionality in the representation, the sparsity of the representation will imply a moderate increase of the amount of data.

VI. Channel representation of input features

It is assumed that the system can obtain at least partial knowledge about its state from a set of observed feature variables, $\{a^h\}$, forming a feature vector $\mathbf{a} = \begin{pmatrix} a^1 & a^2 & \ldots & a^H \end{pmatrix}^T$. In order for an association or learning process to be meaningful, there has to be a sufficiently unique and repeatable correspondence between system states and observed features. One way to state this requirement is as follows: The *sensor space* of states that the feature channels can represent, $\mathcal{A}$, should allow an unambiguous mapping $f : \mathcal{A} \rightarrow \mathcal{R}$. The situation where this requirement is violated is known in learning and robotics as *perceptual aliasing*, see e.g. [14].

A generative model for $\{a^h\}$, useful for systems analysis, can be expressed as localized, non-negative kernel functions $\mathrm{B}^h(\mathbf{x})$. These are functions of a scaled distance between the state vector $\mathbf{x}$ and a set of prototype states $\mathbf{x}^h \in \mathcal{X}$. We exemplify this with the $\cos^2$-kernel,

$$a^h = \mathrm{B}^h(\mathbf{x}) = \begin{cases} \cos^2(\mathrm{d}(\mathbf{x}, \mathbf{x}^h)) & \text{if } \mathrm{d}(\mathbf{x}, \mathbf{x}^h) \leq \pi/2 \\ 0 & \text{otherwise.} \end{cases} \tag{11}$$

The used distance function is defined as

$$d(\mathbf{x}, \mathbf{x}^h) = \sqrt{(\mathbf{x} - \mathbf{x}^h)^T \mathbf{M}^h (\mathbf{x} - \mathbf{x}^h)}. \tag{12}$$

Matrix $\mathbf{M}^h$ is positive semidefinite, and describes the metric for the distance function around state $\mathbf{x}^h$, allowing a scaling with different sensitivities with respect to different state variables. The norm of the metric matrix $\mathbf{M}^h$ is generally very large, which makes the angular space for a nonzero contribution to a particular a^h extremely small compared to the total angular space of the curved hypersurface over which the state vector is moving. The number of such partially overlapping regions may be of the order $H = 10^5$ to $H = 10^7$. Assuming a more or less even coverage of the parts of interest of the hyperspace, means that most outputs a^h will be zero at any given sample point along the training trajectory, ensuring a sparse data set. Equation (11) indicates that a^h will have a maximal value of 1 as $\mathbf{x} = \mathbf{x}^h$. It will go to zero as the weighted distance increases to $\pi/2$, see figure 2. Normally, neither $\mathbf{x}^h$, nor $\mathbf{M}^h$ are explicitly known, but emerge implicitly from the properties of the set of sensors used in the actual case. These are generally different from one sensor or filter to another, which motivates the notion of *channel representation*, as each channel has its specific identity, the identification of which is part of the learning process.

The feature variables a^h will in a typical case be outputs from band-pass filters describing properties of a signal, or properties in an image of an object, such as local orientation, local curvature, color, etc, image features at different positions. If the properties have a confidence measure, it is natural to weight the channel features with this. See discussion in section IV.

In general, there is no requirement for a regular arrangement of channels, be it on the input side or on the output side. The prescription of an orderly arrangement at the output comes from the need to interface the structure to the environment, e.g. to determine its performance. In such a case it will be desirable to map the response channel variables back into scalar variables in order to compare them with the reference, something which is greatly facilitated by a regular arrangement.

Similarly to the state variables, we denote the observation at sample point n by a vector $\mathbf{a}_n = \begin{pmatrix} a_n^1 & a_n^2 & \dots & a_n^H \end{pmatrix}^T$. These observation or feature vectors can be put into a feature matrix $\mathbf{A} = \begin{pmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \dots & \mathbf{a}_N \end{pmatrix}$.

A. *Feature generation*

The feature vectors $\mathbf{a}$, input to the associative structure, may derive directly from the preprocessing parts of a computer vision system, representing orientation, curvature, color, etc. More often, combinations or functions $f(\mathbf{a})$ of a set of features $\mathbf{a}$, will be used as input to the associative structure. A common variety to increase specificity in the percept space is covariant combination between components, or a subset of components, such as $f(\mathbf{a}) = \text{vec}(\mathbf{a}\mathbf{a}^T)$. The symbol vec signifies the trivial transformation of concatenating rows or columns of a matrix into a vector. For simplicity of notation, we will express this as a substitution,

$$\mathbf{a} \leftarrow f(\mathbf{a})\,. \tag{13}$$

This is given a conceptual illustration in figure 7. The final vector $\mathbf{a}$, going into the associative structure will generally be considerably longer than the corresponding size of the sensor channel array. As we are dealing with sparse feature data, the increase of the data volume will be moderate.

VII. System Operation Modes

The channel learning architecture can be run under two different operation modes, providing output in two different representations:

1. Position encoding for *discrete event* mapping
2. Magnitude encoding for *continuous function* mapping

The first variety, using discrete event mapping, is the mode which maximally exploits the advantage of the information representation, to allow implementation and learning of highly non-linear transfer functions, using a linear mapping.

The second variety is similar to more traditional function approximation methods.

A. *Position encoding for discrete event mapping*

In this mode, the structure is trained to map onto a set of channel representations of the system response state

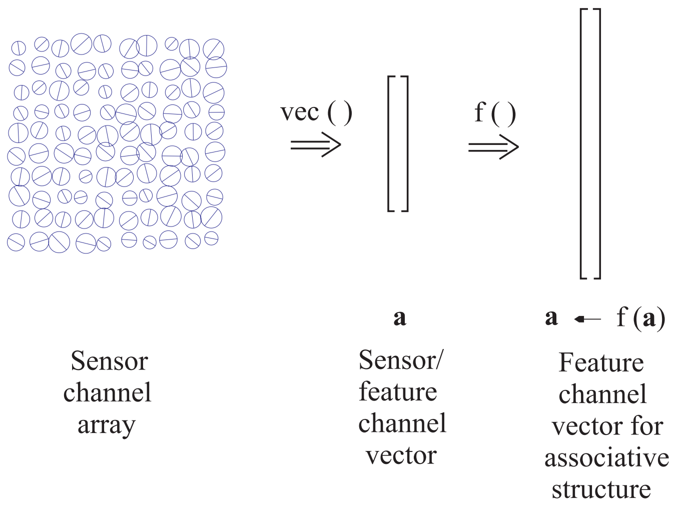
vec ( )
f ( )
a
a ← f (a)
Sensor channel array
Sensor/ feature channel vector
Feature channel vector for associative structure

Fig. 7. Illustration of steps in going from sensor array to feature vector.

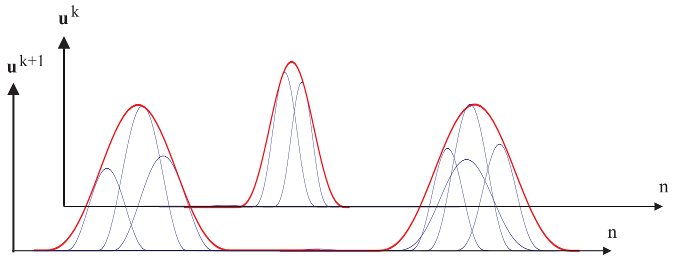

Fig. 8. Intuitive illustration of discrete event mapping (the input domain will generally be multidimensional).

variables, as discussed in subsection V-A. Thus the responses will have non-zero output only within limited regions of the definition range. The major issue is that a multi-dimensional, fragmented feature set is mapped onto a likewise fragmented, version of the system state space. See figure 8 for an intuitive illustration.

There are a number of characteristics of this mode:

- Mapping is made to sets of output channels, whose response functions may be partially overlapping to allow the reconstruction of a continuous variable.
- Output channels are assumed to assume some standard maximum value, say 1, but are expected to be zero most of the time, to allow a sparse representation.
- The system state is not given by the magnitude of a single output channel, but is given by the relation between outputs of adjacent channels.
- Relatively few feature functions, or sometimes only a single feature function, are expected to map onto a particular output channel.
- The channel representation of a signal allows a unified representation of signal value and of signal confidence, where the relation between channel values represents value, and the magnitude represents confidence. Since the discrete event mode implies that both the feature and response state vectors are in the channel representation, the confidence of the feature vector will be propagated to the

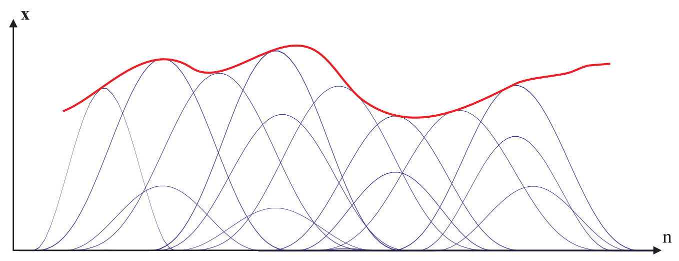

Fig. 9. Intuitive illustration of continuous function mapping.

response vector if the mapping is linear.

The properties just listed, allows the structure to be implemented as a purely linear mapping,

$$\mathbf{u} = \mathbf{C}\mathbf{a}\,. \tag{14}$$

B. *Magnitude encoding for continuous function mapping*

The continuous function mapping mode is used to generate the response state variables directly, rather than a set of channel functions for position decoding. The response state vector, $\mathbf{x}$, is approximated by a set of discrete channel feature functions. See figure 9 for an intuitive illustration.

This mode corresponds to classical function approximation objectives. The mode is used for accurate representation of a scalar continuous function, which is often useful in control systems.

The approximation will be good if the feature functions are sufficiently local, and sufficiently dense. There are a number of characteristics for the Continuous Function Mapping:

- It uses rather complete sets of feature functions, compared to the mapping onto a response mapping in discrete event mode. The structure can still handle local feature dropouts without adverse effects upon well behaved regions.
- Mapping is made to continuous output response variables, which may have a magnitude which varies over a large range.
- A high degree of accuracy in the mapping can be obtained if the feature vector is normalized, as stated below.

In this mode, however, it is not possible to represent both a state value $\mathbf{x}$, and a confidence measure r, except if it is done explicitly. Given the properties of the channel feature vector $\mathbf{a}$, we intuitively accept that a feature vector with a larger magnitude has a greater confidence in its statement than one with a small magnitude. An additional assumption is that features which are active within a domain all have the same confidence. This implies that the confidence measure r can be viewed as a part the feature vector $\mathbf{a}$.

Assuming a linear model of mapping, this gives a product between the expected state $\mathbf{x}$ and the confidence measure,

$$r\mathbf{x} = \mathbf{C}\mathbf{a}\,. \tag{15}$$

By dividing the feature vector $\mathbf{a}$ with r we can normalize with the amount of confidence, or certainty, in $\mathbf{a}$. This is related to the theory of normalized averaging, see e.g. [5].

TABLE I

Overview of characteristics of different operation modes.

Operation mode	Continuous Function Mapping	Discrete Event Mapping
Representation of state	Magnitude encoding	Position encoding
Handling of zero	Zero is not an acceptable value	Zero represents domains of “no information”
Representation of confidence	Implicit representation	Explicit representation
Model	$\mathbf{x} = \mathbf{C}\frac{1}{\mathbf{w}^T\mathbf{a}}\mathbf{a}$	$\mathbf{u} = \mathbf{C}\mathbf{a}$

For the confidence measure a linear model is assumed,

$$r = \mathbf{w}^T\mathbf{a}\,. \tag{16}$$

The model for this mode comes out as

$$\mathbf{x} = \mathbf{C}\frac{1}{\mathbf{w}^T\mathbf{a}}\mathbf{a}\,, \tag{17}$$

where $\mathbf{w} > \mathbf{0}$ is a suitable weight vector. Note that $\mathbf{w}^T\mathbf{a}$ is a weighted l_1-norm of $\mathbf{a}$, since $\mathbf{a}$ is non-negative. An unweighted l_1-norm, $\mathbf{w} = \mathbf{1}$, is often used in RBF networks and probabilistic mixture models, see [1], [15]. Other choices of weighting $\mathbf{w}$ will be discussed in section VIII-F.

B.1 Comparison between modes

Although the the two modes use a similar mapping, there are distinctive differences, as illustrated in table I.

The major difference is in how they deal with *zero*, or more properly with *no information* and with interpolation. This depends upon whether the output is a channel vector or not. If it is, position encoding will be used with use of confidence. This should be viewed as the normal mode of operation for this architecture, as this forms the appropriate representation for continued processing at a next level. A conversion back to scalars, as described in section IV-E, is used mainly for easier interpretation of the states or for control of some linear actuator.

C. *Comparison with other methods*

The idea of localized kernels has been widely used in the field of neural networks. The continuous function model (17) with fixed kernel parameters is a simple form of RBF-network. The discrete event model (14) on the other hand is to our knowledge quite different from kernel machines. Using a channel representation of the output variables, as well as the input variables, we are able to handle problems that other kernel machines cannot. In function approximation, the discrete event mode can *simultaneously* handle discontinuities and continuous regions. The reason is the representation form of the channel reponse vector. The channel vector $\mathbf{u}$ can represent two values near a discontinuity. A global decoding would average the two values, similar to the behaviour in continuous mode and other neural networks. The discrete event mode can consequently be used in cases where we have multiple events, or where there are alternative interpretations. For example, a system designed to estimate object view from image features can give alternative statements at a lower level, an ambiguity which can be resolved at a more global level. Another problem that produces multiple, conflicting, events is perceptual aliasing. These properties will be demonstrated in the experiments section IX.

VIII. Associative Structure

We will now turn to the problem of estimating the linkage matrix $\mathbf{C}$ in (17) and in (14). We take on a unified approach for the two system operation modes. The models can be summarized into

$$\mathbf{u} = \mathbf{C}\frac{1}{s(\mathbf{a})}\mathbf{a}\,, \tag{18}$$

where $s(\mathbf{a})$ is a normalization function, and $\mathbf{u}$ denotes a scalar or a vector, representing either the explicit state variable/variables, or a channel representation thereof. In continuous function mode (17) $\mathbf{u} = \mathbf{x}$ and $s(\mathbf{a}) = \mathbf{w}^T\mathbf{a}$. In discrete event mode (14) $\mathbf{u}$ is a channel representation of $\mathbf{x}$ and $s(\mathbf{a}) \equiv 1$.

In the subsequent discussion, we will limit the scope to a supervised learning framework. Still, the structure can advantageously be used as a core in systems for other strategies of learning, such as reinforcement learning, with a proper embedding [16]. This discussion will assume *batch mode training*. This implies that there are N observation pairs of corresponding feature vectors $\mathbf{a}_n$ and state or response vectors $\mathbf{u}_n$. Let $\mathbf{A}$ and $\mathbf{U}$ denote the matrices containing all feature vector and response vector samples respectively, i.e.

$$\begin{cases} \mathbf{U} = \begin{pmatrix} | & | & & | \\ \mathbf{u}_1 & \mathbf{u}_2 & \dots & \mathbf{u}_N \\ | & | & & | \end{pmatrix} = \begin{pmatrix} - & \mathbf{u}^1 & - \\ - & \mathbf{u}^2 & - \\ & \vdots & \\ - & \mathbf{u}^K & - \end{pmatrix} \\ \mathbf{A} = \begin{pmatrix} | & | & & | \\ \mathbf{a}_1 & \mathbf{a}_2 & \dots & \mathbf{a}_N \\ | & | & & | \end{pmatrix} = \begin{pmatrix} - & \mathbf{a}^1 & - \\ - & \mathbf{a}^2 & - \\ & \vdots & \\ - & \mathbf{a}^H & - \end{pmatrix} \end{cases} \tag{19}$$

For a set of observation samples collected in accordance with (19), the model in (18) can be expressed as

$$\mathbf{U} = \mathbf{C}\mathbf{A}\mathbf{D}_s\,, \tag{20}$$

where

$$\mathbf{D}_s = \operatorname{diag}^{-1}\begin{pmatrix} s(\mathbf{a}_1) & s(\mathbf{a}_2) & \dots & s(\mathbf{a}_N) \end{pmatrix}\,. \tag{21}$$

The linkage matrix $\mathbf{C}$ is computed as a solution to a least squares problem, with the constraint $\mathbf{C} \geq \mathbf{0}$. We will motivate this constraint before we go into the optimization procedure.

A. Regularization properties

An unrestricted least squares solution tends to give a full matrix with negative and positive coefficients, which do their best to push and pull the basis functions to minimize the error for the particular training set, a.k.a. overfitting. This requires some form of regularization to make the problem well-posed.

In many situations a linkage matrix with a small norm is preferred, e.g. in Tikhonov regularization, because a small norm reduces the global noise propagation. However, this is not a primary goal in an architecture based on locality. Rather, it is desirable that a large error, due to drop-outs, heavy noise etc., in one region of the state space should not affect the performance in other, sufficiently distant regions of the state space. This can be achieved using a non-negativity constraint. While non-negativity does not give the smallest possible error on the training set, nor the smallest norm of the linkage matrix, it acts as a regularization. The non-negativity constraint will consequently be referred to as *monopolar regularization*. It is well known that this constraint gives a regularization in image reconstruction applications, see e.g. [17]. Monopolar regularization has the great advantage that it is not determined by prespecified parameters, but dependent upon the feature functions. Monopolarity in addition gives a more sparse linkage matrix, allowing a faster processing.

Figure 10 illustrates typical behavior of Tikhonov regularization and monopolar regularization for the type of monopolar localized functions used in discrete event mode. A response channel function u^k is to be approximated by a linear combination of a set of feature functions a^h. The feature functions are randomly placed and have a random width within a certain range. As before, let $\mathbf{u}^k$ denote a row vector with samples from u^k, and $\mathbf{A}$ denote a matrix where row h contains corresponding samples from a^h. Assume that we want to find a linkage vector $\mathbf{c}^k$ such that $\mathbf{u}^k \approx \mathbf{c}^k\mathbf{A}$. A solution using Tikhonov regularization is computed as

$$\mathbf{c}_{\text{Tikh}} = \arg\min_{\mathbf{c}^k} \|\mathbf{u}^k - \mathbf{c}^k\mathbf{A}\|^2 + \gamma\|\mathbf{c}^k\|^2\,, \tag{22}$$

while a solution using monopolar regularization is computed as

$$\mathbf{c}_{\text{mono}} = \arg\min_{\mathbf{c}^k \geq \mathbf{0}} \|\mathbf{u}^k - \mathbf{c}^k\mathbf{A}\|^2\,. \tag{23}$$

The parameter γ in (22) is chosen such that both methods give the same relative error $\|\mathbf{u}^k - \mathbf{c}^k\mathbf{A}\|/\|\mathbf{u}^k\|$, which in this case became 5%, i.e. both methods perform quite well.

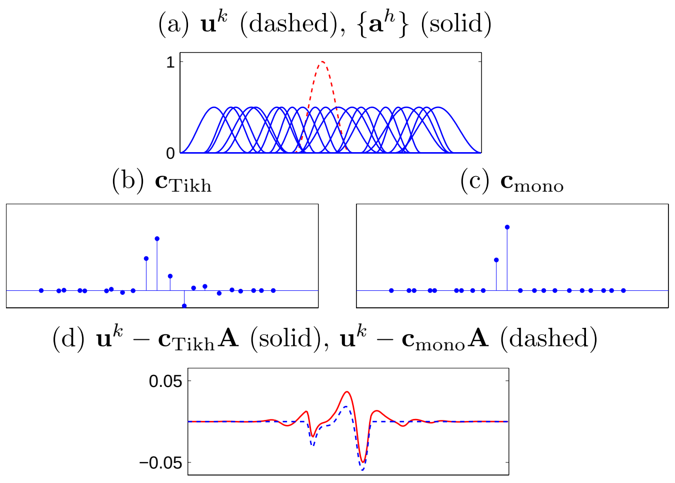

Fig. 10. Comparison between Tikhonov regularization and monopolar regularization. (a) Response channel function $\mathbf{u}^k$, and feature functions $\mathbf{a}_k$. (b,c) Linkage vectors $\mathbf{c}_{\text{Tikh}}$ and $\mathbf{c}_{\text{mono}}$ from (22) and (23) respectively. Each link is plotted at the center of its corresponding function a_k. (d) Response errors.

Tikhonov regularization tends to produce non-sparse, non-local, solutions which give a computationally more complex system. We can also observe ringing effects which typically occur for linear methods. The non-negativity constraint on the other hand gives a solution with a slightly larger norm, $\|\mathbf{c}_{\text{mono}}\|/\|\mathbf{c}_{\text{Tikh}}\| = 1.1$, but it is both sparse and local. Note that only two elements in $\mathbf{c}_{\text{mono}}$ are nonzero. Also worth mentioning is that an unrestricted solution, i.e. setting $\gamma = 0$, gives a slightly better approximation of $\mathbf{u}^k$, but a much worse solution $\mathbf{c}^k$ with respect to generalization power and noise robustness (not shown here).

The monopolar regularization is most efficient in discrete event mode, although it may be useful in continuous function mode as well, especially if the function consists of localized nonzero regions similar to the channel functions. For partially negative scalar functions, a positive offset has to be added and later subtracted.

B. Loosely coupled subsystems

The terms *loosely coupled* and *weakly coupled* are used in control theory to describe systems which consist of a number of subsystems, which only share a subset of the state variables and the feature variables available for the entire system, see e.g. [18], [19]. A subsystem will partially share variables with some limited set of other subsystems, while being totally independent of the remaining subsystems. The degree to which variables are shared among subsystems can be expressed as a *state distance* or *coupling metric* between subsystems. Subsystems at a larger distance according to this metric will not affect each other, e.g. in a process of optimization.

The architecture described here takes advantage of this property. The local behavior of the feature and response channels, together with the monopolar constraint on the linkage matrix, implies that the solution for one local region of the response space does not affect the solution in other, sufficiently distant, regions of the response space. For example, the mapping to a certain response channel will only depend on the features that are active somewhere within the active domain of that response channel. No other features will be linked to that response. This also implies that there can be a bad performance in the mapping accuracy locally in the response space, without this affecting the performance in the remaining part of the space.

This property is also essential as we want to add new features after training the system, such as in the implementation of an incremental learning procedure. A new feature that is never active within the active domain of a response channel will not affect the previously learned links to that response channel.

This is very different from the unconstrained case used in most learning procedures, where the linkage elements are allowed to assume any values. In this case, every added feature, regardless of whether it is active or not within some domain of the response space, may affect the solution for the entire response space. This global, tightly coupled, dependence is found to various degrees in many artificial networks such as multilayer perceptrons or RBF networks, and this makes them less attractive for complex problems.

Another important advantage with loosely coupled subsystems structures is that the iterative procedure to compute the model parameters, in this case the linkage matrix, exhibits fast convergence.

To summarize, there may always be regions of the state space which can not be reconstructed well. Still, large errors within these will not affect the performance within other parts of the state space.

C. Comparison with other methods

The discrete event mode is somewhat similar to fuzzy systems, see e.g. [20], [21], although fuzzy systems are used in low-dimensional problems only. The system in discrete event mode can almost, due to the monopolar constraint, be thought of as built up by a set of weighted fuzzy rules. This makes it easier to interpret the system behaviour, a problem which has recieved some attention within the machine learning community, see e.g. [22]. An ordinary fuzzy system can be thought of as having a sparse binary linkage matrix, and an optimization made with respect to the kernel parameters.

We can also observe that the decoding method in section IV-E corresponds to the defuzzification step in fuzzy systems. An important difference, is that fuzzy systems typically use a global centroid computation, while our decoding is local, and thus allows multiple output values.

If our system was to use Tikhonov regularization instead of monopolar regularization we would get a much less sparse linkage matrix, and the system would be more complex to interpret. In addition, the non-sparseness would make the system more computationally complex and tightly coupled.

D. Optimization procedure

Motivated by the discussion in sections VIII-A and VIII-B, $\mathbf{C}$ is computed as the solution to the constrained weighted least-squares problem

$$\min_{\mathbf{C}\geq\mathbf{0}} e(\mathbf{C}) , \tag{24}$$

where

$$\begin{aligned} e(\mathbf{C}) &= \|\mathbf{U} - \mathbf{CAD}_s\|^2_{\mathbf{W}} \\ &= \text{trace}(\mathbf{U} - \mathbf{CAD}_s)\mathbf{W}(\mathbf{U} - \mathbf{CAD}_s)^T . \end{aligned} \tag{25}$$

The weight matrix $\mathbf{W}$, which controls the relevance of each sample, is chosen $\mathbf{W} = \mathbf{D}_s^{-1}$. The minimization problem (24) does not generally have a unique solution, as it can be under-determined or over-determined.

There are several ways to solve sparse box constrained optimization problems, see e.g. [23], [24]. However, the high dimensionality largely reduces the number of methods which can be efficiently applied. The method used here is related to, but not covered by, the theory in [25]. The solution is derived from the projected Landweber method

$$\begin{cases} \mathbf{C}(0) &= \mathbf{0} \\ \mathbf{C}(i+1) &= \max\left(\mathbf{0}, \mathbf{C}(i) - \nabla e(\mathbf{C}(i))\mathbf{D}_f\right) , \end{cases} \tag{26}$$

where $\mathbf{D}_f$ is the positive definite diagonal matrix

$$\mathbf{D}_f = \text{diag}(\mathbf{v})\text{diag}^{-1}(\mathbf{AD}_s\mathbf{A}^T\mathbf{v}) \quad \text{for some } \mathbf{v} > \mathbf{0} . \tag{27}$$

Since $\mathbf{W} = \mathbf{D}_s^{-1}$ we have

$$\begin{aligned} \nabla e(\mathbf{C}) &= (\mathbf{CAD}_s - \mathbf{U})\mathbf{WD}_s\mathbf{A}^T \\ &= (\mathbf{CAD}_s - \mathbf{U})\mathbf{A}^T , \end{aligned} \tag{28}$$

and we rewrite sequence (26) as

$$\mathbf{C}(i+1) = \max\left(\mathbf{0}, \mathbf{C}(i) - (\mathbf{C}(i)\mathbf{AD}_s - \mathbf{U})\mathbf{A}^T\mathbf{D}_f\right) . \tag{29}$$

We can interpret $\mathbf{D}_s$ and $\mathbf{D}_f$ as normalizations in the sample and feature domain respectively. See section VIII-F for further details. We will consequently refer to $\mathbf{D}_s$ as *sample domain normalization* and $\mathbf{D}_f$ as *feature domain normalization*.

Note that if $\mathbf{v}$ is an eigenvector with eigenvalue λ to the matrix $\mathbf{AD}_s\mathbf{A}^T$, then we get the ordinary gradient search method $\mathbf{D}_f = (1/\lambda)\mathbf{I}$. It is well known that ordinary gradient search without constraints converges for $\mathbf{D}_f = \alpha\mathbf{I}$ if $0 < \alpha < 2/\lambda_{\max}$, where $\lambda_{\max}$ is the largest eigenvalue to $\mathbf{AD}_s\mathbf{A}^T$. An advantage using (26) is that we do not have to estimate the largest eigenvalue, which can be a computationally difficult task for large matrices.

$\mathbf{D}_f$ is sometimes called a *preconditioner*, and if suitably chosen (not necessarily as a diagonal matrix) it can allow a considerable acceleration of the Landweber methods, see e.g. [25]. It is generally a good idea to choose $\mathbf{D}_f$ such that it in some way mimics the behaviour of the inverse of $\mathbf{AD}_s\mathbf{A}^T$. It is easy to see that the choice (27) gives $\mathbf{D}_f\mathbf{AD}_s\mathbf{A}^T\mathbf{v} = \mathbf{v}$. This means that $\mathbf{D}_f$ is a (local) inverse in the neighborhood of $\mathbf{v}$. Hence, we should expect a fast convergence if the solution, i.e. each row in $\mathbf{C}$, has a similar direction as $\mathbf{v}$. Note that we are at least guaranteed that the solution lies in the same hyper-quadrant as $\mathbf{v}$, since they are both non-negative. Although not theoretically proven, experiments show that preconditioner (27) combined with the locality exhibited by the data gives a very fast convergence.

The convergence of sequence (29) and other related methods are discussed and proven in [26]. We reproduce the major steps below. Define $\mathbf{B} := \mathbf{D}_f\mathbf{AD}_s\mathbf{A}^T$, and let $\rho(\cdot)$ denote the *spectral radius*, i.e. the largest, in absolute value, eigenvalue.

Theorem 1 *Assume that $\rho(\mathbf{B}) < 2$. Then sequence (29) converges to a solution of problem (24).*

(See [26] for a proof.) In order to verify the inequality $\rho(\mathbf{B}) < 2$, we use the following auxiliary lemma.

Lemma 1 *Let $\mathbf{G}$ be an non-negative matrix. Assume there exist a vector $\mathbf{w} > 0$ and a scalar $\kappa > 0$ such that $\mathbf{Gw} \leq \kappa\mathbf{w}$. Then the spectral radius $\rho(\mathbf{G}) \leq \kappa$.*

(See [26] for a proof.) As mentioned earlier we have that $\mathbf{Bv} = \mathbf{v}$ and, hence, $\mathbf{B}$ fulfills the conditions in Lemma 1 with $\kappa = 1$ and the convergence follows from Theorem 1. Note that the non-negativity of $\mathbf{A}$ paved the way for the non-negativity of $\mathbf{B}$, which is needed in Lemma 1.

The issue of stopping rules is no different here from other cases of optimization, and we will not go into any details. All Landweber methods have a property of *semiconvergence* for noisy data, meaning that the performance on other data than the training data typically improves at first and later deteriorates. This problem is reduced by regularization, and two types are usually suggested; Tikhonov regularization and early termination. We use the latter with a stop criterion based either on experience or by monitoring the performance on a validation set.

As a final remark, note that problem (24) implies independent optimization of each response, i.e. the optimization is performed locally in the response domain. Hence we may, as an equivalent alternative, optimize each linkage vector $\mathbf{c}^k$ separately as

$$\mathbf{c}^k(i+1) = \max\left(\mathbf{0}, \mathbf{c}^k(i) - (\mathbf{c}^k(i)\mathbf{AD}_s - \mathbf{u}^k)\mathbf{A}^T\mathbf{D}_f\right) , \tag{30}$$

which then solves the problem

$$\min_{\mathbf{c}^k\geq\mathbf{0}} \left\|\mathbf{u}^k - \mathbf{c}^k\mathbf{AD}_s\right\|^2_{\mathbf{W}=\mathbf{D}_s^{-1}} . \tag{31}$$

E. Further Improvements

Coefficients of matrix $\mathbf{C}$ which are below certain thresholds $\mathbf{C}_{\min}$ shall be eliminated altogether, as this gives an even more sparse matrix. The thresholding is computed every now and then during the iterative procedure. The increase of error on new data turns out to be minimal, as

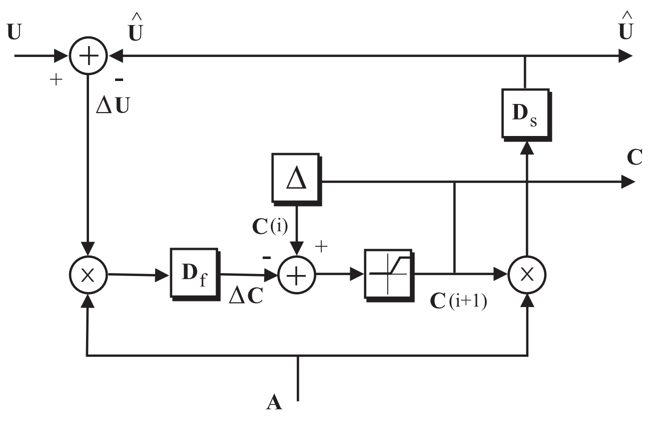

Fig. 11. Block diagram of the associative structure.

the iterative optimization will try to compensate for this removal of coefficients.

Furthermore, the coefficients of matrix $\mathbf{C}$ may be limited in magnitude, as this gives a more robust system as well. One way to achieve this is to use a constraint $\mathbf{C} \leq \mathbf{C}_{\max}$. The assumption is that significant feature values, a^h, shall be within some standard magnitude range, and consequently the maximum value of a^h shall be above some minimum value $a_{\min}$. This leads to an alternative, more computationally efficient, procedure to remove feature values in $\mathbf{A}$, which are below $a_{\min}$. This is related to the thresholds of a sigmoid curve, often assumed for the transfer function of real and artificial neurons.

A block diagram summarizing the associative structure is given in figure 11. The particular features departing from conventional approaches are centered around the nonlinearities acting to produce the monopolar linkage coefficients $\mathbf{C}$, and the normalization matrices $\mathbf{D}_f$ and $\mathbf{D}_s$; subject of next section.

F. Normalization modes

The normalization can be viewed as a way to control the gain in the feedback system loop, which the iterative optimization procedure implies, see figure 11. This normalization can be put in either of the two representation domains, the *sample domain* or the *feature domain*, but with different effects upon convergence, accuracy, etc. For each choice of sample domain normalization $\mathbf{D}_s$ there are non-unique choices of feature domain normalizations $\mathbf{D}_f$ such that sequence (29) converges to a solution of problem (24). $\mathbf{D}_f$ can for example be computed from (27). The choice of normalization depends on the operation mode, i.e. continuous function mapping or discrete event mapping. There are some choices that are of particular interest. These are discussed in sections VIII-F.1 and VIII-F.2 and summarized in section VIII-G.

F.1 Discrete event mapping

Discrete event mode (14) corresponds to a sample domain normalization matrix

$$\mathbf{D}_s = \mathbf{I}\,. \tag{32}$$

Choosing $\mathbf{v} = \mathbf{1} = (1\ 1\ \ldots\ 1)^T$ in (27) gives

$$\mathbf{D}_f = \mathrm{diag}^{-1}(\mathbf{A}\mathbf{A}^T\mathbf{1}) = \begin{pmatrix} \frac{1}{\mathbf{a}^1\mathbf{m}_\mathrm{f}^T} & & \\ & \frac{1}{\mathbf{a}^2\mathbf{m}_\mathrm{f}^T} & \\ & & \ddots \end{pmatrix}, \tag{33}$$

where $\mathbf{m}_\mathrm{f} = \sum_h \mathbf{a}^h$ is proportional to the mean in the feature domain. As $\mathbf{D}_s$ does not contain any components of $\mathbf{A}$ there is no risk that it turns singular in domains of samples having all feature components zero.

This choice of normalization will be referred to as *Normalization entirely in the feature domain.*

F.2 Continuous function mapping

There are several ways to choose $\mathbf{w}$ in the continuous function model (17), depending on the assumptions of error models, and the resulting choice of confidence measure s. One approach is to assume that all training samples have the same confidence, i.e. $s \equiv 1$, and compute $\mathbf{C} \geq \mathbf{0}$ and $\mathbf{w} \geq \mathbf{0}$ such that

$$\begin{cases} \mathbf{1} & \approx & \mathbf{w}^T\mathbf{A} \\ \mathbf{X} & \approx & \mathbf{C}\mathbf{A}\,. \end{cases} \tag{34}$$

Sometimes it may be desirable to have an individual confidence measure for each training sample. Another approach is to design a suitable $\mathbf{w}$ and then compute $\mathbf{C}$ using the optimization framework in section VIII-D with $s(\mathbf{a}) = \mathbf{w}^T\mathbf{a}$.

There are two specific designs of $\mathbf{w}$ that are worth emphasizing. The channel representation implies that large feature channel magnitudes indicate a higher confidence than low values. We can consequently use the sum of the feature channels as a measure of confidence:

$$s(\mathbf{a}) = \mathbf{1}^T\mathbf{a} \quad \Rightarrow \quad \mathbf{x} = \mathbf{C}\frac{1}{\mathbf{1}^T\mathbf{a}}\mathbf{a}\,. \tag{35}$$

As mentioned before, this model is often used in RBF-networks and probabilistic mixture models, see [1], [15]. The corresponding sample domain normalization matrix is

$$\mathbf{D}_s = \mathrm{diag}^{-1}(\mathbf{A}^T\mathbf{1}) = \begin{pmatrix} \frac{1}{\mathbf{a}_1^T\mathbf{1}} & & \\ & \frac{1}{\mathbf{a}_2^T\mathbf{1}} & \\ & & \ddots \end{pmatrix}, \tag{36}$$

and if we choose $\mathbf{v} = \mathbf{1}$ in (27) we get

$$\mathbf{D}_f = \mathrm{diag}^{-1}(\mathbf{A}\mathbf{1}) = \begin{pmatrix} \frac{1}{\mathbf{a}^1\mathbf{1}} & & \\ & \frac{1}{\mathbf{a}^2\mathbf{1}} & \\ & & \ddots \end{pmatrix}. \tag{37}$$

This choice of model will be referred to as *Mixed domain normalization.*

It can also be argued that a feature element which is frequently active should have a higher confidence than a feature element which is rarely active. This can be included in the confidence measure by using a weighted sum of the features, where the weight is proportional to the mean in the sample domain:

TABLE II

Overview of some models, together with their corresponding normalization matrices and preferred operation mode.

Normalization mode	Model	Sample Domain Normalization	Feature Domain Normalization	Operation mode
Normalization entirely in the feature domain	$\mathbf{u} = \mathbf{C}\mathbf{a}$	$\mathbf{D}_s = \mathbf{I}$	$\mathbf{D}_f = \operatorname{diag}^{-1}(\mathbf{A}\mathbf{A}^T\mathbf{1})$	Discrete event mapping
Mixed domain normalization	$\mathbf{x} = \mathbf{C}\frac{1}{\mathbf{1}^T\mathbf{a}}\mathbf{a}$	$\mathbf{D}_s = \operatorname{diag}^{-1}(\mathbf{A}^T\mathbf{1})$	$\mathbf{D}_f = \operatorname{diag}^{-1}(\mathbf{A}\mathbf{1})$	Continuous function mapping
Normalization entirely in the sample domain	$\mathbf{x} = \mathbf{C}\frac{1}{\mathbf{m}_s^T\mathbf{a}}\mathbf{a}$ $\mathbf{m}_s = \mathbf{A}\mathbf{1}$	$\mathbf{D}_s = \operatorname{diag}^{-1}(\mathbf{A}^T\mathbf{A}\mathbf{1})$	$\mathbf{D}_f = \mathbf{I}$	Continuous function mapping

$$s(\mathbf{a}) = \mathbf{m}_s^T\mathbf{a} \quad \text{where} \quad \mathbf{m}_s = \mathbf{A}\mathbf{1} = \sum_n \mathbf{a}_n\,. \tag{38}$$

This corresponds to the sample domain normalization matrix

$$\mathbf{D}_s = \operatorname{diag}^{-1}(\mathbf{A}^T\mathbf{A}\mathbf{1}) = \begin{pmatrix} \frac{1}{\mathbf{m}_s^T\mathbf{a}_1} & & \\ & \frac{1}{\mathbf{m}_s^T\mathbf{a}_2} & \\ & & \ddots \end{pmatrix}, \tag{39}$$

and by using $\mathbf{v} = \mathbf{A}\mathbf{1}$ in (27) we get

$$\mathbf{D}_f = \mathbf{I}\,. \tag{40}$$

This choice of model will be referred to as *Normalization entirely in the sample domain.*

G. Summary of normalization modes

The special cases mentioned in the previous section are summarized in table II. Note the symmetry between the three cases. Normalization entirely in the sample domain may be considered dual to the normalization entirely in the feature domain.

H. Sensitivity analysis for continuous function mode

We will now make some observations concerning the insensitivity to noise of the system, under the assumption of sample normalization in continuous function mode. That is, a response state estimate $\hat{\mathbf{x}}_n$ is generated from a feature vector $\mathbf{a}$ according to model (17), i.e.

$$\hat{\mathbf{x}}_n = \mathbf{C}\frac{1}{\mathbf{w}^T\mathbf{a}_n}\mathbf{a}_n\,. \tag{41}$$

We observe that regardless of choice of normalization vector $\mathbf{w}^T$, the response will be independent of any global scaling of the features, i.e.

$$\mathbf{C}\frac{1}{\mathbf{w}^T\mathbf{a}_n}\mathbf{a}_n = \mathbf{C}\frac{1}{\mathbf{w}^T\gamma\mathbf{a}_n}\gamma\mathbf{a}_n\,. \tag{42}$$

If multiplicative noise is applied, represented by a diagonal matrix $\mathbf{D}_\gamma$, we get

$$\hat{\mathbf{x}}_n = \mathbf{C}\frac{1}{\mathbf{w}^T\mathbf{D}_\gamma\mathbf{a}_n}\mathbf{D}_\gamma\mathbf{a}_n\,. \tag{43}$$

If the choice of weights in $\mathbf{C}$ and $\mathbf{w}$ is consistent, i.e. if the weights used to generate a response at a sample n were to obey the relation $\mathbf{C} = \hat{\mathbf{x}}_n\mathbf{w}^T$, the network is perfectly invariant to multiplicative noise. As we shall see in the experiments to follow, the normalization comes close to this ideal for the entire sample set, provided that the response signal varies slowly. For such situations, the network suppresses multiplicative noise well.

Similarly, a sensitivity analysis can be made for discrete event mode. We will in this presentation only refer to the discussion in section IV for the invariances available in the representation, and to the results from the experimental verification in following section.

IX. Experimental Verification

We will in this section analyze the behavior and the noise sensitivity of several variants of associative networks, both in continuous function mode and in discrete event mode. A generalization of the common CMU twin spiral pattern [27] has been used, as this is often used to evaluate classification networks. We have chosen to make the pattern more difficult in order to show that the proposed learning machinery can represent both continuous function mappings (regression) and mappings to discrete classes (classification). The robustness is analyzed with respect to three types of noise: additive, multiplicative, and impulse noise on the feature vector.

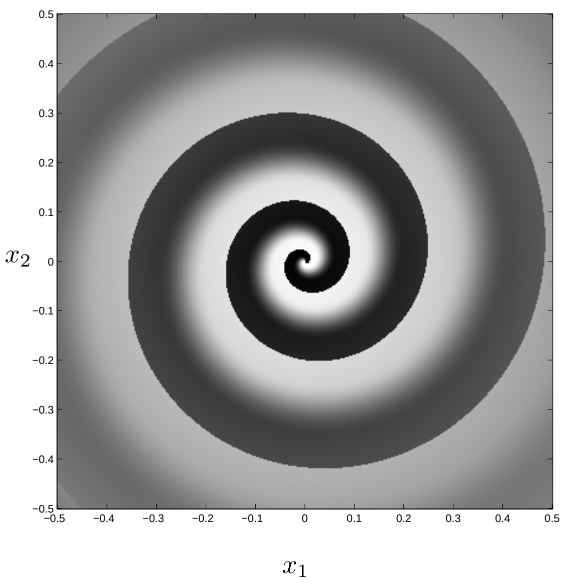

Fig. 12. Desired response function. Black to White correspond to values of $x_3 \in [-1, 1]$.

A. Experimental setup

In the experiments, a three dimensional state space $\mathcal{X} \subset \mathbb{R}^3$ is used. The sensor space $\mathcal{A} \subset \mathbb{R}^2$, and the response space $\mathcal{R} \subset \mathbb{R}$ are orthogonal projections of the state space. The network is trained to perform the mapping $f : \mathcal{A} \to \mathcal{R}$ which is depicted in figure 12. Note that this mapping can be seen as a surface of points $\mathbf{x} \in \mathbb{R}^3$, with $x_3 = f(x_1, x_2)$. The analytic expression for $f(x_1, x_2)$ is:

$$f_s(r, \varphi) = (1/\sqrt{2} - r)\cos(\varphi + \sqrt{1000r})$$
$$f(r, \varphi) = \begin{cases} f_s(r, \varphi) & \text{if} \quad \mathrm{mod}(\varphi + \sqrt{1000r}, 2\pi) < \pi \\ \mathrm{sign}(f_s(r, \varphi)) & \text{otherwise.} \end{cases} \tag{44}$$

Variables $r = \sqrt{x_1^2 + x_2^2}$ and $\varphi = \tan^{-1}(x_1, x_2)$ are the polar coordinates in sensor space $\mathcal{A}$. As can be seen in the figure, the mapping contains both smooth parts (given by the cos function) and discontinuities (introduced by the sign function). The pattern is intended to demonstrate the following properties:

1. The ability to approximate piecewise continuous surfaces.
2. The ability to describe discontinuities (i.e. assignment into discrete classes).
3. The transition between interpolation and representation of a discontinuity.
4. The inherent approximation introduced by the sensor channels.

As sensor channels, a variant of the channels prescribed in expression (11) is used:

$$\mathrm{B}^h(\mathbf{x}) = \begin{cases} \cos^2(\omega d) & \text{if} \quad \omega d \le \frac{\pi}{2} \\ 0 & \text{otherwise} \end{cases} \tag{45}$$

$$\text{where} \quad d = \sqrt{(\mathbf{x} - \mathbf{x}^h)^T \mathbf{M} (\mathbf{x} - \mathbf{x}^h)}, \tag{46}$$

and $\mathbf{M} = \mathrm{diag}(1\ 1\ 0)$. In the experiments $H = 2000$ such sensors are used, with random positions $\{\mathbf{x}^h\}_1^H$ inside the box $([-0.5, 0.5], [-0.5, 0.5]) \subset \mathcal{A}$. The sensors have channel widths of $\omega = \pi/0.14$ giving each an active domain with radius 0.07. Thus, for each state $\mathbf{x}_n$, a feature vector $\mathbf{a}_n = \left(\mathrm{B}^1(\mathbf{x}_n)\ \mathrm{B}^2(\mathbf{x}_n)\ \ldots\ \mathrm{B}^H(\mathbf{x}_n)\right)^T$ is obtained.

During training, random samples of the state vector $\mathbf{x}_n \in \mathcal{X}$ on the surface $f : \mathcal{A} \to \mathcal{R}$ are generated, and used to obtain pairs $\{f_n,\ \mathbf{a}_n\}$ using (44) and (45). The training sets are stored in the matrices $\mathbf{f}$, and $\mathbf{A}$ respectively. The performance is then evaluated on a regular sampling grid. This has the advantage that performance can be visualized as an image. Since real valued positions $\mathbf{x} \in \mathcal{X}$ are used, the training and evaluation sets are disjoint.

The mean absolute error (MAE) between the network output and the ground truth (44), is used as a performance measure,

$$\varepsilon_{\mathrm{MAE}} = \frac{1}{N}\sum_{n=1}^{N} |f(\mathbf{x}_n) - \mathbf{c}\mathbf{a}_n| \,, \tag{47}$$

or, for discrete event mode

$$\varepsilon_{\mathrm{MAE}} = \frac{1}{N}\sum_{n=1}^{N} |f(\mathbf{x}_n) - \mathrm{dec}(\mathbf{C}\mathbf{a}_n)| \,. \tag{48}$$

The rationale for using this error measure is that it is roughly proportional to the number of misclassifications along the black-to-white boundary, in contrast to RMSE which is proportional to the number of misclassifications squared.

B. Associative network variants

We will demonstrate the behavior of the following five variants of associative networks:

1. **Mixed domain normalization bipolar network**

This network uses the model

$$\hat{f} = \frac{1}{\mathbf{1}^T\mathbf{a}}\mathbf{c}\mathbf{a} \,.$$

This model is often used in RBF-networks and probabilistic mixture models, see [1], [15]. This network is optimized according to

$$\mathbf{c} = \arg\min_{\mathbf{c}} \|\mathbf{f} - \mathbf{c}\mathbf{A}\mathbf{D}_s\|^2 + \gamma\|\mathbf{c}\|^2 \,. \tag{49}$$

In the experiments, the explicit solution is used, i.e.

$$\mathbf{c} = \mathbf{f}\mathbf{A}\mathbf{D}_s(\mathbf{A}\mathbf{D}_s\mathbf{D}_s^T\mathbf{A}^T + \gamma\mathbf{I})^{-1} \,. \tag{50}$$

Note that for larger systems, it is more efficient to replace (50) with a gradient descent method.

2. **Mixed domain normalization monopolar network**
Same as above, but with a monopolar constraint on **c**, instead of the Tikhonov regularization used above.

3. **Sample domain normalization monopolar network**
This network uses the model

$$\hat{f} = \frac{1}{\mathbf{m}_s^T\mathbf{a}}\mathbf{c}\mathbf{a} \,,$$

where $\mathbf{m}_s$ is computed from the training set sensor channels according to $\mathbf{m}_s = \mathbf{A}\mathbf{1}$.

4. **Uniform sample confidence monopolar network**
This network uses the model

$$\hat{f} = \frac{1}{\mathbf{w}^T\mathbf{a}}\mathbf{c}\mathbf{a} \,,$$

where the mapping **w** is trained to produce the response 1 for all samples, see (34).

5. **Discrete event mode monopolar network**
This network uses the model

$$\hat{\mathbf{u}} = \mathbf{C}\mathbf{a} \quad \Leftrightarrow \quad \hat{f} = \mathrm{dec}(\mathbf{C}\mathbf{a}) \,,$$

with $K = 7$ channels. The responses should describe the interval $[-1, 1]$ so the decoding step involves a linear mapping, see (7).

C. Varied number of samples

As a demonstration of the generalization abilities of the networks we will first vary the number of samples. The monopolar networks are optimized according to section VIII, with 50 iterations. For the bipolar network we have used $\gamma = 0.005$. This value is chosen to give the same error on the training set as in network #2 using $N = 500$ samples.

The performance on the regular grid is demonstrated in figure 13 for the bipolar network (#1), and in figure 14 for the discrete event network (#5).

If we look at the center of the spiral, we see that both networks fail to describe the fine details of the spiral, although #1 is doing slightly better. For the discrete event network, the failure is a direct consequence of the feature channel sizes. For the bipolar network it is a combined consequence of the size and density of the feature channels.

We can also observe that the discrete event network is significantly better at dealing with the discontinuities. This is also reflected in the error measures, see figure 15. For very low numbers of samples, when both networks clearly fail, the bipolar network is slightly better. We have also plotted the performance of the monopolar mappings in continuous function mode. As can be seen in the plot, these are all slightly worse off than the bipolar network. All three monopolar continuous function mode variants have similar performances on this setup. Differences appear mainly when the sample density becomes non-uniform (not shown here).

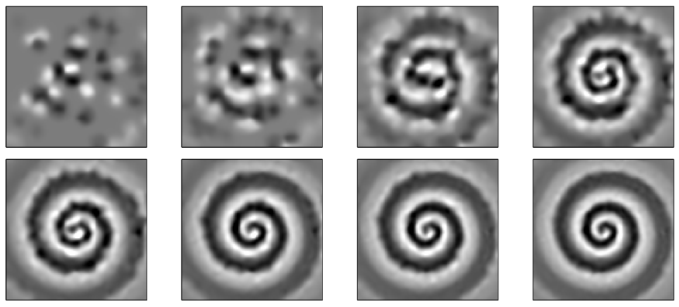
Fig. 13. Performance of bipolar network (#1) under varied number of samples. Top left to bottom right: N = 63, 125, 250, 500, 1000, 2000, 4000, 8000.

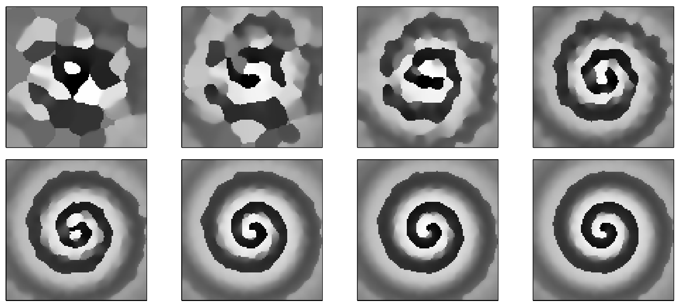
Fig. 14. Performance of discrete event network (#5) under varied number of samples. Top left to bottom right: N = 63, 125, 250, 500, 1000, 2000, 4000, 8000.

D. Varied number of channels

The relationship between the sizes of feature and response channels is important for the performance of the network. The distance between the channels also determines where the decision between interpolation and introduction of a discontinuity is made. We will now demonstrate these two effects by varying the number of channels in the range $[3 \ldots 14]$, and keeping the number of samples high, $N = 8000$.

As can be seen in figure 16, a low number of channels gives a smooth response function. For $K = 3$ no discontinuity is introduced at all, since there is only one interval for the local reconstruction (see section IV-E). As the number of channels is increased, the number of discontinuities increases. Initially this is an advantage, but for a large number of channels, the response function becomes increasingly patchy (see figure 16). In practice, there is thus a trade-off between description of discontinuities, and patchiness. This trade-off is also evident if MAE is plotted against the number of channels, see figure 17 left.

In figure 17, right part, error curves for smaller numbers of samples have been plotted. It can be seen that, for a given number of samples, the optimal choice of channels varies. Better performance is obtained for a small number

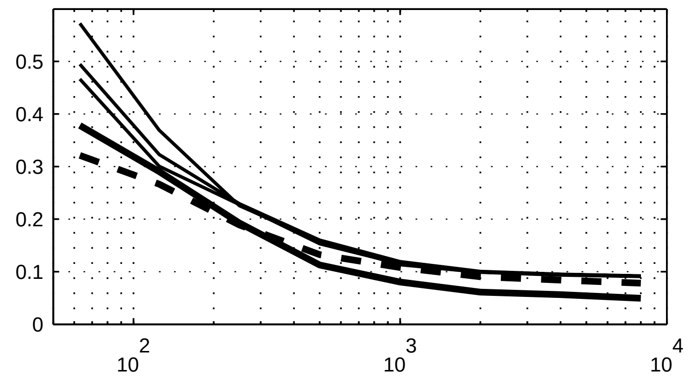

Fig. 15. MAE under varied number of samples. Solid thick is #5, and dashed is #1. Solid thin are #2,#3, and #4. For low number of samples the variants are ordered #2, #3, #4 with #4 being the best one.

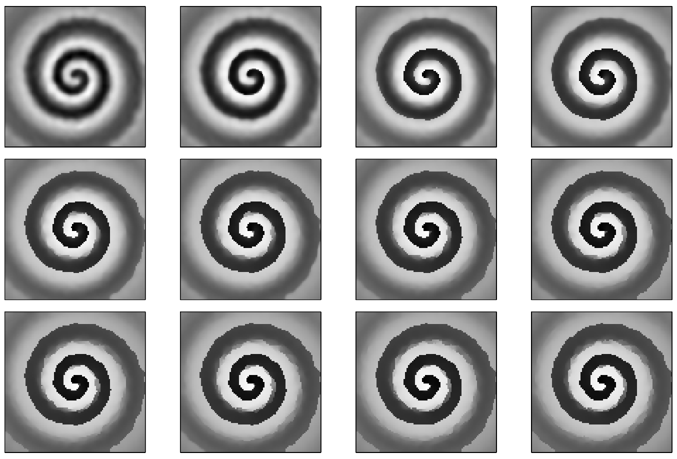

Fig. 16. Performance of discrete event network (#5) under varied number of channels. Top left to bottom right: $K = 3$ to $K = 14$

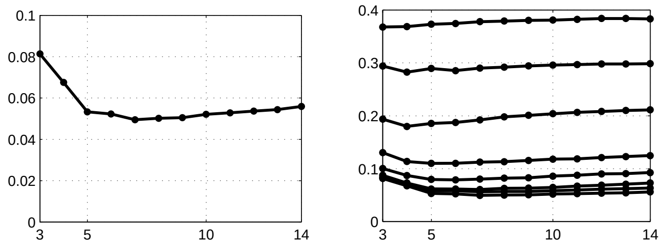

Fig. 17. MAE under varied number of channels. Left MAE for $N = 8000$. Right MAE for $N = 63$, 125, 250, 500, 1000, 2000, 4000, and 8000.

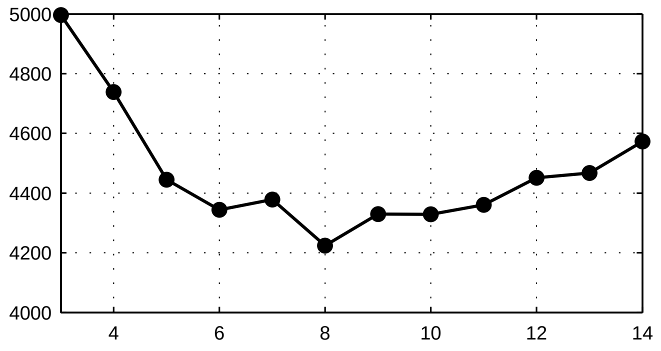

Fig. 18. Number of non-zero coefficients under varied number of channels. Compare this with 2000 non-zero coefficients for the continuous function networks.

of channels, when fewer samples are used. The standard way to interpret this result is that a high number of response channels allows a more complex model, which requires more samples.

If we plot the number of non-zero coefficients in the linkage matrix $\mathbf{C}$, we also see that there is an optimal number of channels, see figure 18. Note that although the size of $\mathbf{C}$ is between 3 and 14 times larger than in continuous function mode, the number of links only increases by a factor 2.1 to 2.5.

E. Noise sensitivity

We will now demonstrate the performance of the associative networks when the feature set is noisy. We will use the following noise models:

1. **Additive noise**: A random value is added to each feature value, i.e.

$$\mathbf{a}_n^* = \mathbf{a}_n + \boldsymbol{\eta}\,,$$

with $\eta^k \in \text{rect}[-p, p]$, and the parameter p is varied in the range $[0, 0.1]$.

2. **Multiplicative noise**: Each feature value is multiplied with a random value, i.e.

$$\mathbf{a}_n^* = \mathbf{D}_\eta \mathbf{a}_n\,,$$

where $\mathbf{D}_\eta$ is a diagonal matrix with with $(\mathbf{D}_\eta)_{kk} = \eta^k \in \text{rect}[1-p, 1+p]$, and the parameter p is varied in the range $[0, 1]$.

3. **Impulse noise**: A fraction of the features is set to 1, i.e.

$$a_n^{k,*} = \begin{cases} 1 & \text{if } f_r < p \text{ where } f_r \in \text{rect}(0,1) \\ a_n^k & \text{otherwise,} \end{cases}$$

and the parameter p is varied in the range $[0, 0.01]$.

The results of the experiments are shown in figure 19. We have consistently used $N = 4000$ samples for evaluation, and corrupted them with noise according to the discussion above. In order to make the amount of regularization comparable we have optimized the γ parameter for network #1 to give the same error on the training set as network #2 at $N = 4000$ samples. This gave $\gamma = 0.08$.

As can be seen from the additive noise experiment, network #5 has a different slope for its dependence upon noise level. The other networks are comparable, and differences are mainly due to how well the networks are able to represent the pattern in the first place (see section IX-C). For the multiplicative noise case, we see that the slope is similar for all networks. Thus we can conclude that the mul-

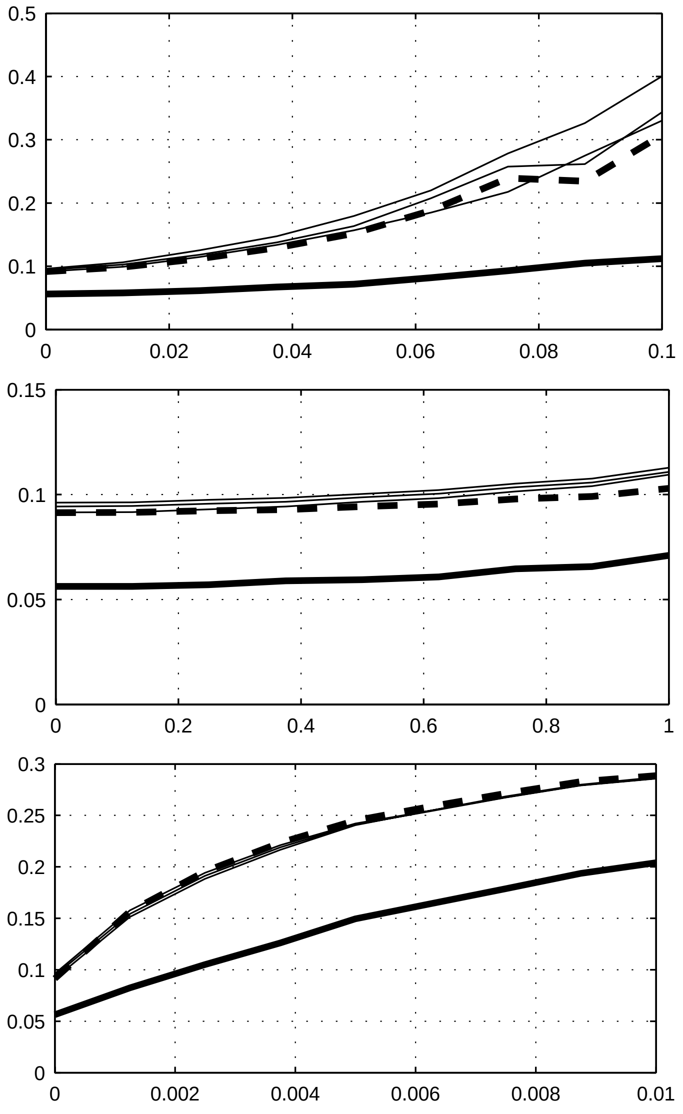

Fig. 19. Noise sensitivity. Top to bottom: additive noise, multiplicative noise, and impulse noise. Solid thick is #5, and dashed is #1. Solid thin are #2,#3, and #4.

tiplicative noise behavior is comparable for all tested networks. For the impulse noise case we can see that for small amounts of noise, network #5 has a less steep slope than the others. For larger amounts of noise however, all networks seem to behave in a similar manner.

The purpose of these experiments has been to demonstrate the abilities of the associative networks to generalize, and to cope with various kinds of sensor noise. Several experiments using image features as inputs have been made, but have to be excluded from this presentation. For details of such experiments, the reader is directed to [28], [4], [3].

X. Concluding Remarks

As shown in the experimental section, the channel learning architecture running in discrete event mode is able to describe simultaneously continuous and transiential phenomena, while still being better than or as good as a linear network at suppressing noise. An increase in the number of response channels does not cause an explosion in the number of used links. Rather, it remains fairly stable at approximately twice the number of links required for a continuous function mapping. This is a direct consequence of the monopolar constraint.

The training procedure shows a fast convergence. In the experiments described, a mere 50 iterations have been required. The fast convergence is due to the monopolar constraint, locality of the features and responses, and the choice of preconditioner in the Landweber method.

The learning architecture using channel information also deals properly with the perceptual aliasing problem, that is, it does not attempt to merge or average conflicting statements, but rather passes them on to the next processing level. This allows a second processing stage to resolve the perceptual aliasing, using additional information not available at the lower level.

The ability of the architecture to handle a large number of models in separate or loosely coupled domains of the state space, promises systems with a combination of the continuous mapping of control systems with the state complexity we have become familiar with from digital systems. Such systems can be used for the implementation of extremely complex, contextually controlled mapping model structures. One such application is for view based object recognition in computer vision [28].

XI. Acknowledgments

The authors would like to acknowledge many fruitful discussions with members of the Computer Vision Laboratory and members of the Dept. of Computer Science, Tommy Elfving and Vladimir Kozlov at the Dept. of Mathematics, and Yair Censor at the Dept. of Mathematics, Univ. of Haifa, Israel, who have also contributed ideas for further pursuit, some of which are the subject of other publications under preparation.

The authors also want to acknowledge the financial support of WITAS – The Wallenberg Laboratory for Information Technology and Autonomous Systems; The Foundation for Strategic Research, project VISIT – VIsual Information Technology; The Swedish Research Council, project *A New Structure for Signal Processing and Learning*; and NUTEK, The Swedish Board of Technical Development.

XII. Appendix: Implementation in Matlab

In order to support the claims made in the text, and to allow a smooth introduction into the use of the learning associative network, there is a toolbox available on the web-site of the Computer Vision Lab [29]. The toolbox is implemented in Matlab, and includes optimization routines, methods for channel encoding of scalars, a simple generative model, and a data set for experimentation. To demonstrate the function of the toolbox, a program which performs the experiments described in section IX is also included. For full scale software implementations, readers are asked to contact the authors.

References

[1] S. Haykin, *Neural Networks–A comprehensive foundation*, Prentice Hall, 2nd edition, 1999, ISBN 0-13-273350-1.

[2] G. H. Granlund, "An Associative Perception-Action Structure Using a Localized Space Variant Information Representation", in *Proceedings of Algebraic Frames for the Perception-Action Cycle (AFPAC)*, Kiel, Germany, September 2000.

[3] K. Nordberg, G. Granlund, and H. Knutsson, "Representation and Learning of Invariance", in *Proceedings of IEEE International Conference on Image Processing*, Austin, Texas, November 1994, IEEE.

[4] Per-Erik Forssén, "Sparse Representations for Medium Level Vision", Lic. Thesis LiU-Tek-Lic-2001:06, Dept. EE, Linköping University, SE-581 83 Linköping, Sweden, February 2001, Thesis No. 869, ISBN 91-7219-951-2.

[5] G. H. Granlund and H. Knutsson, *Signal Processing for Computer Vision*, Kluwer Academic Publishers, 1995, ISBN 0-7923-9530-1.

[6] Per-Erik Forssén, "Observations Concerning Reconstructions with Local Support", Tech. Rep. LiTH-ISY-R-2425, Dept. EE, Linköping University, SE-581 83 Linköping, Sweden, April 2002.

[7] Michael Felsberg, Hanno Scharr, and Per-Erik Forssén, "The B-spline channel representation: Channel algebra and channel based diffusion filtering", Tech. Rep. LiTH-ISY-R-2461, Dept. EE, Linköping University, SE-581 83 Linköping, Sweden, September 2002.

[8] I. Daubechies, "The wavelet transform, time-frequency localization and signal analysis", *IEEE Trans. on Information Theory*, vol. 36, no. 5, pp. 961–1005, September 1990.

[9] Hanno Scharr, Michael Felsberg, and Per-Erik Forssén, "Noise adaptive channel smoothing of low-dose images", in *CVPR Workshop: Computer Vision for the Nano-Scale Workshop*, June 2003, Accepted.

[10] Christopher M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, 1995, ISBN 0-19-853864-2.

[11] K.-R. Müller, G. Rätsch, K. Tsuda, and B. Schölkopf, "An introduction to kernel-based learning algorithms", *IEEE Neural Networks*, vol. 12, no. 2, pp. 181–201, May 2001.

[12] G. Baudat and F. Anouar, "Kernel-based Methods and Function Approximation", in *Internation Joint Conference on Neural Networks (IJCNN)*, Washington DC USA, 2001, pp. 1244–1249.

[13] Gösta Granlund, "Does Vision Inevitably Have to be Active?", in *Proceedings of the 11th Scandinavian Conference on Image Analysis*, Kangerlussuaq, Greenland, June 7–11 1999, SCIA, Also as Technical Report LiTH-ISY-R-2247.

[14] Lonnie Chrisman, "Reinforcement Learning with Perceptual Aliasing: The Perceptual Distinctions Approach", in *National Conference on Artificial Intelligence*, 1992, pp. 183–188.

[15] J. Moody and C. J. Darken, "Fast learning in networks of locally-tuned processing units", *Neural Computation*, vol. 1, pp. 281–293, 1989.

[16] Richard S. Sutton and Andrew G. Barto, *Reinforcement Learning, An Introduction*, MIT Press, Cambridge, Massachusetts, 1998, ISBN 0-262-19398-1.

[17] J. G. Nagy and Z. Strakos, "Enforcing nonnegativity in image reconstruction algorithms", in *Mathematical Modelling, Estimation and Imaging, D.C. Wilson et. al., eds., SPIE Proceedings Series*, 2000, vol. 4121, pp. 182–190.

[18] Z. Gajic and I. Borno, "General transformation for block diagonalization of weakly coupled linear systems composed of n-subsystems", *IEEE Transactions on Circuits and Systems - I: Fundamental Theory and Applications*, vol. 47, no. 6, pp. 909–912, June 2000.

[19] N. Derbel, "Optimal control of linear weakly coupled systems", in *Proceedings of the 34th Conference on Decision & Control*, New Orleans, LA, December 1995, IEEE, pp. 643–648.

[20] Bart Kosko, "Fuzzy systems as universal approximators", *IEEE Transactions on Computers*, vol. 43, no. 11, pp. 1329–1333, November 1994.

[21] S. Mitaim and B. Kosko, "What is the best shape for a fuzzy set in function approximation?", in *5th IEEE International Conference on Fuzzy Systems (FUZZ-96)*, September 1996, pp. 1237–1243.

[22] S. R. Gunn and J.S. Kandola, "Structural Modelling with Sparse Kernels", *Machine learning*, vol. 48, no. 1, pp. 137–163, 2002.

[23] Å. Björck, *Numerical Methods for Least Squares Problems*, SIAM, Society for Industrial and Applied Mathematics, 1996.

[24] M. Adlers, *Topics in Sparse Least Squares Problems*, Ph.D. thesis, Linköping University, Linköping, Sweden, Dept. of Mathematics, 2000, Dissertation No. 634.

[25] M. Piana and M. Bertero, "Projected landweber method and preconditioning", *Inverse Problems*, , no. 13, pp. 441–463, 1997.

[26] Björn Johansson, Tommy Elfving, Vladimir Kozlov, Yair Censor, and Gösta Granlund, "An Oblique-Projected Landweber Method with Application to Learning", 2003, Submitted to Inverse Problems.

[27] CMU Neural Networks Benchmark Collection, `http://www.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/neural/bench/cmu/`.

[28] Gösta H. Granlund and Anders Moe, "Unrestricted recognition of 3-D objects using multi-level triplet invariants", in *Proceedings of the Cognitive Vision Workshop*, Zürich, Switzerland, September 2002.

[29] CVL web page, `http://www.isy.liu.se/cvl/Projects/hiperlearn/`.