

A theoretical framework for multiple
neural network systems

Mike W. Shields, Matthew C. Casey

June 2006



**University
of Surrey**

Department of Computing

Computing
Sciences
Report

CS-06-03

A theoretical framework for multiple neural network systems

Mike W. Shields, Matthew C. Casey*

Department of Computing, School of Electronics and Physical Sciences, University of Surrey, Guildford, Surrey, GU2 7XH, UK

Abstract

Multiple neural network systems have been demonstrated to improve performance on tasks such as classification and regression when compared to single neural networks. Whilst there has been significant focus on understanding the theoretical properties of certain types of these multi-net systems, such as ensembles, there has been little theoretical work on understanding the properties of the generic combination of networks. In this paper we provide an abstract, formal framework in which the generic combination of neural networks can be described, and in which the properties of the system can be rigorously analyzed. We achieve this by describing multi-net systems in terms of partially ordered sets and state transition systems. By way of example, we explore an abstract version of backpropagation applied to a generic multi-net system that can combine an arbitrary number of networks in sequence and in parallel. By using the framework we show with a constructive proof that if it is possible to train the generic system, then training can be achieved by an abstract version of backpropagation.

Keywords: Multi-net systems; ensembles; multiple classifiers; partially ordered sets; state transition systems

1. Introduction

Neural network research has reached the stage where we have at least good understanding of the most popular architectures and algorithms (cf. [4]), and yet there remains a significant gap between the capabilities of such networks compared to the capability, say, of the human brain. Whilst improved learning algorithms allow us to achieve good levels of performance on tasks such as regression or classification, neural networks fall short of the equivalent complexity of biological systems. One way in which the performance and capability of artificial neural systems has been improved is through the combination of multiple networks [19,23]. These *multi-net systems* [33] are simply ways in which networks can be combined into a cohesive architecture, often with a specific learning algorithm (for example [6,13,20]), and whilst they still fall short of biological complexity, they have been used to demonstrate improved performance and capability in a number of areas, including regression [29], classification [23] and, perhaps more relevantly, computational modelling [19].

Whilst these combined systems have proven popular, our formal understanding of their properties and capabilities is not complete. For ensembles, our understanding depends upon the task, either regression or

classification [8]. Other architectures require a more specific configuration of the components and choice of learning algorithm [22]. In general this means that the choice of an optimum topology and parameterization is a matter for experience driven trial-and-error, often without knowing if convergence to a stable solution is possible, or whether a simpler solution is better. To overcome this, a formal understanding of these systems is therefore desirable to remove such uncertainties. Whilst statistical analysis is proving useful for specific architectures [7,14,34,37], there has been little work on expanding this to a generic framework in which all such multi-net systems may be described.

In this paper we provide an abstract, general theory of multi-net systems using a framework in which the generic combination of networks can be described, and in which the properties of the system can be rigorously analyzed. To achieve this we use partially ordered sets to describe the topology and parameterisation of the combination and components, together with notions of transition systems to describe the dynamics of the interaction of the components during the feedforward and learning phases. Such techniques have been used successfully in a number of domains. This novel application of set theory provides a framework in which we can prove properties of combined systems. In particular, we provide a

* Corresponding author. Tel./fax: +44 (0) 1483 689635.

E-mail addresses: m.casey@surrey.ac.uk (Matthew Casey), myramike@gmail.com (Mike Shields)

constructive proof of the existence of a supervised learning algorithm, based upon an abstract notion of backpropagation, which can be used to train a generic multi-net system, such that the parameters of the system converge to meet a specific criterion function. Importantly, this is achieved without resorting to specific details of the components involved, except to provide necessary constraints.

In section 2 of this paper we provide a background to multi-net systems and existing work on formalism. In section 3 we provide the framework to formally define the topology, parameterization and learning algorithm of a multi-net system. In section 4, we consider how an abstract form of backpropagation can be used to train a generic multi-net system for a supervised learning task. In section 5 we conclude with a discussion on the limitations of the proposed framework and put this into context with current work.

2. Multi-net systems

The idea of combining components together to form a cohesive system which is more capable than its individual parts is not a new one, even for neural networks. For example, Hebb's discussion on learning includes the concept of 'superordinate' systems built from integrating *cell assemblies* [18], which for us of combinations of networks. This idea of a combination of neural components has been used successfully for computational models of cognitive abilities (for example, [16,31]), appear to be a natural extension of the neural modelling paradigm, building from neurons, to layers, to networks [19], and some such architectures have even been linked to functional specialism [15].

Particular multi-net architectures, or in general combinations of statistical learning techniques, have demonstrated tangible performance improvement on tasks such as regression [29] and classification [23]. Broad types of combination include ensemble, modular and hybrid systems [33], but more detailed taxonomy do not recognize such clear divisions, with the behaviour of parallel, sequential and hybrid types of architecture dependent upon the process of learning [28]. Here, the choice of learning algorithm can make such systems co-operative (typically ensembles), competitive (modular), static (pre-configured prior to combination) or dynamic (configured in-situ), for the same or similar configurations [32].

Exemplars of modular systems are the mixture-of-experts (ME) and the hierarchical mixture-of-experts (HME) architectures [20,21], in which a task is automatically decomposed into sub-tasks solved by individual expert networks using a competitive algorithm to essentially allocated patterns to experts. With a particular configuration and algorithm, convergence properties are known [22,27], yet such a specific result

does not apply to the more general use of ME, some of which has proven popular for different simulations [2,12]

Ensemble systems have also proven popular with the development of algorithms such as AdaBoost [13] and negative correlation (NC) learning [25]. Whilst properties of ensembles for regression problems are mostly well understood [7,29], especially with the recent linkage of the Ambiguity [24] and the bias-variance-covariance [35] decompositions to learning in NC, which results in being able to understand how to construct an accurate regressor, given the use of a quadratic error function [9], there is only limited theory [14,34] for ensembles of classifiers (*multiple classifier systems*) with no complete understanding of how accurate ensemble classifiers can be constructed.

Whilst there is a good theoretical understanding of ME and HME, and a developing understanding of ensembles, there is no such theory for other types of multi-net system. Examples include sequential systems [11,26], hierarchical clustering techniques, or adaptive combinations [36]. Furthermore, architectures consisting of ad-hoc combinations of networks are also popular in computational models of brain function and behaviour, such as language [1], numeracy [16] and emotion. Yet despite a lack of understanding of the properties of the combined system, we know significantly more about the individual components, which it would be useful to apply to the combined system.

Attempts have been made to define a theoretical framework that can be used to describe a more general class of multi-net system. Parallel (cf. ensembles) and sequential combinations of networks were formalized by Bottou and Gallinari [5], in which they explored ways in which combinations of learning algorithms could be defined. Amari [3] defined a stochastic model of neural networks that was used to describe both single network and multi-net systems (the ME architecture), and which could be extended to other types of combination. In a similar way, the formal definition of the HME architecture and algorithm [21] could also be extended to more general types of multi-net system. Kittler, Hatef, Duin and Matas [23] also recognized the need for a theoretical framework for describing combinations of classifiers, but their work was restricted to a particular configuration, namely the parallel combination of classifiers (or networks), as used in an ensemble. Whilst each of these have abstracted some of the properties of the components, for example the types of component [23], none consider how properties of the whole system might be rigorously analysed. If we are to understand such properties of a more general class of multi-net system, important perhaps for the construction of improved models of the brain and other more complex systems, then we must develop a generic formal understanding of these systems and use knowledge of the components to infer properties of the whole – something

that can only be achieved in a rigorous formal framework.

In this paper we view multi-net systems from the perspective of set theory using partially ordered sets. The starting point for the work reported here is Casey [10]. He used directed trees to define the generic architecture of multi-net systems, together with a framework for the description of the associated learning algorithm. However, like previous work, this lacked sufficient rigor to explore the properties of such systems further. In this paper we build upon this work by considering combined systems generically by developing an abstract model of multi-nets. The model is abstract in that it represents multi-net topologies in terms of partially ordered sets, with weights considered non-numerically in terms of abstract parameterizations, together with activation functions dealt with as functions. We also formalize the notion of a feedforward state of the system (the production of an output given a set of inputs), and the concept of learning as a strategy to change the parameterization of the system to one that satisfies a predicate function (more properly, an extension of a predicate), modelling convergence to some defined state. We provide this formal description so that the abstract properties of the generic class of multi-net systems can be analyzed.

However, whilst providing a formal framework is useful to consistently define multi-net architectures, it is only really useful when used to discover properties of the combined system that were not previously known. Consequently, in this paper we present results from the application of the formalism to understanding how supervised learning can be achieved in a generic system. The system is generic in the sense that we do not specify topological constraints on the components, such as the type of network or neurons. In particular, we describe a simple multi-layer perceptron (MLP) as a multi-net system, go on to define the abstract notion of backpropagation learning in this system, and then prove that, if we have any type of feedforward multi-net system used for a supervised learning task, that if we can precisely define the convergence predicate, then there exists a learning algorithm, based upon the abstract form of backpropagation, that can be used to train the system. Whilst this is perhaps intuitive given that we have defined an MLP as a multi-net, the significance is in the generality of the result: we do not require that the system consists of neurons, perceptrons or indeed, and other type of network, rather, our constraint lies in being able to describe the system using the formalism only, with the concepts of feedforward stable states, but only in the necessity to define a suitable convergence function. The limitation is that, whilst this is a demonstration of the usefulness of abstracting combinations of networks, in order to prove practical, the detail of the strategy and associated predicate must be provided, something that we do not provide, since we only show existence.

3. From multi-nets to state transition systems

In this section we present the basic mathematical concepts underlying the work presented here. The first of these is that of a partially ordered set, which we use instead of directed trees to describe the topology of a multi-net system (and visually via a Hasse diagram). The second is that of a transition system, which provides us with the means to model dynamical aspects of these systems (both feedforward and backward).

3.1. Partial orders

Casey [10] pictures multi-nets as directed trees, but we shall find it convenient to treat these from the point of view of partially ordered sets. A partially ordered set consists of a set V and a relation on V , which we shall denote by \leq . If $u, v \in V$, then we shall interpret $u \leq v$ to mean that u is either equal to or lies below v , where the root of the tree lies above all other nodes. In general, the relation \leq satisfies three conditions; if $u, v, w \in V$, then:

- 1) $u \leq u$ (\leq is *reflexive*);
- 2) If $u \leq v$ and $v \leq u$, then $u = v$ (\leq is *antisymmetric*);
- 3) If $u \leq v$ and $v \leq w$, then $u \leq w$ (\leq is *transitive*).

We write $u < v$ if $u \leq v$ and $u \neq v$, and we write $u \not\leq v$ if $u \leq v$ is false. We add an additional constraint to this formulation to avoid two separate branches of the tree from being connected¹. For $u, v, w \in V$:

$$\text{If } u \leq v, w, \text{ then either } v \leq w \text{ or } w \leq v \quad (1)$$

Finite partial orders (that is to say, partial orders where V is a finite set) may be represented pictorially by a Hasse diagram, in which elements of the set are represented by nodes, and the existence of a sequence of arrows from, say, a to b indicates $a > b$. Fig. 1a) shows a Hasse diagram for an ensemble $V_{ens} = \{t, u, v, w\}$ where three networks are combined in parallel. In this diagram $u < t$, $v < t$, $w < t$, but, say, $v \not\leq w$. In general, we say that $a_0 \in V$ is a root node if $a \leq a_0$ for all $a \in V$, so that v is a root node of V_{ens} . Observe that if $b_0 \in V$ is also a root node, then $a_0 \leq b_0$ and $b_0 \leq a_0$, but then $a_0 = b_0$ by antisymmetry, so root nodes, if they exist, are unique. We write $root(V)$ for the root of V .

Definition 1. A *framework*, is a finite, non-empty partially ordered set with a root node, satisfying (1). We also define a relation \triangleleft on V by:

$$a \triangleleft b \text{ iff } a < b, a < c < b, \text{ for no } c \in V \quad (2)$$

$a \triangleleft b$ means that there is an arrow in the Hasse diagram from b to a .

Fig. 1b) shows a framework V_{hme} . Here we have

¹ This is not used in any of the proofs given in this paper, but serves to identify these partial orders with a tree-like Hasse diagram.

$u \triangleleft t$, $x \triangleleft v$ and $w \triangleleft t$ but not $y \triangleleft t$, since $y < v < t$.

Define $\bullet a = \{b \in V : b \triangleleft a\}$ to be the children of a . For example, in V_{hme} , $\bullet t = \{u, v, w\}$ and $\bullet z = \emptyset$. Note that z is a leaf of V_{hme} and in general, we define:

$$leaf(V) = \{a \in V : \bullet a = \emptyset\} \quad (3)$$

So far we have provided a formal notation and informal depictions (via a Hasse diagram) of how we can describe a multi-net system. Indeed, this description is quite generic and does not yet constrain the characteristics of the nodes themselves. We next look at how the dynamics of a multi-net system can be described.

3.2. Transition systems

In our discussion of the dynamics of feedforward multi-net systems, we shall use the language of transition systems.

A transition system T is composed of a non-empty set Q of *states*, a non-empty set A of *actions* and a relation $\rightarrow \subseteq Q \times A \times Q$, the *transition relation* of T . The elements of \rightarrow are *ordered triples* (q, a, q') , where $q, q' \in Q$ and $a \in A$. We shall write $q \rightarrow^a q'$ to indicate that $(q, a, q') \in \rightarrow$. Intuitively, $q \rightarrow^a q'$ means that if the system is in state q , then the action a may take place, after which the system is in state q' . For finite (Q finite) transition systems, there is a graphical representation in which elements of Q are nodes and $q \rightarrow^a q'$ is indicated by an arrow from q to q' labelled a .

If we consider a feedforward multi-net system, such as an ensemble, then *states* describe static conditions of the multi-net, and *actions* transform one static situation into another. Such an action occurs when an individual node takes the values on its inputs, evaluates the output, which will also depend on its current local parameterization, and transmits it, either to a parent node or to the output of the entire multi-net. In this way the networks within the system take input, change state and pass on their output.

At the beginning of the feedforward sweep, the inputs to the nodes and the local parameterizations have specific values (for example, the weights). Changes to this state as a result of applying the input depend upon these parameters and the node functions. A state, therefore, is embodied by a function associating nodes to values and parameterizations. An action involves an activation of a node, altering the value at its output. These dynamics may be described by *execution sequences*.

If x is a sequence of actions and $q, q' \in Q$, then we define $q \rightarrow^x q'$, providing that:

- 1) If $x = \Lambda$ (the empty sequence), then $q \rightarrow^x q'$ iff $q = q'$;
- 2) If $x = a_1 \cdots a_n$, where $a_1, \dots, a_n \in A$, then $q \rightarrow^x q'$ iff there exists $q_1, \dots, q_n \in Q$ such that $q \rightarrow^{a_1} q_1 \rightarrow \cdots \rightarrow^{a_n} q_n = q'$.

We shall say that an execution $q \rightarrow^x q'$ is *maximal from q* if and only if $q' \rightarrow^a q''$ for no $a \in A$ and $q'' \in Q$, such that q' is the maximal state from q . We shall prove (Theorem 1) that from any state $\underline{\sigma}_0$ in a given multi-net with a given parameterization:

- 1) There exists a maximal execution $\underline{\sigma}_0 \rightarrow^x \underline{\sigma}$;
- 2) If $\underline{\sigma}_0 \rightarrow^x \underline{\sigma}$ and $\underline{\sigma}_0 \rightarrow^y \underline{\sigma}'$ are maximal executions, then $\underline{\sigma} = \underline{\sigma}'$.

The practical consequence of this is that there is a function $G_{\underline{\theta}} : I \rightarrow O$, where I is the set of all input values that can be applied to the leaves of the multi-net; O is the set of all values that can appear as an output at the root of the multi-net, such that $\underline{\sigma}_0$ is a state in which $\underline{\theta}$ is the parameterization and $\underline{x} \in I$ gives the values on the leaves of the multi-net in state $\underline{\sigma}_0$, then $G_{\underline{\theta}}(\underline{x})$ is the value of the output at the root in state $\underline{\sigma}$ where $\underline{\sigma}_0 \rightarrow^x \underline{\sigma}$ is any maximal execution. If $\underline{\theta}$ is the parameterization of the multi-net after training, then $G_{\underline{\theta}}$ describes its functionality.

3.3. Multi-net systems: feedforward state

We now use the idea of a framework and of a transition system to define a feedforward multi-net system.

One of the points of this paper is that many aspects of the combination of networks can be described and analysed with no reference to numerical issues. For example, Theorem 2 (section 4.3) shows that any reasonable training scheme can be achieved by backpropagation. The proof would be much more complicated if we focussed on the numerical properties and such systems. However, once the existence of the proof is established, it is necessary to make it more concrete to be of use.

With this in mind, we take an abstract view of a multi-net system. We assume that each node is a network or some combination (such as a weighted average), and that instead of referring to weights, we suppose that each node $v \in V$ is associated with a set of parameters Θ_v . Likewise, we do not refer to an activation function, but that each node $v \in V$ is associated with a function ϕ_v that takes the

values on the node's inputs and calculates an output, depending on the current value of its local parameters. We shall also assume that each node $v \in V$ is associated with a set O_v of output values.

To be specific, suppose that $v \in V$. If v is not a leaf node, then there are distinct nodes v_1, \dots, v_n such that $v_i \triangleleft v$, for each i so that $\{v_1, \dots, v_n\} = \bullet v$. We can therefore consider inputs to v to be vectors with coordinates v_1, \dots, v_n . Technically, an input vector to v is a function that maps the children of v to the union of the outputs of each child:

$$\underline{x} \bullet v \rightarrow \bigcup_{w \in \bullet v} O_w \quad (4)$$

with the property that $\underline{x}(w) \in O_w$ for each $w \in \bullet v$. We shall write \underline{x}_w for $\underline{x}(w)$. The set of all such vectors \underline{x} constitutes the *input space* of v in V and will be denoted by I_v . For uniformity, in the case of nodes $v \in \text{leaf}(V)$, we shall define $\bullet v = \{\perp_v\}$, so that the inputs to v will belong to a set O_{\perp_v} , which are the overall inputs to the system. The input space of $v \in \text{leaf}(V)$ may now be defined as for non-root nodes. We are now in a position to give a formal definition of a multi-net.

Definition 2. A *multi-net* is defined to be a triple $N = (F, \Theta, \Phi)$, where:

- 1) $F = (V, \leq)$ is a multi-net framework (Definition 1);
- 2) Θ is an indexed family of parameterization sets Θ_v , $v \in V$, and Θ_N the set of all parameterizations of N ;
- 3) Φ is an indexed family of node functions ϕ_v , $v \in V$, where:

$$\phi_v : \Theta_v \times I_v \rightarrow O_v \quad (5)$$

We shall now describe the feedforward dynamics of the system. If we consider the parameterization of the multi-net to be fixed (learning will be dealt with later), then what changes during the feedforward sweep are the values on the outputs of the nodes. We assume for the moment that the external inputs to the multi-net remain unchanged. We may therefore define a *feedforward state* to be a function that maps the framework to the union of the outputs of each node feeding the system output:

$$\underline{\sigma} : V \rightarrow \bigcup_{v \in V^+} O_v \quad (6)$$

where $V^+ = V \cup \{\perp_v : v \in \text{leaf}(V)\}$. $\underline{\sigma}(v)$ gives the value residing on the output of the node v , and $\underline{\sigma}(\perp_v)$ gives the

value on the input to the leaf v .

Let Σ_N denote the set of all feedforward states of the multi-net N . What will change a state is the activation of some node, as it computes its output from the current input. We may therefore consider V as the set of actions. The transition relation for these states will depend on the parameterization of the multi-net. We may represent a parameterization by a function that maps the framework to the union of all the node parameterizations:

$$\underline{\theta} : V \rightarrow \bigcup_{v \in V} \Theta_v \quad (7)$$

satisfying $\underline{\theta}(v) \in \Theta_v$, for all $v \in V$.

We are now interested in describing relations $\rightarrow_{\underline{\theta}} \subseteq \Sigma_N \times V \times \Sigma_N$, which for a given parameterization describes the possible set of state transitions for the system.

In any state $\underline{\sigma} \in \Sigma_N$, the inputs to $v \in V$, that is the values lying on the outputs to the elements $w \in \bullet v$, will have values $\underline{\sigma}(w)$ and so we can define a vector $\underline{z}_{\underline{\sigma}, v} \in I_v$ by

$$\underline{z}_{\underline{\sigma}, v}(w) = \underline{\sigma}(w), \quad w \in \bullet v \quad (8)$$

What we are interested in is the output of the system for a given state, defined as the feedforward state of each component node – we give inputs to the system and propagate these through to the output in a single pass. To achieve this, we define the notion of *stability*. Let us say that a node $v \in V$ is *stable* in state $\underline{\sigma} \in \Sigma_N$ with respect to $\underline{\theta} \in \Theta_N$ if $\phi_v(\underline{\theta}(v), \underline{z}_{\underline{\sigma}, v}) = \underline{\sigma}(v)$, otherwise, it is unstable. A node is stable in a given state with respect to a given parameterization if its output equals the value computed by the node from its inputs in that state and its local parameters.

We may now define $\underline{\sigma} \rightarrow_{\underline{\theta}}^v \underline{\sigma}'$ iff

$$v \text{ is stable at } \underline{\sigma} \text{ with respect to } \underline{\theta} \quad (9)$$

$$\underline{\sigma}' = \underline{\sigma}[v / \phi_v(\underline{\theta}(v), \underline{z}_{\underline{\sigma}, v})] \quad (10)$$

which is the operation of the action $\rightarrow_{\underline{\theta}}^v$ that moves the node v from state $\underline{\sigma}$ to state $\underline{\sigma}'$, where

$$\underline{\sigma}[v/a](w) = \begin{cases} \underline{\sigma}(w) & \text{if } w \neq v \\ a & \text{otherwise} \end{cases} \quad (11)$$

So $\underline{\sigma} \rightarrow_{\underline{\theta}}^v \underline{\sigma}'$ when the output to v at $\underline{\sigma}$ with respect to $\underline{\theta}$ does not match the inputs. The effect of the transition is to update this output according to the node function ϕ_v .

We say that $\underline{\sigma}$ is *stable* w.r.t. to $\underline{\theta}$ if every $v \in V$ is stable at $\underline{\sigma}$ with respect to $\underline{\theta}$. By definition, if $x \in V^+$ and $\underline{\sigma}'$ is stable, then $\underline{\sigma} \rightarrow_{\underline{\theta}}^x \underline{\sigma}'$ is a maximal execution: the final stable output. Write $Stbl_{\underline{\theta}}(\Sigma_N)$ for the set of all states $\underline{\sigma} \in \Sigma_N$ stable with respect to $\underline{\theta}$.

For completeness of the formalism, we need to determine whether we have an output defined for a stable multi-net. Such an output is obtained by feeding inputs to the system and allowing them to propagate up to the output nodes, such that each node itself becomes stable. The output state is the maximal execution of the system.

Proposition 1. For every $\underline{\sigma} \in \Sigma_N$ and $\underline{\theta} \in \Theta_N$, there exists a maximal execution $\underline{\sigma} \rightarrow_{\underline{\theta}}^x \underline{\sigma}'$.

Proof. Define

$$L_{\underline{\sigma}, \underline{\theta}} = \{w \in V : v \leq w \Rightarrow v \text{ is stable w.r.t. } \underline{\theta}\} \quad (12)$$

which is the set of nodes w , such that all are stable. We observe:

- 1) $\underline{\sigma} \in Stbl_{\underline{\theta}}(\Sigma_N)$ iff $L_{\underline{\sigma}, \underline{\theta}} = V$;
- 2) If w is minimal in the set $V \setminus L_{\underline{\sigma}, \underline{\theta}}$, then there exists

$$\underline{\sigma}' \text{ such that } \underline{\sigma} \rightarrow_{\underline{\theta}}^w \underline{\sigma}' \text{ and } L_{\underline{\sigma}, \underline{\theta}} \cup \{w\} \subseteq L_{\underline{\sigma}', \underline{\theta}}.$$

We may now argue by induction on $|V \setminus L_{\underline{\sigma}, \underline{\theta}}|$ (the cardinality of $V \setminus L_{\underline{\sigma}, \underline{\theta}}$). If $|V \setminus L_{\underline{\sigma}, \underline{\theta}}| = 0$, then $L_{\underline{\sigma}, \underline{\theta}} = V$, so $\underline{\sigma}$ is stable, from 1), and $\underline{\sigma} \rightarrow_{\underline{\theta}}^{\wedge} \underline{\sigma}'$ is a maximal execution. If $|V \setminus L_{\underline{\sigma}, \underline{\theta}}| > 0$, then let w be minimal in the set $V \setminus L_{\underline{\sigma}, \underline{\theta}}$. w cannot be stable, since if it were, with $v \in L_{\underline{\sigma}, \underline{\theta}}$ for all $v \leq w$ would mean that $w \in L_{\underline{\sigma}, \underline{\theta}}$, a contradiction. So $\underline{\sigma} \rightarrow_{\underline{\theta}}^w \underline{\sigma}''$ for some $\underline{\sigma}'' \in \Sigma_N$. But by 2), $L_{\underline{\sigma}, \underline{\theta}} \cup \{w\} \subseteq L_{\underline{\sigma}', \underline{\theta}}$, so $|V \setminus L_{\underline{\sigma}', \underline{\theta}}| < |V \setminus L_{\underline{\sigma}, \underline{\theta}}|$.

By induction, there exists a maximal execution $\underline{\sigma}'' \rightarrow_{\underline{\theta}}^x \underline{\sigma}'$, but now $\underline{\sigma} \rightarrow_{\underline{\theta}}^{w.x} \underline{\sigma}'$ is a maximal execution. \square

Given a stable state of the system, we need to know that the system output is predictable, such that whenever we have two states of the system with identical parameterizations that the same inputs lead to the same stable states, i.e. outputs.

Proposition 2. Suppose that $\underline{\sigma}_1, \underline{\sigma}_2 \in Stbl_{\underline{\theta}}(\Sigma_N)$ such that $\underline{x}_{\underline{\sigma}_1} = \underline{x}_{\underline{\sigma}_2}$, then $\underline{\sigma}_1 = \underline{\sigma}_2$.

Proof. Define

$$T_{\underline{\sigma}_1, \underline{\sigma}_2, \underline{\theta}} = \{w \in V : v \leq w \Rightarrow \underline{\sigma}_1(v) = \underline{\sigma}_2(v)\} \quad (13)$$

which is the set of nodes w , such that all the outputs of the nodes from the two states are equal. We observe that $T_{\underline{\sigma}_1, \underline{\sigma}_2, \underline{\theta}} = V \Leftrightarrow \underline{\sigma}_1 = \underline{\sigma}_2$. It is also the case that $leaf(V) \subseteq T_{\underline{\sigma}_1, \underline{\sigma}_2, \underline{\theta}}$, by hypothesis, so $T_{\underline{\sigma}_1, \underline{\sigma}_2, \underline{\theta}} \neq \emptyset$.

We shall assume that $T_{\underline{\sigma}_1, \underline{\sigma}_2, \underline{\theta}} \neq V$ and obtain a contradiction. Suppose that w is minimal in $V \setminus T_{\underline{\sigma}_1, \underline{\sigma}_2, \underline{\theta}}$, then $v \in T_{\underline{\sigma}_1, \underline{\sigma}_2, \underline{\theta}}$ for all $v < w$ and in particular,

$\bullet w \in T_{\underline{\sigma}_1, \underline{\sigma}_2, \underline{\theta}}$. From the stability definition

$$\underline{\sigma}_1(w) = \phi_w(\underline{\theta}, \underline{z}_{\underline{\sigma}_1, \underline{\theta}}) = \phi_w(\underline{\theta}, \underline{z}_{\underline{\sigma}_2, \underline{\theta}}) = \underline{\sigma}_2(w) \quad (14)$$

But $v \in T_{\underline{\sigma}_1, \underline{\sigma}_2, \underline{\theta}}$ for all $v < w$, so $w \in T_{\underline{\sigma}_1, \underline{\sigma}_2, \underline{\theta}}$, the required contradiction. \square

Given these two propositions, we may now describe the function computed by a multi-net in a given configuration. But first we need to define the input and output sets of a network.

Definition 3. If N is a multi-net then define the input space of N to be $I_N = I_{v_1} \times \dots \times I_{v_n}$ and the output space of N to be $O_N = O_{root(V)}$. If $\underline{\sigma} \in Stbl_{\underline{\theta}}(\Sigma_N)$, then define $\underline{x}_{\underline{\sigma}} \in I_N$ by $\underline{x}_{\underline{\sigma}}(\perp_v) = \underline{\sigma}(\perp_v)$, for all $v \in leaf(V)$, and $\underline{y}_{\underline{\sigma}} \in O_N$ by $\underline{y}_{\underline{\sigma}} = \underline{\sigma}(rootV)$.

Theorem 1. Suppose that N is a multi-net and that $\underline{\theta} \in \Theta_N$, then there is a total function

$$G_{N, \underline{\theta}} : I_N \rightarrow O_N \quad (15)$$

such that if $\underline{\sigma} \in Stbl_{\underline{\theta}}(\Sigma_N)$, then

$$G_{N, \underline{\theta}}(\underline{x}_{\underline{\sigma}}) = \underline{y}_{\underline{\sigma}} \quad (16)$$

Proof. The function $G_{N, \underline{\theta}}$ is well defined because if $\underline{\sigma}_1, \underline{\sigma}_2 \in Stbl_{\underline{\theta}}(\Sigma_N)$ and $\underline{x}_{\underline{\sigma}_1} = \underline{x}_{\underline{\sigma}_2}$, then by Proposition 2, $\underline{\sigma}_1 = \underline{\sigma}_2$ and in particular, $\underline{y}_{\underline{\sigma}_1} = \underline{y}_{\underline{\sigma}_2}$.

The function is total because if $\underline{x} \in I_N$ and $\underline{\sigma} \in \Sigma_N$ is any state such that $\underline{x}_{\underline{\sigma}} = \underline{x}$, then by Proposition 1, there exists a maximal execution, $\underline{\sigma} \rightarrow_{\underline{\theta}}^x \underline{\sigma}'$ and clearly $\underline{x}_{\underline{\sigma}'} = \underline{x}_{\underline{\sigma}} = \underline{x}$, since no transition alters the values on the inputs to the multi-net. Hence, $G_{N, \underline{\theta}}(\underline{x})$ is defined and equal to $\underline{y}_{\underline{\sigma}'}$. \square

If all the sets I_{v_i} are equal to some set I , then we say that N is *uniform* and in such cases, we have a function

$\hat{G}_{N,\underline{\theta}} : I \rightarrow O_N$ defined by

$$\hat{G}_{N,\underline{\theta}}(x) = G_{N,\underline{\theta}}(x) \quad (17)$$

where $\underline{x}(\perp_v) = x$, for each $v \in \text{leaf}(V)$.

In this section we have defined the feedforward operation of a multi-net system using partially ordered sets and state transitions. Along the way we have had to provide formal proof of certain properties of these systems to ensure completeness and rigor (such as the stable and predictable output of the system). Whilst this is perhaps lengthy, this formal approach is necessary in order to provide a solid foundation for the exploration of multi-net system properties. However, as yet, we have not discussed the concept of learning in these systems, which we tackle next.

3.4. Multi-net systems: strategies for learning

We now turn to the training of multi-nets, and specifically to systems governed by some criterion function that can be used to measure the convergence of the system to the criterion. The criterion is liberal in the sense that it can specify any required stopping condition. For convenience, we can treat such criterion functions as error functions used in supervised learning systems, where the error is measured to be within a certain value. However, this does not exclude other criteria or types of learning, such as unsupervised learning systems, in which our criterion may be measured in a way not necessarily related to a target response, for example through a simple number of learning cycles.

Taking our abstract view, we consider training to be the application of an operator S to parameterizations. Thus, for a given (uniform) multi-net N , a parameterization $\underline{\theta} \in \Theta$, an input $\underline{x} \in I_N$ and an output $y \in O_N$, the operator will generate a new parameterization $\underline{\theta}' \in \Theta$.

In fact, the elements $\underline{\theta}$, \underline{x} and y are all aspects of a state of the multi-net. As training will take place only at stable states, we may consider S to be represented by a family of functions

$$S_{\underline{\theta}} : \text{Stbl}_{\underline{\theta}}(\Sigma_N) \rightarrow \Theta, \quad (18)$$

with the interpretation that if the multi-net has parameterization $\underline{\theta} \in \Theta$ in state $\underline{\sigma} \in \text{Stbl}_{\underline{\theta}}(\Sigma_N)$, then the parameterization will be changed to $S_{\underline{\theta}}(\underline{\sigma})$.

We shall also assume that the strategy is non-trivial, that it actually does something, that is:

$$\exists \underline{\theta} \in \Theta \exists \underline{\sigma} \in \text{Stbl}(\Sigma_N). S_{\underline{\theta}}(\underline{\sigma}) \neq \underline{\theta} \quad (19)$$

The feedforward process may then be reapplied to give a new stable state $T_{\underline{\theta}}(\underline{\sigma})$ which is defined to be the unique

element $\underline{\sigma}' \in \text{Stbl}_{\underline{\theta}}(\Sigma_N)$ such that $\underline{\sigma} \rightarrow_{S_{\underline{\theta}}(\underline{\sigma})}^x \underline{\sigma}'$ is a maximal execution. $T_{\underline{\theta}}$ is therefore a function which maps a stable state of the multi-net to the stable state once the operator S has been applied, i.e. training.

$$T_{\underline{\theta}} : \text{Stbl}_{\underline{\theta}}(\Sigma_N) \rightarrow \text{Stbl}_{S_{\underline{\theta}}(\underline{\sigma})}(\Sigma_N), \quad (20)$$

Starting with a state $\underline{\sigma} \in \text{Stbl}_{\underline{\theta}}(\Sigma_N)$, we may then repeatedly apply the strategy to determine sequences $\underline{\theta}_1, \dots$ and $\underline{\sigma}_1, \dots$ where:

$$\underline{\theta}_1 = \underline{\theta}, \underline{\theta}_{n+1} = S_{\underline{\theta}_n}(\underline{\sigma}_n) \quad (21)$$

$$\underline{\sigma}_1 = \underline{\sigma}, \underline{\sigma}_{n+1} = T_{\underline{\theta}_n}(\underline{\sigma}_n) \quad (22)$$

We shall say that a sequence $\underline{\theta}_1, \dots$ converges at N providing $\underline{\theta}_N = \underline{\theta}_{N+1}$.

Lemma 1. If $\underline{\theta}_1, \dots$ converges at N , then $\underline{\theta}_n = \underline{\theta}_N$ for all $n \geq N$.

Proof. We argue by induction on $n - N$ that $\underline{\theta}_n = \underline{\theta}_{n+1}$ for all $n \geq N$. The case $n - N = 0$ is the hypothesis $\underline{\theta}_N = \underline{\theta}_{N+1}$.

Suppose that $\underline{\theta}_n = \underline{\theta}_{n+1}$, then $\underline{\theta}_n = \underline{\theta}_{n+1} = S_{\underline{\theta}_n}(\underline{\sigma}_n)$. By definition $\underline{\sigma}_n \rightarrow_{\underline{\theta}_{n+1}} \underline{\sigma}_{n+1}$ is a maximal execution. But $\underline{\theta}_n = \underline{\theta}_{n+1}$ and $\underline{\sigma}_n \in \text{Stbl}_{\underline{\theta}_n}(\Sigma_N)$, so $\underline{\sigma}_n = \underline{\sigma}_{n+1}$ and now $\underline{\theta}_{n+2} = S_{\underline{\theta}_{n+1}}(\underline{\sigma}_{n+1}) = S_{\underline{\theta}_n}(\underline{\sigma}_n) = \underline{\theta}_{n+1}$. \square

Training will have some form of goal, of course, and we shall suppose that this is expressed by a set; strictly speaking, the extension of a predicate:

$$P \subseteq I_N \times O_N \quad (23)$$

P expresses a goal in the sense that training has succeeded if $(\underline{x}, G_{N,\underline{\theta}}(\underline{x})) \in P$; P may be considered as the extension of a predicate defining successful training.

We shall say that the family S is a training strategy with respect to P providing

$$(\underline{x}_{\underline{\sigma}}, y_{\underline{\sigma}}) \in P \Leftrightarrow S_{\underline{\theta}}(\underline{\sigma}) = \underline{\theta}. \quad (24)$$

In other words, the strategy converges to a parameterization in which the multi-net computes a function that satisfies the criterion.

4. Supervised learning for multi-net systems

So far we have provided a formal definition of a multi-net system and its associated learning algorithm. The most important aspect of this approach is that it does not constrain the type of networks that can be combined. All that is required is a suitable topology, set of feedforward functions, parameters and learning strategy functions. However, whilst this definition is perhaps interesting, to

be useful we need to consider example systems in which we can start to use this formal foundation to infer properties of the combined system.

In this section we consider a multi-layer perceptron (MLP) using the backpropagation algorithm [30] as a multi-net system. In this way we can think of individual layers as separate networks. This allows us to generalize the notion of an MLP to be a generic sequential (and parallel for partially connected networks) combination of feedforward networks that can be trained using an abstract algorithm (based on backpropagation).

4.1. System definition

We start by considering the simple MLP as shown in Fig. 2, which, without loss of generality, we have considered as a two-layer network with an arbitrary number of inputs, hidden layer neurons and outputs. Each unit operates using the logistic sigmoid on the weighted summation of its inputs, and follows the standard backpropagation algorithm [30].

We define $N_{mlp} = (F_{mlp}, \Theta_{mlp}, \Phi_{mlp})$ as a multi-net, by

- 1) $F_{mlp} = (V_{mlp}, \leq)$, with $V_{mlp} = \{t, u\}$ and $u \triangleleft t$;
- 2) $\Theta_{mlp} = \{\Theta_t, \Theta_u\}$, $\Theta_u = \mathbb{R}^{|\mathcal{I}_u|}$ and $\Theta_t = \mathbb{R}^{|\mathcal{I}_t|}$ where \mathbb{R} denotes the set of real numbers;
- 3) $\Phi_{mlp} = \{\phi_t, \phi_u\}$, where:

$$\phi_{u,j}(\underline{\theta}_{u,j}, \underline{x}) = \frac{1}{1 + e^{-\sum_i \theta_{u,j}^i x^i}} \text{ for each neuron } j \text{ in } u \quad (25)$$

$$\phi_{t,j}(\underline{\theta}_{t,j}, \underline{x}) = \frac{1}{1 + e^{-\sum_i \theta_{t,j}^i x^i}} \text{ for each neuron } j \text{ in } t \quad (26)$$

4.2. Learning as a backward pass

We now consider how this simple system is trained using backpropagation, before extending this to the concept of a multi-net system. In backpropagation, each node receives some form of instruction from its parent. Depending upon this instruction, and its current parameterization and output, it will modify its parameters and propagate an instruction down to its children. This suggests that at each node v we have a set A_v of instructions implemented as a set of *adjustment functions* J_v . We assume that each set of instructions contains an element 0_v , which is the instruction to do nothing. The adjustment functions give a new node parameterization from the output of the node:

$$j_v : A_v \times O_v \times \Theta_v \rightarrow \Theta_v, \quad (27)$$

To propagate the changes back through the system, we have *instruction propagation functions*:

$$a_{v,u} : A_v \times O_v \times \Theta_v \rightarrow A_u, \quad (28)$$

Intuitively, j_v modifies the parameterization at v according to an instruction, its current output and current parameterization. Based on the same information, $a_{v,u}$ propagates an instruction down to its child node u .

In view of the interpretation of the elements 0_v , we require:

$$j_v(0_v, y, \theta) = \theta, \quad a_{v,u}(0_v, y, \theta) = 0_u \quad (29)$$

for all $y \in O_v$, $\theta \in \Theta_v$. In fact, we would not expect any other instruction to propagate a 0_v or to leave a local parameterization as it is. We require further that:

$$a_{v,u}(a, y, \theta) = 0_u \Leftrightarrow a = 0_v \Leftrightarrow j_v(a, y, \theta) = \theta \quad (30)$$

for all $a \in A_v$, $y \in O_v$, $\theta \in \Theta_v$. We shall refer to this as the *strictness condition*.

We adopt a state-based approach to the description of the backpropagation process. That is to say, we conceive of the multi-net going through a series of changes as the backpropagation sweeps down from the root, much like we treated the feedforward state.

A state is an instantaneous snapshot the system. Given the purpose of the j_v and $a_{v,u}$ functions, we see that what the state must record are the instructions, outputs and parameters currently at each node. Consequently, we define a *feedback state* to be a function that maps the nodes to the union of instructions, outputs and parameterizations for each node:

$$\underline{\rho} : V \rightarrow \bigcup_{v \in V} A_v \times \bigcup_{v \in V} O_v \times \bigcup_{v \in V} \Theta_v \quad (31)$$

such that for each $v \in V$, $\underline{\rho}(v) \in A_v \times O_v \times \Theta_v$. If $\underline{\rho}(v) = (a, y, \theta)$, then we define $a_{\underline{\rho}(v)} = a$, $y_{\underline{\rho}(v)} = y$ and $\theta_{\underline{\rho}(v)} = \theta$. Thus $a_{\underline{\rho}(v)}$ gives the instruction considered by v in state $\underline{\rho}$ and similarly for the current output and current parameter.

We also have a notion of state transition, the expression $\underline{\rho} \rightarrow^v \underline{\rho}'$ says that if the algorithm is applied locally at node v in state $\underline{\rho}$, then the state will be transformed to

$\underline{\rho}'$. $\underline{\rho} \rightarrow^v \underline{\rho}'$ holds precisely when for all $w \in V$:

$$a_{\underline{\rho}'(w)} = \begin{cases} a_{\underline{\rho}(w)} & \text{if } w \neq v \\ a_{v,w}(a_{\underline{\rho}(v)}, y_{\underline{\rho}(v)}, \theta_{\underline{\rho}(v)}) & \text{otherwise} \end{cases} \quad (32)$$

$$y_{\underline{\rho}'(w)} = y_{\underline{\rho}(w)} \quad (33)$$

$$\theta_{\underline{\rho}'(w)} = \begin{cases} \theta_{\underline{\rho}(w)} & \text{if } w \neq v \\ j_v(a_{\underline{\rho}(v)}, y_{\underline{\rho}(v)}, \theta_{\underline{\rho}(v)}) & \text{otherwise} \end{cases} \quad (34)$$

So the application of the algorithm at node v

propagates an instruction down to each of the children of v according to the functions $a_{v,u}$, leaving everything else unchanged. (Outputs at nodes only change in feedforward pass.) Finally, the application of the algorithm at v will change its own local parameters, leaving all others unchanged.

Putting together the node modifications together in the case of backpropagation is not quite as simple as in the case of the feedforward formalism as there is no concept corresponding to a stable state, which guarantees that each node is only modified once during a sweep. We have to impose this explicitly here. Essentially, the idea is that no node v should be modified until every node $w \geq v$ has been modified. We should therefore consider sequences of the form

$$\underline{\rho} \rightarrow^{v_1} \underline{\rho}_1 \cdots \rightarrow^{v_n} \underline{\rho}_n \quad (35)$$

such that $i < j \Rightarrow v_i > v_j$. Note that this means $v_1 = \text{root}(V)$. Any such sequence $v_1 \cdots v_n$ is called a *topological sort of V* and we denote the set of all topological sorts of V by $TS(V)$. We now have a function:

$$R : TS(V) \times \Psi_N \rightarrow \Psi_N \quad (36)$$

given by

$$R(v_1 \cdots v_n, \underline{\rho}) = \underline{\rho}' \Leftrightarrow \underline{\rho} \rightarrow^{v_1} \underline{\rho}_1 \cdots \rightarrow^{v_n} \underline{\rho}_n = \underline{\rho}' \quad (37)$$

On the face of it, it would seem that the application of the algorithm depends on the order in which it is applied to the nodes. Fortunately, this is not the case, as the following result shows.

Proposition 3. With the above notation, if $\alpha, \alpha' \in TS(V)$, then $R(\alpha, \underline{\rho}) = R(\alpha', \underline{\rho})$, for all $\underline{\rho} \in \Psi_N$.

Proof. Let $v, v' \in V$ and define $v \iota v' \Leftrightarrow v \not\leq v' \wedge v' \not\leq v$. If $\alpha, \alpha' \in V^+$, then define $\alpha \equiv^{(1)} \alpha'$ if and only if there exists $\beta, \beta'' \in V^+$ and $v, v' \in V$ such that $\alpha = \beta v v' \beta''$, $\alpha' = \beta' v \beta''$ and $v \iota v'$. Finally, let \equiv be the reflexive transitive closure of $\equiv^{(1)}$. Informally, $\alpha \equiv \alpha'$ if either $\alpha = \alpha'$ or α' may be obtained from α by commuting adjacent nodes $v, v' \in V$ satisfying $v \iota v'$. We shall show:

- 1) If $\alpha \in TS(V)$ and $\alpha \equiv \alpha'$, then $\alpha' \in TS(V)$ and $R(\alpha, \underline{\rho}) = R(\alpha', \underline{\rho})$ for all $\underline{\rho} \in \Psi_N$;
- 2) If $\alpha, \alpha' \in TS(V)$, then $\alpha \equiv \alpha'$.

Together, these imply the proposition. For 1), suppose that $\alpha = v_1 \cdots v_r v v' v_{r+1} \cdots v_n$ and $\alpha' = v_1 \cdots v_r v' v v_{r+1} \cdots v_n$ with $v \iota v'$, so that $\alpha \equiv^{(1)} \alpha'$. If $\alpha' \notin TS(V)$, then we

would have to have $v \leq v'$. But, this would contradict $v \iota v'$. Hence, $\alpha' \in TS(V)$. We also note that if $\underline{\rho} \rightarrow^v \underline{\rho}' \rightarrow^{v'} \underline{\hat{\rho}}$, $\underline{\rho} \rightarrow^{v'} \underline{\rho}'' \rightarrow^{v'} \underline{\hat{\rho}'}$ and $v \iota v'$, then a simple calculation shows that $\underline{\hat{\rho}} = \underline{\hat{\rho}'}$. Hence, $R(\alpha, \underline{\rho}) = R(\alpha', \underline{\rho})$. We have argued that if $\alpha \in TS(V)$ and $\alpha \equiv^{(1)} \alpha'$, then $\alpha' \in TS(V)$ and $R(\alpha, \underline{\rho}) = R(\alpha', \underline{\rho})$. Now 1) follows from transitivity.

For 2), suppose that $\alpha, \alpha' \in TS(V)$. We argue by induction on $n(\alpha') = |V| - \ell(\alpha \wedge \alpha')$, where $|V|$ denotes the cardinality of V , $\alpha \wedge \alpha'$ denotes the longest common prefix of α and α' and $\ell(\alpha \wedge \alpha')$ denotes the length of $\alpha \wedge \alpha'$. If $|V| - \ell(\alpha \wedge \alpha') = 0$, then $\alpha = \alpha'$, so that $\alpha \equiv \alpha'$, by reflexivity. Suppose $|V| - \ell(\alpha \wedge \alpha') > 0$ and suppose that $\alpha = \beta v \gamma$, $\alpha' = \beta v_1 \cdots v_r v \gamma'$, $v_1, \dots, v_r, v \in V$, $\beta, \gamma, \gamma' \in V^+$ with $v \neq v_i$, so that $\beta = \alpha \wedge \alpha'$. As $\beta v_1 \cdots v_r v \gamma' \in TS(V)$, $v \not\leq v_i$, $1 \leq i \leq r$, and as all the v_i occur in γ and $\beta v \gamma \in TS(V)$, $v_i \not\leq v$, $1 \leq i \leq r$, hence, $v_i \iota v$, $1 \leq i \leq r$, and so

$$\begin{aligned} \alpha' &= \beta v_1 \cdots v_r v \gamma' \equiv^{(1)} \beta v_1 \cdots \\ &\cdots \equiv^{(1)} \beta v_{r-1} v v_r \gamma' \equiv^{(1)} \cdots \equiv^{(1)} \beta v v_1 \cdots v_r \gamma' = \alpha'' \end{aligned} \quad (38)$$

Hence $\alpha' \equiv \alpha''$. But now $\alpha'' \in TS(V)$, by the first part of the proof, and $\beta v \leq \alpha, \alpha''$, so that $\ell(\alpha \wedge \alpha'') > \ell(\alpha \wedge \alpha')$ and hence $n(\alpha'') < n(\alpha')$. By induction $\alpha \equiv \alpha''$, but $\alpha' \equiv \alpha''$. Hence $\alpha \equiv \alpha'$, by transitivity. \square

As a result, we have a function $B : \Psi_N \rightarrow \Psi_N$ given by $B(\underline{\rho}) = R(\alpha, \underline{\rho})$ for any $\alpha \in TS(V)$. $B(\underline{\rho})$ is the new feedback state following one sweep of the algorithm.

There is one further matter to consider and that is how the backward pass is initiated. We propose that this depends on a function

$$Q : I_N \times O_N \rightarrow A_{\text{root}(V)} \quad (39)$$

which compares the input with the output and issues an instruction at the node accordingly.

Let us now put all this together. We take an element $\underline{\sigma} \in \text{Stbl}_Q(\Sigma_N)$ and define an element $\underline{\rho}(\underline{\sigma}) \in \Psi_N$ as follows. If $v \in V$, then let

$$a_{\underline{\rho}(\underline{\sigma})(v)} = \begin{cases} Q(\underline{x}_{\underline{\sigma}}, \underline{y}_{\underline{\sigma}}) & \text{if } v = \text{root}(V) \\ 0_v & \text{otherwise} \end{cases} \quad (40)$$

$$y_{\underline{\rho}(\underline{\sigma})(v)} = y_{\underline{\sigma}(v)} \quad (41)$$

$$\theta_{\underline{\rho}(\underline{\sigma})(v)} = \theta_v \quad (42)$$

In words, $\underline{\rho}(\underline{\sigma})$ represents the situation in which a feedforward has just completed with parameterization $\underline{\theta}$, no node except the root is being instructed to change and the root is to receive the instruction computed by Q on the basis of the current input and output of the multi-net.

Although formal, this provides the necessary foundation for a backward pass of learning, but does not deal with convergence.

4.3. Convergence strategy

Definition 4 An abstract backpropagation algorithm (ABPA) for a multi-net N is a triple $B = (Q, A, J)$, where

- 1) $Q : I_N \times O_N \rightarrow A_{root(V)}$;
- 2) A is an indexed family of instruction propagation functions A_v , $v \in V$;
- 3) J is an indexed family of adjustment functions J_v , $v \in V$.

We may now show how an ABPA determines a multi-net learning strategy in that the strategy allows us to perform a series of adjustments to the multi-net which converges to a parameterization that meets the criterion. Suppose that $\underline{\sigma} \in Stbl_{\underline{\theta}}(\Sigma_N)$, then this determines a feedback state $\underline{\rho}(\underline{\sigma})$, to which we may apply the function B to get a new feedback state $B(\underline{\rho}(\underline{\sigma}))$. We now extract the parameterization part of $B(\underline{\rho}(\underline{\sigma}))$. If $v \in V$, then let $(S_B)_{\underline{\theta}}(\underline{\sigma})_v = \theta_{B(\underline{\rho}(\underline{\sigma}))}(v)$. By construction, $(S_B)_{\underline{\theta}} : Stbl_{\underline{\theta}}(\Sigma_N) \rightarrow \Theta$, so that the family of functions $(S_B)_{\underline{\theta}}$ does indeed constitute a training strategy for N . In fact, we have:

Proposition 4. Suppose that $B = (Q, A, J)$ is an ABPA, then S_B is a training strategy with respect to the predicate P_B given by

$$(x, y) \in P_B \Leftrightarrow Q(x, y) = 0_{root(V)} \quad (43)$$

Proof. Suppose that $Q(x, y) = 0_{root(V)}$. By definition of $\underline{\rho}(\underline{\sigma})$, $a_{\underline{\rho}(\underline{\sigma})}(v) = 0_v$ for all $v \in V$. If we now check the definition of state transition (29) we can see that in such circumstances $\underline{\rho}(\underline{\sigma}) \rightarrow^v \underline{\rho}' \Rightarrow \underline{\rho}' = \underline{\rho}(\underline{\sigma})$. Hence, if $v_1 \cdots v_n$ is a topological sort of V and $\underline{\rho}(\underline{\sigma}) \rightarrow^{v_1} \underline{\rho}_1 \cdots \rightarrow^{v_n} \underline{\rho}_n$, then $\underline{\rho}_n = \underline{\rho}(\underline{\sigma})$, that is, $B(\underline{\rho}(\underline{\sigma})) = \underline{\rho}(\underline{\sigma})$. But now, $(S_B)_{\underline{\theta}}(\underline{\sigma})_v = \theta_{B(\underline{\rho}(\underline{\sigma}))}(v) = \theta_{\underline{\rho}(\underline{\sigma})}(v) = \theta_v$, for all $v \in V$,

that is, $(S_B)_{\underline{\theta}}(\underline{\sigma}) = \underline{\theta}$.

Next, suppose that $Q(x_{\underline{\sigma}}, y_{\underline{\sigma}}) \neq 0_{root(V)}$, then $a_{\underline{\rho}(\underline{\sigma})}(root(V)) \neq 0_{root(V)}$ and so

$$j_{root(V)}(a_{\underline{\rho}(\underline{\sigma})}(root(V)), y_{\underline{\rho}(\underline{\sigma})}(root(V)), \theta_{\underline{\rho}(\underline{\sigma})}(root(V))) \neq \theta_{\underline{\rho}(\underline{\sigma})}(root(V)) \quad (44)$$

Hence, if $v_1 \cdots v_n$ is a topological sort of V and $\underline{\rho}(\underline{\sigma}) \rightarrow^{v_1} \underline{\rho}_1 \cdots \rightarrow^{v_n} \underline{\rho}_n$, then $\theta_{\underline{\rho}_1}(root(V)) \neq \theta_{\underline{\rho}(\underline{\sigma})}(root(V))$. But, we observe that for any $v \in V$, if $\underline{\rho} \rightarrow^{v_n} \underline{\rho}'$, then for $w \neq v$, $\theta_{\underline{\rho}'(w)} = \theta_{\underline{\rho}(w)}$. Applying this observation to the execution sequence $\underline{\rho}(\underline{\sigma}) \rightarrow^{v_1} \underline{\rho}_1 \cdots \rightarrow^{v_n} \underline{\rho}_n$, we see that

$$\theta_{B(\underline{\rho}(\underline{\sigma}))}(root(V)) = \theta_{\underline{\rho}_1}(root(V)) \neq \theta_{\underline{\rho}(\underline{\sigma})}(root(V)) \quad (45)$$

Consequently,

$$(S_B)_{\underline{\theta}}(\underline{\sigma})_{root(V)} = \theta_{B(\underline{\rho}(\underline{\sigma}))}(root(V)) \neq \theta_{\underline{\rho}(\underline{\sigma})}(root(V)) = \theta_{root(V)} \quad (46)$$

and so $(S_B)_{\underline{\theta}}(\underline{\sigma}) \neq \underline{\theta}$. \square

This result shows that an ABPA implements a strategy; furthermore, the criterion of the strategy is determined by the function Q .

We may make a further observation here. We shall say that a strategy is strict if and only if, for all $\underline{\theta} \in \Theta_N$ and $\underline{\sigma} \in Stbl(\Sigma_N)$,

$$(S_B)_{\underline{\theta}}(\underline{\sigma}) \neq \underline{\theta} \Rightarrow \forall v \in V. (S_B)_{\underline{\theta}}(\underline{\sigma})_v \neq \theta_v \quad (47)$$

Proposition 5. Suppose that $B = (Q, A, J)$ is an ABPA, then S_B is a *strict* training strategy with respect to P_B .

Proof. We know that S_B is a training strategy with respect to P_B . We must prove strictness. Suppose that $(S_B)_{\underline{\theta}}(\underline{\sigma}) \neq \underline{\theta}$, then $Q(x_{\underline{\sigma}}, y_{\underline{\sigma}}) \neq 0_{root(V)}$. Let $v_1 \cdots v_n$ be a topological sort of V and $\underline{\rho}(\underline{\sigma}) \rightarrow^{v_1} \underline{\rho}_1 \cdots \rightarrow^{v_n} \underline{\rho}_n = B(\underline{\rho}(\underline{\sigma}))$.

As $Q(x_{\underline{\sigma}}, y_{\underline{\sigma}}) \neq 0_{root(V)}$,

$$\theta_{\underline{\rho}_1}(v_1) = j_{root(V)}(Q(x_{\underline{\sigma}}, y_{\underline{\sigma}}), y_{\underline{\rho}(root(V))}, \theta_{\underline{\rho}(root(V))}) \neq \theta_{\underline{\rho}(root(V))} \quad (48)$$

and

$$a_{\underline{\rho}_1}(u) = a_{root(V),u}(Q(x_{\underline{\sigma}}, y_{\underline{\sigma}}), y_{\underline{\rho}(root(V))}, \theta_{\underline{\rho}(root(V))}) \neq 0_{root(V)} \quad (49)$$

by the strictness conditions. Assume by induction that

$$j < r \Rightarrow \theta_{\underline{\rho}_r(v_j)} \neq \theta_{v_j} \text{ and } j \leq r \Rightarrow a_{\underline{\rho}_r(v_j)} \neq 0_{v_j} \quad (50)$$

$$j < r \Rightarrow \theta_{\underline{\rho}_{r+1}(v_j)} \neq \theta_{v_j} \text{ and } j \leq r \Rightarrow a_{\underline{\rho}_{r+1}(v_j)} \neq 0_{v_j} \quad (51)$$

and now, by the strictness conditions

$$\theta_{\underline{\rho}_{r+1}(v_r)} \neq \theta_{v_j} \text{ and } a_{\underline{\rho}_{r+1}(v_{r+1})} \neq 0_{v_j}, \quad (52)$$

completing the induction step. \square

In fact, strictness is all that is required for a converse to the proposition.

Theorem 2. S_B is a strict training strategy with respect to P if and only if there exists an ABPA $B = (Q, A, J)$ such that

- 1) $S = S_B$;
- 2) $P = \{(x, y) : Q(x, y) = O_{root(V)}\}$.

Proof. Suppose that S is a strict strategy for N with respect to a criterion P . Suppose that $\underline{\theta} \in \Theta_N$. Define

$$C = \{(\underline{\sigma}, \underline{\theta}) \in \Sigma_N \times \Theta_N : \underline{\sigma} \in Stbl_{\underline{\theta}}(\Sigma_N)\} \quad (53)$$

and for each $v \in V$, define

$$0_v = \{(\underline{\sigma}, \underline{\theta}) \in C : S_{\underline{\theta}}(\underline{\sigma}) = \underline{\theta}\} \quad (54)$$

$$A_v = (C \setminus 0_v) \cup \{0_v\} \quad (55)$$

For all $v \in V$ and $u \in \cdot v$, define

$$j_v(0_v, y, \theta) = \theta \quad (56)$$

$$a_{v,u}(0_v, y, \theta) = \theta_v \quad (57)$$

For the definition of the functions j_v we need to make an observation. Since S is strict, we must have $|\Theta_v| > 2$ for all $v \in V$, for if $|\Theta_v| = 1$, say $\Theta_v = \{\theta\}$, then we would have to have $S_{\underline{\theta}}(\underline{\sigma})_v = \theta = \underline{\theta}_v$ for all $\underline{\theta} \in \Theta_N$ and $\underline{\sigma} \in Stbl_{\underline{\theta}}(\Sigma_N)$. But by strictness, we would have to have $S_{\underline{\theta}}(\underline{\sigma}) = \underline{\theta}$ for all $\underline{\theta} \in \Theta_N$ and $\underline{\sigma} \in Stbl_{\underline{\theta}}(\Sigma_N)$, contradicting the assumption of non-triviality. It follows that there exists a bijective function $\tau : \bigcup \Theta_v \rightarrow \bigcup \Theta_v$ satisfying $\tau(\Theta_v) = \Theta_v$, for all $v \in V$, and $\tau(\theta) \neq \theta$, for all $\theta \in \Theta_v$ and $v \in V$. If $(\underline{\sigma}, \underline{\theta}) \in A_v$, then define

$$j_v((\underline{\sigma}, \underline{\theta}), y, \theta) = \begin{cases} S_{\underline{\theta}}(\underline{\sigma})_v & \text{if } \theta = \underline{\theta}_v \text{ and } S_{\underline{\theta}}(\underline{\sigma}) \neq \underline{\theta} \\ \tau(\theta) & \text{otherwise} \end{cases} \quad (58)$$

$$a_{v,u}((\underline{\sigma}, \underline{\theta}), y, \theta) = (\underline{\sigma}, \underline{\theta}) \quad (59)$$

Let $B = (Q, A, J)$. We shall prove that B is an ABPA and that $S = S_B$. To show that is an ABPA, we need to establish the strictness conditions. By definition $a_{v,u}(0_v, y, \theta) = \theta_v$, while

$$a_{v,u}((\underline{\sigma}, \underline{\theta}), y, \theta) = (\underline{\sigma}, \underline{\theta}) \neq 0_u \quad (60)$$

So $a_{v,u}(a, y, \theta) = 0_u \Leftrightarrow a = 0_v$.

Also, by definition, $j_v(0_v, y, \theta) = \theta$, while if $j_v((\underline{\sigma}, \underline{\theta}), y, \theta) = \theta$, then as $\tau(\theta) \neq \theta$, it would have to follow that $S_{\underline{\theta}}(\underline{\sigma})_v = \theta$, $\theta = \underline{\theta}_v$ and $S_{\underline{\theta}}(\underline{\sigma}) \neq \underline{\theta}$. But this implies that $S_{\underline{\theta}}(\underline{\sigma})_v = \underline{\theta}_v$, so that $S_{\underline{\theta}}(\underline{\sigma}) = \underline{\theta}$, by strictness, is a contradiction. Hence $j_v(a, y, \theta) = \theta \Leftrightarrow a = 0_v$ and we have established that B is an ABPA. We now show that $S = S_B$.

Suppose first of all that $S_{\underline{\theta}}(\underline{\sigma}) = \underline{\theta}$, then by definition of Q and $\underline{\rho}(\underline{\sigma})$, $a_{\underline{\rho}(\underline{\sigma})(v)} = 0$ for all $v \in V$, and we have noted that in this case $B(\underline{\rho}(\underline{\sigma})) = \underline{\rho}(\underline{\sigma})$, so that $(S_B)_{\underline{\theta}}(\underline{\sigma}) = \underline{\theta} = S_{\underline{\theta}}(\underline{\sigma})$.

Next, suppose that $S_{\underline{\theta}}(\underline{\sigma}) \neq \underline{\theta}$, let $v_1 \cdots v_n \in TS(V)$ and suppose that $\underline{\rho}(\underline{\sigma}) \rightarrow^{v_1} \underline{\rho}_1 \cdots \rightarrow^{v_n} \underline{\rho}_n = B(\underline{\rho}(\underline{\sigma}))$. We shall argue by induction on r , that if $1 \leq r \leq n$, then

$$j < r \Rightarrow \theta_{\underline{\rho}_r(v_j)} = S_{\underline{\theta}}(\underline{\sigma})_{v_j} \quad (61)$$

$$j \geq r \Rightarrow \theta_{\underline{\rho}_r(v_j)} = \underline{\theta}_{v_j}$$

$$j \leq r \Rightarrow a_{\underline{\rho}_r(v_j)} = (\underline{\sigma}, \underline{\theta}) \text{ and } j > r \Rightarrow a_{\underline{\rho}_r(v_j)} = 0_{v_j} \quad (62)$$

This is certainly true when $r=1$. Suppose it is true for $1 \leq r < n$; we show that it is true for $r+1$. We first observe that

$$j < r \Rightarrow \theta_{\underline{\rho}_{r+1}(v_j)} = \theta_{\underline{\rho}_r(v_j)} = S_{\underline{\theta}}(\underline{\sigma})_{v_j} \quad (63)$$

$$j > r \Rightarrow \theta_{\underline{\rho}_{r+1}(v_j)} = \theta_{\underline{\rho}_r(v_j)} = \underline{\theta}_{v_j} \quad (64)$$

$$j \leq r \Rightarrow a_{\underline{\rho}_{r+1}(v_j)} = a_{\underline{\rho}_r(v_j)} = (\underline{\sigma}, \underline{\theta}) \quad (65)$$

$$j > r+1 \Rightarrow a_{\underline{\rho}_{r+1}(v_j)} = a_{\underline{\rho}_r(v_j)} = 0_{v_j} \quad (66)$$

So to complete the induction step, we must show

$$\theta_{\underline{\rho}_{r+1}(v_r)} = S_{\underline{\theta}}(\underline{\sigma})_{v_r} \quad (67)$$

$$a_{\underline{\rho}_{r+1}(v_{r+1})} = (\underline{\sigma}, \underline{\theta}) \quad (68)$$

But, now,

$$\begin{aligned}
\theta_{\underline{\rho}(v_r)} &= j_{v_{r+1}}(a_{\underline{\rho}_r(v_r), y_{\underline{\rho}_r(v_r)}, \theta_{\underline{\rho}_r(v_r)}}) \\
&= j_{v_{r+1}}((\underline{\sigma}, \underline{\theta}), y_{\underline{\rho}_r(v_r)}, \theta_{\underline{\rho}_r(v_r)}) \\
&= S_{\underline{\theta}}(\underline{\sigma})_{v_r}
\end{aligned} \tag{69}$$

and if $v_{r+1} \in \bullet v_j$, so that $j < r+1$ by the defining property of a topological sort, then

$$\begin{aligned}
a_{\underline{\rho}_{r+1}(v_{r+1})} &= a_{v_j, v_{r+1}}(a_{\underline{\rho}_j(v_j), y_{\underline{\rho}_j(v_j)}, \theta_{\underline{\rho}_j(v_j)}}) \\
&= a_{v_j, v_{r+1}}((\underline{\sigma}, \underline{\theta}), y_{\underline{\rho}_j(v_j)}, \theta_{\underline{\rho}_j(v_j)}) \\
&= (\underline{\sigma}, \underline{\theta})
\end{aligned} \tag{70}$$

and we are done. \square

4.4. Discussion

The significance of this result is as follows. If it is possible to express *precisely* the aims of training a multi-net, that is, as an extension to a predicate, then if training is possible at all, it may be achieved somehow by abstract backpropagation.

Here then, we have considered the general form of an MLP as a multi-net system. Within the definition, although we have illustrated this with a specific topology and set of functions, we have not used these parameters to provide the constructive proof. Indeed, one aspect that we have not considered is the generality of our description of an MLP as a multi-net. We have considered, without loss of generality, a system in which there are two components (the hidden layer and the output layer), with the leaf node feeding sequentially the root node. Because of the generality of this situation, we could consider a similar system in which there are two or more leaf nodes feeding the root. This corresponds to the class of MLPs that consist of partial connections between the hidden and output layers. Furthermore, since we have not constrained the operation of the root node in terms of how it combines the outputs of the leaves, this may be achieved simply via a weighted combination without activation, or indeed weight adjustment during learning. Such a combination also describes a simple ensemble. Therefore, with one example we have considered the class of systems that encompasses fully connected MLPs, partially connected MLPs and some ensembles. Indeed, this can be further extended to those ensemble systems that use more complex combination techniques, or indeed even the ME system. A similar result was considered by Brown in his thesis [7] when he proposed the linkage between NC learning [25], Dyn-Co [17] and ME [20].

However, whilst we have shown the existence of a supervised training algorithm for this general class of multi-net systems, we have not made this concrete in any way. The proof of existence is powerful, but consequently limited. To be of use, making such an algorithm concrete is essential, but obviously relies upon

the ability of defining appropriate functions and parameters, something which is not trivial in itself.

5. Conclusion

In this paper we have provided a formal framework in which the general class of multi-net systems can be described. Furthermore, we have proven that, given an appropriately constructed partially ordered set, that there exists a learning algorithm that can be used to train the system to a given criterion, although we do not know what this algorithm might be. By way of proof, we have used backpropagation to demonstrate how such an algorithm can be constructed for this generic class of neural system.

We feel that a key contribution of this paper is that it takes a formal, abstract view of the area, abstract in that no reference is made in the model to numbers. Of course, in practical applications we need to make this concrete, but we believe that a considerable amount of neural network theory can be elucidated in its absence. Essentially, we are offering a different, and we believe novel, perspective on the problem area. This poses an interesting *mathematical* problem. Give an abstract multi-net system and criterion, is it possible to encode the various parameters and functions so that the latter are computable, that is recursive, and that the resulting system is isomorphic (does exactly the same thing as the abstract system)? A positive result would enhance the consequences of Theorem 2.

The next stages of this research are to consider what implications this has on existing multi-net architectures, and in particular whether this helps us to understand better multiple classifier systems, as well as exploring what properties of individual system components can be used to inform us about properties of the system as a whole. In the first instance this means looking at the properties of the combined system in order to systematically them break down to their component parts (sub-multi-nets), so that these can then be put back together to infer properties of the whole once again.

Acknowledgments

The authors would like to thank David Pitt for suggesting that we work together.

References

- [1] S.S.R.Abidi and K.Ahmad, Conglomerate Neural Network Architectures: The Way Ahead for Simulating Early Language Development, *Journal of Information Science and Engineering* 13 (2) (1997) 235-266.
- [2] K.Ahmad, M.C.Casey and T.Bale, Connectionist Simulation of Quantification Skills, *Connection Science* 14 (3) (2002) 165-201.
- [3] S.-I.Amari, Information Geometry of the EM and em Algorithms for Neural Networks, *Neural Networks* 8 (9) (1995) 1379-1408.
- [4] C.M.Bishop, *Neural Networks for Pattern Recognition* (Clarendon Press, Oxford, UK, 1995).
- [5] L.Bottou and P.Gallinari, A Framework for the Cooperation of Learning Algorithms, in: R.P.Lippmann, J.E.Moody, and

- D.S.Touretzky, ed., *Advances in Neural Information Processing Systems* (1991) 781-788.
- [6] L.Breiman, Bagging Predictors, *Machine Learning* 24 (2) (1996) 123-140.
- [7] G.Brown, Diversity in Neural Network Ensembles, Unpublished doctoral thesis, University of Birmingham, Birmingham, UK, 2004.
- [8] G.Brown, J.L.Wyatt, R.Harris and X.Yao, Diversity Creation Methods: A Survey and Categorisation, *Information Fusion* 6 (1) (2005) 5-20.
- [9] G.Brown, J.L.Wyatt and P.Tino, Managing Diversity in Regression Ensembles, *Journal of Machine Learning Research* 6 (2005) 1621-1650.
- [10] M.C.Casey, Integrated Learning in Multi-net Systems, Unpublished doctoral thesis, University of Surrey, Guildford, UK, 2004.
- [11] M.C.Casey and K.Ahmad, In-situ Learning in Multi-net Systems, in: Z.R.Yang, R.Everson, and H.Yin, ed., *Proceedings of the 5th International Conference on Intelligent Data Engineering and Automated Learning (IDEAL 2004)*, Lecture Notes in Computer Science 3177 (Springer-Verlag, Heidelberg, 2004) 752-757.
- [12] M.N.Dailey and G.W.Cottrell, Organization of Face and Object Recognition in Modular Neural Network Models, *Neural Networks* 12 (7-8) (1999) 1053-1073.
- [13] Y.Freund and R.E.Schapire, Experiments with a New Boosting Algorithm, in: *Machine Learning: Proceedings of the 13th International Conference (Morgan Kaufmann, 1996)* 148-156.
- [14] G.Fumera and F.Roli, A Theoretical and Experimental Analysis of Linear Combiners for Multiple Classifier Systems, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27 (6) (2005) 942-956.
- [15] M.S.Gazzaniga, Organization of the Human Brain, *Science* 245 (1989) 947-952.
- [16] S.Grossberg and D.V.Repin, A Neural Model of How the Brain Represents and Compares Multi-digit Numbers: Spatial and Categorical Processes, *Neural Networks* 16 (8) (2003) 1107-1140.
- [17] J.V.Hansen, Combining Predictors: Meta Machine Learning Methods and Bias/Variance & Ambiguity Decompositions, University of Aarhus, Aarhus, Denmark, 2000.
- [18] D.O.Hebb, *The Organization of Behavior: A Neuropsychological Theory* (John Wiley & Sons, New York, 1949).
- [19] R.A.Jacobs, M.I.Jordan and A.G.Barto, Task Decomposition through Competition in a Modular Connectionist Architecture: The What and Where Vision Tasks, *Cognitive Science* 15 (1991) 219-250.
- [20] R.A.Jacobs, M.I.Jordan, S.J.Nowlan and G.E.Hinton, Adaptive Mixtures of Local Experts, *Neural Computation* 3 (1) (1991) 79-87.
- [21] M.I.Jordan and R.A.Jacobs, Hierarchical Mixtures of Experts and the EM Algorithm, *Neural Computation* 6 (2) (1994) 181-214.
- [22] M.I.Jordan and L.Xu, Convergence Results for the EM Approach to Mixtures of Experts Architectures, *Neural Networks* 8 (1995) 1409-1431.
- [23] J.Kittler, M.Hatef, R.P.W.Duin and J.Matas, On Combining Classifiers, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20 (3) (1998) 226-239.
- [24] A.Krogh and J.Vedelsby, Neural Network Ensembles, Cross Validation, and Active Learning, in: G.Tesauro, D.S.Touretzky, and T.K.Leen, ed., *Advances in Neural Information Processing Systems* (1995) 231-238.
- [25] Y.Liu and X.Yao, Ensemble Learning via Negative Correlation, *Neural Networks* 12 (10) (1999) 1399-1404.
- [26] B.Lu and M.Ito, Task Decomposition and Module Combination Based on Class Relations: A Modular Neural Network for Pattern Classification, *IEEE Transactions on Neural Networks* 10 (5) (1999) 1244-1256.
- [27] J.Ma, L.Xu and M.I.Jordan, Asymptotic Convergence Rate of the EM Algorithm for Gaussian Mixtures, *Neural Computation* 12 (12) (2000) 2881-2908.
- [28] D.Partridge and N.Griffith, Multiple Classifier Systems: Software Engineered, Automatically Modular Leading to a Taxonomic Overview, *Pattern Analysis and Applications* 5 (2) (2002) 180-188.
- [29] M.P.Perrone, Improving Regression Estimation: Averaging Methods for Variance Reduction with Extensions to General Convex Measure Optimization, Unpublished doctoral thesis, Brown University, Providence, RI, 1993.
- [30] D.E.Rumelhart, G.E.Hinton and R.J.Williams, Learning Internal Representations by Error Propagation, in: D.E.Rumelhart and J.L.McClelland, ed., Volume 1: Foundations (MIT Press, Cambridge, MA., 1986) 318-362.
- [31] D.E.Rumelhart and D.Zipser, Feature Discovery by Competitive Learning, in: D.E.Rumelhart and J.L.McClelland, ed., Volume 1: Foundations (MIT Press, Cambridge, MA., 1986) 151-193.
- [32] A.J.C.Sharkey, Types of Multinet System, in: F.Roli and J.Kittler, ed., *Proceedings of the Third International Workshop on Multiple Classifier Systems (MCS 2002)* (Springer-Verlag, Berlin, Heidelberg, New York, 2002) 108-117.
- [33] A.J.C.Sharkey, Multi-Net Systems, in: A.J.C.Sharkey, ed., (Springer-Verlag, London, 1999) 1-30.
- [34] K.Tumer and J.Ghosh, Analysis of Decision Boundaries in Linearly Combined Neural Classifiers, *Pattern Recognition* 29 (2) (1996) 341-348.
- [35] N.Ueda and R.Nakano, Generalization Error of Ensemble Estimators, in: *Proceedings of the IEEE International Conference on Neural Networks* (1996) 90-95.
- [36] N.M.Wanas, L.Hodge and M.S.Kamel, Adaptive Training Algorithm for an Ensemble of Networks, in: *Proceedings of the 2001 International Joint Conference on Neural Networks (IJCNN'01)* (IEEE Computer Society Press, Los Alamitos, CA., 2001) 2590-2595.
- [37] L.Xu and M.I.Jordan, On Convergence Properties of the EM Algorithm for Gaussian Mixtures, *Neural Computation* 8 (1) (1996) 129-151.

Figure captions

Fig. 1. a) an ensemble system where the output of three components (u , v , and w) are combined using a weighted summation; b) the Hasse diagram for the equivalent system (V_{ens}), with the addition of a root node (t) in place of the weighted summation; c) a HME system consisting of two levels of experts (u , x , y), combined by two gating networks (w , z); d) the Hasse diagram for the equivalent system (V_{hme}), with the addition of two nodes (t , v) in place of the weighted summation.

Fig. 2. a) an arbitrary two layer MLP; b) the same MLP depicted with each layer as a node; c) the Hasse diagram for the equivalent system (V_{mlp}).

Figures

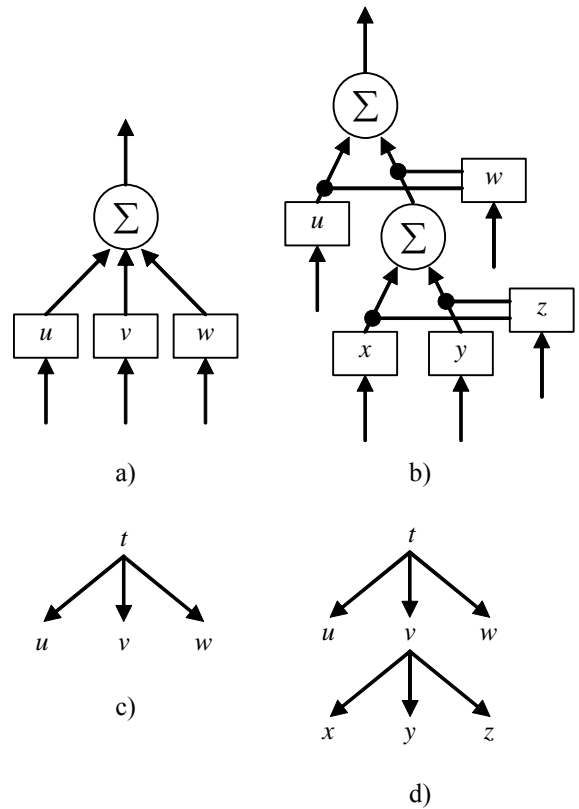


Fig. 1.

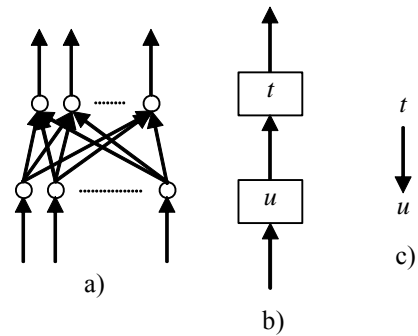


Fig. 2.