Constructive Neural Network Algorithms for Feedforward Architectures Suitable for Classification Tasks

Maria do Carmo Nicoletti, João R. Bertini Jr., David Elizondo, Leonardo Franco, and José M. Jerez

Abstract. This chapter presents and discusses several well-known constructive neural network algorithms suitable for constructing feedforward architectures aiming at classification tasks involving two classes. The algorithms are divided into two different groups: the ones directed by the minimization of classification errors and those based on a sequential model. In spite of the focus being on two-class classification algorithms, the chapter also briefly comments on the multiclass versions of several two-class algorithms, highlights some of the most popular constructive algorithms for regression problems and refers to several other alternative algorithms.

1 Introduction

Conventional neural network (NN) training algorithms (such as Backpropagation) require the definition of the NN architecture before learning starts. The common way for developing a neural network that suits a task consists of defining several different architectures, training and evaluating each of them, and then choosing the one most appropriate for the problem based on the error produced between the target and actual output values. Constructive neural network (CoNN) algorithms,

Maria do Carmo Nicoletti
CS Dept, UFSCar, S. Carlos, SP, Brazil
carmo@dc.ufscar.br

João R. Bertini Jr.
ICMC, USP, S. Carlos, SP, Brazil
bertini@icmc.usp.br

David Elizondo
De Monfort University, Leicester, UK
elizondo@dmu.ac.uk

Leonardo Franco and José M. Jerez
University of Málaga, Málaga, Spain
lfranco@lcc.uma.es,jja@lcc.uma.es

however, define the architecture of the network along with the learning process. Ideally CoNN algorithms should efficiently construct small NNs that have good generalization performance. As commented in (Muselli, 1998), "…the possibility of adapting the network architecture to the given problem is one of the advantages of constructive techniques… [This] has also important effects on the convergence speed of the training process. In most constructive methods, the addition of a new hidden unit implies the updating of a small portion of weights, generally only those regarding the neuron to be added".

The automated design of appropriate neural network architectures can be approached by two different groups of techniques : evolutionary and non-evolutionary. In the evolutionary approach, a NN can be evolved by means of an evolutionary technique, i.e. a population-based stochastic search strategy such as a GA (see (Schaffer *et al.*, 1992) (Yao, 1999)). In the non-evolutionary approach, the NN is built not as a result of an evolutionary process, but rather as the result of a specific algorithm designed to automatically construct it, as is the case with a constructive algorithm.

CoNN algorithms, however, are not the only non-evolutionary approach to the problem of defining a suitable architecture for a neural network. The strategy implemented by the so called pruning methods can also be used and consists in training a larger than necessary network (which presumably is an easy task) and then, pruning it by removing some of its connections and/or nodes (see (Reed, 1993)).

As approached by Lahnajärvi and co-workers in (Lahnajärvi *et al.*, 2002), pruning algorithms can be divided into two main groups. Algorithms in the first group estimate the sensitivity of an error function to the removal of an element (neuron or connection); those with the least effect can be removed. Algorithms in the second group generally referred to as penalty-term as well as regularization algorithms, "add terms to the objective function that reward the network for choosing efficient and small solutions". The same group of authors also detected in the literature what they call combined algorithms that take advantage of the properties of both, constructive and pruning algorithms, in order to determine the network size in a flexible way (see (Fiesler, 1994), (Gosh & Tumer, 1994)).

There are many different kinds of neural network; new algorithms and variations of already known algorithms are constantly being published in the literature. Similarly to other machine learning techniques, neural network algorithms can also be characterized as supervised, when the target values are known and the algorithm uses the information, or as unsupervised, when such information is not given and/or used by the algorithm. The two main classes of NN architecture are feedforward, where the connections between neurons do not form cycles, and feedback (or recurrent), where the connections may form cycles. NNs may also differ in relation to the type of data they deal with; the two more popular being categorical and quantitative. Both, supervised and unsupervised learning with categorical targets are referred to as classification. Supervised learning with quantitative target values is known as regression. Classification problems can be considered a particular type of regression problems.

This chapter reviews some well-known CoNN algorithms for feedforward architectures, suitable for classification tasks, aiming to present the main ideas they are based upon as well as to stress their main similarities and differences. An empirical evaluation of several two-class and multiclass CoNN algorithms in a variety of knowledge domains can be seen in (Nicoletti & Bertini, 2007). Although focusing mainly on a particular class of constructive neural network algorithms, the chapter also aims at establishing the importance of using a constructive approach when working with neural networks, independent of the application problem at hand. For this reason towards the end of the chapter a few other approaches will be briefly presented.

The remainder of the chapter is organized as follows: Section 2 presents the main characteristics of constructive neural network algorithms. Section 3 approaches several two-class CoNN algorithms by grouping them into (3.1) algorithms directed by the minimization of classification errors and (3.2) algorithms based on the sequential model. Section 4 briefly describes the multiclass versions of a few two-class CoNN previously presented as well highlighting the main characteristics of a few other multiclass proposals. Section 5 introduces some CoNN algorithms for regression problem as well as other combined approaches. In Section 6 several CoNN algorithms that do not quite conform to the main focus of this chapter and others that do not qualify to be part of the groups characterized before are briefly presented and finally, in Section 7, the conclusions of this chapter are presented.

2 Main Characteristics of CoNN Algorithms

Several constructive algorithms that focus on feedforward architectures have been proposed in the literature. During the learning phase they all essentially repeat the same process: incrementally adding and training hidden neurons (generally Threshold Logic Units – TLUs) until a stopping criterion is satisfied. Generally they all begin the process having as the initial network only the input layer; output neuron(s) are then added and trained and, depending on their performance, the algorithm starts to add and connect hidden neurons to the current architecture and train them, aiming at improving the accuracy of the network performance.

The final result at the end of the constructive process implemented by these algorithms is a neural network that had its architecture defined along with its training. In spite of sharing the same basic mechanism, CoNN algorithms differ from each other in many different ways, such as:

1. Number of nodes they add per layer at each iteration;
2. Direction in which they grow the network:
   - forward, from input towards output nodes or
   - backward, from output towards input nodes;
3. Functionality of the added neurons (do they all play the same role?);
4. Stopping criteria;
5. Connectivity pattern of the newly added neuron;

6. Algorithm used for training individual neuron, such as
    - The Fisher Discriminant
    - Pocket algorithm (Gallant, 1986a)
    - Pocket with Ratchet Modification (PRM) (Gallant, 1986a, 1990)
    - MinOver (Krauth and Mėzard, 1987)
    - Quickprop (Fahlman, 1988)
    - AdaTron (Anlauf & Biehl, 1989)
    - Thermal Perceptron algorithm (Frean, 1992)
    - Loss minimization (Hrycej, 1992)
    - Modified Thermal algorithm (Burgess, 1994)
    - Maxover (Wendemuth, 1995)
    - Barycentric Correction Procedure (BCP) (Poulard, 1995);
7. Type of input patterns they deal with: binary (or bipolar) valued, categorical or real valued attributes;
8. Type of problems they solve:
    - classification (two-class or multi-class), where the input is assigned to one of two or more classes
    - regression problems, characterized by a continuous mapping from inputs to an output or
    - clustering, where the patterns are grouped according to some similarity measure;
9. Topology of the connections among neurons (initially fixed or dynamically constructed);
10. 'Shape' of the feedforward architecture (e.g. tower-like, cascade-like, etc…).

Among the most well known CoNN algorithms for two-class classification problems are the Tower and the Pyramid (Gallant, 1986b), the Tiling (Mėzard & Nadal, 1989), the Upstart (Frean, 1990), the Perceptron Cascade (Burgess, 1994), the PTI and the Shift (Amaldi & Guenin, 1997), the Irregular Partitioning Algorithm (IPA) (Marchand *et al.*, 1990; Marchand & Golea, 1993), the Target Switch (Campbell & Vicente, 1995), the Constraint Based Decomposition (CBD) algorithm (Drăghici, 2001) and the BabCoNN (Bertini Jr. & Nicoletti, 2008a).

Smieja, (1993), Gallant, (1994), Bishop, (1996), Mayoraz and Aviolat (1996), Campbell (1997) and Muselli (1998) discuss several CoNN algorithms in detail.

3 A Closer Look at Several Two-Class CoNN Algorithms

In order to review several of the two-class CoNN algorithms previously mentioned in a systematic way, this section divides them into two main categories: (3.1) those directed by the minimization of classification errors and (3.2) those based on the sequential learning model.

3.1 Algorithms Directed by the Minimization of Classification Errors

Since most of the CoNN algorithms can be characterized as belonging to this category, a few sub-categories have been adopted in this review aimed at grouping algorithms that share similar characteristics.

3.1.1 Growing NNs with Single-Neuron Hidden Layers

The Tower and the Pyramid (Gallant, 1986b, 1994) algorithms are two-class CoNN algorithms that can be viewed as incremental versions of the PRM algorithm; both can be characterized as forward methods since they grow the NN layer by layer, from the input towards the output layer. In the Tower algorithm each new hidden node added to the network is connected to all input nodes and to the last previously added hidden node, making the architecture of the network look like a tower.

The basic idea of the Tower algorithm is very simple. Initially the PRM algorithm is used for training the only node of the first hidden layer of the network which can be considered the output node. If p is the number of attributes that describe a training pattern, the first hidden node receives p + 1 inputs i.e., the input values associated with the p attributes plus the constant value associated with the bias. As is well known a single neuron can learn with 100% precision only from linearly separable training sets. If that is the case, the first neuron added to the network and subsequently trained will correctly learn to separate both classes and the Tower algorithm is reduced to the PRM algorithm. However, if that is not the case, the Tower algorithm continues to add hidden layers to the network (each containing only one TLU) until a stopping criterion is satisfied.

Generally three stopping criteria can be implemented: 1) the NN correctly classifies the training set; 2) adding a new hidden layer does not contribute to increasing the network accuracy and 3) a predefined maximum number of hidden neurons has been reached. Considering that the first step adds the first hidden node to the network, the k^{th} step adds the k^{th} hidden node. After the first hidden node is added, all the subsequent hidden nodes added will receive an extra input value, which corresponds to the output of the last added hidden node. This extra dimension, added after the first hidden node was created represents the behavior of the newly created hidden node.

The Pyramid algorithm is very similar to the Tower algorithm. The only difference between them is that each new hidden node created by the Pyramid method is connected to every hidden node previously added to the network, as well as to the input nodes, making the network look like a pyramid. Each step of the learning phase expands the training patterns in one dimension (which represents the last hidden neuron added). Considering that the first step adds the first hidden node to the network, the k^{th} step adds the k^{th} hidden node that receives the p+1 input values from the input layer plus bias, as well as the output of the (k−1) hidden nodes previously added.

Gallant (1994) presents a detailed description of both algorithms as well as the proof of their convergence, by stating and proving the following theorem: "with arbitrarily high probability, the Tower (Pyramid) algorithm will fit noncontradictory sets of training examples with input values restricted to {+1,–1}, provided enough cells are added and enough iterations are taken for each added cell. Furthermore each added cell will correctly classify a greater number of training examples than any prior cell."

3.1.2 Growing NNs with Hidden Neurons Performing Different Functions

The Tiling algorithm (Mézard & Nadal, 1989) is a CoNN algorithm originally proposed for Boolean domains that trains a multilayer feedforward NN where hidden nodes are added to a layer in a way comparable to the process of laying tiles. The neurons in each hidden layer in a Tiling NN perform one out of two different functions and their names reflect their functionality. Each layer has a *master neuron* that works as the output neuron for that layer. If the master neuron does not correctly classify all training patterns, however, the Tiling algorithm starts to add and train *ancillary neurons*, one at a time, aiming at obtaining a faithful representation of the training set. The output layer has only one master neuron. The *faithfulness criterion* employed by the Tiling algorithm establishes that no two training patterns, belonging to different classes, should produce the same outputs at any given layer.

As commented by Gallant (1994), "The role of these units (ancillary) is to increase the number of cells for layer L so that no two training examples with *different classifications* have the *same set of activations* in layer L. Thus each succeeding layer has a different representation for the inputs, and no two training examples with different classifications have the same representation in any layer. Layers with this property are termed *faithful layers*, and faithfulness of layers is clearly a necessary condition for a strictly layered network to correctly classify all training examples".

In order to construct the network, the first step of the Tiling method is to train the master neuron of the first hidden layer, using the original training set, aiming at minimizing the classification error. If the master neuron does not classify the training set correctly, ancillary neurons are added to this layer and subsequently trained, one at a time, in order to obtain a faithful representation. Tiling constructs an NN in successive layers such that each new layer has a smaller number of neurons than the previous layer and layer L only receives connections from hidden layer L – 1.

The way the Tiling method operates assures that the master neuron in layer L classifies the training set with higher accuracy than the master neuron in layer L – 1. Assuming a finite training set with non-contradictory patterns, the Tiling algorithm is guaranteed to converge to zero classification errors (under certain assumptions) (Gallant, 1994). Once the first layer is finished, the Tiling algorithm goes on adding layers (with one master and a few ancillary neurons) until one out of four stopping criteria is satisfied: 1) the network converges; 2) the master neuron added degrades the performance of the network; 3) a pre-defined maximum number of layers is reached or 4) a pre-defined number of ancillary neurons per layer is reached and the training set still does not have a faithful representation in this layer. The original Tiling algorithm as described in (Mézard & Nadal, 1989) uses the same TLU training algorithm for each neuron (master or ancillary) added to the NN; in the same article the authors state and prove a theorem that ensures its convergence.

The unnamed algorithm proposed in (Nadal, 1989) and described by its author as an algorithm similar in spirit to the Tiling algorithm was named by Smieja in (Smieja, 1993) as *Pointing*. The Pointing algorithm constructs a feedforward neural network with the input layer, one output neuron and an ordered sequence of single hidden neurons, each one connected to the input layer and to the previous hidden neuron in the sequence. Despite its author claiming that the algorithm is very similar to the Tiling one, although more constrained (which is true), actually the Pointing algorithm corresponds to the Tower algorithm as proposed by Gallant (Gallant, 1986b).

The Partial Target Inversion (PTI) algorithm is a CoNN algorithm proposed in (Amaldi & Guenin, 1997) that shares strong similarities with the Tiling algorithm. The PTI grows a multi-layer network where each layer has one master neuron and a few ancillary neurons. Following the Tiling strategy as well, the PTI adds ancillary neurons to a layer in order to satisfy the faithfulness criteria; the neurons in layer c are connected only to neurons in layer $c-1$. If the training of the master neuron results in a weight vector that correctly classifies all training patterns or if the master neuron of layer c does not classify a larger number of patterns than the master neuron of layer $c-1$, the algorithm stops. If a training pattern, however, was incorrectly classified by the master neuron, and the master neuron correctly classifies a greater number of patterns than the master of the previous layer, the algorithm starts adding ancillary neurons to the current layer aiming at its faithfulness. When the current c layer becomes faithful the algorithm adds a new layer, $c+1$, initially only having the master neuron. The process continues until stopping criteria are met, such as when the number of master (or ancillary) neurons has reached a pre-defined threshold. The only noticeable difference between the Tiling and the PTI is the way the training set, used for training the ancillary neurons in the process of turning a layer faithful, is chosen.

For training the master neuron of layer c, the PTI uses all the outputs from the previous layer. Considering that the layer c needs to be faithful, the first ancillary neuron added to layer c will be trained with those patterns (used to train the master neuron) that made layer c unfaithful.

In addition, the patterns that activate the last added neuron (master, in this case) have their classes inverted. The authors justify this procedure by saying that "When trying to break several unfaithful classes simultaneously, it may be worthwhile to flip the target of the prototypes on layer c corresponding to several unfaithful classes. In fact, the specific targets are not important; the only requirement is that outputs corresponding to a different $c-1$ layer representation trigger a different output for at least one unit in the layer c". For clarification if after the addition of the first ancillary neuron the layer is not faithful yet, another ancillary neuron needs to be added to layer c. The second ancillary neuron will be trained with the $c-1$ outputs that provoked an unfaithful representation of layer c, this time, however, taking also into consideration the master and first ancillary neuron previously added. The patterns that activated the first ancillary neuron have their class inverted when they are included in the training set for the second ancillary neuron. The name PTI (Partial Target Inversion) refers to the fact that when constructing training sets only a partial number of patterns have their class (target) inverted.

3.1.3 Growing NNs Based on *Wrongly-On* and *Wrongly-Off* Errors

Generally CoNN algorithms based on discriminating between the two types of errors (*wrongly-on* and *wrongly-off*) tend to grow the neural network in a backward way, i.e. from the output towards the input layer.

The Upstart (Frean, 1990) is a constructive NN algorithm that dynamically grows a neural network whose structure resembles a binary tree. The algorithm starts the construction of the NN from the output layer towards the input layer and during the construction of an Upstart network, a neuron adds two other ancillary neurons in order to correct its misclassifications.

Let u_n be a neuron that classifies training patterns but produces *wrongly-off* errors (i.e. positive training patterns are misclassified by u_n as negative). The Upstart algorithm deals with *wrongly-off* errors by adding a *wrongly-off corrector* as a ‘child’ neuron u_{n+} which will try to correct the errors made by its parent u_n. The main tasks of neuron u_{n+} are 1) to correct the classification of positive training patterns that have been misclassified by u_n as negative and 2) to keep unchanged the other classifications made by u_n (i.e. u_{n+} should be inactive for any other pattern). The neuron u_{n+} is trained with the subset of training patterns that were *wrongly-off* plus the set of negative patterns.

Similarly, to deal with *wrongly-on* errors (i.e. negative training patterns that are misclassified by u_n as positive) the Upstart algorithm creates a *wrongly-on corrector* as a child neuron u_{n-} aiming: 1) to correct the classification of *wrongly-on* training patterns and 2) keep unchanged the other classifications made by neuron u_n (i.e. u_{n-} should stay inactive for any other pattern). Neuron u_{n-} is trained with the subset of training patterns that were *wrongly-on* plus the set of positive patterns.

When a neuron u_n does not correctly classify all training patterns, it gives rise to *wrongly-off* errors, *wrongly-on* errors or both. A hidden neuron in an Upstart network, therefore, can have up to two children. As can be seen in (Frean, 1990), two useful results follow from this training method because the children neurons have a simpler problem to solve than their parents. Children neurons can always make fewer misclassifications than their parents and connecting a child neuron to its parent with the appropriate weight will always reduce the misclassifications made by the parent .

For a child neuron (either a *wrongly-on* or a *wrongly-off corrector*) that has been added in order to correct misclassifications made by its parent, it is mandatory that this child only changes the activations of patterns that provoked the error; for this reason, the inactive output of the neuron should be 0. For a detailed description of Upstart, see also (Kubat, 2000).

Another algorithm that adopts the same error-correction strategy of the Upstart is the Perceptron Cascade (PC) algorithm (Burgess, 1994); PC also borrows the architecture of neural networks created by the Cascade Correlation algorithm (Fahlman & Lebiere, 1990).

PC begins the construction of the network by training the output neuron. If this neuron does not classify all training patterns correctly, the algorithm begins to add hidden neurons to the network. Each new added hidden neuron is connected to all previous hidden neurons as well as to the input neurons. The new hidden neuron is then connected to the output neuron; each time a hidden neuron is added, the output neuron needs to be retrained. The addition of a new hidden neuron enlarges the space in one dimension. The algorithm has three stopping criteria: 1) the network converges i.e. correctly classifies all training patterns; 2) a pre-defined maximum number of hidden neurons has been achieved and 3) the most common, the addition of a new hidden neuron degrades the network's performance.

Similarly to the Upstart algorithm, hidden neurons are added to the network in order to correct *wrongly-on* and *wrongly-off* errors. Following the same strategy employed by Upstart, what distinguishes a neuron created for correcting *wrongly-on* or for correcting *wrongly-off* errors caused by the output neuron is the training set used for training the neuron. To correct *wrongly-on* errors the training set used should have all negative patterns plus the patterns which produce *wrongly-on* errors. For correcting *wrongly-off* errors, the training set should have all positive patterns plus the negative patterns which produce *wrongly-off* patterns. Unlike Upstart, however, the PC algorithm only adds and trains one neuron at a time in order to correct the most frequent *wrongly-on* and *wrongly-off* errors produced by the output neuron.

Like the Upstart and Perceptron Cascade algorithms, the Shift algorithm (Amaldi & Guenin, 1997) also constructs the network beginning with the output neuron. This algorithm, however, creates only one hidden layer, iteratively adding neurons to it; each added neuron is connected to the input neurons and to the output neuron. The error correcting procedure used by the Shift method is similar to the one used by the Upstart method, in the sense that the algorithm also identifies *wrongly-on* and *wrongly-off* errors. However, it adds and trains a hidden neuron to correct the most frequent between these two types of errors. Also, the training set used for training a *wrongly-off* (or *wrongly-on*) corrector differs slightly from the training sets used by the Upstart algorithm.

3.2 Algorithms Based on the Sequential Model

The general CoNN model identified as sequential learning was proposed in (Marchand *et al.*, 1990) for Boolean domains and is an improved version of a previous proposal described in (Ruján & Marchand, 1989). In (Marchand & Golea, 1993) the algorithm is extended to deal with real data. Contrary to many of the CoNN algorithms, the sequential learning model is not directed by the minimization of the classification error.

Typically, a sequential learning algorithm adds to the network hidden neurons that are partial classifiers i.e. they separate a group of training patterns belonging to the same class from the remaining patterns in the training set; this was formally stated in (Muselli, 1998) as: Let Q^+ and Q^- be two subsets of the input space; a neuron will be called a *partial classifier* if it provides an output of +1 for at least one pattern of Q^+ and output of –1 for all elements of Q^-.

Any algorithm that implements the sequential learning model should use an efficient strategy for training individual TLUs as partial classifiers. Constructive algorithms based on the sequential learning model produce networks having partial classifiers as hidden nodes. In the paper (Marchand *et al.*, 1990) the authors describe a mechanism that uses the Perceptron as an auxiliary process for identifying the weight vector. However, as pointed out in (Poulard & Hernandez, 1997), this mechanism is time consuming and becomes prohibitive for large training sets and the authors suggest the BCPMax instead (Poulard & Labreche, 1995), which associates the BCP (Poulard, 1995) with a Pattern Exclusion Procedure (PEP); initially designed for the BCP the PEP finds the best value of the neuron's threshold for a fixed value of the weight vector: the one maximizing the number of excluded patterns.

The sequential learning model has been implemented by a few algorithms, namely Marchand *et al.*´s own proposal known in the literature as the Irregular Partitioning Algorithm (IPA), the Carve algorithm (Young & Downs, 1998), the Target Switch algorithm (Campbell & Vicente, 1995), the Oil Spot Algorithm (Mascioli & Martinelli, 1995), the Constraint Based Decomposition (CBD) algorithm, proposed in (Drăghici, 2001) and the Decomposition Algorithm for Synthesis and Generalization (DASG) recently introduced in (Subirats, Jerez and Franco, 2008).

The IPA algorithm creates a neural network with an input layer, a single hidden layer and the output neuron. The connections between the hidden layer and the output neuron have weights and the output neuron has a bias. The hidden layer is created by sequentially adding neurons to it; each added neuron represents a hyperplane that separates the greatest number of patterns belonging to the same class from the rest of the training set. Once the hyperplane is found, the identified patterns belonging to the same class are removed from the training set and the procedure is repeated. The process ends when the training set only has patterns belonging to the same class.

The Target Switch algorithm was originally designed to deal with binary patterns. The algorithm can induce two different network structures, namely a cascade that uses linear neurons and a tree-like structure that uses threshold neurons. The algorithm is based on the concept of *dichotomy* which, for a classification problem with a training set $E = E^- \cup E^+$ can be summarised as: "A set of weights and thresholds which correctly store all the E^+ patterns and some of the E^- will be said to induce a *(+)dichotomy* while a *(–)dichotomy* will correspond to correct storage of all the E^- patterns and some of the E^+ " (Campbell, 1997).

For growing either type of structure, neurons are always added in pairs, one for inducing a *(+)dichotomy* and the other for inducing a *(–)dichotomy*. The patterns belonging to E^- that are correctly stored by the *(+)dichotomy* and those belonging to E^+ that are correctly stored by the *(–)dichotomy* are the patterns that will be correctly separated by the neuron introduced in order to connect the above mentioned pair.

When the architecture is a cascade type, the introduced neuron is a linear neuron that implements a summation function on the pair outputs. If the result is positive or negative the current pattern is correctly classified otherwise a misclassification is produced, which will be dealt with by the next pair of neurons to be added. For growing neural networks with a tree structure the introduced neuron is a threshold neuron that implements a threshold function on the pair outputs. Considering that the first iteration adds one threshold neuron (the output), each following iteration will add two more threshold neurons to those already added in the previous iteration.

To obtain the dichotomies the authors propose the use of any Perceptron-like TLU training algorithm. Roughly speaking, the idea is to run the TLU training algorithm and then shift the resulting hyperplane in order to correctly classify all patterns of a given class.

The Constraint Based Decomposition (CBD) is another algorithm that follows the sequential model. The algorithm builds an architecture with four layers which are named input, hyperplane, AND and OR layers respectively. The whole training set is used for training the first hidden neuron in the hyperplane layer. The next hidden neuron to be added will be trained with those training patterns that were misclassified by the first hidden neuron. The algorithm goes on adding neurons to the first layer until no pattern is left in the training set. For training a neuron u_i, one pattern from each class is randomly chosen and removed from the training set E. These patterns are put in the training set E_{u_i}. After u_i has been trained with E_{u_i}, the algorithm starts to add patterns to E_{u_i}, one at a time, in a random manner. Each time a pattern is added to the set, u_i is retrained with the updated E_{u_i}. However, if the addition of a new pattern to E_{u_i} results in misclassification, the last pattern added is removed from E_{u_i} and marked as ´used´ by the neuron. Before adding a new hidden neuron, the algorithm considers all patterns in E for the current neuron. A new neuron will be added when all training patterns left have been tried for the current neuron. The neurons of the AND layer are connected only to relevant neurons from the hyperplane layer and in the OR layer the output neurons are connected only to neurons from the AND layer which are turned on for the given class.

The recently introduced DASG algorithm belongs also to the class of sequential learning algorithm. It works with binary inputs by decomposing the original Boolean function (or partially defined Boolean function) into two new lower complexity functions, which in turn are decomposed until all obtained functions are threshold functions that can be implemented by a single neuron. The final solution incorporates all functions in a single hidden layer architecture with an output neuron that computes and OR or AND Boolean function.

The BabCoNN (Barycentric-based CoNN) (Bertini Jr. & Nicoletti, 2008a) is a new two-class CoNN that borrows some of the ideas of the BCP (Barycentric Correction Procedure, see (Poulard, 1995), (Poulard & Labreche 1995)) and can be considered a representative of the sequential model. Like the Upstart,

Table 1 Overview of fourteen two-class CoNN algorithm characteristics

Algorithm	Group	# HL Growth direction	New Neuron Connected to	Special Feature	Stopping criteria
Tower	One HN per HL	Various Forward	Previously added HN and INs	Weight update	CON AD NHL
Pyramid	One HN per HL	Various Forward	All previously added HNs and INs	Dimension of weight space increases	CON AD NHL
Tiling	Neurons perform different functions	Various Forward	Previous layer	Faithful layers – divide and conquer	CON AD NHL NHN
PTI	Neurons perform different functions	Various Forward	Previous layer	Faithful layers – inversion of classes	CON AD NHL NHN
Upstart	Wrongly-on/off correctors	Binary tree Backward	Parent neuron	Children correct the father's mistakes	CON AD NHL
Shift	Wrongly-on/off correctors	One Backward	INs	Weighted connections are used to correct the output error	CON AD NHL
Perceptron cascade	Wrongly-on/off correctors	Cascade-like Backward	Previously added HNs and INs	Output increases the dimension of its weight space every time a neuron is added	CON AD NHL
Cascade correlation	Wrongly-on/off correctors	Cascade-like Backward	Previously added HNs and INs	Suitable for regression tasks	CON AD NHL
Offset	Neurons perform different functions	Two Forward	Previous layer	Parity machine	CON AD NHL
IPA	Sequential	One Forward	INs	Sequentially classifies the training set	TSC
Target switch	Sequential	Cascade (tree-like) Backward	Previously added HNs and INs	(+) and (–) dichotomies	TSC
CBD	Sequential	Three Forward	Previous layer	AND/OR layers	TSC
BabCoNN	Sequential	One Backward	Input	HN fires −1, 0 or 1	TSC
DASG	Sequential	One Forward	Input	AND/OR output function	TSC

Perceptron Cascade (PC) and Shift, the BabCoNN also constructs the network, beginning with the output neuron. However, it creates only one hidden layer; each hidden neuron is connected to the input layer as well as to the output neuron, like with the Shift algorithm.

Although the Upstart, PC and Shift construct the network by adding new hidden neurons specialized in correcting *wrongly-on* and *wrongly-off* errors, the BabCoNN employs a different strategy. The BCP is used for constructing a hyperplane in the hyperspace defined by the training set; the classified patterns are removed from the set and the process is repeated again with the updated training set. Due to the way the algorithm works a certain degree of redundancy is inserted in the process, in the sense of a pattern being correctly classified by more than one hidden neuron. This has been fixed by the BabCoNN classification process, where hidden neurons have a particular way of firing their output.

Table 1 summarizes the main characteristics of fourteen two-class algorithms previously discussed. For presenting the table the following abbreviations were adopted: Forward (the NN is grown from input towards output layer); Backward (the NN is grown from output towards input layer); INs: all neurons in the input layer; HN: a hidden neuron; HL: a hidden layer; #HL: number of hidden layers. The following abbreviations were adopted for stopping criteria: CON (convergence); AD (accuracy decay); NHL (number of hidden layers exceeds a given threshold); NHN (number of hidden neurons per hidden layer exceeds a given threshold); TSC (all training patterns have been correctly classified).

4 A Brief Approach to Multiclass Classification Using CoNN

A multiclass classification problem is a classification problem involving *m* (> 2) classes usually treated as *m* two-class problems. Generally multiclass CoNN start by training as many output neurons as there are classes in the training set, using one of two strategies: individual (I) and winner-takes-all (WTA).

The multiclass versions of a few two-class algorithms have been proposed in (Parekh *et al.*, 1995), (Yang *et al.*, 1996), (Parekh *et al.*, 1997a), (Parekh *et al.*, 1997b), (Parekh *et al.*, 2000) and they are the MTower, MPyramid, MTiling, MUpstart and MPerceptron Cascade, which can be considered extensions of their two-class counterparts.

The MTower algorithm deals with an m-class problem by adding and training *m* hidden neurons per hidden layer at each iteration. In an MTower architecture each of the *m* neurons in a certain hidden layer has connections with all the neurons of the input layer as well as with all the *m* neurons of the previous hidden layer. The MPyramid also deals with an m-class problem by adding and training *m* hidden neurons per hidden layer at each iteration. The *m* hidden neurons in each hidden layer, however, are connected to all the hidden neurons of all the hidden layers as well as to the input neurons.

Although the two-class Upstart algorithm constructs the neural network as a binary tree of TLUs starting with the output neuron, its multiclass version, the MUpstart, creates a network with a single hidden layer where each single hidden neuron is directly connected to every neuron in the output layer. The input layer is fully connected to the hidden neurons as well as to the output neurons. As mentioned before, each neuron added to a hidden layer by the Tiling algorithm can be a master (one per hidden layer) or an ancillary neuron (a few per layer). The MTiling also constructs a multi layer neural network where the first hidden layer has connections to the input layer and each subsequent hidden layer has connections only to the previous hidden layer. For training data containing *m* classes (> 2), MTiling adds *m* master neurons and as many ancillary neurons as necessary to make the layer faithful. The output layer has exactly *m* neurons.

The multiclass MPerceptron-Cascade is very similar to the MUpstart. Their main difference is the architecture of the neural network they induce. While the MUpstart adds the new hidden neurons in a single layer, the MPerceptron-Cascade adds the new hidden neurons in new layers. The MBabCoNN (Bertini Jr. & Nicoletti, 2008b) is the multiclass version of BabCoNN and constructs a network beginning with the output layer containing as many neurons as there are classes in the training set (each output neuron is associated to a class). The algorithm allows the neurons to be trained using any TLU algorithm combined with either strategy, individual or WTA. After adding *m* output neurons, the algorithm starts to add neurons to its single hidden layer in order to correct the classification mistakes made by the output neurons. A detailed description of MBabCoNN and an empirical evaluation of its performance *versus* the performance of several multiclass CoNN algorithms is described in one chapter of this book and is an extended version of the paper (Bertini Jr. & Nicoletti, 2008b).

5 CoNN Algorithms for Regression Problems and Combined Approaches

In spite of their strong focus on classification tasks, many CoNN proposals specifically aim at regression problems (see (Kwok & Yeung, 1997a), (Ma & Khorasani, 2003), (Ma & Khorasani, 2004)). A review of the CoNN algorithms for regression problems, approached from the perspective of a state-space search can be seen in (Kwok & Yeung, 1997a).

In their proposed taxonomy Kwok & Yeung group the algorithms into six different categories, each named after its most representative algorithm (1) Dynamic node creation (DNC) (Ash, 1989); (2) Projection pursuit regression, based on the statistical technique proposed in (Friedman & Stuetzle, 1981); (3) Cascade-Correlation, that mostly groups variants of the cascade-correlation architecture proposed in (Fahlman & Lebiere, 1990); (4) Resource-allocating networks (RAN) (Platt, 1991); (5) Group methods of data handling, a class of algorithms inspired by the GMDH proposed by Ivakhnenko and described in (Farlow, 1984) and (6) Miscellaneous, a category that groups CoNN that have ‘multivaluated state transition mappings while still retraining the whole network upon hidden unit addition’. In the last category, however, the authors only talk about a hybrid algorithm, proposed in (Nabhan & Zomaya, 1994) that employs both, a constructive and a pruning strategy.

Kwok & Yeung in (Kwok & Yeung, 1997b) conducted a very careful investigation on the objective functions for training hidden neurons in CoNN for multilayer feedforward networks for regression problems, aiming at deriving a class of objective functions whose value and the corresponding weight updates could be computed in O(N) time, for a training set with N patterns.

In spite of the many CoNN algorithms surveyed in (Kwok & Yeung, 1997a), the most popular for regression problems is no doubt the Cascade Correlation algorithm (CasCor) and maybe the second most popular is the DNC. While the DNC algorithm constructs neural networks with a single hidden layer, the CasCor creates them with multiple hidden layers, where each hidden layer has one hidden neuron. The popularity of CasCor can be attested by the various ways this algorithm has inspired new variations and also has been used in the combined approaches between learning methods.

A similar approach to CasCor called Constructive Backpropagation (CBP) was proposed in (Lehtokangas, 1999). The RCC, a recurrent extension to CasCor is described in (Fahlman, 1991) and its limitations are presented and discussed in (Giles *et al.*, 1995). In (Kremer, 1996) the conclusions of Giles *et al.* in relation to RCC are extended. An investigation into problems and improvements in relation to the basic CasCor can be found in (Prechelt, 1997), where CasCor and five of its variations are empirically compared using 42 different datasets from the benchmark PROBEN1 (Prechelt, 1994).

CasCor has also inspired the proposal of the Fixed Cascade Error (FCE), described in (Lahnajärvi *et al.*, 1999c), (Lahnajärvi *et al.*, 2002), which is an enhanced version of a previous algorithm proposed by the same authors known as Cascade Error (CE) (see (Lahnajärvi *et al.*, 1999a), (Lahnajärvi *et al.*, 1999b)). While the general structure of both algorithms is the same, they differ in the way the hidden neurons are created.

The Rule-based Cascade-correlation (RBCC) proposed in (Thivierge *et al.*, 2004) is a collaborative symbolic-NN approach which is partially inspired by the KBANN (Knowledge-Based Artificial Neural Networks) model proposed in (Towel *et al.*, 1990), (Towel, 1991) where the NN used is a CasCor network. In the KBANN an initial set of rules is translated into a neural network which is then refined using a training set of patterns; the refined neural network can undergo a further step and be converted into a set of symbolic rules which could, again, be used as the starting point for constructing a neural network and the whole cycle would be repeated.

According to the authors the RBCC is a particular case of the Knowledge-based Cascade-correlation algorithm (KBCC) (Shultz & Rivest, 2000) (Shultz & Rivest, 2001). The KBCC extends the CasCor by allowing as hidden neurons during the growth of a NN not only single neurons, but previously learned networks as well. In (Thivierge et al., 2003) an algorithm that implements simultaneous growing and pruning of CasCor networks is described; the pruning is done by removing irrelevant connections using the Optimal Brain Damage (OBD) procedure (Le Cun *et al.*, 1990).

In (Islam & Murase, 2001) the authors propose the CNNDA (Cascade Neural Network Design Algorithm) for inducing two-hidden-layer NNs. The method automatically determines the number of nodes in each hidden layer and can also reduce a two-hidden-layer network to a single-layer network. It is based on the use of a temporary weight freezing technique. The Fast Constructive-Covering Algorithm (FCCA) for NN construction proposed in (Wang, 2008) is based on geometrical expansion. It has the advantage of each training example having to be learnt only once, which allows the algorithm to work faster than traditional training algorithms.

6 Miscellaneous

A few constructive approaches have also been devised for RBF (Radial Basis Function) networks, such the Orthogonal Least Squares (OLS) (Chen *et al.*, 1989) (Chen *et al.*, 1991) and the Growing Radial Basis Function (GRBF) networks (Karayiannis & Weiqun, 1997).

Although CoNN algorithms seem to have a lot of potential in relation to both the size of the induced network and its accuracy, it is really surprising that their use, particularly in the area of classification problems, is not as widespread as it should be, considering their many advantages. In regression problems, however, CoNNs have been very popular, particularly the Cascade-Correlation algorithm and many of its variations. In what follows some of the most recent works using CoNN are mentioned.

In (Lahnajärvi *et al.*, 2004) four CasCor-based CoNN algorithms, have been used for evaluating the movements of a robotic manipulator. In (Huemer *et al.*, 2008) the authors describe a method for controlling machines, such as mobile robots, using a very specific CoNN. The NN is grown based on a reward value given by a feedback function that analyses the on-line performance of a certain task. In fact since conventional NNs are commonly used in controlling tasks (Alnajjar & Murase, 2005), this is a potential application area for CoNN algorithms as well.

In (Giordano *et al.*, 2008), a committee of CasCor neural networks was implemented as a software filter, for the online filtering of CO_2 signals from a bioreactor gas outflow. The knowledge-based CasCor proposal (KBCC) previously mentioned has been used in a few knowledge domains, such as simulation of cognitive development (see e.g. (Mareschal & Schultz, 1999) and (Sirois & Shultz, 1998)), vowel recognition (Rivest & Shultz, 2002) and for gene-splice-junction determination (Thivierge & Shultz, 2002), a benchmark problem from the UCI Machine Learning Repository (Asuncion & Newman, 2007). A more in depth investigation into the use of the knowledge-based neural learning implemented by the KBCC in developmental robotics can be seen in (Shultz *et al.*, 2007).

A few other non-conventional approaches to CoNN can be found in recent works, such as the one described in (García-Pedrajas & Ortiz-Boyer, 2007), based on cooperative co-evolution, for the automatic induction of the structure of an NN for classification purposes; the method partially tries to avoid the problems of greedy approaches. In (Yang *et al.*, 2008) the authors combined the ridgelet function with feedforward neural networks in the ICRNN (Incremental Constructive Ridgelet Neural Network) model. The ridgelet function was chosen as the activation function due to its efficiency in describing linear-, curvilinear- and hyperplane-like structures in its hidden layer; the structure of the network is induced via a constructive method.

The CoNN classifier known as the Recursive Deterministic Perceptron (RDP) (Tajine & Elizondo, 1996) is a generalization of the Perceptron, capable of solving any two-class classification problem. It works by transforming any non-linearly separable two-class problem into a linearly one, which can be easily learnt by the Perceptron. Its multiclass version (Tajine *et al.* 1997) is a generalization that allows separation of the m-classes in a deterministic way. Results show that in certain domains, both the multiclass version and the backpropagation have similar performance (Elizondo *et al.*, 2008).

The Switching Neural Network (SNN) is a connectionist model recently proposed in (Muselli, 2006) suitable for classification problems. The first layer of an SNN contains converters, called latticizers that change the representation of the input vectors into binary strings. The two other layers of the SNN represent a Boolean function that solves, in the lattice domain, the original classification problem. As proposed in (Ferrari & Muselli, 2008) the construction of an SNN can be done by a constructive algorithm known as Switch Programming (SP) which is based on solutions of a linear programming problem. Good simulation results suggest that this proposal is worthy of a deeper investigation.

The constructive proposals CLANN and its multiclass version M-CLANN described in (Tsopzé *et al.*, 2007) and (Nguifo *et al.*, 2008) respectively are based on concept lattices and aim at a semantic interpretation of the involved neurons and consequently at an 'interpretable' (in the sense of comprehensibility) neural network. CLANN and M-CLANN can be approached as representation-translators, in the same sense as the KBANN model is (Towel *et al.*, 1990), (Towel, 1991).

A different approach to CoNN can be found in (Barreto-Sanz *et al.*, 2008), where the authors propose the FGHSON (Fuzzy Growing Hierarchical Self-Organizing Networks), an adaptive network method capable of representing the underlying structure of the data, in a hierarchical fuzzy way.

Transformation of original data features usually helps to find interesting low dimensional data that can reveal previously unseen structures. This process aims to ease the problem for a classifier. The simplest of these transformations is the linear projection. Many methods search for the optimal and the most informative linear projection. Friedman (Friedman, 1987) proposed a framework to find interesting data transformations by maximizing an index of projection pursuit. Grochowski and Duch in (Grochowski & Duch, 2008) proposed the QPC network, a constructive neural network that can implement this framework. The algorithm introduces a new index based on the quality of projected clusters that can be used to define specific representations for the hidden layer of a neural network and may help to construct the network.

The recently introduced C-Mantec algorithm (Subirats, Franco et al, 2008) that works by error correction using the thermal perceptron (Frean, 1992) incorporates competition between neurons in the hidden layer and it has been shown to lead to very compact architectures. The intrinsic dynamics of the algorithm has been applied for detecting and filtering noisy instances, reducing overfitting and improving the generalization ability.

7 Conclusions

This chapter presents an overview of several CoNN algorithms and highlighted some of their applications and contributions. Although focusing on feedforward architectures for classification tasks, the chapter also tries to present a broad view of the area, discussing several of the most recent contributions.

An interesting aspect of CoNN research is its chronological aspect. It may be noticeable that most of the CoNN algorithms for classification tasks were proposed in the nineties and since then not many new proposals have been published. Another point to consider also is the lack of real world applications involving the use of CoNN algorithms; this can be quite surprising, considering the many that are available and the fact that several have competitive performances in comparison to other more traditional approaches. The tendency in the area is for diversifying both the architecture and the constructive process itself, by means of including collaborative techniques. What has been surveyed in this chapter is just a part of the research work going on in the area of CoNN algorithms. As mentioned in the Introduction, there is a very promising area characterized as the group of evolutionary techniques that has been contributing a lot to the development of CoNNs and was not the subject of this chapter.

Acknowledgments. To FAPESP and CAPES for the support provided to M. C. Nicoletti and to J. R. Bertini Jr. respectively. L. Franco, D. Elizondo and J.M. Jerez acknowledge support from MICIIN through grant TIN2008-04985 and to Junta de Andalucía grants P06-TIC-01615 and P08-TIC-04026.

References

Alnajjar, F., Murase, K.: Self-organization of spiking neural network generating autonomous behavior in a real mobile robot. In: Proceedings of The International Conference on Computational Intelligence for Modeling, Control and Automation, vol. 1, pp. 1134–1139 (2005)

Amaldi, E., Guenin, B.: Two constructive methods for designing compact feedfoward networks of threshold units. International Journal of Neural System 8(5,6), 629–645 (1997)

Anlauf, J.K., Biehl, M.: The AdaTron: an adaptive perceptron algorithm. Europhysics Letters 10, 687–692 (1989)

Ash, T.: Dynamic node creation in backpropagation networks, Connection Science, vol. Connection Science 1(4), 365–375 (1989)

Asuncion, A., Newman, D.J.: UCI Machine Learning Repository. University of California, School of Information and Computer Science, Irvine (2007), `http://www.ics.uci.edu/~mlearn/MLRepository.html`

Barreto-Sanz, M., Pérez-Uribe, A., Peña-Reyes, C.-A., Tomassini, M.: Fuzzy growing hierarchical self-organizing networks. In: Kůrková, V., Neruda, R., Koutník, J. (eds.) ICANN 2008,, Part II. LNCS, vol. 5164, pp. 713–722. Springer, Heidelberg (2008)

Bertini Jr., J.R., Nicoletti, M.C.: A constructive neural network algorithm based on the geometric concept of barycenter of convex hull, Computational Intelligence: Methods and Applications. In: Rutkowski, R.L., Tadeusiewicz, R., Zadeh, L.A., Zurada, J. (eds.) Academic Publishing House Exit, pp. 1–12. IEEE Computational Intelligence Society, Poland (2008a)

Bertini Jr., J.R., Nicoletti, M.C.: MBabCoNN – A multiclass version of a constructive neural network algorithm based on linear separability and convex hull. In: Kůrková, V., Neruda, R., Koutník, J. (eds.) ICANN 2008,, Part II. LNCS, vol. 5164, pp. 723–733. Springer, Heidelberg (2008)

Bishop, C.M.: Neural networks for pattern recognition. Oxford University Press, USA (1996)

Burgess, N.: A constructive algorithm that converges for real-valued input patterns. Int. Journal of Neural Systems 5(1), 59–66 (1994)

Campbell, C.: Constructive learning techniques for designing neural network systems. In: Leondes, C. (ed.) Neural Network Systems Technologies and Applications, vol. 2. Academic Press, San Diego (1997)

Campbell, C., Vicente, C.P.: The target switch algorithm: a constructive learning procedure for feed-forward neural networks. Neural Computation 7(6), 1245–1264 (1995)

Chen, S., Billings, S., Luo, W.: Orthogonal least squares learning methods and their application to non-linear system identification. International Journal of Control 50, 1873–1896 (1989)

Chen, S., Cowan, C., Grant, P.M.: Orthogonal least squares learning algorithm for radial basis function networks. IEEE Transactions on Neural Networks 2, 302–309 (1991)

Drăghici, S.: The constraint based decomposition (CBD) training architecture. Neural Networks 14, 527–550 (2001)

Elizondo, D., Ortiz-de-Lazcano-Lobato, J.M., Birkenhead, R.: On the generalization of the m-class RDP neural network. In: Kůrková, V., Neruda, R., Koutník, J. (eds.) ICANN 2008, Part II. LNCS, vol. 5164, pp. 734–743. Springer, Heidelberg (2008)

Fahlman, S.E.: Faster-learning variations on backpropagation: an empirical study. In: Touretzky, D.S., Hinton, G.E., Sejnowski, T.J. (eds.) Proceedings of the 1988 Connectionist Models Summer School, pp. 38–51. Morgan Kaufmann, San Mateo (1988)

Fahlman, S., Lebiere, C.: The cascade correlation architecture, Advances in Neural Information Processing Systems, vol. 2, pp. 524–532. Morgan Kaufman, San Mateo (1990)

Fahlman, S.: The recurrent cascade-correlation architecture, Advances in Neural Information Processing Systems, vol. 3, pp. 190–196. Morgan Kaufman, San Mateo (1991)

Farlow, S.J. (ed.): Self-organizing methods in Modeling: GMDH Type Algorithms. In: Statistics: Textbooks and Monographs, vol. 54. Marcel Dekker, New York (1984)

Ferrari, E., Muselli, M.: A constructive technique based on linear programming for training switching neural networks. In: Kůrková, V., Neruda, R., Koutník, J. (eds.) ICANN 2008, Part II. LNCS, vol. 5164, pp. 744–753. Springer, Heidelberg (2008)

Fiesler, E.: Comparative bibliography of ontogenic neural networks. In: Proceedings of The International Conference on Artificial Neural Networks (ICANN), Sorrento, vol. 94, pp. 793–796 (1994)

Frean, M.: The upstart algorithm: a method for constructing and training feedforward neural networks. Neural Computation 2, 198–209 (1990)

Frean, M.: A thermal perceptron learning rule. Neural Computation 4, 946–957 (1992)

Friedman, J.H., Stuetzle, W.: Projection pursuit regression. Journal of the American Statistical Association 76(376), 817–823 (1981)

Friedman, J.: Exploratory projection pursuit. Journal of the American Statistical Association 82, 249–266 (1987)

Gallant, S.I.: Optimal linear discriminants. In: Proceedings of The Eighth International Conference on Pattern Recognition, pp. 849–852 (1986a)

Gallant, S.I.: Three constructive algorithms for network learning. In: Proceedings of The Eighth Annual Conference of the Cognitive Science Society, Amherst, Ma, pp. 652–660 (1986b)

Gallant, S.I.: Perceptron based learning algorithms. Proceedings of the IEEE Transactions on Neural Networks 1(2), 179–191 (1990)

Gallant, S.I.: Neural Network Learning & Expert Systems. The MIT Press, England (1994)

García-Pedrajas, N., Ortiz-Boyer, D.: A cooperative constructive method for neural networks for pattern recognition. Pattern Recognition 40, 80–98 (2007)

Ghosh, J., Tumer, K.: Structural adaptation and generalization in supervised feed-forward networks. Journal of Artificial Neural Networks 1(4), 431–458 (1994)

Giles, C.L., Chen, D., Sun, G.-Z., Chen, H.-H., Lee, Y.-C., Goudreau, M.W.: Constructive learning of recurrent neural network: limitations of recurrent cascade correlation and a simple solution. IEEE Transactions on Neural Networks 6(4), 829–836 (1995)

Giordano, R.C., Bertini Jr., J.R., Nicoletti, M.C., Giordano, R.L.C.: Online filtering of CO2 signals from a bioreactor gas outflow using a committee of constructive neural networks. Bioprocess and Biosystems Engineering 31(2), 101–109 (2008)

Grochowski, M., Duch, W.: Projection pursuit constructive neural networks based on quality of projected cluster. In: Kůrková, V., Neruda, R., Koutník, J. (eds.) ICANN 2008,, Part II. LNCS, vol. 5164, pp. 754–762. Springer, Heidelberg (2008)

Hrycej, T.: Modular Learning in Neural Networks. Addison Wiley, New York (1992)

Huemer, A., Elizondo, D., Gongora, M.: A reward-value based constructive method for the autonomous creation of machine controllers. In: Kůrková, V., Neruda, R., Koutník, J. (eds.) ICANN 2008,, Part II. LNCS, vol. 5164, pp. 773–782. Springer, Heidelberg (2008)

Islam, M.M., Murase, K.: A new algorithm to design compact two-hidden-layer artificial networks. Neural Networks 14, 1265–1278 (2001)

Karayiannis, N.B., Weiqun, G.: Growing radial basis neural networks: merging supervised and unsupervised learning with network growth techniques. IEEE Transactions on Neural Networks 8(6), 1492–1506 (1997)

Krauth, W., Mézard, M.: Learning algorithms with optimal stability in neural networks. Journal of Physics A 20, 745–752 (1987)

Kremer, S.: Comments on constructive learning of recurrent neural networks: limitations of recurrent cascade correlation and a simple solution. IEEE Transactions on Neural Networks 7(4), 1047–1049 (1996)

Kubat, M.: Designing neural network architectures for pattern recognition. The Knowledge Engineering Review 15(2), 151–170 (2000)

Kwok, T.-Y., Yeung, D.-Y.: Constructive algorithms for structure learning in feedforward neural networks for regression problems. IEEE Transactions on Neural Networks 8(3), 630–645 (1997a)

Kwok, T.-Y., Yeung, D.-Y.: Objective functions for training new hidden units in constructive neural networks. IEEE Transactions on Neural Networks 8(5), 1131–1148 (1997b)

Lahnajärvi, J.J.T., Lehtokangas, M.I., Saarinen, J.P.P.: Fast constructive methods for regression problems. In: Proceedings of the 18th IASTED International Conference on Modelling, Identification and Control (MIC 1999), Innsbruck, Austria, pp. 442–445 (1999a)

Lahnajärvi, J.J.T., Lehtokangas, M.I., Saarinen, J.P.P.: Comparison of constructive neural networks for structure learning. In: Proceedings of the 18th IASTED International Conference on Modelling, Identification and Control (MIC 1999), Innsbruck, Austria, pp. 446–449 (1999b)

Lahnajärvi, J.J.T., Lehtokangas, M.I., Saarinen, J.P.P.: Fixed cascade error – a novel constructive neural network for structure learning. In: Proceedings of the Artificial Neural Networks in Engineering Conference (ANNIE 1999), St. Louis, USA, pp. 25–30 (1999c)

Lahnajärvi, J.J.T., Lehtokangas, M.I., Saarinen, J.P.P.: Evaluation of constructive neural networks with cascaded architectures. Neurocomputing 48, 573–607 (2002)

Lahnajärvi, J.J.T., Lehtokangas, M.I., Saarinen, J.P.P.: Estimating movements of a robotic manipulator by hybrid constructive neural networks. Neurocomputing 56, 345–363 (2004)

Le Cun, T., Denker, J.S., Solla, S.A.: Optimal brain damage. In: Touretzky, D.S. (ed.) Advances in Neural Information Processing Systems, vol. 2, pp. 598–605. Morgan Kauffman, San Mateo (1990)

Lehtokangas, M.: Modelling with constructive backpropagation. Neural Networks 12, 707–716 (1999)

Ma, L., Khorasani, K.: A new strategy for adaptively constructing multilayer feedforward neural networks. Neurocomputing 51, 361–385 (2003)

Ma, L., Khorasani, K.: New training strategies for constructive neural networks with application to regression problems. Neural Networks 17, 589–609 (2004)

Marchand, M., Golea, M., Ruján, P.: A convergence theorem for sequential learning in two-layer perceptrons. Europhysics Letters 11(6), 487–492 (1990)

Marchand, M., Golea, M.: On learning simple neural concepts: from halfspace intersections to neural decision lists. Network: Computation in Neural Systems 4(1), 67–85 (1993)

Mareschal, D., Schultz, T.R.: Development of children´s seriation: a connectionist approach. Connection Science 11, 149–186 (1999)

Mascioli, F.M.F., Martinelli, G.: A constructive algorithm for binary neural networks: the oil-spot algorithm. IEEE Transaction on Neural Networks 6, 794–797 (1995)

Mayoraz, E., Aviolat, F.: Constructive training methods for feedforward neural networks with binary weights. International Journal of Neural Networks 7, 149–166 (1996)

Mézard, M., Nadal, J.: Learning feedforward networks: the tiling algorithm. J. Phys. A: Math. Gen. 22, 2191–2203 (1989)

Muselli, M.: Sequential constructive techniques. In: Leondes, C. (ed.) Neural Network Systems Techniques and Applications, vol. 2, pp. 81–144. Academic, San Diego (1998)

Muselli, M.: Switching neural networks: a new connectionist model for classification. In: Apolloni, B., Marinaro, M., Nicosia, G., Tagliaferri, R. (eds.) WIRN 2005 and NAIS 2005. LNCS, vol. 3931, pp. 23–30. Springer, Heidelberg (2006)

Nabhan, T.M., Zomaya, A.Y.: Toward generating neural network structures for function approximation. Neural Networks 7(1), 89–99 (1994)

Nadal, J.-P.: Study of a growth algorithm for a feedforward network. International Journal of Neural Systems 1(1), 55–59 (1989)

Nguifo, E.M., Tsopzé, N., Tindo, G.: M-CLANN: Multi-class concept lattice-based artificial neural network for supervised classification. In: Kůrková, V., Neruda, R., Koutník, J. (eds.) ICANN 2008,, Part II. LNCS, vol. 5164, pp. 812–821. Springer, Heidelberg (2008)

Nicoletti, M.C., Bertini Jr., J.R.: An empirical evaluation of constructive neural network algorithms in classification tasks. International Journal of Innovative Computing and Applications 1(1), 2–13 (2007)

Parekh, R.G., Yang, J., Honavar, V.: Constructive neural network learning algorithm for multi-category classification, TR ISU-CS-TR95-15a, Iowa State University, IA (1995)

Parekh, R.G., Yang, J., Honavar, V.: MUpstart a constructive neural network learning algorithm for multi-category pattern classification. In: Proceedings of the IEEE/INNS International Conference on Neural Networks (ICNN 1997), vol. 3, pp. 1924–1929 (1997a)

Parekh, R.G., Yang, J., Honavar, V.: Pruning strategies for the MTiling constructive learning algorithm. In: Proceedings of the IEEE/INNS International Conference on Neural Networks (ICNN 1997), 3rd edn., pp. 1960–1965 (1997b)

Parekh, R.G., Yang, J., Honavar, V.: Constructive neural-network learning algorithms for pattern classification. IEEE Transactions on Neural Networks 11(2), 436–451 (2000)

Platt, J.: A resource-allocating network for function interpolation. Neural Computation 3, 213–225 (1991)

Poulard, H.: Barycentric correction procedure: a fast method for learning threshold unit. In: Proceedings of WCNN 1995, vol. 1, pp. 710–713 (1995)

Poulard, H., Estève, D.: A convergence theorem for barycentric correction procedure, Technical Report 95180, LAAS-CNRS, Toulouse, France (1995)

Poulard, H., Labreche, S.: A new threshold unit learning algorithm, Technical Report 95504, LAAA-CNRS, Toulouse, France (1995)

Poulard, H., Hernandez, N.: Training a neuron in sequential learning. Neural Processing Letters 5, 91–95 (1997)

Prechelt, L.: PROBEN1 A set of neural-network benchmark problems and benchmarking rules, Fakultät für Informatik, Univ. Karlsruhe, Germany, Tech. Rep. 21/94 (1994)

Prechelt, L.: Investigation of the CasCor family of learning algorithms. Neural Networks 10(5), 885–896 (1997)

Ruján, P., Marchand, M.: Learning by minimizing resources in neural networks. Complex Systems 3, 229–241 (1989)

Reed, R.: Pruning algorithms – a survey. IEEE Transaction on Neural Networks 4(5), 740–747 (1993)

Rivest, F., Shultz, T.R.: Application of knowledge-based cascade-correlation to vowel recognition. In: Proceedings of The International Joint Conference on Neural Networks, pp. 53–58 (2002)

Schaffer, J.D., Whitely, D., Eshelman, L.J.: Combinations of genetic algorithms and neural networks: a survey of the state of the art. In: Proceedings of the International Workshop of Genetic Algorithms and Neural Networks, pp. 1–37 (1992)

Shultz, T.R., Rivest, F.: Knowledge-based cascade-correlation. In: Proceedings of The IEEE-INNS-ENNS International Joint Conference on Neural Networks, vol. 5, pp. 641–646 (2000)

Shultz, T.R., Rivest, F.: Knowledge-based cascade-correlation: using knowledge to speed learning. Connection Science 13, 1–30 (2001)

Shultz, T.R., Rivest, F., Egri, L., Thivierge, J.-P., Dandurand, F.: Could knowledge-based neural learning be useful in developmental robotics? The case of KBCC. International Journal of Humanoid Robotics 4(2), 245–279 (2007)

Sirois, S., Shultz, T.R.: Neural network modeling of developmental effects in discrimination shifts. Journal of Experimental Child Psychology 71, 235–274 (1998)

Smieja, F.J.: Neural network constructive algorithms: trading generalization for learning efficiency? Circuits, Systems and Signal Processing 12, 331–374 (1993)

Subirats, J.L., Jerez, J.M., Franco, L.: A new decomposition algorithm for threshold synthesis and generalization of Boolean Functions. IEEE Transactions on Circuits and Systems I 55, 3188–3196 (2008)

Subirats, J.L., Franco, L., Molina, I.A., Jerez, J.M.: Active learning using a constructive neural network algorithm. In: Kůrková, V., Neruda, R., Koutník, J. (eds.) ICANN 2008,, Part II. LNCS, vol. 5164, pp. 803–811. Springer, Heidelberg (2008)

Tajine, M., Elizondo, D.: Enhancing the perceptron neural network by using functional composition. Technical Report 96-07, Computer Science Department, Université Louis Pasteur, Strasbourg, France (1996)

Tajine, M., Elizondo, D., Fiesler, E., Korczak, J.: Adapting the 2-class recursive deterministic perceptron neural network to m-classes. In: Proceedings of The IEEE International Conference on Neural Networks (ICNN 1997), Los Alamitos, vol. 3, pp. 1542–1546 (1997)

Thivierge, J.-P., Dandurand, F., Shultz, T.R.: Transferring domain rules in a constructive network: introducing RBCC. In: Proceedings of The IEEE International Joint Conference on Neural Networks, vol. 2, pp. 1403–1408 (2004)

Thivierge, J.-P., Shultz, T.R.: Finding relevant knowledge: KBCC applied to DNA splice-junction determination. In: Proceedings of The IEEE International Joint Conference on Neural Networks, pp. 1401–1405 (2002)

Thivierge, J.-P., Rivest, F., Shultz, T.R.: A dual-phase technique for pruning constructive networks. In: Proceedings of The IEEE International Joint Conference on Neural Networks, vol. 1, pp. 559–564 (2003)

Towell, G.G., Shavlik, J.W., Noordewier, M.O.: Refinement of approximate domain theories by knowledge-based neural networks. In: Proceedings of the Eight National Conference on Artificial Intelligence, Boston, MA, pp. 861–866 (1990)

Tsopzé, N., Nguifo, E.M., Tindo, G.: Concept-lattice-based artificial neural network. In: Diatta, J., Eklund, P., Liquiére, M. (eds.) Proceedings of the Fifth International Conference on Concept Lattices and Applications (CLA 2007), Monpellier, France, pp. 157–168 (2007)

Towell, G.G.: Symbolic knowledge and neural networks: insertion, refinement and extraction. Doctoral dissertation, Madison, WI. University of Wisconsin, USA (1991)

Wang, D.: Fast constructive-covering algorithm for neural networks and its implement in classification. Applied Soft Computing 8, 166–173 (2008)

Yang, J., Parekh, R.G., Honavar, V.: MTiling – a constructive network learning algorithm for multi-category pattern classification. In: Proceedings of the World Congress on Neural Networks, pp. 182–187 (1996)

Yang, S., Wang, M., Jiao, L.: Incremental constructive ridgelet neural network. Neurocomputing 72, 367–377 (2008)

Yao, X.: Evolving neural networks. Proceedings of the IEEE 87(9), 1423–1447 (1999)

Young, S., Downs, T.: CARVE – a constructive algorithm for real-valued examples. IEEE Transactions on Neural Network 9(6), 1180–1190 (1998)

Wendmuth, A.: Learning the unlearnable. Journal of Physics A 28, 5423–5436 (1995)