2006 International Joint Conference on Neural Networks
Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada
July 16-21, 2006

The Generalisation of the Recursive Deterministic Perceptron

David Elizondo, Ralph Birkenhead, and Eric Taillard

Abstract— **The Recursive Deterministic Perceptron (RDP) feed-forward multilayer neural network is a generalisation of the single layer perceptron topology. This model is capable of solving any two-class classification problem as opposed to the single layer perceptron which can only solve classification problems dealing with linearly separable sets (two classes X and Y of $I\!R^d$ are said to be linearly separable if there exists a hyperplane such that the elements of X and Y lie on the two opposite sides of $I\!R^d$ delimited by this hyperplane). For all classification problems, the construction of an RDP is done automatically and convergence is always guaranteed. Three methods for constructing RDP neural networks exist: Batch, Incremental, and Modular. The Batch method has been extensively tested. However, no testing has been done before on the Incremental and Modular methods. Contrary to the Batch method, the complexity of these two methods is not NP-Complete. A study on the three methods is presented. This study will allow the highlighting of the main advantages and disadvantages of each of these methods by comparing the results obtained while building RDP neural networks with the three methods in terms of the level of generalisation. The networks were trained and tested using the following standard benchmark classification datasets: IRIS and SOYBEAN.**

I. Introduction

One of the biggest limitations of the single layer perceptron topology (SLPT), introduced by Rosenblatt [14], is its inability to handle classification problems dealing with non-linearly separable (NLS) sets. The Recursive Deterministic Perceptron feed-forward neural network [8, 15] is a multilayer generalisation of this topology, which provides a solution to any two-class classification problem (even if the two classes are NLS). The RDP neural network is guaranteed to converge, does not need any parameters from the user, does not suffer from catastrophic interference, and provides transparent extraction of knowledge as a set of rules [11]. These rules can be generated, using a computational geometry approach, as a finite union of open polyhedral sets.

The approach taken by the RDP is to augment the affine dimension of the input vectors, by adding to these vectors the outputs of a sequence of intermediate neurons as new components. Each intermediate neuron (IN) corresponds to an SLPT. This allows for additional degrees of freedom for transforming the original NLS problem into a linearly separable (LS) one (two subsets X and Y of $I\!R^d$ are said to be linearly separable if there exists a hyperplane such that the elements of X and Y lie on the two opposite sides of $I\!R^d$ delimited by this hyperplane). These INs are added progressively, one at each time step. The algorithm stops when the two classes become LS.

David Elizondo and Ralph Birkenhead are with the Centre for Computational Intelligence, School of Computing, De Montfort University, The Gateway, Leicester, LE1 9BH, UK, (email: rab@dmu.ac.uk, elizondo@dmu.ac.uk).

Eric Taillard is with the EIVD, University of Applied Sciences of Western Switzerland, Route de Cheseaux 1, Case postale, CH-1401 Yverdon, Switzerland, (email: Eric.Taillard@heig-vd.ch).

Three methods exist for constructing an RDP neural network. These methods are Batch, Incremental, and Modular learning, and were introduced by one of the authors in [16]. They produce a multilayer topology which, contrary to the SLPT, is capable of solving any classification problem even if the classes considered are NLS. The Batch method, limited by its NP-Complete strategy for creating an RDP network and producing small topologies, has been extensively tested in [8]. However, no testing has been done on the Incremental and Modular methods which offer polynomial time complexity with slightly larger topologies. In this paper, for the first time, a comparison study on the performance of the three methods is presented. This study highlights the main advantages and disadvantages of each of the methods in terms of the level of generalisation obtained by using the three methods to build RDP neural networks for two machine learning benchmark classification problems. It is expected that the results obtained in this research will show the potential for using the Incremental and Modular methods for building RDP neural networks and thus help to popularise the use of RDP neural networks for solving classification problems.

To illustrate the principle for building an RDP neural network the NLS 2-input Exclusive-OR (XOR) problem can be used. For this illustration, the Batch learning method will be used. This problem consists of classifying the two classes $X = \{(0,0),(1,1)\}$ and $Y = \{(0,1),(1,0)\}$, which are NLS. To perform the NLS to LS transformation, a subset of patterns which is LS from the rest of the patterns is selected. For example, the subset $\{(0,0)\} \subset X \cup Y$ can be selected, since $\{(0,0)\}$ and $\{(0,1),(1,0),(1,1)\}$ are LS (by the hyperplane $P_t = \{(x_1,x_2) \in I\!R^2 \mid 2*x_1+2*x_2-1 = 0\}$). Therefore:

$$2*0+2*0-1<0$$

and

$$2*1+2*0-1>0, 2*0+2*1-1>0, 2*1+2*1-1>0.$$

Thus, the intermediate neuron IN1 corresponding to the SLPT of weight vector $\vec{w} = (2,2)$ and threshold $t = -1$ "associated" to the hyperplane P_t is created. The output of IN1 allows us to add, to the input vectors, one column by assigning the value -1 to the input pattern $\{(0,0)\}$, and the value 1 to the remaining three input patterns $\{(0,1),(1,0),(1,1)\}$. So, this SLPT produces the following sets of augmented input vectors: $X' = \{(0,0,\underline{-1}),(1,1,\underline{1})\}$ and $Y' = \{(0,1,\underline{1}),(1,0,\underline{1})\}$. Now, X and Y are LS by the hyperplane $P_2 = \{(x_1, x_2, x_3) \in I\!R^3 \mid -2*x_1 - 2*x_2 + 4*x_3 - 1 = 0)\}$. Hence:

$$-2*0+-2*0+4*-1-1 < 0, -2*1+-2*1+4*1-1 < 0,$$

and

$$-2*1+-2*0+4*1-1 > 0, -2*0+-2*1+4*1-1 > 0.$$

Next, a second intermediate neuron IN2 (output neuron) which corresponds to the SLPT with the weight vector $\vec{w} = (-2, -2, 4)$, and threshold $t = -1$, associated to the hyperplane P_2, is created. The final result is a two layer RDP neural network solving the XOR classification problem since the output value of this neural network is -1 for the vector patterns $\{(0,0), (1,1)\}$, and 1 for the remaining vector patterns $\{(0,1), (1,0)\}$. Figure 1 shows a graphical representation of the final RDP neural network which solves the XOR classification problem.

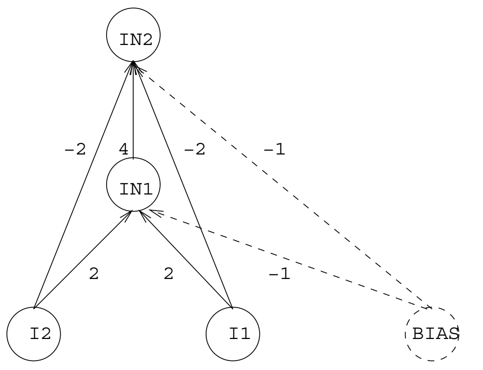

Fig. 1. RDP neural network for solving the XOR classification problem (the intermediate layer contains only one component, IN1, and IN2 corresponds to the output neuron of the RDP

Contrary to other learning methods such as backpropagation, the construction of an RDP neural network, with a 100% correct decision boundary on all training data sets, is always guaranteed. The minimum that can be expect from any learning method is to be able to correctly map all of its training data set. Actually, for any given classification problem, there are an infinite number of RDP neural networks which solve it (all with a 100% correct decision boundary on all training data sets) exist. The choice of a particular RDP allows control of the level of generalisation.

The RDP is a linear method that combines, in a cascade fashion, a series of linear single layer perceptrons to build a neural network. Other linear learning methods on LS and NLS data include Support Vector Machines (SVM) [4, 1, 2, 3]. They are trained by finding a hyperplane that linearly separates the data set by solving a constrained quadratic programming optimisation problem. In the case of NLS data, the data is mapped into some other Euclidean space. Thus, SVM is still doing a linear separation but in a different space. A kernel function must be found and used on a SVM. In its simplest form a kernel function calculates the dot product of two training vectors. This helps on the evaluation of the correct classification of each training vector.

Some preliminary results on the level of generalization between the Batch RDP learning method and other growing methods including Cascade Correlation, RuleNeg, and Rulex, are presented in [15]. The levels of generalization obtained-with the Batch RDP were comparable with those obtained with the other growing methods.

This paper is divided into seven sections. In the second section some of the notions used throughout this paper are introduced. Section three introduces the three methods for building RDP neural networks. These methods include, Batch, Incremental, and Modular. In section four, the procedure used to compare the three learning methods is presented. Two machine learning benchmarks (Iris and Soybean) were used and datasets were generated using cross validation. The three learning methods are compared in terms of their level of generalisation. Section five presents some results and discussion. A summary and some conclusions are presented in section six. In the last section, some future research ideas are proposed.

II. Background

In this section, some of the standard notions used throughout this paper are introduced, together with some definitions and properties.

A. Preliminaries

The following standard notions are used: Let $E, F \subset I\!R^d$,

- $Card(E)$ stands for the cardinality of a set E. $E \setminus F$ is the set of elements which belongs to E and does not belong to F.
- $E \oplus F$ is the set of elements of the form $\vec{e} + \vec{f}$ with $\vec{e} \in E$ and $\vec{f} \in F$.
- $E \ominus F$ stands for $E \oplus -(F)$, i.e. the set of elements of the form $\vec{e} - \vec{f}$ with $\vec{e} \in E$ and $\vec{f} \in F$. If $E = \{(1,2), (-1,2.5)\}$ and $F = \{(2.2,3), (3.1,1)\}$, then $E \ominus F$ corresponds to $\{(-1.2,-1), (-2.1,1), (-3.2,-0.5), (-4.1,1.5)\}$. $Im(E,G) = \{(x_1, ..., x_d, x_{d+1}) \in G \mid (x_1, ..., x_d) \in E$.

Let $\vec{p}_1, \vec{p}_2$ be the standard position vectors representing two points P_1 and P_2 in $I\!R^d$,

- The set $\{t\vec{p}_1 + (1-t)\vec{p}_2 \mid 0 \leq t \leq 1\}$ is called the segment between $\vec{p}_1, \vec{p}_2$ and is denoted by $[\vec{p}_1, \vec{p}_2]$.
- The dot product of two vectors $\vec{u} = (u_1, ..., u_d), \vec{v} = (v_1, ..., v_d)$ is defined as $\vec{u}^T\vec{v} = u_1v_1 + ... + u_dv_d$. $Adj(\vec{u}, r) = (u_1, ..., u_d, r)$ and by extension $Adj(S, r) = \{Adj(\vec{x}, r) \mid \vec{x} \in S\}$.
- $\mathcal{P}(\vec{w}, t)$ stands for the hyperplane $\{\vec{x} \in I\!R^d \mid \vec{w}^T\vec{x} + t = 0\}$ of $I\!R^d$ of the normal $\vec{w}$, and the threshold t. $I\!P$ will stand for the set of all hyperplanes of $I\!R^d$.

The fact that two sub-sets X and Y of $I\!R^d$ are linearly separable is denoted by $X \mid\mid Y$ or $X \mid\mid Y\ (P)$. Thus if $X \mid\mid Y\ (\mathcal{P}(\vec{w}, t))$, then ($\forall \vec{x} \in X,\ \vec{w}^T\vec{x} + t > 0$ and $\forall \vec{y} \in Y,\ \vec{w}^T\vec{y} + t < 0$) or ($\forall \vec{x} \in X,\ \vec{w}^T\vec{x} + t < 0$ and $\forall \vec{y} \in Y,\ \vec{w}^T\vec{y} + t > 0$).

Let $X, Y \subset I\!R^d$, $I\!P(X,Y) = \{P \in I\!P \mid X \mid\mid Y\ (P)\}$.

TABLE I

BATCH LEARNING ALGORITHM

```
Batch($X, Y$)
Batch(X, Y) -data: two disjoint finite subsets $X, Y$ of $I\!R^d$,
-result: A RDP $P = [(\vec{w}_0, t_0), ..., (\vec{w}_{n-1}, t_{n-1})]$
which transforms $X$ and $Y$ into two LS classes. (An RDP linearly
separating $X, Y$, by adding one IN to the RDP constructed by this
algorithm is obtained.
This IN corresponds to the output neuron.)
$i := 0; X_0 := X; Y_0 := Y; X'_0 := X; Y'_0 := Y; S_0 = X \cup Y;$
WHILE $not(X_i \parallel Y_i)$ do
    BEGIN
        SELECT: Select a non empty subset $Z_i$ from $X'_i$ or from $Y'_i$
        (if it exits)
            such that $Z_i \parallel (S_i \setminus Z_i)(\mathcal{P}(\vec{w}_i, t_i))$ ;
            (i.e., $(Z_i \subset X'_i$ or $Z_i \subset Y'_i)$ and $Z_i \parallel (S_i \setminus Z_i)(\mathcal{P}(\vec{w}_i, t_i))$
        CASE: $Z_i \subset X'_i$
            $S_{i+1} := Adj(Z_i, -1) \cup Adj(S_i \setminus Z_i, 1);$
            $X'_{i+1} := Im(X'_i, S_{i+1}) \setminus Im(Z_i, S_{i+1});$
            $Y'_{i+1} := Im(Y'_i, S_{i+1});$
            $X_{i+1} := Im(X_i, S_{i+1});$
            $Y_{i+1} := S_{i+1}) \setminus X_{i+1};$
            $i := i + 1;$
        CASE: $Z_i \subset Y'_i$
            $S_{i+1} := Adj(Z_i, -1) \cup Adj(S_i \setminus Z_i, -1);$
            $Y'_{i+1} := Im(Y'_i, S_{i+1}) \setminus Im(Z_i, S_{i+1});$
            $X'_{i+1} := Im(X'_i, S_{i+1});$
            $X_{i+1} := Im(X_i, S_{i+1});$
            $Y_{i+1} := S_{i+1}) \setminus X_{i+1};$
            $i := i + 1;$
    END;
```

Let $P \in I\!P(X, Y)$, $\mathcal{C}_Y(X, P)$ is the half space delimited by P and containing X *(i.e.* $\mathcal{C}_Y(X, P) = \{\vec{v} \in I\!R^d \mid \vec{u}^T\vec{v} + t > 0\}$ *if for some* $\vec{x} \in X$, $\vec{u}^T\vec{x} + t > 0$*).*

III. METHODS FOR BUILDING RDP NEURAL NETWORKS

The three methods for building RDP neural networks are Batch, Incremental and Modular. The Batch method follows a selection strategy based on searching homogeneous LS subsets (i.e., whose elements belong to the same class) from a set of NLS points. With the Incremental approach, all the previous knowledge learned before when adding new knowledge to the RDP does not have to be retrained. The modular approach allows to combine several RDP models, within a single RDP, without having to do any further training.

A. The Batch Method

1) Description of the Method: The Batch method (table I) uses an LS subset selection strategy which consists on selecting a set of LS points which belong to the same class and have maximum cardinality. This algorithm stops after at most $n - 1$ steps, where n corresponds to the number of learning patterns

2) Example: In this subsection, the use of the Batch learning method, presented in Table I, is illustrated by applying it to an NLS 2D classification toy example. The NLS 2D classification problem consists of two classes 1 ($+$), 2 ($\diamond$) (see Fig. 2).

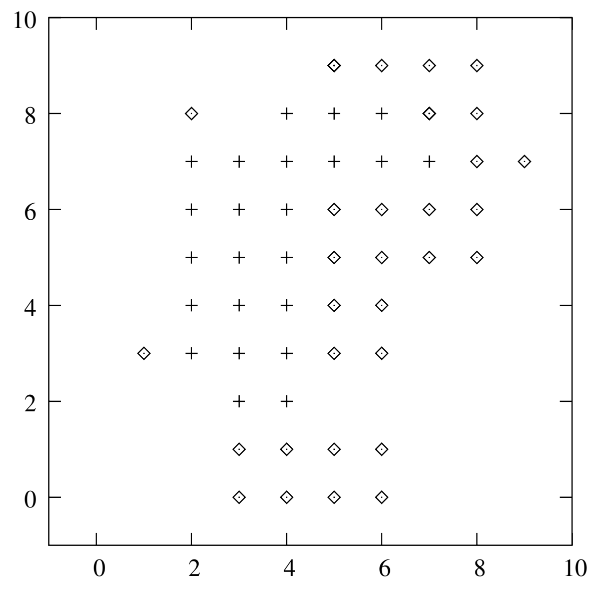

Fig. 2. 2D plot of the two class classification problem used to illustrate the Batch learning algorithm ($+$ = class 1, $\diamond$ = class 2)

$$
\begin{aligned}
A = \; & \{(3,2),(4,2),(2,3),(3,3),(4,3),(2,4),(3,4),(4,4),\\
& (2,5),(3,5),(4,5),(2.6),(3,6),(4,6),(2,7),(3,7),\\
& (4,7),(5,7),(6,7),(7,7),(4,8),(5,8),(6.8)\},
\end{aligned}
$$

$$
\begin{aligned}
B = \; & \{(3,0),(4,0),(5,0),(6,0),(3,1),(4.1),(5,1),(6,1),\\
& (5,2),(6,2),(1,3),(5,3),(6,3),(5,4),(6,4),(5,5),\\
& (6,5),(7,5),(8,5),(5,6),(6,6),(7,6),(8,6),(8,7),\\
& (9,7),(2,8),(7,8),(8,8),(5,9),(6,9),(7,9),(8,9)\}.
\end{aligned}
$$

After applying the Batch learning method in Table I to this problem, an RDP containing seven INS which linearly separates A and B is obtained. Table II shows the LS subsets selected for each IN (at each step, the selected LS subset was of maximal cardinality). Table III shows the weight vectors and thresholds found for each IN. A projection of the selected LS subsets used in the different steps for building the RDP is shown in Fig. 3. Fig. 4 shows the RDP topology found by the learning algorithm described in Table I for solving this problem.

B. The Incremental Method

1) Description of Method: To do the Incremental or progressive learning, the network is trained with a subset of the training data set. Once the RDP network is trained, training can be restarted with a new point. If this new point is not well classified by the existing RDP, then the new knowledge can be interpolated without disturbing the previously acquired knowledge. This can be done with the addition of a new IN.

TABLE II

LS SUBSETS SELECTED BY THE BATCH LEARNING ALGORITHM DESCRIBED IN TABLE 2 APPLIED TO THE TWO-DIMENSIONAL TWO-CLASS CLASSIFICATION PROBLEM.

Step #	Selected subset	Class
1	$\{(4,0),(5,0),(6,0),(4,1),(5,1),(6,1),(5,2),$ $(6,2),(5,3),(6,3),(6,4),(7,5),(8,5),(7,6),$ $(8,6),(8,7),(9,7),(8,8)\}$	A
2	$\{(2,8,0),(5,9,0),(6,9,0),(7,9,0),(8,9,0)\}$	A
3	$\{(3,0,0,0),(3,1,0,0),(1,3,0,0)\}$	A
4	$\{(3,2,0,0,0),(4,2,0,0,0),(2,3,0,0,0),$ $(3,3,0,0,0),(4,3,0,0,0),(2,4,0,0,0),$ $(3,4,0,0,0),(4,4,0,0,0),(2,5,0,0,0),$ $(3,5,0,0,0),(4,5,0,0,0),(2,6,0,0,0),$ $(3,6,0,0,0),(2,7,0,0,0),(3,7,0,0,0)\}$	B
5	$\{(4,6,0,0,0,1),(4,7.0,0,0,1),(4,8,0,0,0,1),$ $(5,8,0,0,0,1)\}$	B
6	$\{(5,4,0,0,0,1,1),(5,5,0,0,0,1,1),$ $(6,5,0,0,0,1,1),(5,6,0,0,0,1,1),$ $(6,6,0,0,0,1,1)\}$	A

TABLE III

RDP WEIGHT VECTORS AND THRESHOLD VALUES OBTAINED BY THE BATCH LEARNING ALGORITHM (TABLE 2) FOR EACH OF THE INS

i (Step)	$\vec{w}_i$ (Weight vector)	t_i, (Threshold)
1	(19.1, -12.4)	-4.0
2	(-3.5.21.3, 0.0)	-15.0
3	(-6.7. -6.9, - 1 .O. 0.0)	4.0
4	(6.2. 1.8, 1.0, 1.0, 3.0)	-1.0
5	(21.6, -6.6, 1.0, 7.0, 8.0, -6.0)	2.0
6	(0.4, -1.5, -3.0.0.0, - 1 .o, 2.0, 1.0)	-1.0
7	(5.8, 1.3,4.0,7.0.5.0, 1.0. 1.0,3.0)	-5.0

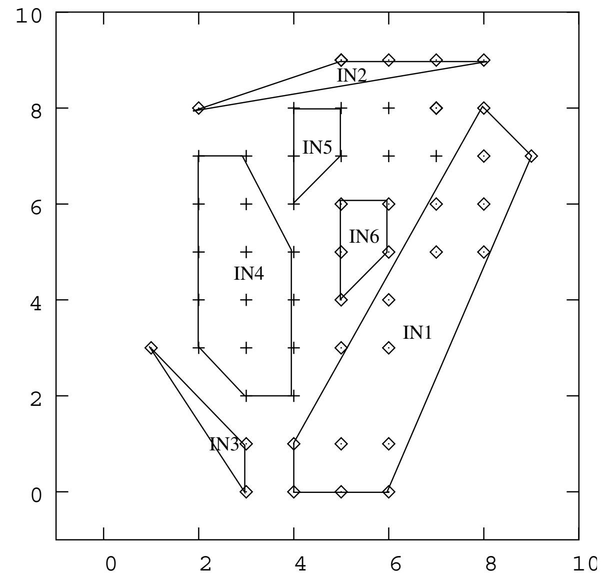

Fig. 3. Projection in a plane of the LS subsets selected by the Batch learning algorithm described in Table I to create the INs necessary to construct the RDP

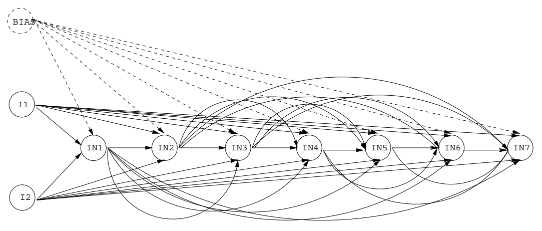

Fig. 4. The topology of the RDP for solving the 2D classification problem obtained by applying the Batch learning algorithm described in Table I (IN7 corresponds to the output neuron)

2) Example: The Incremental learning method is illustrated by applying it to the same problem used to illustrate the Batch learning method (see Fig. 2). First a data subset is selected

$$\begin{aligned} A_1 = \ & \{(3,2),(2,3),(3,3),(2,4),(3,4),(4,4),(2,5), \\ & (3,5),(4,5),(2,6),(3,6),(4,6),(2,7),(3,7), \\ & (4.7),(5,7),(6,7),(4,8),(5,8),(6,8)\} \end{aligned}$$

from sets A and

$$\begin{aligned} B_1 = \ & \{(3,0),(4,0),(5,0),(6,0).(3,1),(4.1),(5,1), \\ & (6,1),(5,2),(6,2),(5.3),(6,3),(5.4),(6,4), \\ & (5,5),(6,5),(7,5),(8,5),(6,6),(7,6),(8,6), \\ & (8,7),(9,7),(7,8),(8,8),(8,9)\} \end{aligned}$$

and B which are linearly separable by the hyperplane $P((-42,24),-85)$. Therefore, $A_1\|_p B_1$ where $P = [((-42,24),-85)]$. $A = A_1 \bigcup (p_1,p_2,p_3)$ where p_1, p_2 and p_3 correspond respectively to $(4,2),(4,3)$, and $(7,7)$ and $B = B_1 \bigcup p_4,p_5,p_6,p_7,p_8,p_9$ where p_4,p_5,p_6,p_7,p_8, and p_9 correspond respectively to $(1,3),(5,6),(2,8),(5.9),(6,9)$, and $(7,9)$ (Fig. 5). Next, the remaining points $p1,...,p_9$ are "interpolated" and the RDP containing ten INS shown in Table IV which linearly separates A and B is obtained.

C. The Modular Method

1) Description of Method: The idea behind modular neural networks is to divide the original problem into smaller sub-problems, each of which is to be solved by a sub-network. These sub-networks are then assembled together into a global network which solves the original problem. The following theorem shows how to join the sub-networks.

Theorem 1: Let A_1, A_2, B_1, B_2 be finite subsets of $I\!R^d$. Let P_1,P_2,P_3,P_4 be RDPs on $I\!R^d$ such that $A_1\|^{>}_{P_1}B_1$, $A_1\|^{>}_{P_2}B_2$, $A_2\|^{>}_{P_3}B_1$, $A_2\|^{>}_{P_4}B_2$ Let

$$Q = [((1,-1,1,-1),3),((3,5,2,5,5),-3)].$$

Thus, if $P = Q \circ [P_1,P_2,P_3,P_4]$, then $(A_1 \bigcup A_2)\|^{>}_{P}(B_1 \bigcup B_2)$

The complete proof of the this property is given in [16]. Assuming that $\forall(\vec{x}) \in (A_1 \bigcup A_2 \bigcup B_1 \bigcup B2)\mathcal{F}(P_1)((\vec{x}) \neq 0, \mathcal{F}(P_2)((\vec{x}) \neq 0, \mathcal{F}(P_3)((\vec{x}) \neq 0, \mathcal{F}(P_4)((\vec{x}) \neq 0$. Each

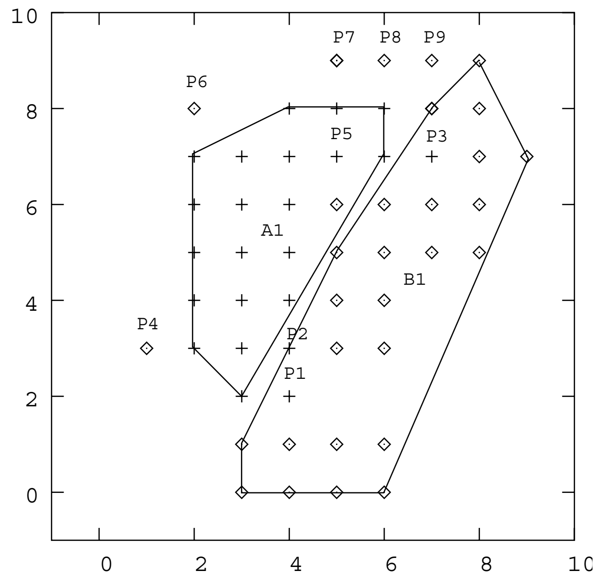

Fig. 5. Cluster subsets A1 and B1 used to build the first RDP and individual points added, one at the time, $p_1, ..., p_9$ used to illustrate the Incremental learning method.

TABLE IV

RDP WEIGHT VECTORS AND THRESHOLD VALUES OBTAINED BY THE INCREMENTAL LEARNING ALGORITHM FOR EACH OF THE INS

i Step)	$\vec{w}_i$ (Weight vector)	t_i (Threshold)
1	(-41.9994,24)	119.997
2	(-83.994,48, -35.0012)	226.972
3	(-167.988.96, -70.0024, -21.9913)	455.899
4	(335.97, -192, 140.005,43.98 26,23.947)	32.095
5	(671.94, -384,280.01,87.96 52, 47.894, -919.96)	-551.61
6	(1343.88, -768,560.02, 175.93, 95.788, -1839.92, -304.161)	480.416
7	(2687.76, -1536, 1120.04, 351.86, 191.576, -3679.84, -608.322, - 1887.79)	-94.549
8	(5375.52, -3072,2240.08,103.12, 383.152, -7359.68, -1216.54, -3775.58, -832.398)	1154.148
9	(10751, -6144.4480.16, 1407.44, 766.304, -14719.4, -2433.28, -7551.16, -1664.8, -2175.64)	1927.436
10	(-21502, 12288, -8960.32, -2814.88, -1532.61, 29438.8.4866.56, 15102.3, 3329.6, 4351.28, 1796.44)	-2062.32

TABLE V

OUTPUTS PRODUCED BY THE RDPS P_1, P_2, P_3, AND P_4

	A_1	A_2	B_1	B_2
P_1	1	± 1	-1	± 1
P_2	± 1	1	± 1	-1
P_3	1	± 1	± 1	-1
P_4	± 1	1	-1	± 1

RDP P_1, P_2, P_3, and P_4 will produce the outputs described in table V for each of the data subsets:

Let:

$$S_1 = \{(1,-1,1,-1),(1,-1,1,1),(1,1,1,-1),\\ (1,1,1,1),(-1,1,-1,1),(-1,1,1,1),\\ (1,1,-1,1)\}$$

and

$$S_2 = \{(-1,-1,-1,-1),(-1,-1,1,-1),(-1,1,-1,-1),\\ (-1,1,1,-1),(-1,-1,-1,1),(1,-1,-1,-1),\\ (1,-1,-1,1)\}$$

Then $S_1 \bigcap S_2 = 0$,

$$\{(\mathcal{F}(P_1)(\vec{a}),\mathcal{F}(P_2)(\vec{a}),\mathcal{F}(P_3)(\vec{a}),\mathcal{F}(P_4)(\vec{a})|\vec{a} \in A_1 \cup A_2\}\\ \subseteq S_1$$

and

$$\{(\mathcal{F}(P_1)(\vec{b}),\mathcal{F}(P_2)(\vec{b}),\mathcal{F}(P_3)(\vec{b}),\mathcal{F}(P_4)(\vec{b})|\vec{b} \in B_1 \cup B_2\}\\ \subseteq S_2$$

Let $Q = [((1,-1,1,-1),3),((3,5,2,5,5),-3)]$, then $S_1 \|^{>}_{Q} S_2$. Therefore, if $P = Q \circ [P_1,P_2,P_3,P_4]$, then $(A_1 \cup A_2) \ \|^{>}_{P} \ (B_1 \cup B_2)$

2) Example: To illustrate the modular construction of an RDP, the same classification problem used to illustrate the Batch and Incremental methods is used. The two original classes are decomposed into the following subclasses:

$$A_1 = \{(3,2),(4,2),(2,3),(3,3),(4,3),(2,4),(3,4),(4,4),\\ (2,5),(3,5),(4,5),(2,6),(3,6),(4,6),(2,7),(3,7)\},$$

$$A_2 = \{(4,7),(5,7),(6,7),(7,7),(4,S),(5,8),(6,8)\},$$

$$B_1 = \{(3,0),(4,0),(5,0),(6,0),(3,1),(4,1),(5,1),(6,1),\\ (5,2),(6,2),(1,3),(5,3),(6,3),(5,4),(6,4),(5,5),\\ (6,5),(7,5),(8,5),(5,6),(6,6),(7,6),(8,6),(8,7),\\ (9,7)\},$$

$$B_2 = \{(2,8),(7,8),(8,8),(5,9),(6,9),(7,9),(8,9)\},$$

as shown in Fig. 6 ($A = A_1 \bigcup A_2$ and $B = B_1 \bigcup B_2$). Next a RDP to linearly separate each subclass from the other subclasses, as shown below, is created:

$$A_1 \|^{>}_{P_1} B_1, \ A_1 \|^{>}_{P_2} B_2, \quad A_2 \|^{>}_{P_3} B_1, \ A_2 \|^{>}_{P_4} B_2$$

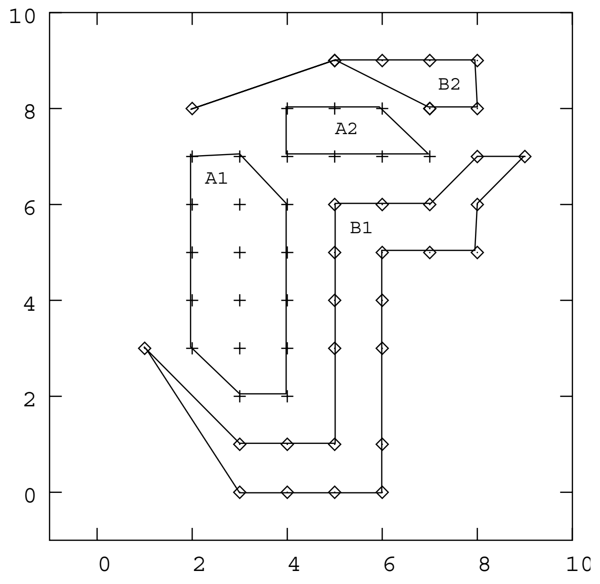

Fig. 6. Clusters of the data subsets A_1, A_2, B_1, and B_2 used to illustrate the modular learning algorithm for constructing RDP neural networks.

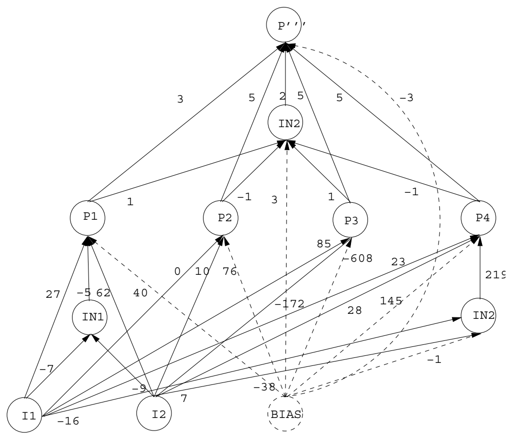

Fig. 7. RDP topology for solving the two-dimensional classification problem obtained by applying the modular learning algorithm.

where $P_1 = [((-7,-9),-38),\ ((27,-5,62),40)]$, $P_2 = [((0,10),76)]$, $P_3 = [((85,\ -172),\ -608)]$, and $P_4 = [((-16,7),-1),\ ((23,28,219),145)]$. Once the four RDP modules are created, they can be unified into a single RDP network by using the RDP computed above to linearly separate S_1 and S_2. The final topology of the FLDP that combines the four modules for linearly separating classes A and B is shown in Fig. 7. The RDPs P_1, P_2, P_3, P_4 can be constructed in a parallel fashion since their constructions are independent from each other.

IV. Comparison Procedure

Two machine learning benchmark data sets were used in the comparison study. These benchmarks included Iris, and Soybean [6]. These data sets are widely used to compare different classification algorithms. The IRIS dataset classifies

TABLE VI
Inputs and outputs used on the IRIS classification problem.

Attributes (In cm)	Output	Output Classes
Sepal Length Sepal Width Petal Length Petal Width	Iris plant type	Iris Setosa Iris Versicolour Iris Virginica

TABLE VII
Inputs and outputs used in the SOYBEAN classification problem.

Attributes		Output	Output classes
Date	leaf-shread	Disease type	brown-spot
plant-stand	stem		alternarialeaf-spot
precipitation	stem-cankers		frog-eye-leaf-spot
temperature	canker-lesion		
hail	fruiting-bodies		
crop-hist	external decay		
area-damaged	fruit-pods		
severity	fruit spots		
seed-tmt	seed		
germination	plant-growth		

a plant as being an Iris Setosa, Iris Versicolour or Iris Virginica. The dataset describes every iris plant using four input parameters (Table VI). The dataset contains a total of 150 samples with 50 samples for each of the three classes. All the samples of the Iris Setosa class are linearly separable from the rest of the samples (Iris Versicolour and Iris Virginica). Therefore, only the samples belonging to the Iris Versicolour and the Iris Virginica classes were used in this study. Some of the publications that used this benchmark include: [12] [13] [5] and [8].

The SOYBEAN classification problem contains data for the disease diagnosis of the Soybean crop. The dataset describes the different diseases using symptoms. The original dataset contains 19 diseases and 35 attributes. The attribute list was limited to those attributes that had non trivial values in them (Table VII). Thus there were only 20 out of the 35 attributes that were included in the tests. Only two classes were selected based on the number of samples available and used to build the network. The brown spot, alternaria leaf spot and frog eye leaf spot, each having 40 samples were selected and were used. The networks developed were trained to separate the disease class alternaria leaf spot from the other two diseases brown spot and frog eye leaf spot.

The technique of cross validation was applied to split the benchmarks into training and testing data sets. The datasets were randomly divided into 'n' equal sized testing sets that were mutually exclusive [17]. The remaining samples were used to train the networks. In this study, the classification benchmark data sets were divided into ten equally sized data sets. Sixty percent of the samples were used for training the networks and the remaining forty percent were used for testing purposes. Thus, for each of the three training algorithms, ten different neural networks were developed and tested using different combinations of test sets that were picked up from the equally divided sample sets.

To train the modular networks, each of the cross validation training data sets was further divided into four subsets. The first two subsets contained each half of the data for class one. The remaining two subsets contained each half of the data for class two.

This study was based on the comparison between the level of generalisation, with respect to previously unseen data, obtained with each of the three learning algorithms for building RDP neural networks.

V. Results and Discussion

The three methods for constructing RDP neural networks were compared based on their level of generalisation on previously unseen data. The IRIS, and SOYBEAN benchmark data sets were used for building neural networks using these methods. The technique of cross validation was applied to split the benchmarks into training and testing data sets.

Tables VIII, and IX show the level of generalisation, in terms of percentage of well classified samples, obtained using the different methods for constructing RDP neural networks. Some of the training subsets were linearly separable (marked with an asterisk). Thus, a single layer perceptron network was used to train them. No incremental or modular networks were developed using these LS subsets. Several of the originally NLS training sets became LS after splitting them into smaller subsets to use on training the modular method. This simplified the learning procedure.

The results obtained in terms of generalisation show that both the Batch and the Incremental methods offer comparable performance. The average (Δ) results on the level of generalisation obtained on both methods, using the two benchmarks, only differ by 1%. The occasional deviations from this behaviour could be attributed to the failure of the perceptron based linear separability determination algorithm for which the number of maximum weight updates allowed has to be specified before hand. The results obtained using the Iris data set are very similar for both methods with the Incremental method outperforming the Batch one for data sets 4 and 6 by 2.5% and 7.5% respectively. The results obtained with both methods using the Soybean data set vary more than the ones obtained with the Iris dataset. Here the Incremental method outperforms the Batch one on data sets 4,6 and 9, and opposite can be observed on data sets 2, 5, 7, and 10. Therefore, the incremental approach, with an $O(n \ log \ n)$ complexity, is preferable to the Batch approach with an NP-Complete complexity [8]. The Incremental method is also best suited for problems of dynamic nature where the network needs to be trained for new data without loosing the training already achieved (catastrophic interference). Both the Batch and the Incremental methods outperform the two modular approaches.

The modular approaches offer the possibility of a parallel implementation where a divide and conquer approach is used to break up a large problem into smaller problems. If a modular approach is to be used to take advantage of this, the Modular/Batch approach provides better results than the Modular/Incremental approach.

TABLE VIII

Results obtained with the three learning methods and the Iris benchmark in terms of the level of generalisation (* - implies that the two data sets used for the training of the neural network were linearly separable data sets, Δ represents the average over the data sets that were not linear separable).

	Iris			
Data Set	Batch	Incr	Mod Batch	Mod Incr
1	100	100	97.5	97.5
2	92.5*	92.5*	92.5*	92.5*
3	100	100	100	100
4	92.5	95	90	90
5	97.5	97.5	95	90
6	92.5	100	97.5	95
7	97.5*	97.5*	97.5*	97.5*
8	97.5	97.5	95	90
9	97.5	97.5	95	92.5
10	100	100	100	95
Δ	97.2	98.4	96.25	93.75

TABLE IX

Results obtained with the three learning methods and the soybean benchmark in terms of the level of generalisation (* - implies that the two data sets used for the training of the neural network were linearly separable data sets, Δ represents the average over the data sets that were not linear separable).

	Soybean			
Data Set	Batch	Incr	Mod Batch	Mod Incr
1	100*	100*	100*	100*
2	93.7	85.4	87.5	75
3	79.2*	79.2*	79.2*	79.2*
4	91.6	95.8	83	52.1
5	98	79.2	75	52.1
6	83.3	93.7	100	77.1
7	98	95.8	79	75
8	95.8*	95.8*	95.8*	95.8*
9	83.3	95.8	72.5	91.7
10	90	85.4	79	77.1
Δ	91.13	90.2	82.3	71.4

These results show that the smaller topologies, obtained with the exhaustive Batch method, do not necessarily produce the best results in terms of level of generalisation and that maybe some more degrees of liberty are needed to improve the accuracy of the network with respect to previously unseen data. The results obtained are consistent for the two benchmarks.

VI. Future Research

The Batch growing method for building RDP neural networks can produce small topologies. This is done by using the approach of maximum cardinality subset on the selection of linearly separable subsets. This approach has been proven to be NP-Complete [8]. Therefore, heuristic methods can be applied to overcome this complexity problem [7]. For instance, the problem can be simplified by pre-clustering the datasets to obtain a rough solution. This solution can then be refined using a meta heuristic technique. For example, a 'D' point classification strategy can be used for the initial clustering of the data and then a local search strategy like the Tabu Search can be employed to determine the solution to the NP-complete problem.

The implementation of the Incremental method in this study involved the addition of a new intermediate neuron every time a point on the training data set was not classified correctly by the existing model. An algorithm for adjusting the last hyperplane on the network to classify a new point, before opting to add a new intermediate neuron to the topology, was proposed by [16]. It would be of interest to see if the number of intermediate neurons on the topology generated by using the Incremental method can be reduced by this method. It will also be of interest to see how the generalisation level is affected by this topology reduction strategy.

It is mentioned in [16] that the Incremental algorithm is suited to dynamic classification problems and the Batch method is suited to static classification problems. This needs to be further explored to identify the exact nature of the problems that can be solved by the two methods.

The implementation of the Modular method uses either the Batch or the Incremental method to solve the generated subsets of the original data set. The use of a combination of the Batch and the Incremental methods to solve sub problems of a single Modular problem can be explored.

To simplify the experiments in this study, the data sets used for the Modular method were randomly split into subsets. It will be interesting to use instead a clustering technique to split the data sets into smaller subsets. The Modular method provides a network of bigger topology while at the same time simplifing the problem by breaking it into sub problems and solving them. A heuristic can be devised to identify the number of splits that optimises the compromise between this simplicity of the Modular building algorithm and the topology generated by it. The performance of this method can also be enhanced by using a parallel implementation.

The perceptron algorithm was used in this work to test for linear separability among the data subsets. This algorithm is guaranteed to converge after a finite number of steps if the two classes are linearly separable. If the classes are not linearly separable, it will not converge. Other algorithms for testing linear separability, such as the Class of Linear Separability [9] and the Simplex method, might provide different results in terms of generalisation. See [10] for a survey on methods for testing linear separability.

These results provide good basis to further develop this study and to compare the topology size (number of intermediate neurons), and convergence time obtained with the three RDP methods.

References

[1] A. Atiya. Learning with kernels: Support vector machines, regularization, optimization, and beyond. *IEEE TNN*, 16, May 2005.

[2] B. Boser, I. Guyon, and V. Vapnik. A traning algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, 1992.

[3] C. Cortes and V. Vapnik. Support-vector network. *Machine Learning*, 20:273–297, 1995.

[4] N. Cristianini and J. Shawe-Taylor. *An introduction to support vector machines*, volume I. Cambridge University Press, 2003.

[5] B. W. Dasarathy. Nosing around the neighborhood: A new system structure and classification rule for recognition in partially exposed environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2(1):67–71, 1980.

[6] C.L. Blake D.J. Newman, S. Hettich and C.J. Merz. UCI repository of machine learning databases, 1998.

[7] J. Dréo, A. Pétrowski, P. Siarry, and E. Taillard. *Metaheuristics for Hard Optimization Methods and Case Studies*. Springer, 2006.

[8] D. Elizondo. *The Recursive Determinist Perceptron (RDP) and Topology Reduction Strategies for Neural Networks*. PhD thesis, Université Louis Pasteur, Strasbourg, France, January 1997.

[9] D. Elizondo. Searching for linearly separable subsets using the class of linear separability method. In *Proceedings of the IEEE-IJCNN*, pages 955–960, April 2004.

[10] D. Elizondo. The linear separability problem: Some testing methods. *Accepted for Publication: IEEE TNN*, 2006.

[11] D.A. Elizondo and M.A. Gongora. Current trends on knowledge extraction and neural networks. In W. Duch et al., editor, *Proceedings of the IEEE-ICANN*. Springer, September 2005.

[12] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annual Eugenics*, 7(II):179–188, April 1936.

[13] G. W. Gates. The reduced nearest neighbor rule. *IEEE Transactions on Information Theory*, pages 431–433, May 1972.

[14] F. Rosenblatt. *Principles of Neurodynamics*. Spartan, Washington D.C., 1962.

[15] M. Tajine and D. Elizondo. The recursive deterministic perceptron neural network. *Neural Networks*, 11:1571–1588, 1997.

[16] M. Tajine and D. Elizondo. Growing methods for constructing recursive deterministic perceptron neural networks and knowledge extraction. *Artificial Intelligence*, 102:295–322, 1998.

[17] S. M. Weiss and C. A. Kulikowski. *Computer Systems That Learn*. Morgan Kaufmann Publishers, San Mateo, California, 1991.