Analysis of Multilayer Neural Networks with Direct and Cross-Forward Connection

Stanisław Płaczek and Bijaya Adhikari

Vistula University, Warsaw, Poland
stanislaw.placzek@wp.pl,bijaya.adhikari1991@gmail.com

Abstract. Artificial Neural Networks are of much interest for many practical reasons. As of today, they are widely implemented. Of many possible ANNs, the most widely used ANN is the back-propagation model with direct connection. In this model the input layer is fed with input data and each subsequent layers are fed with the output of preceeding layer. This model can be extended by feeding the input data to each layer. This article argues that this new model, named cross-forward connection, is optimal than the widely used Direct Conection.

1 Introduction

Artificial Neural Networks have broad implementation in Machine Learning, engineering and scientific applications. Their abilities to provide solutions to problems involving imprecisions and uncertainties with trivial implementation have enabled us to find solutions to real life problems as [1]:

1. Result approximation and data interpolation
2. Pattern recognition nad feature classification
3. Data compression
4. Trend prediciton
5. Error identification
6. Control

The problems mentioned above are solved by implementing ANN as universal approximator function with multidimensional variables. The function can be represented as:

$$Y = F(X) \tag{1}$$

where:

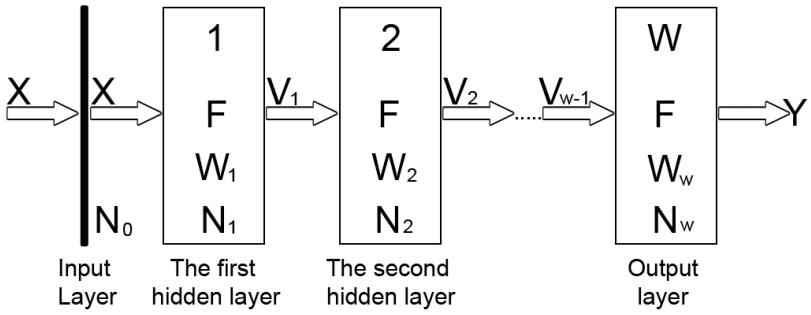
- X-input vector
- Y-output vector

Selecting a network to solve a specific problem is a tedious task. Decision regarding following thing must be made prior to attempting a solution.

- Structure of Neural Network, number of hidden layers and number of neurons in each layer. Conventionally, the size of input and output layers are defined by dimension of X and Y vectors respectively.
- Structure of individual neurons encompassing activation function, which takes requirement of learning algorithm into account.
- Data transfer methods between layers
- Optimization criteria and type of learning algorithm

Structure of Network can be defined in arbitrary way to accomplish complex tasks. The structure plays vital role in determining the functionality of ANN. This paper will compare and contrast two multilayer network structures.

- Direct Connection: This structure consists of at-least one hidden layer. Data tis fed from preceeding layer to succeeding one.



$N_0, N_1, N_2 \ldots N_W$-number of neuron in layer j=0,1...W.

X-input vector with dimension N_0

V_j- output vector of layer j=1,2,...,w-1 with dimension N_j

Y - output vector with dimension N_w

W_j- j=1...W- weight coefficient of matrix

Fig. 1. Structure of Direct Connection ANN

- Cross Forward Connection. In this structure, the input signal is passed on to every layer in the network. Therfore, a layer j=1,2,3.....W,where W is the output layer, has two inputs : vector X and Vector V_{j-1}, output of preceeding layer.

Structure of Cross Forward Connection is simpler than that of Direct Connection, in terms of neuron distribution in hidden layers. Learning time, as second parameter, is shorter for Cross Forward Connection . In later part of the paper,

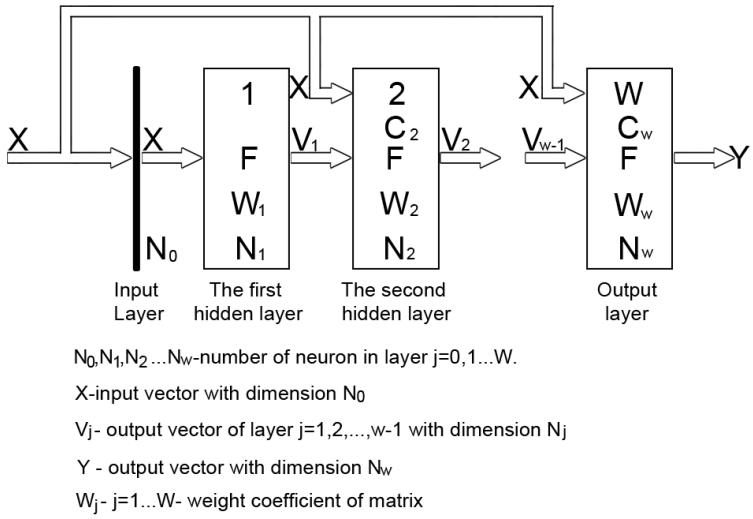


Fig. 2. Structure of Forward Connection ANN

we will analyze a particular optimization problem for ANN where total number of neurons, N, and number of layers , W, are given. Our target is to maximize the total number of subspaces which are created by neurons of every hidden layers. We will solve this complex problem with respect to the relation between dimensionality of feature space, N_0, and neurons' number in all hidden layers, N_i. This problem can be divided into two sub-problems.

- $N_i \leq N_0$ – liner optimization problem,
- $Ni > N0$ – non-linear optimization problem.

Where: i= 1,2,3,. . . . . W-1.

We can solve liner target function using liner-programming method. The non-linear task, with liner constrains, can be solved using Kuhn- Tucker conditions. As examples, we solved both sub-problems and discussed different ANN structures. In conclusion, we summarize our results giving recommendation for different ANN structures.

2 Criteria of ANN Structure Selection

The threshold function for the each neuron is defined as follows:

$$g(x) = \begin{cases} 1, & \text{if } x > 0 \\ -1, & \text{if } x \leq 0 \end{cases} \tag{2}$$

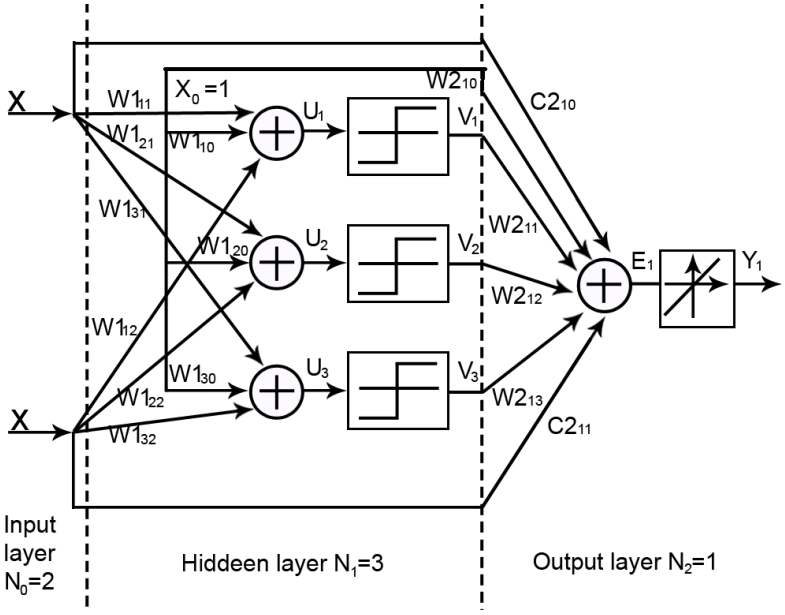


Fig. 3. Two Layer ANN with Cross Forward Connection

We say that the network in Fig. 3 has structure 2-3-1. Where:

- N_0=2; number of neurons in input layer.
- N_1=3; number of neurons in hidden layer.
- N_2=1; number of neurons in output layer.

Signal transfer from input layer to output layer in this structure can be represented in the following way.

$$U = W_1 \cdot X \tag{3}$$

$$V = F_1(U) \tag{4}$$

$$E = W_2 \cdot V + C_2 \cdot X \tag{5}$$

$$Y = F_2(E) \tag{6}$$

Where,

- $X[0 : N_0]$ -input signal
- $W_1[1{:}N_1;0{:}N_0]$ - weight coefficients matrix of hidden layer
- $U[1{:}N_1]$-analog signal of hidden layer
- $V[1{:}N_1]$-output signal of hidden layer

- $W_2[1{:}N_2;0{:}N_1]$ - weight coefficients matrix of output layer
- $E[1{:}N_2]$-analog signal of output layer
- $Y[1{:}N_2]$-output signal of output layer
- $C_2[1:N_2;0:N_0]$ -weight coefficients matrix of Cross connection

This network will be used for pattern recoginition after being trained by teacher datas.

The architecure of ANN in fig(3) could be represented using hyper-spaces. Lets imagine a hyperspace having dimension of the number of neurons in the input layer. The first hidden layer, depicted in equation (3) and (4), divides feature space, X, into subspaces.

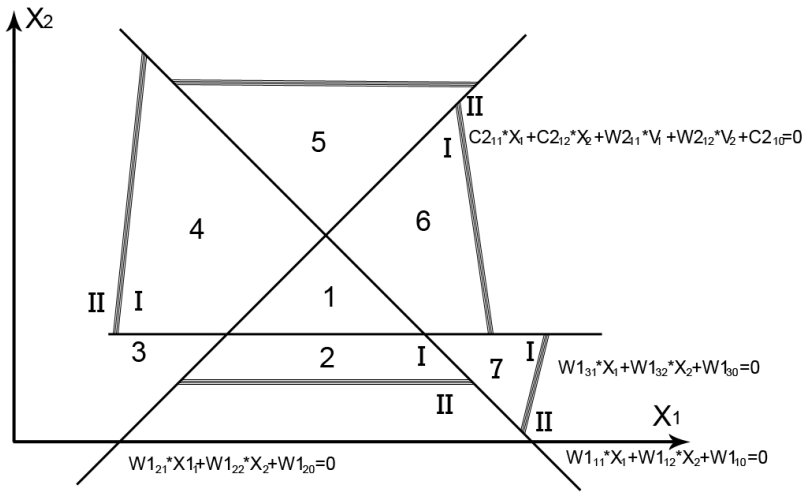


Fig. 4. Structure of division of two dimensional input space by three neurons of the first hidden layer.

Two dimensional feature space is divided into seven sub-spaces. These sub-spaces correspond to internal structure of input data.

The function Φ (p,q) gives the maximum number of p dimensional sub-spaces formed q number of $p-1$ dimensional hyper-planes. The function has following recursive form.[3]

$$\Phi(p,q) = \Phi(p-1,q) + \Phi(p-1,q-1) \tag{7}$$

By definition of $\phi(p,q)$, it is clear that

$$\Phi(p,1) = 2 \tag{8}$$

and

$$\Phi(1, q) = q + 1 \tag{9}$$

In context of Neural Networks, q – number of neurons in the first hidden layer,N_i, and p – dimension of input vector, N_0.

Table 1. Number of sub spaces formed by division of p dimensional input Vector by q neurons present in the first hidden layer

q \ p	1	2	3	4	5	6	7	8	9	10
1	2	2	2	2	2	2	2	2	2	2
2	3	4	4	4	4	4	4	4	4	4
3	4	7	8	8	8	8	8	8	8	8
4	5	11	15	16	16	16	16	16	16	16
5	6	16	26	31	32	32	32	32	32	32
6	7	22	42	57	63	64	64	64	64	64
7	8	29	64	99	120	127	128	128	128	128
8	9	37	93	163	219	247	255	256	256	256
9	10	46	130	256	382	466	502	511	512	512
10	11	56	176	386	638	848	968	1013	1023	1024

Now, re-writing (7), we get:

$$\Phi(p, q) = \Phi(1, q) + \sum_{k=1}^{p-1} \Phi(k, q - 1) \tag{10}$$

Solving recursion (10), we get :

$$\Phi(p, q) = C_{q-1}^{p} + 2 \sum_{k=0}^{p-1} \cdot C_{q-1}^{k} \tag{11}$$

where,

$$C_n^k = \frac{n!}{k! \cdot (n - k)!} \tag{12}$$

In the equations above:

- p-dimension of input vector.
- q- number of neurons in hidden layer

Lets consider an example, for a network having three neurons in first hidden layer and input vector of dimension 2. From (11), We get Φ(2,3)=7.

The number of subspaces formed due to division of the neurons in input layer by the the neurons in the first hidden layer depends solely on the number of neurons. The table presented above shows number of subspaces for different values of p and q.

Coming back to the structure of Cross-Forward Connection, according to Fig.3, input signals to the second hidden layer can be divided into two subsets:

– input received from the output of previous layer-Vector V
– raw input received - vector X

All input signals are multiplied by the adjustable weights of associated neurons i.e. matrices W_2 and C_2 respectively.

For ANN presented in fig.3, we can write:

$$e_k = \sum_{i=1}^{N_1} W_{2k,i} \cdot V_i + \sum_{j=0}^{N_0} C_{2k,j} \cdot X_j \tag{13}$$

And, finally,

For e_k=0,

$$\sum_{j=0}^{N_0} C_{2k,j} \cdot X_j = -\sum_{i=1}^{N_1} W_{2k,i} \cdot V_i \tag{14}$$

The input space, X, in (14) represents the set of parallel hyper-planes. The number of hyper-planes depend on V_i. For two dimension space, the second layer of ANN is composed of four parallel lines formed by all possible combination of values of V_i and V_j i.e.,0,0; 0,1; 1,0; 1,1.

Every subspace which is formed by the hidden layer is further divided into two smaller sub-spaces by output neuron. For N_0, dimensional input space and N_1 number of neurons in the first hidden layer, the maximum number of subspaces is given by:

$$\Psi(N_0, 2) = \Phi(N_0, N_1) \cdot \Phi(N_0, N_2) \tag{15}$$

For, W>2 ,number of sub-spcaes is:

$$\Psi(N_0, W) = \prod_{i=1}^{W} \Phi(N_0, N_i) \tag{16}$$

The number of subspaces of initial feature space in fig 3 is:

$$\Psi_{2,2} = \Phi(2, 3) \cdot \Phi(2, 1) = 7 * 2 = 14$$

For example, to divide input space into 14 subspaces, we require 3 neurons in the first hidden layer and 1 in output layer. Whereas, we need 5 neurons in the first hidden layer and 1 neuron in output layer to obtain the same number of subspaces in the standard Direct Connection. It could be concluded that the ANN with cross forward connection is more optimal than the regular straight Forward Fonnection.

3 Learning Algorithm for Cross Forward Connection Network

Less number of neurons helps convergence of algorithm during learning process. We use standard back propagation algorithm. Aim function(goal of learning

process) is defined as

$$e^2 = \frac{1}{2} \cdot \sum_{k=1}^{N_w} (y_i - z_i)^2 \tag{17}$$

where, z_i is the value provided by the teacher and y_i is the output computed by the network.

And new value of weight coefficient is:

$$W_{ij}(n+1) = W_{ij}(n) - \alpha \cdot \frac{\partial e^2}{\partial W_{ij}}\Big|_n + \beta[W_{ij}(n) - W_{ij}(n-1)] \tag{18}$$

and

$$C_{ij}(n+1) = C_{ij}(n) - \alpha \cdot \frac{\partial e^2}{\partial C_{ij}}\Big|_n + \beta[C_{ij}(n) - C_{ij}(n-1)] \tag{19}$$

4 Structure Optimization of Cross Forward Connection Network

ANN structure optimization is very complicated task and can be solved in different ways. Experience has taught us that ANN with 1 or 2 hidden layer is able to solve most of the practical problems. The problem of ANN optimization structure can be described as :

– maximizing number of subspaces, $\Psi(N_0, W)$.

when total number of neurons,N, and number number of layers, W, are given.

4.1 Optimization task for ANN with one hidden layer

For ANN with 1 hidden layer, the input neurons' number,N_0,is defined by the input vector structure X and is known as apriori. The output neurons' number N_2 is given by the output vector structure, Y - known as task definition. We can calculate the neurons' numbers in the hidden layer N_1 using equation 16. According to the optimization criterion and formula 16, the total number of subspaces for ANN with one hidden layer is given by:

$$\Phi(N_0, W) = \Phi(N_0, 2) = \Phi(N_0, N_1) \cdot \Phi(N_0, N_2) \tag{20}$$

Finally we can calculate number of neurons in one hidden layer N_1.

4.2 Optimization task for more than one hidden layer

For ANN with 2 or more hidden layers, optimization is more complicated. As the first criterion, we assume that:

– the number of layers W is given and,
– total number of neurons N is given for all hidden layers.

N can be calculated using:

$$N = \sum_{i=1}^{W-1} N_i = N_1 + N_2 + N_3 + ... + N_{W-1} \tag{21}$$

In practice we have to calculate neuron's distribution between $\{1 : W - 1\}$ layers. To find neuron's distribution, we have to maximize the number of subspaces according to the equation 22 with 23 as constraint.

$$\psi(N_0, W-1)^{opt} = \max_{N_1, N_2 \ldots N_{W-1}} \prod_{i=1}^{w-1} \Phi_i(N_0, N_i) \tag{22}$$

$$N = \sum_{i=1}^{W-1} N_i = N_1 + N_2 + N_3 + ... + N_{W-1} \tag{23}$$

From 11 and 22,

$$\begin{aligned} \Phi(N_0, N_i) = C_{N_i-1}^{N_0} + 2 \sum_{k=0}^{N_0-1} \cdot C_{N_i-1}^{k} \\ for\ i\ \epsilon\ [1; W-1] \end{aligned} \tag{24}$$

Please note that:

$$\begin{aligned} C_{N_i-1}^{N_0} = 0 \\ when\ N_i - 1 - N_0 < 0 \\ N_i \leq N_0 \end{aligned} \tag{25}$$

Taking 22, 23, 24, and 25 into account, our optimization task can be written as:

$$\Psi(N_0, W-1)^{opt} = \max_{N_1, N_2 \ldots N_{W-1}} \left\{ \prod_{i=1}^{W-1} [C_{N_1-1}^{N_0} + 2 \sum_{k=0}^{N_0-1} C_{N_i-1}^{k}] \right\} \tag{26}$$

with constraints

$$N = \sum_{i=1}^{W-1} N_i \tag{27}$$

$$C_{N_i-1}^{N_0} = 0 \ \ for \ \ N_i \leq N_0 \tag{28}$$

$$C_{N_i-1}^{k} = 0 \ \ for \ \ N_i \leq k \tag{29}$$

The optimization problem in 26 is non-linear and solution space can be divided into :

1. For all hidden layers $N_i \leq N_0$ and $N_i \leq k$ — linear task
2. For all hidden layers $N_i > N_0$ and $N_i > k$ — non-linear task

Set of hidden layers can be divided into two subspaces:

- $S1 = \{N_1, N_2, N_3, ..., N_j\}$ where $j \leq W-1$.For S_1, $N \leq N_0$ and $Ni \leq K$
- $S2 = \{N_{j+1}, N_{j+2}, N_{j+3}, ..., N_{W-1}\}$.For S_1, $N_i > N_0$ and $Ni > K$

Where W = number of layers and W-1 = number of hidden layers. This is a mixed structure, for which final solution can be found using mixture of both methods from point 1 and 2.

4.3 Neuron distribution in the hidden layers, where neurons' number for all hidden layers is less or equal than initial feature space

In this case, we have

$$N_i \leq N_0 \; for \; i \; \epsilon \{ \; 1; W-1\} \tag{30}$$

So, the total number of subspaces is defined by

$$\Phi(N_0, N_i) = \frac{(N_i - 1)!}{N_0!(N_i - 1 - N_0)!} + 2 \cdot \sum_{k=0}^{N_0 - 1} \frac{(N_i - 1)!}{k!(N_i - 1 - k)!} \tag{31}$$

or,

$$\Phi(N_0, N_i) = 0 + 2 \cdot 2^{N_i - 1} = 2^{N_i} \tag{32}$$

Our optimization target can be written as,

$$\begin{gathered} \Psi(N_0, W-1)^{opt} = \max_{N_i \; \epsilon \; [1, W-1]} \left\{ \prod_{i=1}^{W-1} 2^{N_i} \right\} = \max_{N_i \; \epsilon \; [1, W-1]} \left\{ \; 2^{\sum_{i=1}^{W-1} N_i} \; \right\} \\ for \; N = \sum_{i=1}^{W-1} N_i \\ N_i \leq N_0 \; and \; N_i, N_0 \geq 0 \end{gathered} \tag{33}$$

Equation 33 is monotonously increasing and can be written as

$$\begin{gathered} \Psi(N_0, W-1)^{opt} = \max_{N_i \; \epsilon \; [1, W-1]} \left\{ \sum_{i=1}^{W-1} N_i \right\} \\ For \; N = \sum_{i=1}^{W-1} N_i \\ N_i \leq N_0 \; and \; N_i, N_0 \geq 0 \end{gathered} \tag{34}$$

Under the given number of layers, total number of neurons have to satisfy the new constraints

$$N_i \leq N_0 \; and \; N \leq (W-1)N_0 \tag{35}$$

Example:
For ANN with $N_0 = 3, N_1 \leq 3, N_2 \leq 3, N_3 = 1$, $W = 3$, find optimum neurons distribution between two hidden layers N_1, N_2.

It is known that for output layer $N_3 = 1$ and therefore we will only consider two hidden layer for optimization process. For all N_i, where $i = 1, 2$ and $N_i \leq N_0$, using 35 we can write:

$$\begin{aligned} &N \leq (W-1) \cdot N_0 = (3-1) \cdot 3 = 6 \\ &\text{Taking } N_0 = 3 \text{ using 34 we achieve} \\ &\Psi(N_0, W-1) = \Psi(3,2) = max\{N_1 + N_2\} \\ &\text{and constraints} \\ &N_1 \leq 3 \\ &N_2 \leq 3 \\ &we \; use \; N_1 + N_2 = 4 < 6 \end{aligned} \tag{36}$$

To solve this optimization task, we can use linear programming methods or use figure 5.

Using only discrete values of N_1, N_2 for N=4, we can find three solutins $(N_1, N_2) = \{(1,3), (2,2), (3,1)\}$

The following equations indicate the number of subspaces for different number of neurons.

$$\begin{aligned} \Phi(N_0, N_1) &= \Phi(3,1) = 2^1 = 2 \\ \Phi(N_0, N_1) &= \Phi(3,2) = 2^2 = 4 \\ \Phi(N_0, N_1) &= \Phi(3,3) = 2^3 = 8 \end{aligned} \tag{37}$$

Finally, we have three optimal solutions with three different ANN structure. Every structure generates 16 subspaces and are euqivalent. Table 2.

Table 2. Solution of linear programming for N=4

N_0	N_1	N_2	$\Phi(N_0, N_1)$	$\Phi(N_0, N_2)$	$\Psi(N_0, W-1)$
3	1	3	2	8	16
3	2	2	4	4	16
3	3	1	8	2	16

In conclusion, we can say that for every given total number of neurons,N, we have many possible neurons distribution between layers. Optimal number of subspaces in the initial feature space has the same value, Ψ.

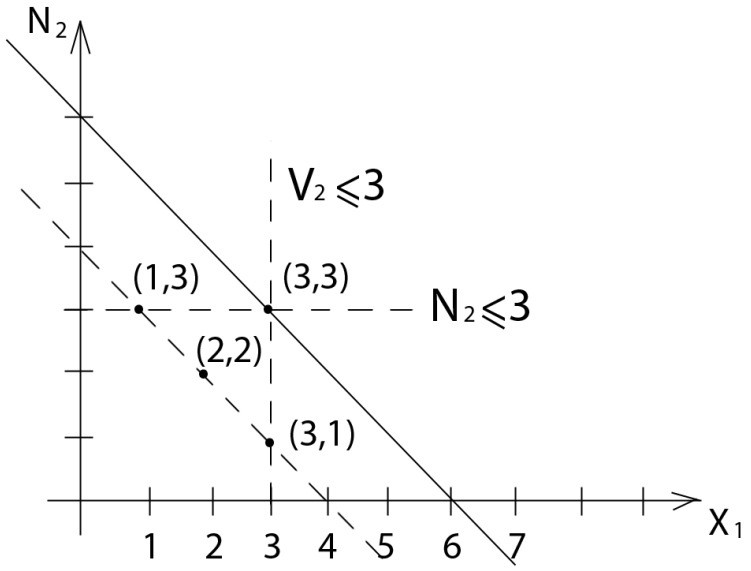


Fig. 5. Graphical solution of linear programming when total number of neurons, N=6 and N=4

4.4 Neurons distribution in the hidden layers, where neurons' number for all hidden layers is greater than initial feature space

Lets assume number of layers, W=3. It implies that we have only two hidden layers. According formula 24.

$$\begin{aligned} \Phi(N_0, N_i) =& C_{N_i-1}^{N_0} + 2\sum_{k=0}^{N_0-1} C_{N_i-1}^{k} \\ & \text{for i } \epsilon\ [1 : W-1] \ and\ N_i > N_0 \\ & \text{For whole ANN, total number of subspaces is given by} \\ \Psi(N_0, W-1) =& \Psi(N_0, 2) = \Phi_1(N_0, N_1) \cdot \Phi_2(N_0, N_2) \\ & \text{and } N_1 + N_2 = N \\ & \text{so, } \ N_1 + N_2 > 2N_0 \end{aligned} \tag{38}$$

Taking all assumptions into account we can write,

$$\begin{aligned} &\Phi(N_0, N_1) = C_{N_i-1}^{N_0} + 2 \cdot (C_{N_i-1}^{0} + C_{N_i-1}^{1} + ... + C_{N_i-1}^{N_0-1})\ for\ N_0 < N_i \\ &\Phi(N_0, N_1) < C_{N_i-1}^{N_0} + 2 \cdot 2^{N_i-1} < 2^{N_i} \end{aligned} \tag{39}$$

In this situation we do not know how many suspaces there are for $\Phi(N_0, N_1)$. To find neurons distribution between the hidden layers we should know relations between N_0, N_i and N.

Example:
For N_0=3, W=3 N=8, and N=10, N=12 find neuron distribution in the layers, were $N_i > 3$. We should maximize the quality criterion

$$\Psi(N_0, W-1)^{OPT} = \max_{N_1, N_2 \ldots N_{W-1}} \prod_{i=1}^{W_1} \left[C_{N_i-1}^{N_0} + 2 \cdot \sum_{k=0}^{N_0-1} C_{N_i-1}^{k} \right] \tag{40}$$

For example,

$$\Psi(3,2)^{OPT} = \max_{N_1, N_2} \prod_{i=1}^{2} \left[C_{N_i-1}^{3} + 2 \cdot \sum_{k=0}^{2} C_{N_i-1}^{k} \right] \tag{41}$$

After simple algebraic operations, we achieve

$$\begin{aligned} &\Psi(3,2)^{OPT} = \max_{N_1, N_2} \left\{ \frac{N_1^3 + 5N_1 + 6}{6} \cdot \frac{N_2^3 + 5N_2 + 6}{6} \right\} \\ &N_1 > 3 \\ &N_2 > 3 \\ &N_1 + N_2 = 8 > 6 \end{aligned} \tag{42}$$

We solve the equation using Kuhn-Tucker conditions. Taking 42 into account. we can write the following Lagrange equation

Table 3. Solution for non-linear Kuhn Tucker conditions for total number of neurons, N=8–12

N	$N_1 > 3$	$N_2 > 3$	$\Phi(3,21)$	*Solution*
8	4	4	225	max
9	5	4	390	max
	4	5	390	max
10	6	4	630	
	5	5	676	max
	4	6	630	
11	4	7	960	
	5	6	1092	max
	6	5	1092	max
	7	4	960	
12	4	8	1395	
	5	7	1664	
	6	6	1774	max
	7	5	1664	
	8	4	1395	

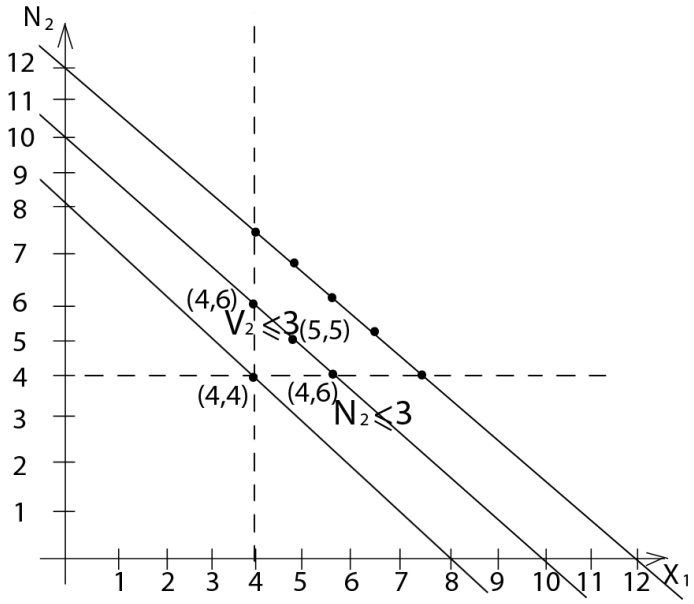


Fig. 6. Graphical solution of Kuhn Tucker conditions. Line $N = N_1 + N_2$ is a solving line with one or more solutions. Only one point is max. Figure shows three solution lines for $N_1 + N_2 = 8$, $N_1 + N_2 = 10$, $N_1 + N_2 = 12$

$$L = \frac{N_1^3 + 5N_1 + 6}{6} \cdot \frac{N_2^3 + 5N_2 + 6}{6} - \lambda_1 \cdot (N_1 - 4) - \lambda_2 \cdot (N_2 - 4) - \lambda_3 \cdot (N_1 + N_2 - 8)$$
$$N_1 - 4 \geq 0$$
$$N_2 - 4 \geq 0$$
$$N_1 + N_2 - 8 = 0 \tag{43}$$

5 Conclusion

For most practical purposes, ANNs with one hidden layer are sufficient. Learning Algorithms for the networks are time consuming and depend on number of layers and number of neurons in each layer. The running time of learning algorithm has dependency, greater than linear, on the number of neurons. Hence, the running time increases faster than the total number of neurons.

Cross Forward connection provides us an opportunity to decrease the number of neurons and thus, the running time of learning algorithm.

We implemented both Direct Connection Neural Networks and Cross Forward Neural Networks with one hidden layer and used them for pattern recognition.

Our implementation required three input neurons and two output neurons. We varied the number of neurons in hidden layer and trained both networks for limited number of epoches and noted the sum of squared errors of each output neurons. The procedure was repeated 20 times and the average sum of square of errors were recorded. Datas for two cases are presented in table 4 and 5.

Table 4. Comparision for Direct Connection and Cross Forward Connection with $N_0 = 3$, $N_1 = 1$,$N_W = 2$

Epoches	10	50	100	500	1000	5000	10000	50000
$\sum \epsilon^2$ for Direct Connection	12.40415	9.10857	8.58351	8.48001	8.38696	8.260625	8.14166	8.0152
$\sum \epsilon^2$ for Cross Forward	2.22719	0.33131	0.12325	0.02912	0.00808	0.00148	0.00076	0.00014

Table 5. Comparision for Direct Connection and Cross Forward Connection with $N_0 = 3$, $N_1 = 4$,$N_W = 2$

Epoches	10	50	100	500	1000	5000	10000	50000
$\sum \epsilon^2$ for Direct Connection	6.91134	0.28018	0.11306	0.01864	0.00542	0.000092	0.000052	0.00009
$\sum \epsilon^2$ for Cross Forward	1.02033	0.12252	0.064224	0.01945	0.00441	0.000823	0.000381	0.00007

Table 4 and 5 clearly demonstrate that for the given number of neurons in the hidden layer, Cross-Forward Connection are optimal. If we closely examine the error term in table four for Direct Connection and the same in table 5 for Cross Forward Connection we will notice that they are fairly comparable. It demonstrates that Cross Forward Connecton Structure with one neuron neuron in hidden layer is almost as good as Direct Connection with four neurons in hidden layer. Thus, Cross-Forward connection reduce the required number of neurons in ANNs.

In addition using optimizations criterion for Cross Forward Connection structures, we have solved two different tasks. For linear one , where $N_i \leq N_0$ for i=1,2,... W-1, we e achieved an equivalent ANN structures with the same number of total subspaces $\Psi(N_0, W-1)$. This means that for given total number of neurons ,N, and number of layers W , there are multiple equivalent ANN structures (Table 2). In practice this ANN structures can be used for tasks with very big dimensionality of input vector X (initial feature space). For nonlinear optimization task, where $N_i > N_0$ for i=1,2,3..... W-1, the target function is nonlinear with liner constraints. There could be one or more optimum solutions. Final solution depends on dimensionality of feature space N_0 and relation between N, N_i and W. In our example, for ANN with $N_0 = 3$, W=3, and

N=8,9,10,11,12,..... we achieved one optimum solution for even N_0s and two solutions for odd N_0s (Table 3).

References

1. Stanisaw Osowski, Sieci Neuronowe do Przetwarzania Informacji. Oficyna Wydawnicza Politechniki Warszawskiej, Warszawa 2006.
2. S. Osowski, Sieci neuronowe w ujeciu algorytmicznym.WNT, Warszawa 1996.
3. O.B.Lapunow, On Possibility of Circuit Synthesis of Diverse Elements, Mathematical Institut of B.A. Steklova, 1958.
4. Toshinori Munakate, Fundationals of the New Artificial Intelligence. Second Edition, Springer 2008.
5. Colin Fyle, Artificial Neural networks and Information Theory, Departmeeent of Ciomputing and information Systems, The University of Paisley, 2000.
6. Joarder Kamruzzaman, Rezaul Begg, Artificial Neural Networks in Finance and Manufacturing, Idea Group Publishing, 2006.
7. A. Mariciak, J. Korbicz, J. Kus, Wstepne przetwarzanie danych, Sieci Nuronowe tom 6, Akademicka Oficyna Wydawnicza EXIT 2000.
8. A. Marciniak, J. Korbicz, Neuronowe sieci modularne, Sieci Nuronowe tom 6, Akademicka Oficyna Wydawnicza EXIT 2000.
9. Z. Mikrut, R. Tadeusiewicz, Sieci neuronowe w przetwarzaniu i rozpoznawaniu obrazow, Sieci Nuronowe tom 6, Akademicka Oficyna Wydawnicza EXIT 2000.
10. L. Rutkowski, Metody i techniki sztucznej inteligencji, Wydawnictwo Naukowe PWN, warszawa 2006.
11. Juan R. Rabunal, Julian Dorado, Artificial Neural Networks in Real-Life Applications, Idea Group Publishing 2006.