Technical University of Denmark

Matrix representation of a Neural Network

Christensen, Bjørn Klint

Publication date:
2003

Document Version
Publisher's PDF, also known as Version of record

Matrix representation of a Neural Network

by
Bjørn Christensen
June 2003

Contents

1 Introduction

This paper describes the implementation of a three-layer feedforward backpropagation neural network.

The paper does not explain *feedforward*, *backpropagation* or what a *neural network* is. It is assumed, that the reader knows all this. If not please read chapters 2, 8 and 9 in *Parallel Distributed Processing*, by David Rummelhart *(Rummelhart 1986)* for an easy-to-read introduction.

What the paper does explain is how a *matrix representation* of a neural net allows for a very simple implementation.

The matrix representation is introduced in *(Rummelhart 1986, chapter 9)*, but only for a two-layer linear network and the feedforward algorithm. This paper develops the idea further to three-layer non-linear networks and the backpropagation algorithm.

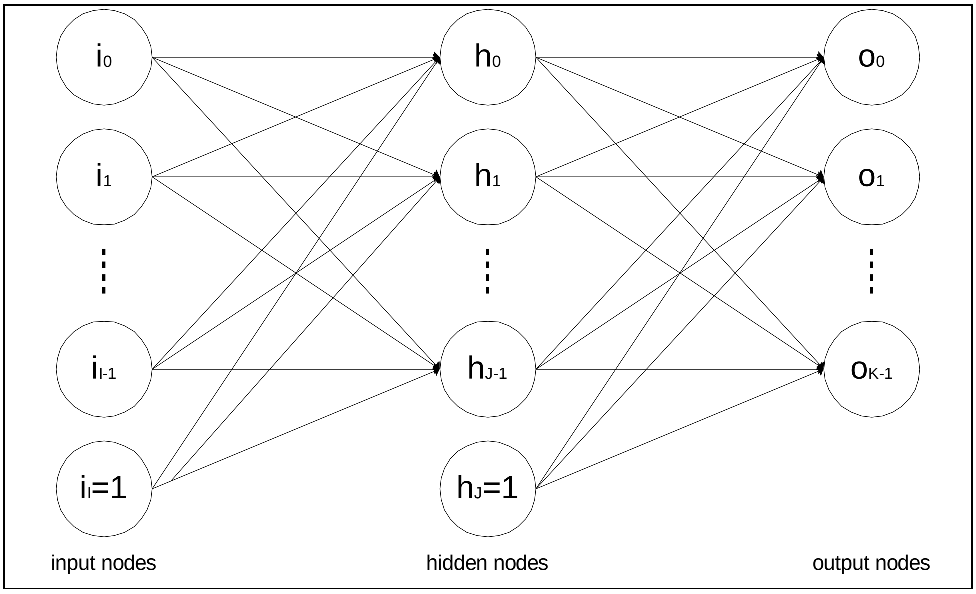

Figure 1

Figure 1 shows the layout of a three-layer network. There are I input nodes, J hidden nodes and K output nodes all indexed from 0. Bias-node for the hidden nodes is called i_I, and bias-node for the output nodes is called h_J.

2 The Weight Matrix

In the matrix representation the input, hidden and output nodes are represented by three vectors $\mathbf{i}$, $\mathbf{h}$ and $\mathbf{o}$ respectively. The weights connecting each layer are represented by a matrix. $\mathbf{V}$ connects the input layer with the hidden layer, and $\mathbf{W}$ connects the hidden layer with the output layer. See Figure 2.

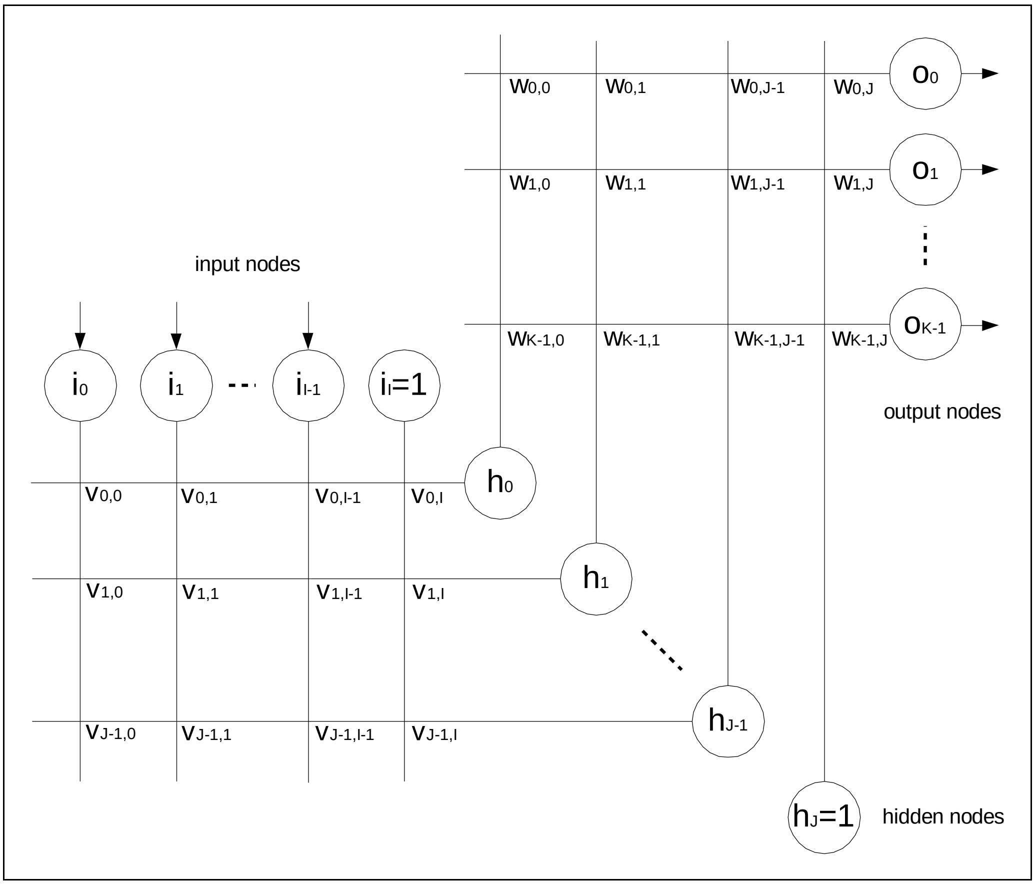

Figure 2

Definitions:

$\mathbf{i}$ - input nodes vector,	dimension = I,	indexed by $i \in [0,I-1]$
$\mathbf{h}$ - hidden nodes vector,	dimension = J,	indexed by $j \in [0,J-1]$
$\mathbf{o}$ - output nodes vector,	dimension = K,	indexed by $k \in [0,K-1]$
$\mathbf{ib}$ - input nodes vector incl bias,	dimension = I+1,	indexed by $i \in [0,I]$
$\mathbf{hb}$ - hidden nodes vector incl bias,	dimension = J+1,	indexed by $j \in [0,J]$
$\mathbf{V}$ – weight matrix from $\mathbf{Ib}$ to $\mathbf{h}$,	dimension = $J \times (I+1)$,	$\mathbf{V}_{j,i}$ weight $\mathbf{ib}_i \rightarrow \mathbf{h}_j$
$\mathbf{W}$ – weight matrix from $\mathbf{hb}$ to $\mathbf{o}$,	dimension = $K \times (J+1)$,	$\mathbf{W}_{k,j}$ weight $\mathbf{hb}_j \rightarrow \mathbf{o}_k$

Bold denotes a vector or a matrix in the text.

2.1 Feedforward

The feedforward algorithm produces an output vector given an input vector. As described in *(Callan 1999, chapter 1)*, the input to hidden node h_j is

$$net_j = V_{j,I} + \sum_{i=0}^{I-1} i_i V_{ji} \quad (1)$$

The naming convention of Callan is changed to match the definitions in this paper which follows *(Rummelhart 1986).* Note that Callan has interchanged the indices in the weight matrix.

The value of hidden node h_j is

$$\mathbf{h}_j = f(net_j) \quad (2)$$

Where f() is the activation function defined as

$$f(x) = \frac{1}{1 + e^{-x}} \quad (3)$$

Using matrix-vector multiplication, the value of all hidden nodes $\mathbf{h}$ can be calculated in a single operation

$$\mathbf{h} = F(\mathbf{V} \cdot \mathbf{ib}) \quad (4)$$

Where F() is the vector function that takes f() on all elements of it's argument. Similarly the output vector is calculated from

$$\mathbf{o} = F(\mathbf{W} \cdot \mathbf{hb}) \quad (5)$$

$\mathbf{ib}$ and $\mathbf{hb}$ are produced from

$$\mathbf{ib}_i = \mathbf{i}_i \text{ for } i \in [0..I-1], \qquad \mathbf{ib}_I = 1 \quad (6)$$

$$\mathbf{hb}_j = \mathbf{h}_j \text{ for } j \in [0..J-1], \qquad \mathbf{hb}_J = 1 \quad (7)$$

2.2 Backpropagation

The backpropagation algorithm calculates the changes of the weights based on the error between the output **o** as calculated the by the feedforward algorithm, and the desired target output value **t**.

Definition:

t - target vector,	dimension = K,	indexed by $k \in [0, K-1]$

From equation 11 in *(Rummelhart 1986, chapter 8)*, we have the change of each weight:

$$\Delta W_{kj} = \eta \delta_k h_j \tag{8}$$

Where η is the *learning rate*, δ_k is the *error signal* form output node o_k, and hj is the value of node h_j. This formula applys to all weights in the network. The error signal from the output layer and the hidden layer are calculated differently. Rummelhart defines the error signal as follows

Error signal from output node k
$$\delta o_k = o_k(1 - o_k)(t_k - o_k) \tag{9}$$

Error signal from hidden node j
$$\delta h_j = h_j(1 - h_j) \sum_{k=0}^{K-1} \delta o_k W_{kj} \tag{10}$$

Definition:

δ**o**- error signal from output nodes	dimension = K,	indexed by $k \in [0, K-1]$
δ**h**- error signal from hidden nodes	dimension = J,	indexed by $j \in [0, J-1]$

Now we are ready to calculate the weight change Δ**W** for all weights in a singe operation:

$$\begin{aligned} \Delta \mathbf{W} &= \eta \left[\begin{array}{cc|cc} \Delta W_{0,0} & \Delta W_{0,1} & \Delta W_{0,J-1} & \Delta W_{0,J} \\ \Delta W_{1,0} & \Delta W_{1,1} & \Delta W_{1,J-1} & \Delta W_{1,J} \\ \hline \Delta W_{K-1,0} & \Delta W_{K-1,1} & \Delta W_{K-1,J-1} & \Delta W_{K-1,J} \end{array} \right] \\ &= \eta \left[\begin{array}{cc|cc} \delta o_0 h_0 & \delta o_0 h_1 & \delta o_0 h_{J-1} & \delta o_0 h_J \\ \delta o_1 h_0 & \delta o_1 h_1 & \delta o_1 h_{J-1} & \delta o_1 h_J \\ \hline \delta o_{K-1} h_0 & \delta o_{K-1} h_1 & \delta o_{K-1} h_{J-1} & \delta o_{K-1} h_J \end{array} \right] \\ &= \eta \left[\begin{array}{c} \delta o_0 \\ \delta o_1 \\ \hline \delta o_{K-1} \end{array} \right] \otimes \left[\begin{array}{cc|cc} h_0 & h_1 & h_{J-1} & h_J \end{array} \right] \\ &= \eta \cdot \delta \mathbf{o} \otimes \mathbf{hb} \end{aligned} \tag{11}$$

So Δ**W** is η times the outer product of δ**o** and **hb**. $\otimes$ is the operator for the outer product. Similarly for Δ**V** we get

$$\Delta \mathbf{V} = \eta \cdot \delta \mathbf{h} \times \mathbf{ib} \tag{12}$$

We need to express $\delta\mathbf{o}$ and $\delta\mathbf{h}$ in vector form. Let us define a vector multiplication operator % that multiplies the components of vector $\mathbf{x}$ and $\mathbf{y}$ as follows:

$$\mathbf{x}\%\mathbf{y} = \begin{bmatrix} x_1 * y_1 \\ x_2 * y_2 \\ \vdots \\ x_n * y_n \end{bmatrix} \tag{13}$$

Calculating $\delta\mathbf{o}$ is easy:

$$\delta\mathbf{o} = \mathbf{o}\%(\mathbf{1}-\mathbf{o})\%(\mathbf{t}-\mathbf{o}) \tag{14}$$

Where $\mathbf{1}$ is a vector of 1's with same dimension as $\mathbf{o}$.

Calculating $\delta\mathbf{h}$ requires a little more work. Let us define a vector s as:

$$s = \mathbf{W}^T \cdot \delta\mathbf{o} \tag{15}$$

$\mathbf{W}^T$ is the transpose of $\mathbf{W}$. Expanding $\mathbf{W}$ and $\delta\mathbf{o}$ gives us:

$$s = \begin{bmatrix} W_{0,0} & W_{0,1} & \cdots & W_{0,J-1} & W_{0,J} \\ W_{1,0} & W_{1,1} & \cdots & W_{1,J-1} & W_{1,J} \\ \vdots & & & & \vdots \\ W_{K-1,0} & W_{K-1,1} & \cdots & W_{K-1,J-1} & W_{K-1,J} \end{bmatrix}^T \cdot \begin{bmatrix} \delta o_0 \\ \delta o_1 \\ \vdots \\ \delta o_{K-1} \end{bmatrix} \tag{16}$$

$$= \begin{bmatrix} W_{0,0} & W_{1,0} & \cdots & W_{K-1,0} \\ W_{0,1} & W_{1,1} & \cdots & W_{K-1,1} \\ \vdots & & & \vdots \\ W_{0,J-1} & W_{1,J-1} & \cdots & W_{K-1,J-1} \\ W_{0,J} & W_{1,J} & \cdots & W_{K-1,J} \end{bmatrix} \cdot \begin{bmatrix} \delta o_0 \\ \delta o_1 \\ \vdots \\ \delta o_{K-1} \end{bmatrix}$$

$$= \begin{bmatrix} \delta o_0 W_{0,0} + \delta o_1 W_{1,0} + \ldots + \delta o_{K-1} W_{K-1,0} \\ \delta o_0 W_{0,1} + \delta o_1 W_{1,1} + \ldots + \delta o_{K-1} W_{K-1,1} \\ \delta o_0 W_{0,J-1} + \delta o_1 W_{1,J-1} + \ldots + \delta o_{K-1} W_{K-1,J-1} \\ \delta o_0 W_{0,J} + \delta o_1 W_{1,J} + \ldots + \delta o_{K-1} W_{K-1,J} \end{bmatrix}$$

$$= \begin{bmatrix} s_0 \\ s_1 \\ \vdots \\ s_{J-1} \\ s_J \end{bmatrix} \quad where \ s_j = \sum_{k=0}^{K-1} \delta o_k W_{kj}$$

The component s_j is operand in the error signal δh_j. Note that we do not need an error signal from h_J since it is a bias node. Therefore we skip the last component of s, and create a vector $\mathbf{s}$ defined as:

$$\mathrm{s}_j = s_j \text{ for } j \in [0..J-1] \tag{17}$$

Dimension of $\mathbf{s}$ is J. Finally we can write $\delta\mathbf{h}$ as:

$$\delta\mathbf{h} = \mathbf{h}\%(\mathbf{1}-\mathbf{h})\%\mathbf{s} \tag{18}$$

Here $\mathbf{1}$ has same dimension as $\mathbf{h}$.

Let us sum up the important equations of the backpropagation algorithm:

Error signal from output nodes	$\delta\mathbf{o} = \mathbf{o}\%(\mathbf{1}-\mathbf{o})\%(\mathbf{t}-\mathbf{o})$	(19)
Changes of weights connected to output nodes	$\Delta\mathbf{W} = \eta \cdot \delta\mathbf{o} \times \mathbf{hb}$	(20)
Propagated errors from output nodes	$s = \mathbf{W}^T \cdot \delta\mathbf{o}$	(21)
Error signal from hidden nodes	$\delta\mathbf{h} = \mathbf{h}\%(\mathbf{1}-\mathbf{h})\%\mathbf{s}$	(22)
Changes of weights connected to hidden nodes	$\Delta\mathbf{V} = \eta \cdot \delta\mathbf{h} \times \mathbf{ib}$	(23)

2.3 Squared Error

The squared error is a measure of the correctness of the network. It is calculated for each input pattern p. Rummelhart defines the squared error as:

$$E_p = \frac{1}{2}\sum_{k=0}^{K-1}(t_k - o_k)^2 \qquad (24)$$

In vector notation using the inner product this is

$$E_p = \frac{1}{2}(\mathbf{t} - \mathbf{o}) \cdot (\mathbf{t} - \mathbf{o}) \qquad (25)$$

Summing the squared errors for all patterns, we get the squared error E for an epoch

$$E = \sum_{p=1}^{n} E_p \qquad (26)$$

Where n is the number of patterns in the training set.

3 References

Rummelhart 1986 — Parallel Distributed Programming: Explorations in the Microstructure of Cognition. Vol 1
David E Rummelhart, James L McClelland
MIT-press 1986

Callan 1999 — The Essence of Nearal Networks
Robert Callan
Prentice Hall 1999