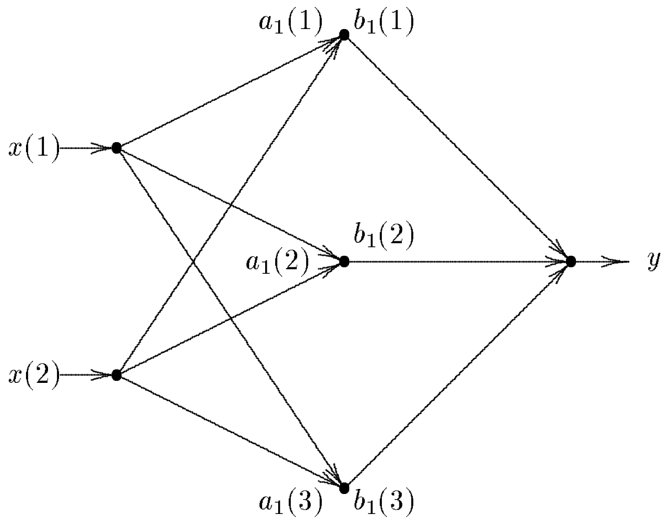
Introduction

These notes deal mainly with the mathematics related to so-called feedforward multilayer perceptrons, which form an important part of all the artificial neural networks.

1. A simple feedforward network

Consider the following network with 3 levels of which 1 is hidden.



Here we are given two real numbers $x(1)$ and $x(2)$, then one calculetes new numbers $a_1(1)$, $a_1(2)$ and $a_1(3)$ using the formula $a_1(j) = W_1(j,1)x(1) + W_1(j,2))x(2) - \tau_1(j)$, for $j = 1, 2, 3$, then one calculates the numbers $b_1(1)$,

$b_1(2)$ and $b_1(3)$ with the formula $b_1(j) = \sigma_1(a_1(j))$. Finally one gets $y = W_2(1)b_1(1) + W_2(2)b_1(2) + W_2(3)b_1(3) - \tau_2$. It turns out to be important that the activation function σ_1 is not linear (or a polynomial), one can for exampe choose $\sigma_1(t) = \frac{1}{1+e^{-t}}$. This network has one input-layer, one hidden layer and one output-layer.

Let us generalize the situation to the case where there are L layers. The number of nodes at the level j is denoted by d_j. Thus the input is a vector $\mathbf{x} \in \mathbb{R}^{d_0}$ and the output is a vector $\mathbf{y} \in \mathbb{R}^{d_L}$. The calculations in the network proceeds as follows. Let $\mathbf{b}_0 = \mathbf{x}$. If $\mathbf{b}_{j-1}$ has been calculated where $\mathbf{b}_{j-1}$ is a vector in $\mathbb{R}^{d_{j-1}}$, then $\mathbf{a}_j = W_j\mathbf{b}_{j-1} - \tau_j$ where W_j is a $d_j \times d_{j-1}$ matrix of weights and $\mathbf{t}_j$ is a d_j-dimensional (column) vector. Now $\mathbf{b}_j = \sigma_j(\mathbf{a}_j)$ where σ_j is some function, often a real-valued function of a real argument that is extended to a vector function by applying it componentwise. It is of course possible to have different node functions at all the nodes at the same level, but not very much is perhaps gained by this. Proceeding in this way one finally gets $\mathbf{y} = \mathbf{b}_L$.

One could, of course, use different node functions at different nodes on the same layer, but this is in most cases not of very much use.

2. Supervised and unsupervised learning

In the case of supervised learning the problem could be how to choose the weight matrices W_j and the threshold vectors τ_j (and to a lesser extent how to choose the node functions σ_j in the network above so that the output of the vector becomes what one wants it to be.

One case of unsupervised learning is that one wants to modify the weights and thresholds in such a way that the output of the network are the principal componets of the input vectors, that is the projection onto the space spanned by the eigenvectors associated with the largest eigenvalues in the correlation matrix of the input vectors.

3. Some incomplete comments on sources

As general references one can mention [**3**] and [**5**]. The presentation of the results in chapter 2 follows mostly that of [**6**] and that in 3 that in [**8**, Chap. 17].

The results on the conjugate gradiemt method can be found in [4], and Theorem 17 is taken from [**1**].

Chapter 5 follows mostly [**2**] and for the results in chapter 6 see, e.g., [**9**].