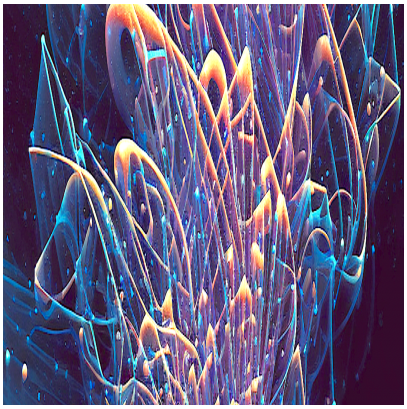


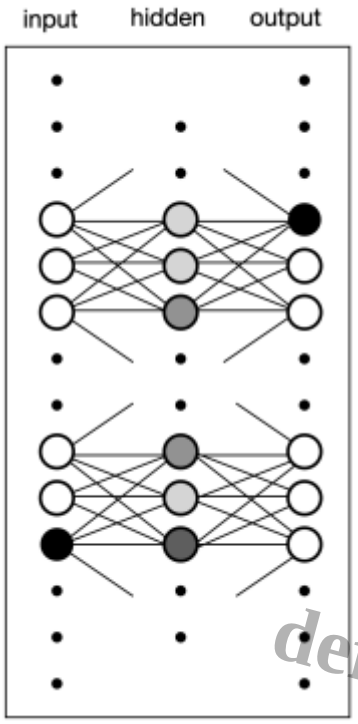
Just one neuron

Description

I revisited our earlier (1990) article on neural networks â??Spatial applications of neural networks in computer-aided design.â• Neural networks were novel in architecture and CAD. What follows is an update of the part of that article in starting to explain how neural networks function.

Neural network layers

Neural network (NN) models store information as numbers attached to nodes and arcs (links) in a network. Nodes are sometimes called â??unitsâ• or â??neuronsâ• to reinforce the analogy with biological neural systems (i.e. brains). There are input nodes, output nodes and hidden nodes, all connected by directed arcs. Input, output and hidden nodes are generally organised in layers. The layers are stacked in the direction from inputs to outputs. Every node in a layer is connected to every node in the next layer. There can be several layers in a neural network.



A very helpful video by Grant Sanderson explains the process for recognising hand-written numerical integers 0-9.

He uses slick animated graphics to demonstrates NN processes. The input layer of the NN consists of the 784 individual pixel values from a 28Ã?28 scan of a hand written integer. So for that exercise the input layer consists of 784 nodes. There are 2 hidden layers in his example, each with an arbitrary number of nodes (16 each) and the output layer, consists of 10 nodes, one for each integer (0-9).

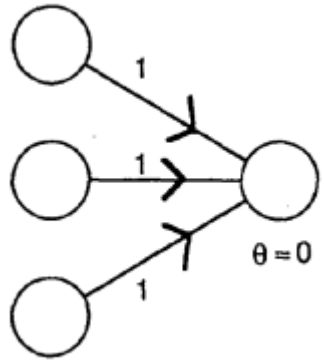
My example that follows is much simpler than the network described by Sanderson, and Iâ??ll focus here on what happens to a single node. To demonstrate operations on a single node, it is simplest to think of each node as receiving binary inputs that produce binary outputs. The output value of a node is the result of adding the products of the weights of arcs directed inwards to the node. Before the node delivers its output, the weighted sum is adjusted by a kind of bias function â?? sometimes called a threshold function.

In my simplified model here I start with a particular a threshold value. That will be an integer 0, 1, 2, 3, â?¦. If the sum of the weighted inputs to a node is greater than the threshold then the node will â??fireâ?• (producing an output value of 1). Net values less than or equal to the threshold produce a 0 as output.

In a full NN, the output from one node provides the input to a node with which it is connected. The NN algorithm inspects each node in turn and passes output values through the network, effectively propagating activation values through the network to produce a configuration of output values in

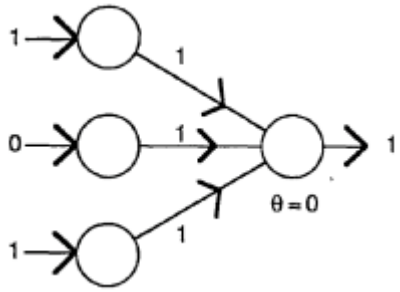
response to a given set of inputs.

Here I will consider just 3 input nodes and one output node that processes binary values. Each input node is connected to the output node with a directed arc of weight 1. The output node has a threshold value ($\hat{I}_{\cdot}$) associated with it of 0.

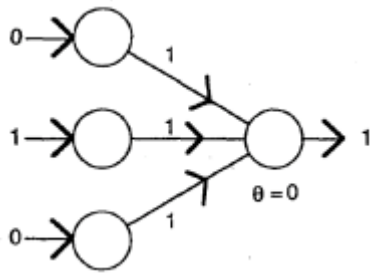


To summarise so far, the node output is calculated by summing the product of each input node with each weight (to produce a net input) and comparing this with the threshold of the output node. If the weighted input is a greater value than the threshold then this mini network produces a 1 as output. If it is less than or equal to the threshold the network generates a 0 output value.

Following this procedures, as shown below the input pattern {1 0 1} produces 1 as output (the weighted sum is 2, which is greater than the threshold value 0).



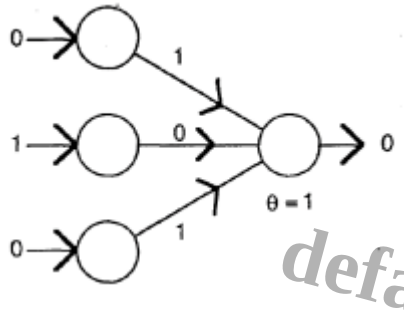
The input pattern {0 1 0} also produces 1 (the weighted sum is 1, which is greater than the threshold value 0).



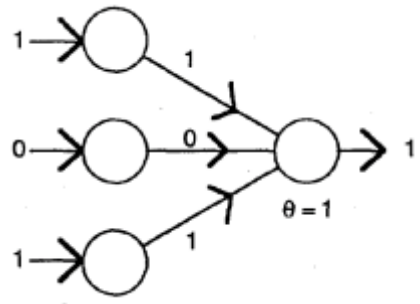
Suppose in this input case {0 1 0} we would rather the node generate a different response, e.g. produce a 0 as output instead of 1. By inspecting the node above it should be apparent that we wish to decrease the influence of weights from input units with value 1. The following algorithm accomplishes this:

1. Calculate the value of the output unit produced from the given input pattern.
2. Compare this calculated value with the preferred output presented to the node.
3. Inspect each of the input values in tum
    - **If** the input value is 1, the calculated output is 1 and the preferred output is 0
    - **then** subtract 1 from the weight of the arc emanating from that node and add 1 to the threshold of the output node.

Decreasing the weight and raising the threshold of the output unit adjusts the node to produce a 0 as output. This results in the revised network here.

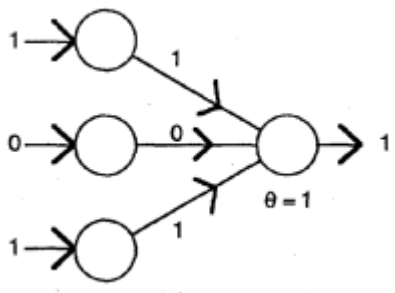


If we now present this network with the input pattern {0 1 0} it produces the required output pattern of {0}. The training algorithm needs also to check that the previous input {1 0 1} still gives the desired output of { 1 }. As shown here the output is still {1}.



We also need a rule for adjusting weights and thresholds in the event that we require the output to be 1 but the NN calculates a 0.

- **If** the input value is 1, the predicted output is 0 and the required output is 1
- **then** add 1 to the weight of the arc emanating from that unit and subtract 1 from the threshold of the output node.



There is no change to weights and thresholds if the training and predicted outputs are the same. The resultant network above provides the required match for both input-output patterns.

Scaling up

Full-fledged NNs operate with real numbers and not just integers or binaries, and a node will have many inputs. For a network as a whole there are two operations: that in which the system is trained on various output patterns, and the simulation or matching operation where we present the NN with the input only and it generates an output.

The training operation involves cycling through the set of given input-output patterns and adjusting weights and thresholds according to rules similar to those above. Training is a much longer process than simulation. The length of time it takes the training algorithm to calculate its weights and thresholds increases with the number of training examples presented. Running the network to produce an output consistent with a new input during simulation is a simpler operation and computation time is independent of the number of training examples.

In the training operation it is usual to cycle through the algorithm several times as the adjustment of weights and thresholds may cause the network to â??forgetâ?• a pattern it has just stored via its weights and thresholds.

Where there is no overlap between the input patterns (that is, where the input patterns are linearly independent, as in my simple example) the algorithm is guaranteed to find a system of weights and thresholds to store all input and output pattern pairs. Where any input node has the same value (1 or 0) in more than one training pattern (the most usual case) there is no such certainty.

Randomness improves accuracy

Stochastic methods improve the performance of the NN in the most general case. The difference between the net input and the threshold to an output unit actually carries some useful information. The differences serve not only to gauge whether the output should be 0 or 1, but it also gives some indication of the strength of conviction in the network in the output value. It is a measure of the probability that the output unit will â??fire.â?•

In simulation we could therefore produce a graded output to indicate non-binary measures of certainty in the output. There is a convenient function for distributing the probability from 0 to 1, calculated from the differences between the net inputs and the output threshold. Here is the logistic function for mapping the difference between the input and the threshold to a unit (*Diff*) onto a probability value (*p*) depending on a constant *T* which determines the slope of
the curve. The function is

$$p = \frac{1}{1 + e^{-Diff/T}}$$

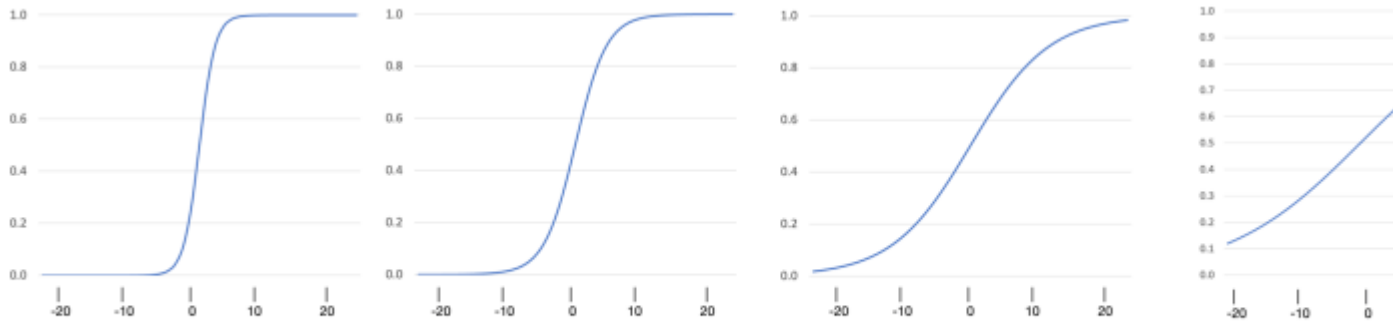
Where there are similarities in the input patterns, that is, where different input patterns have certain nodes in common, the probabilisitic values of the output nodes can accommodate the overlapping nature of the corresponding outputs.

Getting stuck

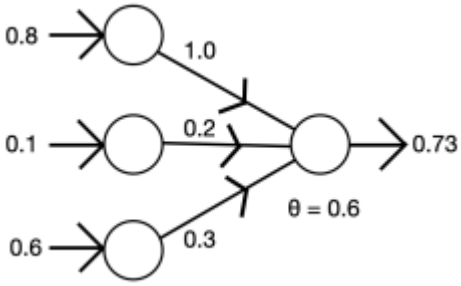
One of the problems that a neural network may encounter is that during successive cycles the system may change certain weights in response to a particular input-output pair in one cycle, only to undo the effect in another cycle in response to another input-output pair. The system can become â??lockedâ?• into a set of weights that is not optimal and which is influenced by the order in which input-output pairs are presented to the NN. One further sophistication in the training procedure seeks to address this problem.

A common technique is to introduce controlled randomness into the training process. As described above, when the NN trains, it adjusts weights and thresholds on the basis of the predicted value of an output node and the desired (i.e. training) output. Using the logistic function the training algorithm can calculate the likelihood of the predicted value being a 1 or 0 and during subsequent training iterations generate a 1 or 0 at random according to this probability. Surprisingly, this introduction of randomness over a large number of training cycles across a large network produces a more accurate result. Over many learning cycles it produces a smoother series of weights and thresholds. It allows the system to train more slowly. The analogy is often made between this process and thermodynamic annealing.

The constant factor *T* in the logistic function is analogous to temperature. The higher the temperature of the system the more random the predicted values at output units and the greater the means of escaping from a set of weights and thresholds that is sub-optimal. With this stochastic approach the difference between net values and thresholds serves as a good measure of the degree of certainty attached to the output.



The slope of the curve varies according to the constant T. Very small values of T produce a stepped function. To illustrate a more general input-output calculation, here is network with non-binary inputs, weights, threshold and output for T = 1.0.



What hidden layers are for

Hidden layers of nodes in the network increases the number of parameters by which patterns can be reconstructed. With large numbers of hidden units there is more chance that the system will generalize on features rather than patterns of binary values with absolute locations, e.g. on a grid as in my previous post. Sanderson explains features in terms of the relative slopes and positions of various lines in hand drawn digits. In the case of the rudimentary floor plans in my previous post the features might include the relative positions of floor plan elements (courtyards, recesses, wings, etc).

With hidden units the algorithms for finding a set of weights and thresholds to account for input and output pairs must resolve a complex optimization problem. Thatâ??s for another time.

References

- Coyne, Richard, and Arthur Postmus. â??Spatial applications of neural networks in computer-aided design.â?• *Artificial Intelligence in Engineering* 5, no. 1 (1990): 9-22.
- Hinton, G. E., and T. J. Sejnowski. â??Learning and relearning in Boltzmann machines.â?• In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume I, Foundations*, edited by D. E. Rumelhart, and J. L. McClelland, 282-314. Cambridge, MA: MIT Press, 1986.
- Rumelhart, D. E., and J. L. (eds) McClelland. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*: Volume 1, Fundamentals. Cambridge, Mass.: MIT Press, 1987.
- McClelland, J. L., D. E. Rumelhart, and G. E. Hinton. â??The appeal of parallel distributed processing.â?• In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1, Foundations*, edited by D. E. Rumelhart, and J. L. McClelland, 3-44. Cambridge, MA: MIT Press, 1986.
- Sanderson, Grant. â??Neural Networks: The basics of neural networks, and the math behind how they learn.â?• *3Blue1Brown*, 2023. Accessed 12 February 2023. https://www.3blue1brown.com/topics/neural-networks

Note

- Other methods are also used for calculating the probability of an output value than the logistic function, notably Tanh and ReLU (Rectified Linear Unit).
- Featured image is by MidJourney, prompted with the cover of the book by Rumelhart and McClelland and the words â??neural network.â?• Here is the full image.

Category

1. Artificial Intelligence

Date Created
March 25, 2023
Author
rcoyne99