Artificial neuron

From Wikipedia, the free encyclopedia

An **artificial neuron** is a mathematical function conceived as a model of biological neurons. Artificial neurons are the constitutive units in an artificial neural network. Depending on the specific model used they may be called a **semi-linear unit**, **Nv neuron**, **binary neuron**, **linear threshold function**, or **McCulloch–Pitts (MCP) neuron**. The artificial neuron receives one or more inputs (representing dendrites) and sums them to produce an output (representing a neuron's axon). Usually the sums of each node are weighted, and the sum is passed through a non-linear function known as an activation function or transfer function. The transfer functions usually have a sigmoid shape, but they may also take the form of other non-linear functions, piecewise linear functions, or step functions. They are also often monotonically increasing, continuous, differentiable and bounded.

The artificial neuron transfer function should not be confused with a linear system's transfer function.

Contents

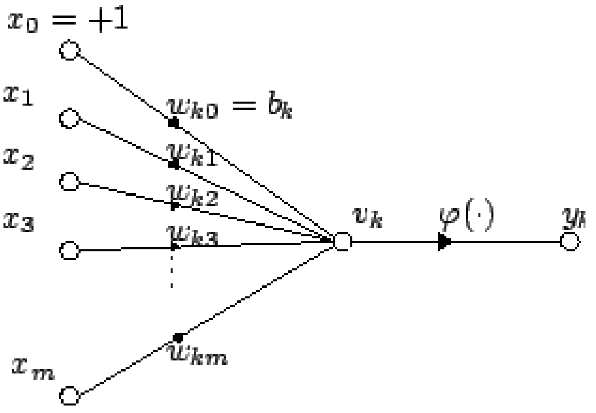
Basic structure

For a given artificial neuron, let there be $m + 1$ inputs with signals x_0 through x_m and weights w_0 through w_m. Usually, the x_0 input is assigned the value +1, which makes it a *bias* input with $w_{k0} = b_k$. This leaves only m actual inputs to the neuron: from x_1 to x_m.

The output of the kth neuron is:

$$y_k = \varphi\left(\sum_{j=0}^{m} w_{kj} x_j\right)$$

Where φ (phi) is the transfer function.



The output is analogous to the axon of a biological neuron, and its value propagates to the input of the next layer, through a synapse. It may also exit the system, possibly as part of an output vector.

It has no learning process as such. Its transfer function weights are calculated and threshold value are predetermined.

Comparison to biological neurons

Artificial neurons are designed to mimic aspects of their biological counterparts.

- *Dendrites* – In a biological neuron, the dendrites act as the input vector. These dendrites allow the cell to receive signals from a large (>1000) number of neighboring neurons. As in the above mathematical treatment, each dendrite is able to perform "multiplication" by that dendrite's "weight value." The multiplication is accomplished by increasing or decreasing the ratio of synaptic neurotransmitters to signal chemicals introduced into the dendrite in response to the synaptic neurotransmitter. A negative multiplication effect can be achieved by transmitting signal inhibitors (i.e. oppositely charged ions) along the dendrite in response to the reception of synaptic neurotransmitters.

- *Soma* – In a biological neuron, the soma acts as the summation function, seen in the above mathematical description. As positive and negative signals (exciting and inhibiting, respectively) arrive in the soma from the dendrites, the positive and negative ions are effectively added in summation, by simple virtue of being mixed together in the solution inside the cell's body.

- *Axon* – The axon gets its signal from the summation behavior which occurs inside the soma. The opening to the axon essentially samples the electrical potential of the solution inside the soma. Once the soma reaches a certain potential, the axon will transmit an all-in signal pulse down its length. In this regard, the axon behaves as the ability for us to connect our artificial neuron to other artificial neurons.

Unlike most artificial neurons, however, biological neurons fire in discrete pulses. Each time the electrical potential inside the soma reaches a certain threshold, a pulse is transmitted down the axon. This pulsing can be translated into continuous values. The rate (activations per second, etc.) at which an axon fires converts directly into the rate at which neighboring cells get signal ions introduced into them. The faster a biological neuron fires, the faster nearby neurons accumulate electrical potential (or lose electrical potential, depending on the "weighting" of the dendrite that connects to the neuron that fired). It is this conversion that allows computer scientists and mathematicians to simulate biological neural networks using artificial neurons which can output distinct values (often from −1 to 1).

History

The first artificial neuron was the Threshold Logic Unit (TLU), or Linear Threshold Unit,[1] first proposed by Warren McCulloch and Walter Pitts in 1943. The model was specifically targeted as a computational model of the "nerve net" in the brain.[2] As a transfer function, it employed a threshold, equivalent to using the Heaviside step function. Initially, only a simple model was considered, with binary inputs and outputs, some restrictions on the possible weights, and a more flexible threshold value. Since the beginning it was already noticed that any boolean function could be implemented by networks of such devices, what is easily seen from the fact that one can implement the AND and OR functions, and use them in the disjunctive or the conjunctive normal form. Researchers also soon realized that cyclic networks, with feedbacks through neurons, could define dynamical systems with memory, but most of the research concentrated (and still does) on strictly feed-forward networks because of the smaller difficulty they present.

One important and pioneering artificial neural network that used the linear threshold function was the perceptron, developed by Frank Rosenblatt. This model already considered more flexible weight values in the neurons, and was used in machines with adaptive capabilities. The representation of the threshold values as a bias term was introduced by Bernard Widrow in 1960 – see ADALINE.

In the late 1980s, when research on neural networks regained strength, neurons with more continuous shapes started to be considered. The possibility of differentiating the activation function allows the direct use of the gradient descent and other optimization algorithms for the adjustment of the weights. Neural networks also started to be used as a general function approximation model. The best known training algorithm called backpropagation has been rediscovered several times but its first development goes back to the work of Paul Werbos.[3][4]

Types of transfer functions

The transfer function of a neuron is chosen to have a number of properties which either enhance or simplify the network containing the neuron. Crucially, for instance, any multilayer perceptron using a *linear* transfer function has an equivalent single-layer network; a non-linear function is therefore necessary to gain the advantages of a multi-layer network.

Below, *u* refers in all cases to the weighted sum of all the inputs to the neuron, i.e. for *n* inputs,

$$u = \sum_{i=1}^{n} w_i x_i$$

where **w** is a vector of *synaptic weights* and **x** is a vector of inputs.

Step function

The output *y* of this transfer function is binary, depending on whether the input meets a specified threshold, *θ*. The "signal" is sent, i.e. the output is set to one, if the activation meets the threshold.

$$y = \begin{cases} 1 & \text{if } u \geq \theta \\ 0 & \text{if } u < \theta \end{cases}$$

This function is used in perceptrons and often shows up in many other models. It performs a division of the space of inputs by a hyperplane. It is specially useful in the last layer of a network intended to perform binary classification of the inputs. It can be approximated from other sigmoidal functions by assigning large values to the weights.

Linear combination

In this case, the output unit is simply the weighted sum of its inputs plus a *bias* term. A number of such linear neurons perform a linear transformation of the input vector. This is usually more useful in the first layers of a network. A number of analysis tools exist based on linear models, such as harmonic analysis, and they can all be used in neural networks with this linear neuron. The bias term allows us to make affine transformations to the data.

See: Linear transformation, Harmonic analysis, Linear filter, Wavelet, Principal component analysis, Independent component analysis, Deconvolution.

Sigmoid

A fairly simple non-linear function, a sigmoid function such as the logistic function also has an easily calculated derivative, which can be important when calculating the weight updates in the network. It thus makes the network more easily manipulable mathematically, and was attractive to early computer scientists who needed to minimize the computational load of their simulations. It is commonly seen in multilayer perceptrons using a backpropagation algorithm.

Pseudocode algorithm

The following is a simple pseudocode implementation of a single TLU which takes boolean inputs (true or false), and returns a single boolean output when activated. An object-oriented model is used. No method of training is defined, since several exist. If a purely functional model were used, the class TLU below would be replaced with a function TLU with input parameters threshold, weights, and inputs that returned a boolean value.

```
class TLU defined as:
 data member threshold : number
 data member weights : list of numbers of size X
 function member fire( inputs : list of booleans of size X ) : boolean defined as:
  variable T : number
  T ← 0
  for each i in 1 to X :
   if inputs(i) is true :
    T ← T + weights(i)
   end if
  end for each
  if T > threshold :
   return true
  else:
   return false
  end if
 end function
end class
```

Limitations

Simple artificial neurons, such as the McCulloch–Pitts model, are sometimes described as "caricature models", since they are intended to reflect one or more neurophysiological observations, but without regard to realism.[5]

Comparison with biological neuron coding

New research has shown that unary coding is used in the neural circuits responsible for birdsong production. [6] [7] The use of unary n biological networks is presumably due to the inherent simplicity of the coding. Another contributing factor could be the fact that unary coding provides a certain degree of error correction.[8]

See also

- ADALINE
- Biological neuron models
- Binding neuron
- Connectionism
- Neural network
- Nv network
- Perceptron

References

1. Martin Anthony (January 2001). *Discrete Mathematics of Neural Networks: Selected Topics* (http://books.google.com/books?id=qOy4yLBqhFcC&pg=PA3). SIAM. pp. 3–. ISBN 978-0-89871-480-7.
2. Charu C. Aggarwal (25 July 2014). *Data Classification: Algorithms and Applications* (http://books.google.com/books?id=gJhBBAAAQBAJ&pg=PA209). CRC Press. pp. 209–. ISBN 978-1-4665-8674-1.
3. Paul Werbos, Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences. PhD thesis, Harvard University, 1974
4. Paul Werbos, Backpropagation through time: what it does and how to do it. Proceedings of the IEEE, Volume 78, Issue 10, 1550–1560, Oct 1990, doi10.1109/5.58337
5. F. C. Hoppensteadt and E. M. Izhikevich (1997). *Weakly connected neural networks*. Springer. p. 4. ISBN 978-0-387-94948-2.
6. Squire, L.; Albright, T.; Bloom, F.; Gage, F.; Spitzer, N., eds. (October 2007). *Neural network models of birdsong production, learning, and coding* (http://web.archive.org/web/20150412190625/https://clm.utexas.edu/fietelab/Papers/birdsong_review_topost.pdf) (PDF). New Encyclopedia of Neuroscience: Elservier. Archived from the original (https://clm.utexas.edu/fietelab/Papers/birdsong_review_topost.pdf) (PDF) on 2015-04-12. Retrieved 12 April 2015.
7. Moore J.M. et al., Motor pathway convergence predicts syllable repertoire size in oscine birds. Proc. Nat. Acad. Sc. USA 108: 16440–16445, 2011.PubMed (https://www.ncbi.nlm.nih.gov/pubmed/21918109)doi:10.1073/pnas.1102077108 (https://dx.doi.org/10.1073%2Fpnas.1102077108)
8. Potluri, P., Error correction capacity of unary coding. 2014. [1] (http://arxiv.org/abs/1411.7406)

Further reading

- McCulloch, W. and Pitts, W. (1943). *A logical calculus of the ideas immanent in nervous activity.* Bulletin of Mathematical Biophysics, 5:115–133. [2] (http://link.springer.com/article/10.1007%2FBF02478259)
- A.S. Samardak, A. Nogaret, N. B. Janson, A. G. Balanov, I. Farrer and D. A. Ritchie. "Noise-Controlled Signal Transmission in a Multithread Semiconductor Neuron" // Phys.Rev.Lett. 102 (2009) 226802, [3] (http://prl.aps.org/abstract/PRL/v102/i22/e226802)

External links

- Artifical neuron mimicks function of human cells (https://www.youtube.com/watch?v=NhTZnnJJP64)
- [4] (http://www.mind.ilstu.edu/curriculum/modOverview.php?modGUI=212) A good general overview

Retrieved from "https://en.wikipedia.org/w/index.php?title=Artificial_neuron&oldid=673303746"

Categories: Artificial neural networks | American inventions