1 Basics of biological neural computation

Contents

Abstract

Artificial neural networks should be good at things that our brains are good at (i.e. parallel processing - e.g. vision), and bad at things our brains are bad at (e.g. long multiplication). In this chapter we look at the basic theory of how the human brain works, and the theory of neurons; in particular, we idealise neurons, removing the complicated details that are not essential for understanding the main principles.

1.1 Synapses vs. RAM

1.1.1 Disadvantages

- Synapses are *slow*

1.1.2 Advantages

- Synapses are very small and very low-power.
- Synapses adapt using locally available signals.

1.2 How the brain works

- Each neuron receives inputs from other neurons
  - A few neurons also connect to receptors.
  - Cortical neurons use "spikes" to communicate.
- The effect of each input line on the neuron is ***controlled by a synaptic weight*** (weights can be positive or negative).
- The ***synaptic weights adapt*** so the whole network learns to performs useful computations:
  - Recognising objects.
  - Understanding language.
  - Making plans.
  - Controlling the body.
- Humans have about 10^{11} *neurons*, each with about 10^4 *weights*.
  - A huge number of weights can affect the computations in a very short time.

1.3 Modularity and the brain

- Different bits of the cortex do different things (e.g. speech recognition).
  - Specific task increase blood flow to specific regions.
  - Local damage to the brain has specific effects; although the cortex looks pretty much the same all over (indeed, early brain damage makes functions relocate and in response to experience the brain had the ability to turn into special purpose hardware).

1.4 Simple models of neurons

To model things, it's beneficial to idealise them removing the complicated details that are not essential for understanding the main principles. Once we understand the basic principles, it's easy to add complexity to make the model more faithful. In the following types of neurons (aka. "units"), $z := b + \sum_i x_i w_i$

1.4.1 Linear neurons

$$y = z$$

Simple, but computationally limited. The output y is a function of the bias, b of the neuron, plus the sum of all its incoming connections on the activity on an input neuron x_i times the weight on that neuron, w_i.

1.4.2 Binary threshold neurons (McCulloch-Pitts, 1943)

$$y = \mathbb{I}_{[z \geq 0]}$$

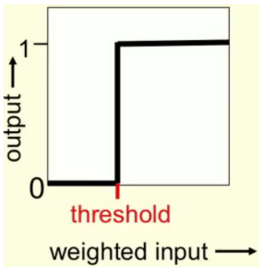

Figure 1.1: First compute a weighted sum of the inputs, then send out a fixed size spike of activity if the weighted sum exceeds a threshold. McCulloch and Pitts thought that each spike is like the truth values of a proposition, and each neuron combines truth values to compute the truth value of another proposition.

1.4.3 Rectified Linear Neuron (aka. linear threshold neurons)

$$y = \begin{cases} z & \text{if } z > 0 \\ 0 & \text{otherwise} \end{cases}$$

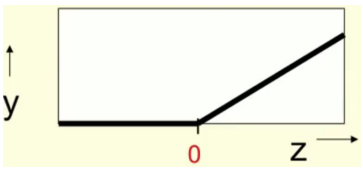

Figure 1.2: They compute a linear weighted sum of their inputs, and the output is a *non-linear* function of the total input.

1.4.4 Sigmoid Neurons

These are the *most commons neurons modelled by artificial neuron networks*, introducing non-linearity in the model.

$$y = \frac{1}{1 + e^{-z}}$$

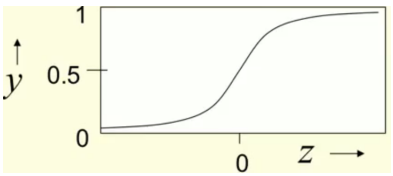

Figure 1.3: Sigmoid neurons give a real-valued output that is a smooth and bounded function of their total input. Typically they use the logistic function, and have nice derivatives which make learning easy.

1.4.5 Stochastic Binary Neurons

$$p := \mathbb{P}(s = 1) = \frac{1}{1 + e^{-z}}$$

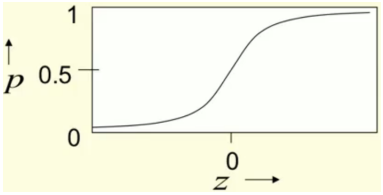

Figure 1.4: Stochastic Binary Neurons treat the output of the logistic equation as the probability of producing a spike in a short time window.

Remark. Rectified linear units, the output is treated as the *Poisson rate* for spikes.