Neural Nets and the Perceptron (Part 1, Artificial Neurons)

Daniel Lawson — University of Bristol

Lecture 09.1.1 (v1.0.2)

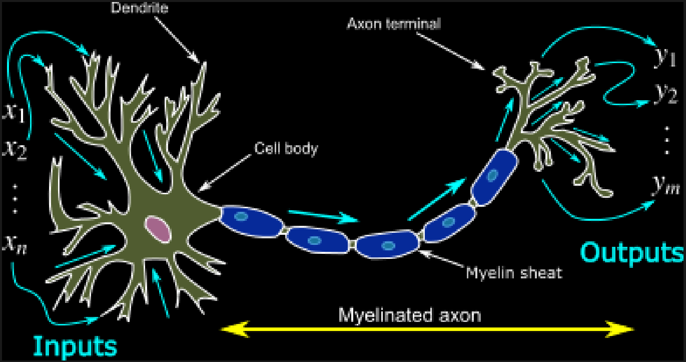
Signposting

- This Block is split into two Lectures:
  - 09.1 (this lecture) on the theory
  - 09.2 on practicalities
- Lecture 09.1 is further split into two parts:
  - Part 1: Introduction and the perceptron
  - Part 2: Multi-layer Networks
- This is Part 1, which covers:
  - Introduction
  - Neurons
  - Single layer perceptron
  - Learning algorithms

ILOs

- ILO2 Be able to use and apply basic machine learning tools
- ILO3 Be able to make and report appropriate inferences from the results of applying basic tools to data

Neurons



- Dendrites take inputs
- Axons fire on activation
- Form a **dynamical system**

Artificial Neurons

- Take a number of input signals
- Activation function transforms to output
- Output sent as input to downstream neurons
- (Typically) constructed to form a **directed system** for learning

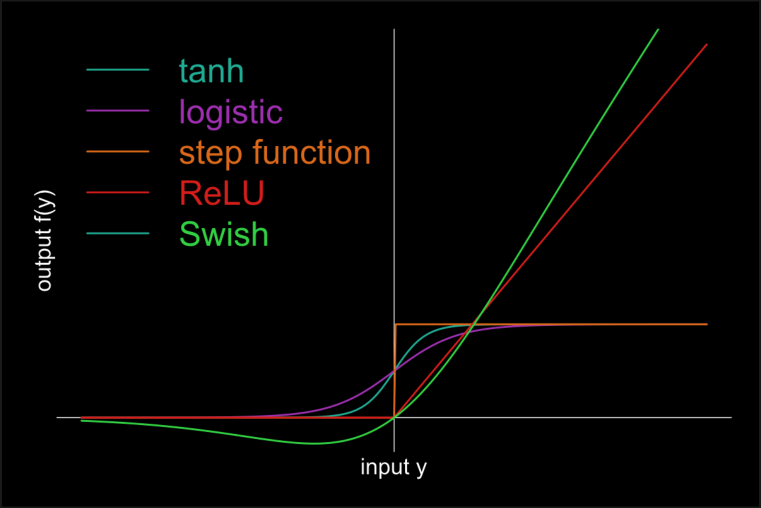
Activation functions

- Neuron i is modelled as:
  - A nonlinear **activation function** f:
  - a base rate $W_{0,i}$,
  - and weights $W_{j,i}$ for each input neuron a_j with output x_{a_j}:

$$f\left(W_{0,i} + \sum_{j=1} W_{j,i} x_{a_j}\right),$$

- f is a mapping $\mathbb{R} \to [r_{min}, r_{max}]$ (which may not be bounded).
- There are many common choices, e.g.:
  - tanh: $f(y) = (1 + \tanh(y))/2$
  - logistic: $f(y) = 1/(1 + e^{-y})$
  - Step function: $f(y) = \mathbb{I}(y > 0)$
  - Rectified linear unit (ReLU): $f(y) = \mathbb{I}(y > 0)y$

Activation functions



Activation functions

- The important features of activation functions are:
  - **Non-linearity**. A deep neural network can be trivially replicated by a one layer neural network if the activations are linear.
  - **Derivatives**. Learning requires evaluating derivatives, which should be *cheap*, and *informative*.
  - **Smoothness**. Simple discontinuities can be handled, complex ones make learning slow.
- In practice:
  - ReLU contains the important complexity whilst being very fast to learn;
  - It may exhibit convergence problems when $y << 0$;
  - For small networks, complex activation helps.
- A notable modern alternative is **Swish**[1]:
  - $f(y) = y/\left(1 + \exp(-\beta y)\right)$
  - **ReLU-like**: Converges to zero for $x \to -\infty$ and to x for $x \to \infty$
  - Has **unbounded derivative** for $x < 0$ so learning still works
  - Strangely, monotonicity seems not to be important?

[1] Ramachandran, Zoph and Le Searching for Activation Functions

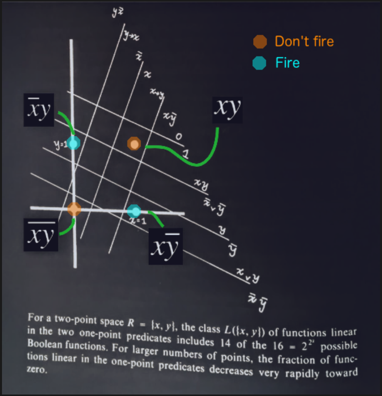
Logical functions

- Every boolean function can be implemented by a neural network[2].
- For simplicity $f(x \leq 0) = 0$, and $f(x > 0) = 1$, i.e. the neuron "fires" on activation. Then, the following can be implemented on a single node:
  - AND: $f(x_1, x_2) = -1.5 + x_1 + x_2$
  - OR: $f(x_1, x_2) = -0.5 + x_1 + x_2$
  - NOT: $f(x_1) = 0.5 - x_1$
- Neural networks with more general activation functions can still implement these functions.

[2] McCulloch and Pitts (1943) A logical calculus of the ideas immanent in nervous activity

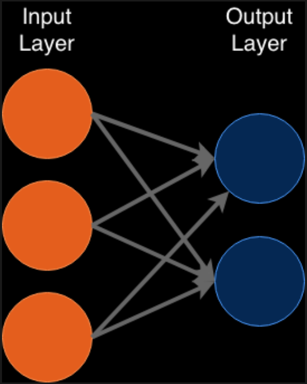
Logical function problems

- But not every function can be implemented in a single layer perceptron[3]:
  - XOR: only x_1 or x_2 can be active



[3] Minsky and Papert 1969 Perceptrons

Single Layer perceptron (SLP)



- Has just two layers:
  - data layer (e.g. features)
  - output layer (e.g. classes)
- No **hidden** layers!
- Weights learned
- Making a linear classification rule

Mathematical description of SLP

- N Inputs x_i and M outputs y_j
- Activation function f and with weights W_{ij}:

$$f(\mathbf{x}) = f\left(W_{0j} + \sum_{i=1}^{N} W_{ij} x_i\right)$$

- W_{0j} allows for an offset (mean) in the activation, just like in linear regression
- Loss is the square error over all output variables j:

$$L(W) = \sum_{j=1}^{M} L_j = \sum_{j=1}^{M} \left[y_j - f\left(W_{0j} + \sum_{i=1}^{N} W_{ij} x_i\right)\right]^2$$

$$= \sum_{j=1}^{M} \delta_{ij}^2(\mathbf{w}_j)$$

  - $\delta_{ij}(\mathbf{w}_j)$ is the error for input i output j.

Learning the SLP

- Learn through Gradient Descent:
  - i.e. Differentiate the loss with respect to the **weights** for $i = 0, \ldots, N$:

$$\nabla_W L = \left(\frac{\partial L}{\partial W_{10}}, \ldots, \frac{\partial L}{\partial W_{ij}} \ldots, \frac{\partial L}{\partial W_{NM}} \right)^T$$

  - where:

$$\frac{\partial L}{\partial W_{ij}} = \frac{\partial L}{\partial f} \frac{\partial f}{\partial W_{ij}} = -2\delta_{ij} \frac{\partial f}{\partial W_{ij}},$$

- Leading to the update rule:

$$W_{ij} \leftarrow W_{ij} + \alpha \frac{\partial f}{\partial W_{ij}} \delta_{ij}$$

  - We are taking a step of size α in a direction towards the multivariate **minima of the loss**
  - Choose step size α to take steps that move *fast enough* whilst not *overshooting*.
  - In practice α is learned adaptively.

Summary

- Neural Networks are possibly the most important development in AI.
- They are a subject of intense mathematical discussion.
- These basic building blocks are straightforward and provide intuition.
- We've only scratched the surface here.

Reflection

- What are the key similarities and differences between real and artificial neurons?
- Why are the properties of activation functions (non-linearity, smoothness, derivatives) important?
- Are perceptrons universal approximators? What implications does this have for their use?
- By the end of the course, you should:
  - Understand a neural network at a basic level
  - Be able to appropriately select deep learning methods and architecture
  - Be able to work with the mathematics underpinning perceptrons

Signposting

- Next Lecture: Part 2, getting to deep neural networks
- **References**:
- Chapter 11 of The Elements of Statistical Learning: Data Mining, Inference, and Prediction (Friedman, Hastie and Tibshirani).
- Russell and Norvig Artificial Intelligence: A Modern Approach
  - Chapter 20 Section 5: Neural Networks
- Swish: Ramachandran, Zoph and Le Searching for Activation Functions
- Important historical papers:
  - McCulloch and Pitts (1943) A logical calculus of the ideas immanent in nervous activity
  - Minsky and Papert 1969 Perceptrons