4 Neural Networks Architectures

Contents

Abstract

There exist many types of architectures used in neural networks; that is the ways in which the neurons are connected. Here we describe a few of the more popular architectures.

4.1 Feed-forward neural networks

These are the commonest type of neural network in practical applications.

- They compute a series of transformations that change the similarities between cases.
- Each unit in one layer is connected to *every* unit on the next layer.
- Each layer get a new representation of the input - in order to achieve this, the activities of the neurons in each layer are a *non-linear* function of the activities in the layer below.

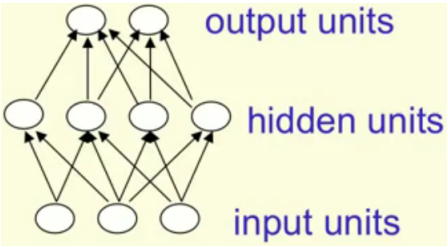

Figure 4.1: A feed-forward network with one hidden layer. The first layer is the input, and the last layer is the output.

Remark. If there is more than one hidden layer, we call them "*deep*" neural networks. This are powerful:

- Intially, each neuron in the first hidden layer is making a decision by weighing up the results from the inputs. In this way, a neuron in the second layer can make a decision at a more complex and more abstract level than neurons in the first layer. And even more complex decisions can be made by neurons in the third layer.
- In this way, a multi-layer network can engage in sophisticated decision making.

Examples include:

- convolutional neural networks (typically used in image classification).

4.2 Recurrent neural networks (RNNs)

The idea in these RNNs is to have neurons which fire for some limited duration of time, before becoming inactive. That firing can stimulate other neurons, which may fire a little while later, also for a limited duration. That causes still more neurons to fire, and so over time we get a cascade of neurons firing.

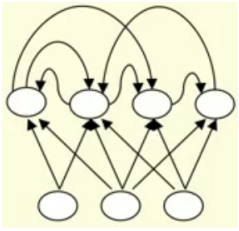

Figure 4.2: **RNNs**: They have directed cycles in their connection graph (i.e. you can sometimes get back to where you started by following the arrows).

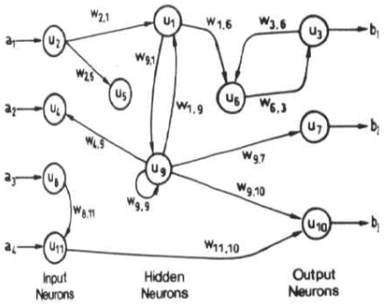

Figure 4.3: RNNs with multiple hidden layers are just a special case that has some of the hidden-to-hidden connections missing.

Remark. These are much *more powerful* and *more biologically realistic*; indeed, there is a lot of interest at present in finding efficient ways of training recurrent nets as they can have complicated dynamics that makes them very difficult to train.

4.2.1 RNNs for modelling sequences

- RNNs are a very natural way to model sequential data (e.g. good for time series).
- RNNs are equivalent to very deep nets with one hidden layer per time slice (except they use the same weights at every time slice, and they get input at every time slice).

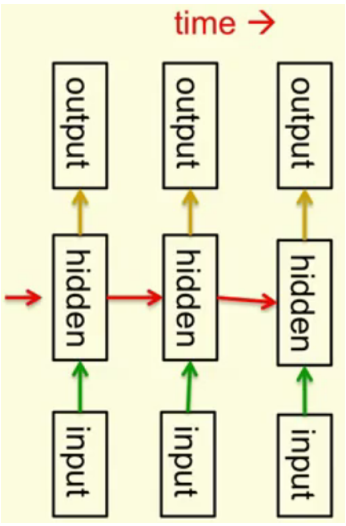

Figure 4.4: They have the ability to remember information in their hidden state for a long time (but its very hard to train them to use this potential).

4.2.2 Examples usage of RNNs[1]

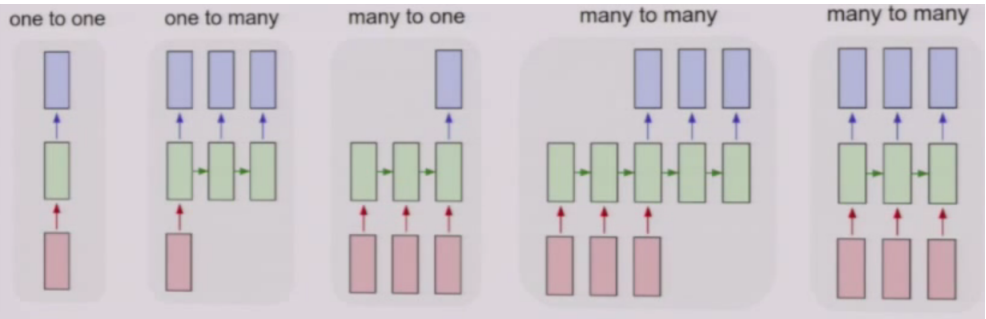

Figure 4.5: Each rectangle is a vector and arrows represent functions (e.g. matrix multiply). Input vectors are in red, output vectors are in blue and green vectors hold the RNN's state (more on this soon). **From left to right: (1)** Vanilla mode of processing without RNN, from fixed-sized input to fixed-sized output (e.g. image classification). **(2)** Sequence output (e.g. image captioning takes an image and outputs a sentence of words). **(3)** Sequence input (e.g. sentiment analysis where a given sentence is classified as expressing positive or negative sentiment). **(4)** Sequence input and sequence output (e.g. Machine Translation: an RNN reads a sentence in English and then outputs a sentence in French). **(5)** Synced sequence input and output (e.g. video classification where we wish to label each frame of the video).

4.3 Symmetrically connected neural networks

These are similar to RNNs, except the connections between units (or neurons) are symmetrical - that is they have the same weight in both directions.

- John Hopfield (and others) realised that symmetric networks are much easier to analyse than recurrent networks.
- They are also more restricted in what they can do, because they obey an *energy function* (for example, they cannot model cycles).

Definition 4.1. Symmetrically connected nets *without* hidden units are called **Hopfield nets**.

Definition 4.2. Symmetrically connected networks *with* hidden units asre called **Boltzmann machines**

Remark. Boltzmann machines are much more powerful models than Hopfield nets, but are less powerful than RNNs, and have a beautifully simple learning algorithm (these will be study later in the course).

4.4 Designing a neural network for learning[2]

- While the design of the input and output layers of a neural network is often straightforward, there can be quite an art to the design of the hidden layers.
- In particular, it is not possible to sum up the design process for the hidden layers with a few simple rules of thumb. Instead, neural networks researchers have developed many design heuristics for the hidden layers, which help people get the behaviour they want out of their nets.
- For example, such heuristics can be used to help determine how to trade off the number of hidden layers against the time required to train the network. We'll meet several such design heuristics later in the course.

[1] `http://karpathy.github.io/2015/05/21/rnn-effectiveness/`
[2] `http://neuralnetworksanddeeplearning.com/chap1.html#the_architecture_of_neural_networks`