Learning from Data: Layered Neural Networks

Amos Storkey, School of Informatics

November 6, 2006

http://www.anc.ed.ac.uk/∼amos/lfd/

Layered Neural Networks (or MLPs)

- Background
- Single Neurons
- Relationship to logistic regression.
- Multiple Neurons.
- The transfer function.
- Different output types.
- Whole Model

Neural Networks

- The field of neural networks grew up out of simple models of neurons.
- Research was done into what *networks* of these neurons could achieve.
- Neural networks proved to be a reasonable modelling tool.
- Which is funny really as they never were good models of neurons...
- or of neural networks.
- But when understood in turns of learning from data, they proved to be powerful.

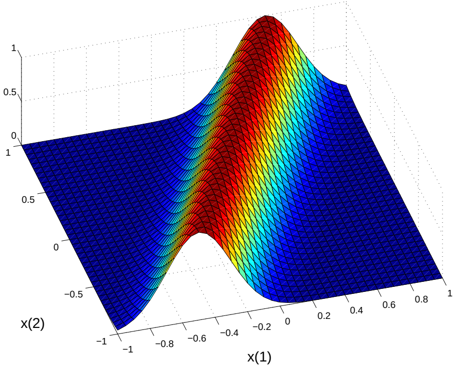
Recap: Previous Models

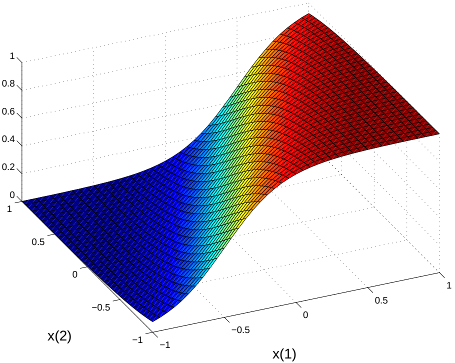
- Up to now:
$$y = f\left(\sum_j w_j \Phi_j(\mathbf{x})\right) \tag{1}$$
where f is the identity model (regression) or the logistic function (logistic regression).
- The problem: Curse of dimensionality - what are good basis functions Φ to choose?
- Extend this by adapting the Φ_j as well as w_j.

Simple Neural Model

- Input: $\mathbf{x}$
- Output: $g(a(\mathbf{x}))$ for *activation* $a(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$ and *transfer function* $g(\cdot)$.
- Most commonly $g(.)$ is logistic, but could be Gaussian shaped.
- $\mathbf{w}$ is called a *weight vector* and b is called the bias.
- These are the parameter of a neuron.
- Note if the output of a neuron is understood as a class probability, and $g(\cdot)$ is logistic, this is just a logistic regression model.
- It has all the same properties!

Single Neuron Function





A single neuron returns functions of the projected distance along some line. Left Gaussian transfer, right sigmoid transfer

Logistic Regression

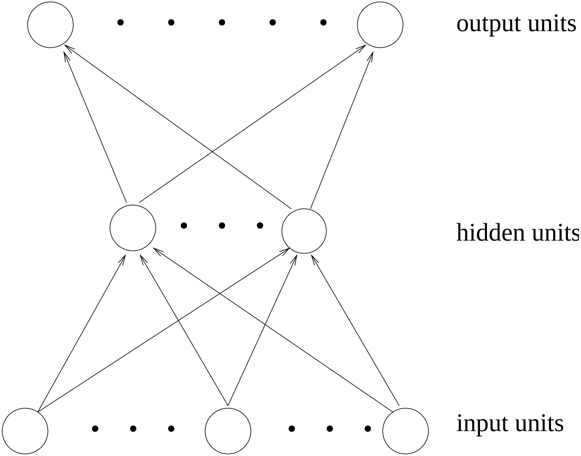
- The logistic regression model only copes with linear decision boundaries.
- We want to model more complicated systems, with nonlinear class boundaries.
- We can use features $\Phi(\mathbf{x})$ to get nonlinear decision boundaries. But what features? Ideally features should select different parts of the input space.
- But the neuron/logistic regression model is precisely a model which selects two different parts of the input space!
- Set each feature $\Phi_j(\mathbf{x})$ to be a neuron model too.

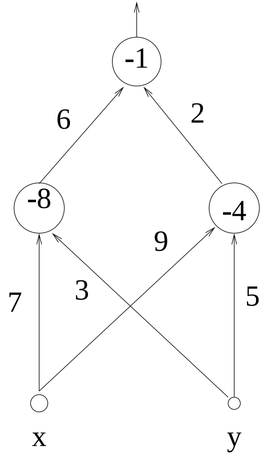
Layered Neural Models

- Suppose we have a layer of K simple neurons, all taking the same input, but producing a different output.
- The output of this first neural layer (the hidden layer) can now be viewed as a new input space.
- And if the parameters of the hidden layer neurons were chosen well, we may find that two classes we are interested in do have nearly linear decision boundaries in the output space of this hidden layer.
- Then we are away! A standard logistic regression model will solve our problems in this case.
- So we just need to put one neuron in the next layer (the output layer) to produce the final classification.

MLP architecture

Example: 1 hidden layer (bottom to top)





Output is

$$g\,(6g(7x+3y-8)+2g(9x+5y-4)-1)$$

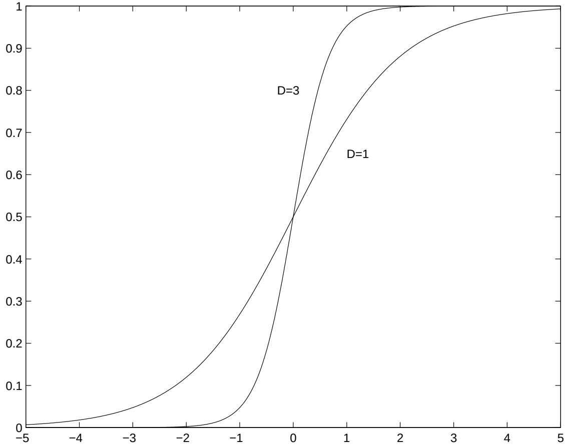
The Transfer Function

- The function used to add nonlinearity in the hidden layer need not be a logistic; it need not return values in the range $[0, 1]$.
- This function is called the *transfer function*
- Choose transfer function $g(\cdot)$ so that outputs of the net are continuous and differentiable functions of the weights.
- The logistic or sigmoid function is the most common choice (the tanh function is equivalent up to an additive or multiplicative constant).

$$g(z) = \frac{1}{1 + e^{-z}}$$

$$\tanh\left(\frac{z}{2}\right) = 2g(z) - 1$$

Logistic Function



The logistic function has a simple derivative $g'(z) = g(z)(1 - g(z))$.

Different outputs

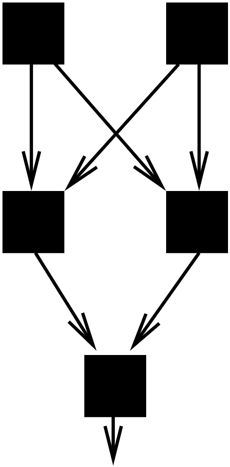
- What if we want a real valued output?
- Answer: We can set the output neuron to have a linear transfer function.
- What if we want a multivariate output?
- Answer: we can have many neurons in the output layer, all returning different variables.
- What if we wanted many classes?
- We pipe our multivariate output **y** through a logit or softmax model:

$$P(n) = \frac{\exp(y_n)}{\sum_i exp(y_i)}$$

Combining Neurons

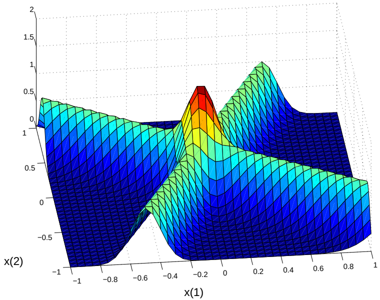
- So what happens when we combine the output of different neurons.
- Suppose we have two neurons, and we sum the output of those two neurons.
- What sort of functions can we now represent?

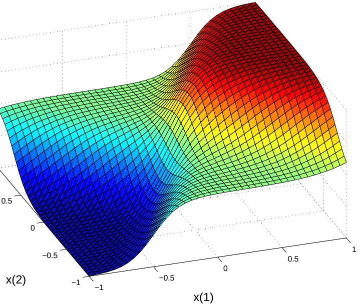
Combined Output of two Neurons



Representation of two hidden neurons

Examples





Multiple neurons return more complicated functions

The Model

- Input layer: $\mathbf{x}$.
- K Hidden layer neurons. Neuron i: $h_i(\mathbf{x}) = g(a_i(\mathbf{x}))$ for *activation* $a_i(\mathbf{x}) = \mathbf{w}_i^T\mathbf{x} + b_i$.
- Output neuron: $r(\sum_{i=1}^{K} v_i g(\mathbf{w}_i^T\mathbf{x} + b_i) + b)$.
- Typically g is logistic and r is linear/logistic.

So what are Neural Networks

- They are simply nonlinear functions with many parameters.
- There is a weight and a bias parameter for each unit.
- That's it really.
- Don't over-glamorise them.

Summary

- Neural network history
- Simple neurons and logistic regression
- What can a simple neuron do?
- Combining neurons
- Layered Neural Networks/Multilayer perceptrons