

ETC3250/5250: Neural networks 1

Semester 1, 2020

Professor Di Cook

Econometrics and Business Statistics
Monash University

Week 8 (a)

What number is this?

This is a three, right?

3

What number is this?

Is this also a three?



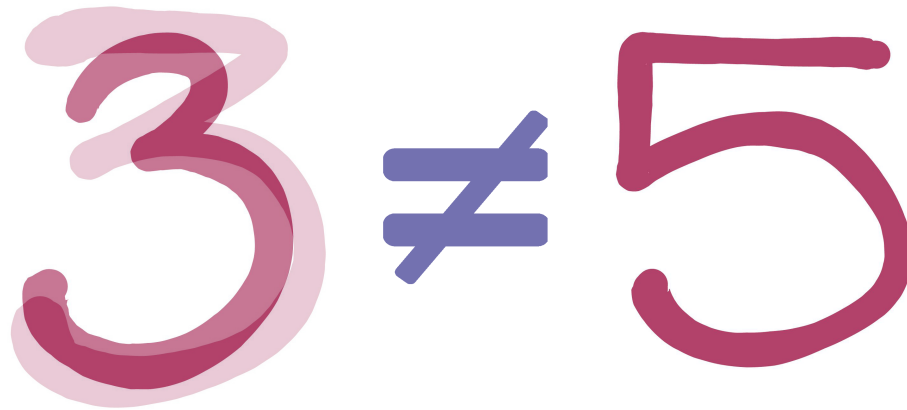
What number is this?

But what about this number? Not a three?



The human brain

The human brain can efficiently recognise that although the images of the two threes are different, they are the same number, and are both distinct from the five.



MNIST data

The MNIST data was presented to AT&T Bell Lab's to build automatic mail sorting machines.

Goal: Analyse handwritten digits and predict numbers written, given a 28×28 grid of pixels for each of the 60000 training images. Digits range from 0-9.



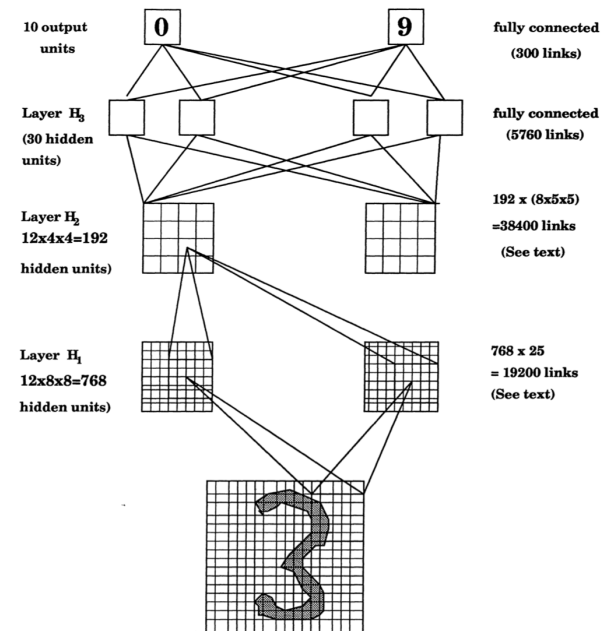
Sample images from MNIST test dataset .

MNIST data

How do we do this?

Humans are good at detecting different features about the images, such as thickness of line, angles, edges, completeness of circles, etc.

It is evident a complex relationship is presented in the images. **Neural networks** can help us automatically capture these complexities.



So, what are neural networks?

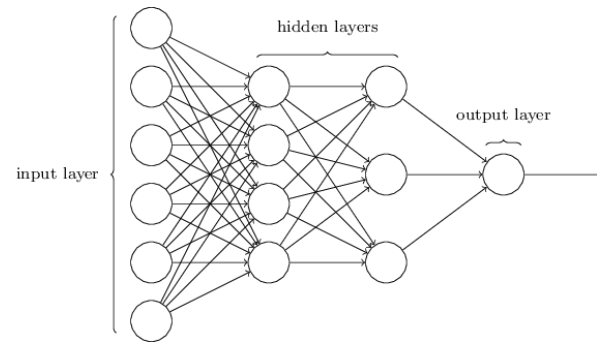
Idea: Capture a complex relationship between outputs and inputs by creating layers of derived variables.

y = output variable

x = original input variable

$f(x)$ = derived variable input

$$y = f_1(f_2(\dots(f_d(x))))$$



Source: Hands on Machine Learning with R

How do we build a neural network?

To build a feedforward neural network, we need **four key components**:


1. Input data (*in this case, the MNIST data*)
2. A pre-defined network architecture;
3. A feedback mechanism to enable the network to learn; and
4. A model training approach.




1. Preparing the data

Data preparation

There are some data cleaning steps we need to keep in mind before we use neural networks.

 Data needs input to be *numeric*. This means if our data has categorical variables, we will need to represent these as *dummy variables* (revise, Week 2!). This is also called **one-hot encoding** in ML literature.

 Neural nets are sensitive to scale of the feature values - hence they should be *standardised* first (have mean zero and unit variance).

 If response is categorical (such as "0" through "9" response in MNIST data) - needs to be recoded as binary matrix. Can use **keras** function `to_categorical()`.

Data preparation

We can obtain the MNIST data from the `dsLabs` package. Data prep follow [Hands on Machine Learning in R](#). This arose when AT&T Bell Lab's was asked to help build automatic mail-sorting machines for the USPS around 1990.

```
library(dplyr)
library(keras)

# Import MNIST training data
mnist <- dsLabs::read_mnist()
mnist_x <- mnist$train$images
mnist_y <- mnist$train$labels

# Rename columns and standardize feature values
colnames(mnist_x) <- paste0("V", 1:ncol(mnist_x))
mnist_x <- mnist_x / 255
p <- ncol(mnist_x)

# One-hot encode response
mnist_y <- to_categorical(mnist_y, 10)
```

2. Network Architecture

Network architecture

When building architecture for the neural network, we are concerned about two key features:

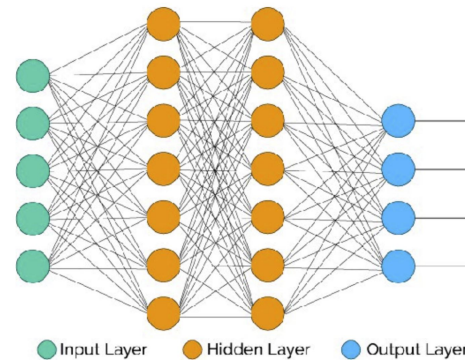
-  The number of layers and nodes, and
-  How signal is activated throughout the network.

Layers and nodes

Our complex relationships are captured using layers and nodes. There are two different types of layers, namely,

- Input and output layers, and
- Hidden layers.

Source: Gabriela de Quiroz



Hidden layers

- ▮ No well-defined approach for selecting the number of hidden layers
- this is just one of many hyperparameters we will need to tune! 2-5 layers works well most of the time for regular tabular data.
- ▮ The more hidden layers - the longer the model will take to train (as we are adding more parameters!)

Output layers

Choice of nodes for the output layer is determined by the ML task.

- 📊 If you are doing regression - a single node.
- 📊 Classification - a node for each class if multiclass.
- 📊 If binary, single node for probability of predicting success.

Think! How many output nodes will MNIST data neural network contain? 🤔

Building network structure in R

We use the **keras** package to build neural networks in R. This is *very different* to other forms of ML algorithms in R. In **keras**, we first define the network structure as a standalone from our data.

```
library(keras)
model <- keras_model_sequential() %>%
  layer_dense(units = 16,
              input_shape = p) %>%
  layer_dense(units = 16) %>%
  layer_dense(units = 10)
```

Activation - how do the layers speak?

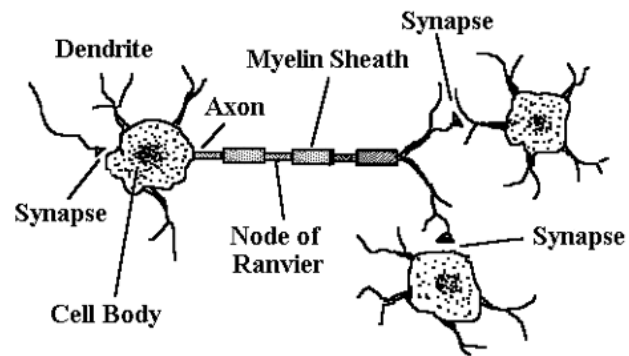
Now that we have our structure in place, we need to determine how to pass signal throughout the network.

Stay tuned! It uses methods we already know: **logistic** and **linear regression**.

But first, some history

"A logical calculus of the ideas immanent in nervous activity" (1943)
Warren S. McCulloch & Walter Pitts

Mathematical model for a neuron.



Logistic regression

Remember the logistic function:

$$\begin{aligned} y &= \frac{e^{\beta_0 + \sum_{j=1}^p \beta_j x_j}}{1 + e^{\beta_0 + \sum_{j=1}^p \beta_j x_j}} \\ &= \frac{1}{1 + e^{-(\beta_0 + \sum_{j=1}^p \beta_j x_j)}} \end{aligned}$$

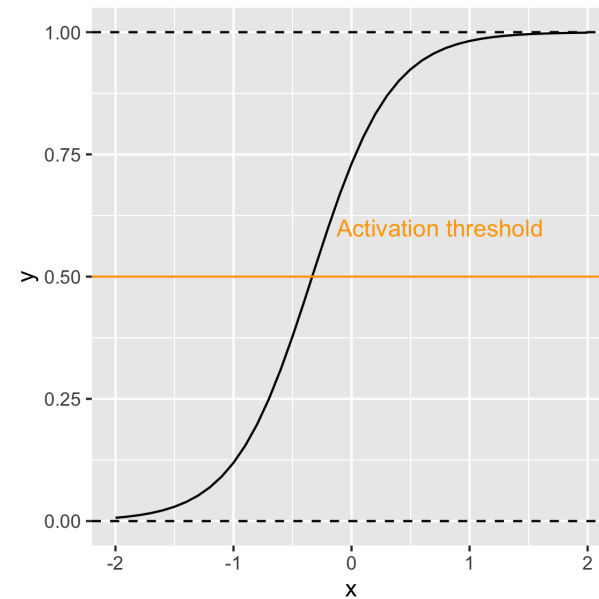
Alternatively,

$$\log_e \frac{y}{1-y} = \beta_0 + \sum_{j=1}^p \beta_j x_j$$

Logistic regression

What the **logistic function** looks like:

$$y = \frac{1}{1 + e^{-(\beta_0 + \sum_{j=1}^p \beta_j x_j)}}$$





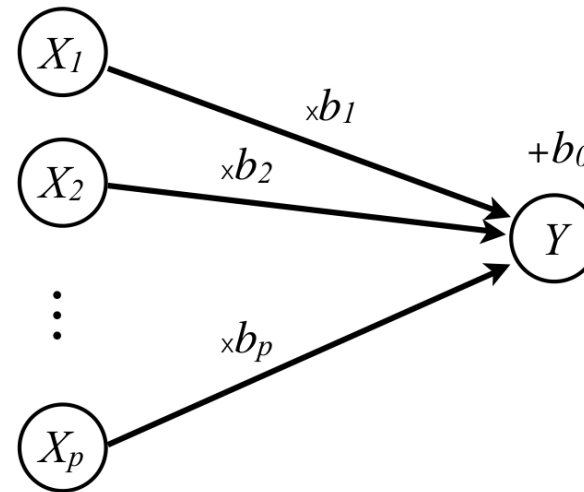
Hang on to this idea....

Linear regression as a network

$$\hat{y} = \beta_0 + \sum_{j=1}^p \beta_j x_j$$

Drawing as a network model:

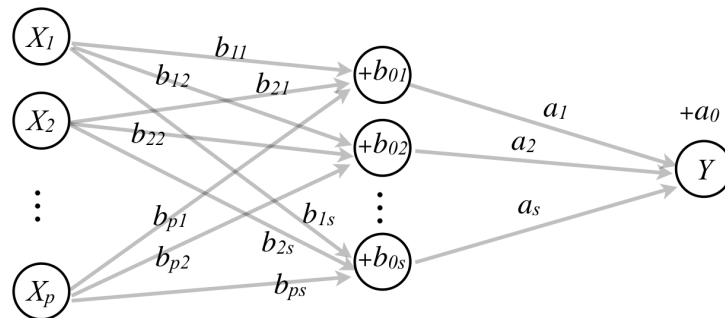
p inputs (predictors), multiplied by weights (coefficients), summed, add a constant, predicts output (response)



Network explanation - hidden layers

$$\hat{y} = \alpha_0 + \sum_{k=1}^s (\alpha_k (\beta_{j0} + \sum_{j=1}^p \beta_{jk} x_j))$$

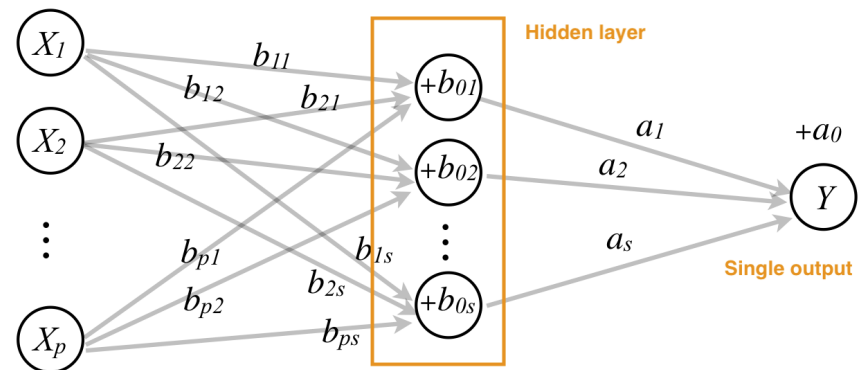
A linear regression model nested within a linear regression model allows for intrinsic dimension reduction, or expansion.



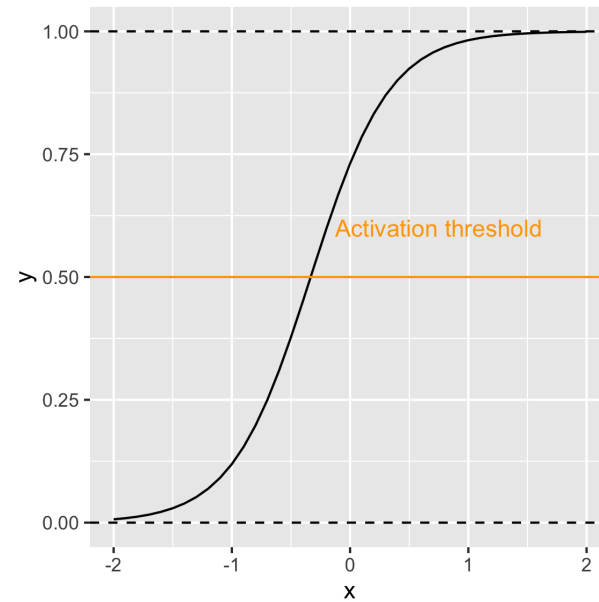
Two layer perceptron

This is a single output, 2 layer, perceptron (neural network), with a linear threshold.

$$\hat{y} = \alpha_0 + \sum_{k=1}^s (\alpha_k (\beta_{j0} + \sum_{j=1}^p \beta_{jk} x_j))$$



Back to logistic regression: When the proportion gets to 0.5, it **activates** an event to happen ($Y = 1$).



Activation functions

$$\hat{y} = g(\alpha_0 + \sum_{k=1}^s (\alpha_k f(\beta_{0k} + \sum_{j=1}^p \beta_{jk} x_j)))$$

$$\text{Let } u = \beta_0 + \sum_{j=1}^p \beta_j x_j$$

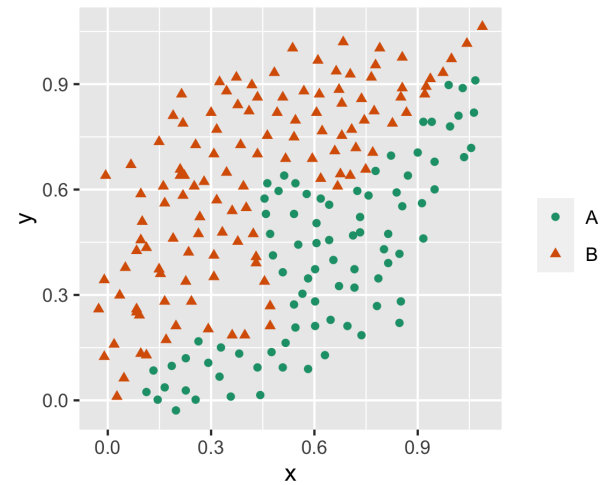
||| Logistic: $\frac{1}{1+e^{-u}}$

||| Gaussian radial: $\frac{1}{\sqrt{2\pi}} e^{-u^2/2}$

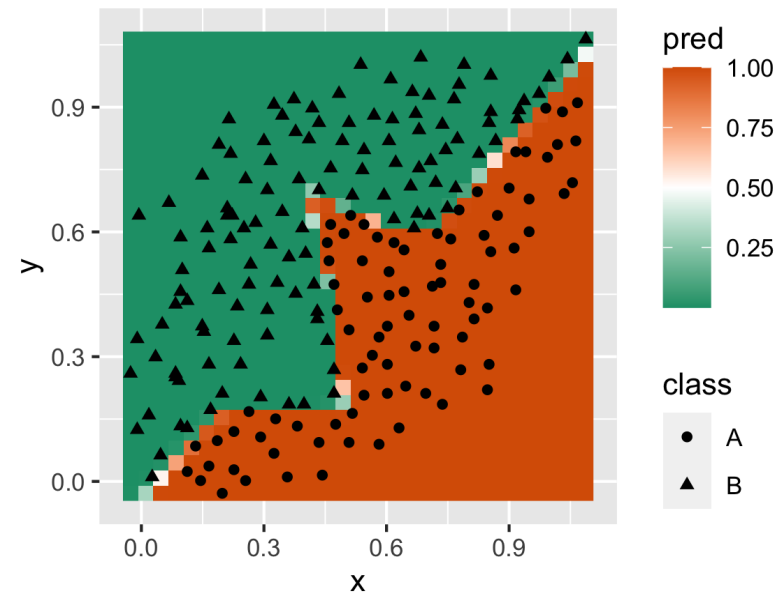
||| Hyperbolic tangent: $\frac{e^u - e^{-u}}{e^u + e^{-u}}$

Example

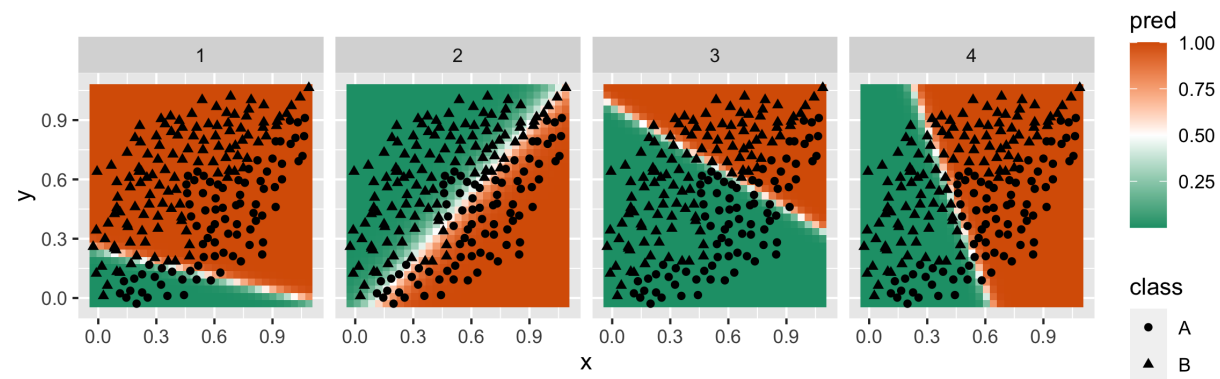
Consider the wiggly data



Overall fit is good



Individual layers



Adding in activation information in R

To add in activation information to our model in `keras`, we simply adjust our model stucture previously by adding in the activation functions we would like to use for each layer.

Note - using different activations doesn't affect what the network can learn, just the speed at which it learns.

```
model <- keras_model_sequential() %>%  
  layer_dense(units = 16,  
              activation = "relu",  
              input_shape = p) %>%  
  layer_dense(units = 16, activation = "relu") %>%  
  layer_dense(units = 10, activation = "softmax")
```


 Made by a human with a computer,
with help from Sarah Romanes

Slides at <https://iml.numbat.space>.

Code and data at <https://github.com/numbats/iml>.

Created using R Markdown with flair by [xaringan](#), and
[kunoichi](#) (female ninja) style.



This work is licensed under a Creative Commons Attribution-
ShareAlike 4.0 International License.

