# Neural Networks Tutorial 1
# COMS3007

Benjamin Rosman, Devon Jarvis

**Instructions:** Use your notes and any resources you find online to answer the following questions. You need to submit a PDF of your answers to questions 1-5 on Moodle. No coding is required for the submitted answers. This can be done in groups of *up to four*. Make sure all your names and student numbers appear on the document!

1. Consider a neural network with three input nodes, one hidden layer with two nodes, and one output node. All activation functions are sigmoids $\sigma(\cdot)$. The initial weights are as follows (including bias nodes which take the form of $x_0 = 1$):

$$\Theta^{(1)} = \begin{bmatrix} 1 & -1 & 0.5 & 1 \\ 2 & -2 & 1 & -1 \end{bmatrix}, \quad \Theta^{(2)} = \begin{bmatrix} -1 & 2 & 1 \end{bmatrix}.$$

   (a) Draw the network and label all weights.

   (b) Neural networks can be written out as functions of their input variables, as combinations of weighted sums and non-linearities. Write out this network in the form $f(x_1, x_2) = ...$

   (c) What is the network output for the following input data?

      i. $(0, 0, 0)$

      ii. $(3, 2, 1)$

      iii. $(-1, 1, -1)$

   (d) Repeat the previous question with the three different inputs for a ReLU activation function.

2. A deep *linear* neural network is a network with multiple layers but all activation functions are *linear*. Demonstrate by means of an example that a linear network with $n$ layers can compute the same function as one with $n-1$ layers. You can do this by choosing some weights for a simple network, and then computing what the weights would need to be in a shallower network to achieve the same computation.

3. As we saw in class, neural networks can be used to learn boolean functions. Design a neural network (choose the architecture, weights and activation functions) to take two inputs $x_1$ and $x_2$, and to output the XOR function, i.e. $x_1 XOR x_2$. Recall that this function has output 1 when the inputs differ. It may be useful to remember that this is the function that was shown to not be able to be modelled by a single-layer perceptron. Hint: use your notes to design the two separate pieces of the network, and then combine them.

4. Now design a neural network (choose the architecture, weights and activation functions) to take two inputs $x_1$ and $x_2$, and output a function that returns 0 when $x_1 = x_2 = 1$, and 1 otherwise (i.e. when either $x_1 = 0$, $x_2 = 0$, or they both are 0). Hint: use your notes to design the three separate pieces of the network, and then combine them.

5. A hospital manager wants to predict how many beds will be needed in the geriatric ward the next week. She asks you to design a neural network for making this prediction. She has weekly data for the last five years that cover:

   - The number of people in the geriatric ward each week.
   - The weather (average day and night temperatures).
   - The season of the year (spring, summer, autumn, winter).
   - Whether or not there was an epidemic on.

   (a) Design a suitable neural network for this problem, considering how you would set up the input data and the output data, as well as the preprocessing of the data. What kind of problem is this? How many hidden layers and nodes in each layer would you choose? Do you expect the system to work?

   (b) How would you change this network if you were required to predict one of the following outcomes: *the hospital is under full*, *the hospital is over full*, or *the hospital is at capacity*.

6. The practical component of this lab will be run over two weeks. This week we will focus on creating a neural network and how to propagate data through the network. Next week we will focus on learning the parameters of the network. We will be using the MNIST Handwritten digit data set for these two labs. In the lab handout folder I have included a file *helpers.py* with some helper functions which you may find useful.

   (a) Create a function which randomly initializes the weights and biases of a layer of the network. The function should accept the number of neurons in a layer and the number of neurons in the following layer and return a tuple containing the weight matrix and bias vector between these two layers. For example $init\_layer(10, 5)$ will output a tuple: $(weight\_matrix, bias\_vector)$ where $weight\_matrix$ is a $10 \times 5$ matrix and $bias\_vector$ is a 5-dimensional vector. Hint: np.random.rand().

   (b) Using the fuction created in Question 6a, create a function which randomly initializes the weights and biases of a neural network. The function should accept a list of numbers specifying the number of neurons in each layer of the network and output a list containing the weight and bias tuple connecting each layer. For example $init\_random\_network([100, 50, 5])$ will output $[(100 \times 50\ matrix, 50d\ vector), (50 \times 5\ matrix, 5d\ vector))]$. I have included a function in the helper file which checks that your network matches the desired format given the layer dimensions. To use it call $helpers.check\_my\_network(desired\_dimensions, initialized\_network)$. In the above example $desired\_dimensions$ will be $[100, 50, 5]$ and $initialized\_network$ will be the output of $init\_random\_network$. Use the function you created to initialize a network with layer sizes $[784, 200, 100, 10]$.

   (c) Utilize the function $helpers.load\_mnist()$ to obtain the data for the lab. This function outputs the training data, training labels, test data and test labels separately. Check that the shapes for the training data, training labels, test data and test labels arrays are $(60000, 784)$, $(60000, 10)$, $(10000, 784)$ and $(10000, 10)$ respectively.

   (d) Write a function which propagates a batch of data through a neural network. This function should accept a list of tuples specifying the network parameters (like you would obtain from the function in Question 6b) and an array of data and produce the output from the network for each data point. Use this function with the network initialized in Question 6b and the training data array obtained in Question 6c. Use a sigmoid activation function on every layer except the last. You should obtain an array of shape $(60000, 10)$ from the network.