1) The universal approximation theorem states that a feedforward neural network (NN) with a single hidden layer can approximate any function over some compact set, provided that it has enough neurons on that layer. This suggests that the number of neurons is more important than the number of layers. But in practice deep learning is obviously very successful at various prediction tasks. Why is that? Shouldn't all deep NNs be equivalent to single layered NNs with enough neurons? Why do we need depth when we could theoretically rewrite that neural network with a single layer?

Although you can approximate the function, there will be many possibilities. It is harder for your model to learn it. Deep neural networks are good at learning different features because the inner layers can spot them. Each layer ends up learning different features based on the previous layer. With one shallow and big network, you will have to learn those features directly from the input, which is possible but very hard.

2) In neural networks, what is the role of the activation function and why do we need it?

The activation function acts a nonlinear function and helps us avoid the linearities. Without it the neural net can be simplified as one layer of the size of the biggest layer in the network.

3) For a softmax of the values (3,4,1,7) what is a possible result?

   a) 0.0171, 0.0465, 0.0023, 0.9341
   b) 0.0011, 0.0085, 0.0003, 0.8362
   c) 0.0023, 0.0171, 0.0465, 0.9341
   d) 0.0211, 0.0785, 0.0103, 0.9362

   Notice that other options do not add up to one or have values that do not correspond to the magnitudes of the inputs.

4) Let’s say you have three problems—a linear regression, a logistic regression, and a small neural net. Which one of them is more likely to benefit from a newly discovered super-fast large matrix multiplication algorithm? Why?

   Linear regression because fast matrix multiply help computing inverse in Normal Eqs

5) Consider the dataset: D = \{x^{(1)}, \ldots, x^{(100)}\} with x in \mathbb{R}^3. The problem is a 3-class classification problem. Consider a neural network architecture with 2 hidden layers of dimension 4 and 5 - using a sigmoid and a softmax respectively.

   a) How would you graphically represent this neural network?
b) What are the feedforward equations for this neural network?

For a single training example $x_i$ in $\mathbb{R}^3$ so $x_i$ is $(x_{i_1}, x_{i_2}, x_{i_3})^T$.

\[
\begin{align*}
  z_1 &= x^T W_1 + 1_{100 \times 1} b_1 \quad (1) \quad W_1: (3, 4) \quad b_1: (1, 4) \\
  a_1^T &= g(z_1) \quad (2) \\
  z_2 &= a_1^T W_2 + 1_{100 \times 1} b_2 \quad (3) \quad W_2: (4, 5) \quad b_2: (1, 5) \\
  a_2^T &= g(z_2) \quad (4) \\
  f &= \text{softmax}(a_2^T) \quad (5) \quad \text{Softmax: 3 vectors of 5 + 1 weights (bias included)}
\end{align*}
\]

c) Describe the relationship between the graphical representation and the feedforward equations. What do the nodes represent? What do the edges represent?

- 1 node = 1 component of a vector (x or activation)
- 1 edge = 1 vector component * 1 matrix weight
  - Edges that point to same node mean that the scalars they represent are summed together to obtain the value of the node

d) What is the total number of parameters in this neural network?

Number of weights per equation:
(1) $4 \times 3 + 4 = 16$
(2) 0
(3) $5 \times 4 + 5 = 25$
(4) 0
(5) $3 \times 5 + 3 = 18$
Total = 59 parameters