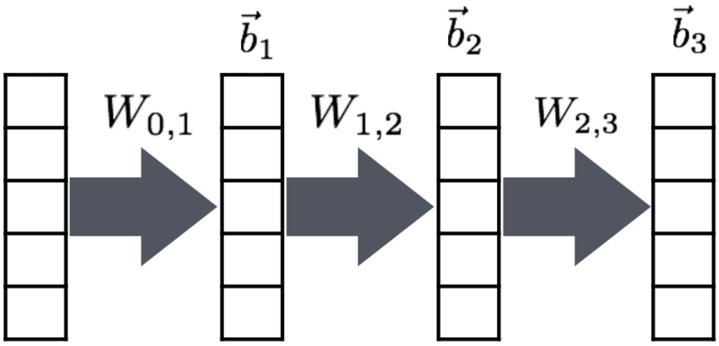
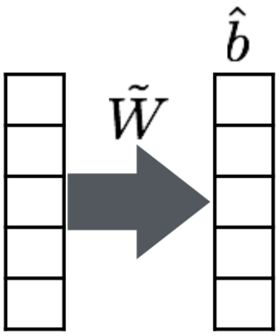
Two networks are said to be equivalent if, they have the same number of input and output nodes, for all inputs, the output of both networks are identical.

Given two neural networks as shown below, all activation functions are identity functions,

$$z = \sigma(z)$$

If the weights and biases are given in the first network with two hidden layers, how do you adjust the weights and biases of the second network with no hidden layers to make the two network equivalent. Provide closed form solution, using mathematical symbols.

Latex or MSWord your solutions and equations for submission in a pdf file.

Network 1	Network 2
 $\vec{b}_1$ $\vec{b}_2$ $\vec{b}_3$ $W_{0,1}$ $W_{1,2}$ $W_{2,3}$	

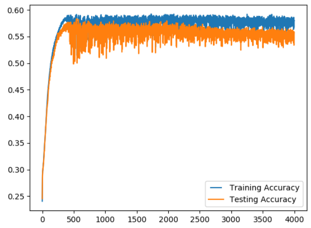
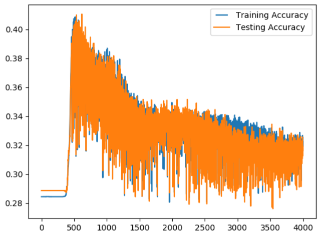
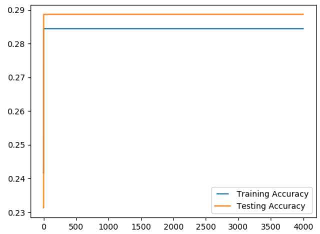
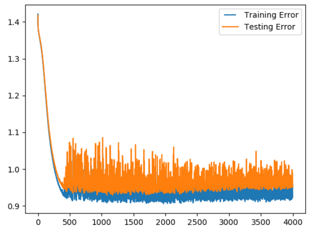
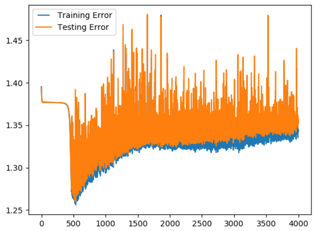
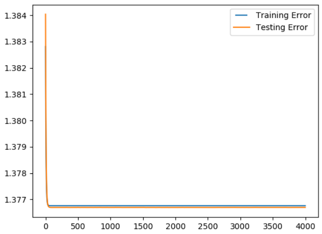
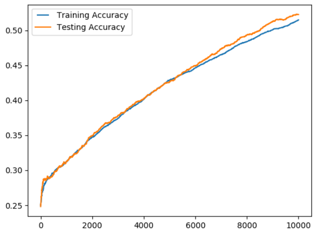
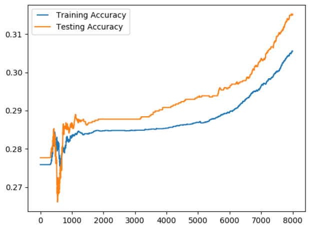
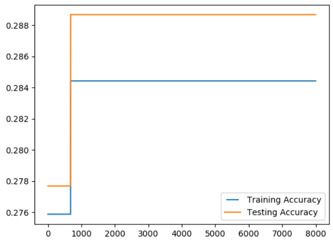
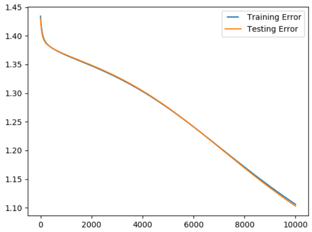
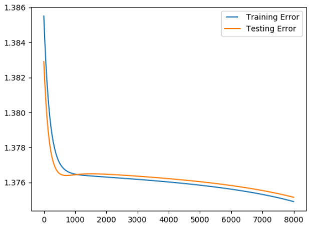
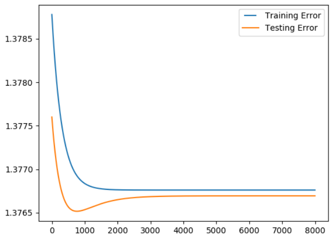
Target format: $O = \widetilde{W}x + \hat{b}$

$$h_1 = \widetilde{W}_{0,1}x + \vec{b}_1$$
$$h_2 = \widetilde{W}_{1,2}h_1 + \vec{b}_2$$
$$O = \widetilde{W}_{2,3}h_2 + \vec{b}_3$$

$$O = \widetilde{W}_{2,3}\left(\widetilde{W}_{1,2}\left(\widetilde{W}_{0,1}x + \vec{b}_1\right) + \vec{b}_2\right) + \vec{b}_3$$
$$O = \widetilde{W}_{2,3}(\widetilde{W}_{1,2}\widetilde{W}_{0,1}x + \widetilde{W}_{1,2}\vec{b}_1 + \vec{b}_2) + \vec{b}_3$$
$$O = \widetilde{W}_{2,3}\widetilde{W}_{1,2}\widetilde{W}_{0,1}x + \widetilde{W}_{2,3}\widetilde{W}_{1,2}\vec{b}_1 + \widetilde{W}_{2,3}\vec{b}_2 + \vec{b}_3$$

$$\widetilde{W} = \widetilde{W}_{2,3}\widetilde{W}_{1,2}\widetilde{W}_{0,1}$$
$$\hat{b} = \widetilde{W}_{2,3}\widetilde{W}_{1,2}\vec{b}_1 + \widetilde{W}_{2,3}\vec{b}_2 + \vec{b}_3$$

Question 2 (abc)

14-100-40-4 (0.025 epsilon)	14-28x6-4 (0.025 epsilon)	14-14x28-4 (0.025 epsilon)
		
		
14-100-40-4 (0.001 epsilon)	**14-28x6-4 (0.001 epsilon)**	**14-14x28-4 (0.001 epsilon)**
		
		

Give a half page discussion on why the three networks 14-100-40-4 net,14-28x6-4 net,14-14x28-4 net perform differently. Which one performs better and why.

I will refer to the nets 14-100-40-4, 14-28x6-4 and 14-14x28-4 nets A, B and C respectively.

In terms of training, Net A is the fastest and easiest to train. In terms of performance, Net B should have the best performance in the long run. Net C is the most difficult to train and has the worst performance of the lot, but should eventually solve the problem as well.

Ignoring the runtime, Net A is the easiest to train simply because it provides a much higher dimensionality in the second layer than the input dimensionality. This gives the network space to move around and find relationships between the input nodes. The weights can converge much faster to some local minima. Net C on the other hand does not provide the increased dimensionality. If the relationship between the inputs and outputs are non-linear, it will have a much harder time finding the relationship across hidden layers. It is easier for relationships to be found when projected into a higher dimension. Net B at least provides higher dimensions than the input, but not a lot more. Hence it still takes a longer time to train the dataset.

Because the nets with more hidden layers are more sensitive to changes, they also train better with a lower learning rate. If the learning rate is too high, it will become impossible to converge to a local minima.

In the results, we can see that there are instances where the training rate is too high, as can be seen in Nets A and B. In fact, in Net B, it starts to move away from the local minimum and get worse over time. This can usually be solved by decreasing the training rate over time. In the end, I could not fix my neural net so I did not manage to get the correct results.