Problem 1

	Linear Regression	1-Layer NN	2-Layer NN
train	0.540372939483	0.54219662568	0.486512954545
test	0.659845351796	0.664220754302	0.603092361843

We could see that the performance of **Linear Regression** model and **1-Layer NN** model is similar because a neural network with just input and output layers can only simulate a linear function. Thus with enough training it should behave just like a linear regression model. The advantage of using a 1-layer NN instead of a linear regression model is that when matrix cannot be inverted the NN model will give a relatively reasonable result.

Additionally, the **2-Layer NN** model behaves significantly better than previous two since it is capable of simulate non-linear functions due to the existences of activation functions.

Problem 2

learning rate	epochs	hidden size
0.01	20	5

For **learning rate**, I've tried 0.1 and 0.001 but they are either too large and jumping around some optimal point, or too small to reach the local minimum. And it takes about **15** epochs for the model to find the minimum loss so set **epochs** to 20 is safe. The 2-layer NN model takes a long time to generate results and thus I tried to reduce the **hidden layer size** to 5. It doesn't affect the loss and significantly reduces execution time.

Problem 3

	linear	step	relu	sigmoid
train	0.671182979131	0.757162306217	0.709770368808	0.486512954545
test	0.79755382189	0.890642643561	0.816859165173	0.603092361843

The other three activation functions behave worse than *sigmoid* function. A *linear* activation function is not really a activation function because the network is still linear. And *step* function is too discrete thus many information is lost. *relu* is the combination of them and its loss value gets in the middle.