Comprehension Questions for Lectures 1 – 3

1. How Can Linear Regression be modified to fit nonlinear curves? (Recall that the equation for Linear Regression for two dimensions is $y = \theta_0 + \theta_1 x$).
   Let $y = \theta_0 + \theta_1 x + \theta_2 x^2$. Then gradient descent would fit a parabola to the data. We could also have more complicated terms, such as $\theta_n \sin \theta_{n+1} x$ or $\theta_m x^{1/2}$

2. How Can Logistic Regression be modified to fit nonlinear decision boundaries? (Recall that the equation for Logistic Regression with two features is $y = 1/1+e^{-z}$ where $z = \theta_0 + \theta_1 x_1 + \theta_2 x_2$).
   Let $z = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \ldots$ would allow for the fitting of nonlinear curves.

3. Can the modifies regression algorithm in #1 fit closed curves (e.g. an ellipse) to the data? No. The first is by using the equation. There is no way that I can write an equation for an ellipse or circle without writing y on the right hand side of the equation, but y is a prediction, not a feature. Thus, there is no way to fit a closed curve to the data.

4. Can the modifies regression algorithm in #1 fit closed decision boundaries (e.g. an ellipse) to the data?
   Let $z = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \theta_4 x_1^2 + \theta_5 x_2^2$ would fit an ellipse on the $x_1 x_2$ plane (note that our y axis would come out of this plane.

5. What is feature extraction?
   Feature extraction is the use of building high level features that have a stronger correlation to our test labels from more basic/primitive features.

6. What are the problems with training Logistic Regression on the MNIST dataset?
   Logistic Regression is not invariant under translation, but most solutions for image recognition problems should be. (e.g. A 9 is a 9 regardless of where I write it on this page).

7. Why do we use ReLU activation units instead of Linear activation units for the hidden layers?
   Let H1, H2, and H3 be three layers in a neural network with n, m, and l neuros respectively. If we use linear activation units, the weights from H1 to H2 can be represented by an m x n matrix that we shall denote as A_{mxn} with corresponding linear transform T. Similarly, the weights from H2 to H3 can be represented by an l x m matrix that we shall denote as B_{lxn} with corresponding linear transform U. We know from linear algebra that $AB = C_{mxl}$, where C is another matrix. C will have a corresponding linear transform $S = U \circ T$. This makes H2 unnecessary as we can directly map H1 to H3 with the set of weights contained in C. Thus, in order to make hidden layers useful, we must use an activation function that is not linear.

   Challenge Question: What effectively happens if we train a neural network with all neurons being linear? (i.e. what would this be equivalent to?) What if we change the output layer to sigmoid?