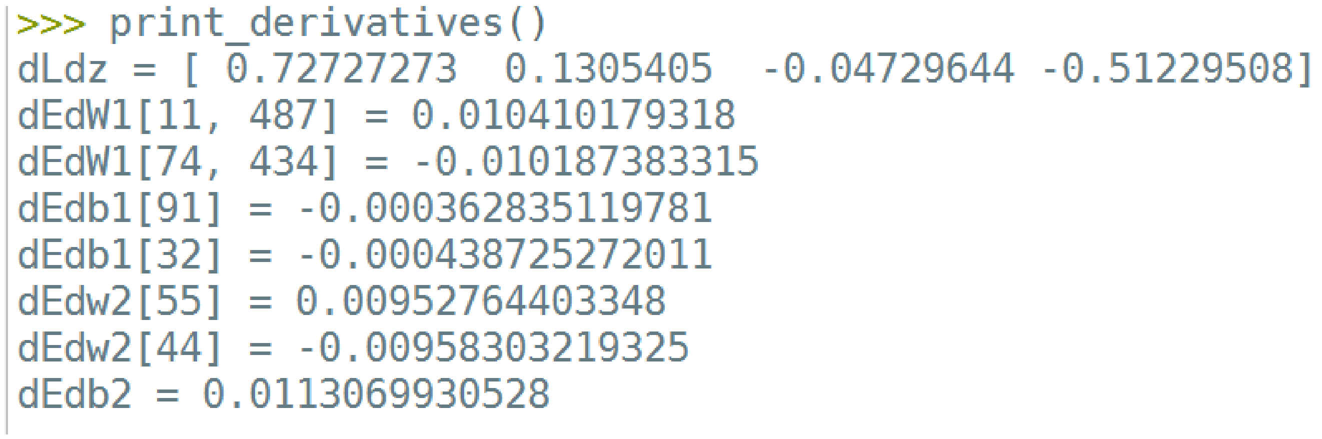
Part 1



```
>>> print_derivatives()
dLdz = [ 0.72727273  0.1305405  -0.04729644 -0.51229508]
dEdW1[11, 487] = 0.010410179318
dEdW1[74, 434] = -0.010187383315
dEdb1[91] = -0.000362835119781
dEdb1[32] = -0.000438725272011
dEdw2[55] = 0.00952764403348
dEdw2[44] = -0.00958303219325
dEdb2 = 0.0113069930528
```

Figure 1: Output of model.print_derivatives() function.

Part 2.1

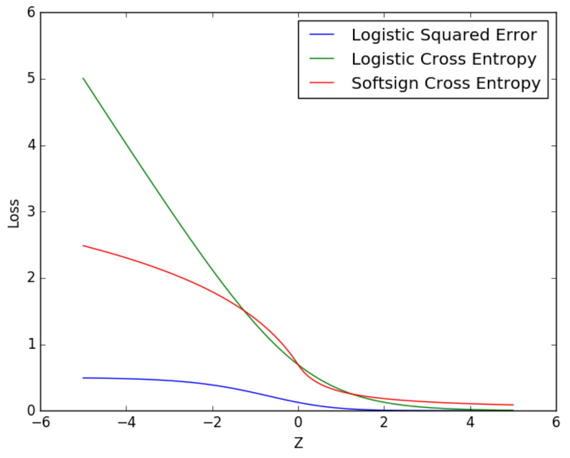
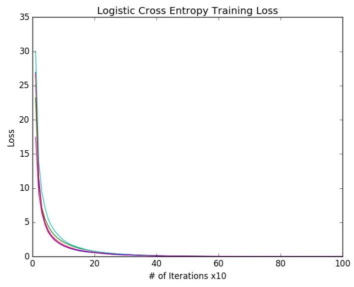


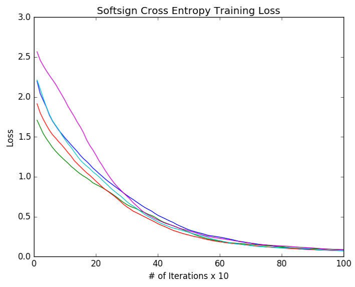
Figure 2: Activation-Function-Loss Functions as functions of z with $t = 1$

From Figure 2, it seems as though the Logistic Cross Entropy Loss function would cause the training loss to decrease the fastest given a very bad initialization. For $z \leq -1$, the slopes of the Logistic Cross Entropy function is much steeper than the squared error lost.Again by looking at the plot, we can see the squared error loss would cause a bad initialization to decrease the slowest.

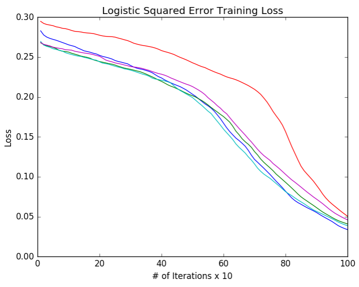
Figure 3 shows the results of training MLPS using the three loss functions. Each model was run 6 times each. Figure 3.a represents training the model using Logistic Cross Entropy. By around 200 iterations, the training loss has already been reduced to near zero whereas in Figure 3.c shows some training loss by the end of the training. The results of the training support the prediction.



(a) Logistic Cross Entropy Model



(b) Softsign Cross Entropy Model



(c) Logistic Squared Error Model

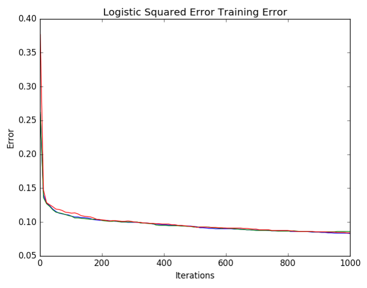
Figure 3: Training Loss of the three models

Part 2.2

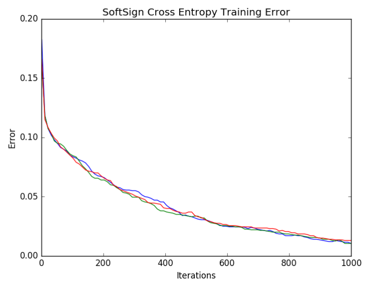
Referring to Figure 2, the Logistic Squared Error model has no distinction between extremely wrong classification and slightly wrong classifications (the plateau from $z \leq -2$), leading to believe that extremely bad classification will not have any effect on the training. Similarly, the Logistic Cross Entropy model has greater gradients for wrong predictions, leading to lower training error Therefore, the training error will be the greatest with Logistic Squared Error and the smallest with Logistic Cross Entropy.

For Validation Loss, usually a model that over fit the training data will perform poorly on the validation set, and have greater validation loss. In other words, the model that best generalizes the data and isn't greatly affected by outliers would prove to have a lower validation cost. Therefore, the logistic squared error would have the lower validation cost.

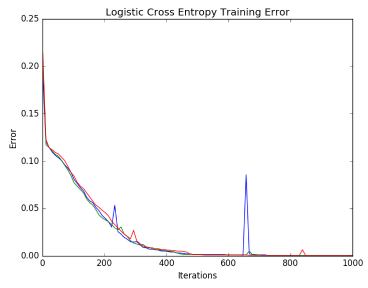
Figures 3 and 4 show the training error and validation loss during the training of each of the models with noise. Each model was run 3 times. We can see that the predictions aligned well with the results. Note the different axis scales.



(a) Logistic Cross Entropy Model

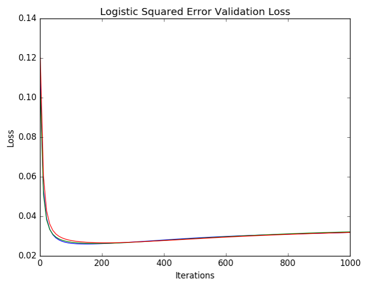


(b) Softsign Cross Entropy Model

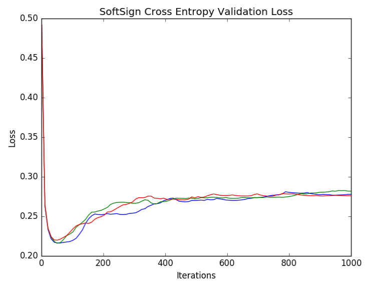


(c) Logistic Squared Error Model

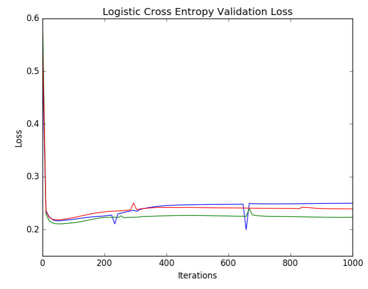
Figure 4: Training Error of the three models



(a) Logistic Cross Entropy Model



(b) Softsign Cross Entropy Model



(c) Logistic Squared Error Model

Figure 5: Validation Loss of the three models

Part 2.3

For getting saturated units unstuck, it would be better to use the softsign nonlinearity. Equations 1 and 2 represent the the derivatives for the logistic and softsign nonlinearity respectively. As $z \to \infty$, (1) approaches zero much quicker than (2). In otherwords, for large magnitudes of the input, the softsign nonlinearity will have a larger gradient and therefore a stronger gradient signal.

$$\frac{\delta \mathcal{L}_{log}}{\delta z} = \frac{e^{-z}}{(1+e^{-z})^2} \tag{1}$$

$$\frac{\delta \mathcal{L}_{soft}}{\delta z} = \frac{1}{(1+|z|)^2} \tag{2}$$