Neural Net HW - Revised on 2/22/17

Problem 2 (Scaling) has been rewritten and clarified.

1. FeedFoward (5 points)

Below is the output from nnet after we fit a model. Let's assume we used a tanh() activation function throughout. Let $x_i, i = 1, 2, \dots$ be the input variables and let \$h_1, h_2, \dots \$ be the output from the hidden layer.

```
a 2-2-1 network with 9 weights
options were - linear output units
 b->h1 i1->h1 i2->h1
  1.27   0.23  -0.34
 b->h2 i1->h2 i2->h2
-32.11  21.98 -48.67
  b->o  h1->o  h2->o
-28.67  43.01   1.37
```

(a) Draw a diagram of this neural network architecture. Label all the edges with the corresponding weights.

(b) Provide an expression for the output value of the first hidden unit as a function of the values of the input features. *This should have the form $h_1 = f(x_1, x_2, \dots)$ for a suitable explicit function f.*

(c) Provide an expression for the value at the output node as a function of the values at the hidden units. *This should have the form $z = g(h_1, h_2, \dots)$ for a suitable explicit function g.*

(d) Provide an expression for the value at the output node as a function of the input values. *This should have the form $z = F(x_1, x_2, \dots)$ for a suitable explicit function F.*

2. Scaling (5 points)

This problem uses the Advertising data.

```
library(nnet)
set.seed(1234)
Advertising <- read.csv("../Data/Advertising.csv")
```

(a) A linear regression Sales ~ TV + Radio + Newspaper can be fitted using either R's lm() function or with a neural net without hidden layer and with linear output, using the code below.

Modify this code as you see fit.

Always set the seed to 1234 before running nnet().

```
lm_fit <-lm(Sales ~ TV + Radio + Newspaper, data=Advertising)
set.seed(1234)
linear_fit <- nnet(Sales ~ TV + Radio + Newspaper,
                   size = 0,  data=Advertising, skip = T,
                   trace = F,maxit = 1000, linout = T)
```

Demonstrate that the lm() approach and the nnet() approach essentially give the same model (e.g. the same coefficients, the same predictions, the same residuals, etc.).

(b) What is the sum of squared errors (SSE) of this linear model?

(c) Now add a hidden layer with 3 units. **Before running nnet(), set the seed to 1234.** Then fit a model using nnet(), with **maxit = 1000** and **decay** at the default value. You'll need to keep the argument `linout=TRUE` so that nnet does regression and not classification. What is the SSE once the method has converged?

*Because we fit neural nets with backpropogation, the iterative optimization algorithm can be sensitive to random initial starting guesses and can get stuck in local optima. One way the nnet library helps us to avoid these pitfalls is by **rescaling** the feature space first. To rescale, one calculates the mean and variance of each feature and converts each measurement into a feature-specific z score, using the formula*

$$z_{ij} = \frac{x_{ij} - \mu_j}{\sigma_j}$$

where x_{ij} is the entry in position (i, j) of a data frame or matrix, μ_j is the mean of column j, and σ_j is the standard deviation of column j. This is generally useful if the different features are measured on entirely different units or scales.

(d) Use the `scale()` method to create a new dataframe **Advert2** from the Advertising data with the code below. We replace the changed **Sales** column with the original one, since we do not want to change the responses.

```
Advert2 <- as.data.frame(scale(Advertising))
Advert2$Sales <- Advertising$Sales
```

Use **head()** to inspect the new dataframe and describe what has changed.

(e) Make a linear model that fits Sales to the scaled predictors in **Advert2**, using nnet() as in (a). Demonstrate that this linear model has the exact same responses as the linear model from (a). Then make a neural network that fits Sales to the scaled predictors in **Advert2**, again with a 3 unit hidden layer. **Before running nnet(), set the seed to 1234.**

(f) Compute the SSE for the neural network with 3 units that was made in (e) and compare it to the SSE of the linear model that was made in (a) and to the SSE of the neural network mdel that was made in (c). Explain what has happened.

3. MNIST Revisited (10 Points)

We'll use the MNIST image classification data (0 vs. 1) that we looked at in the last homework. The data (**mnist_data.RData**) are in Canvas.

(a) Pick two features that have some non-zero variability. Fit a logistic regression model with these two features. Evaluate the model with the AUC score.

(b) Create a neural net with one unit in the hidden layer. Train the neural net with the same two features as the previous part and evaluate the model with AUC.

(c) With the same two features, train a neural net using more units in the hidden layer. How do your results compare?

(d) Finally, make several neural nets with at least four features and as many hidden units as you desire. How do the results compare?

4. Overfitting (5 Points)

Make a dataframe with $k = 11$ columns and $N = 100$ observations, where all entries are independent standard normal random variables. Let z be the last column. Use **set.seed(20305)**.

(a) Fit z to the other 10 columns using multiple regression. What is the sum of squares of the residuals?

(b) Fit z to the other 10 columns, using a neural network with two hidden units and setting $maxit = 2000$ and $decay = .01$. Be sure to set **linout = T** since this is a regression problem. Does this model fit the data better? How do you know?

(c) Redo this experiment with the same data and with 5 and 10 hidden units and explain what you see.