# COMP 652 - Homework 2

Assigned: Sep 22, 2008
Due: Sep 29, 2008
Late: Oct 1, 2008

**1. Gradient of cross-entropy for logistic function. (10 pts.)**

Recall from class that the cross-entropy error function is

$$J_{CE} = -\sum_{i=1}^{m} y_i \log h(\mathbf{x_i}) + (1 - y_i) \log(1 - h(\mathbf{x_i}))$$

This assumes a binary-classification data set $D = \{(\mathbf{x_i}, y_i)\}$, where $\mathbf{x_i}$ is the input vector for the $i^{th}$ instance, $y_i \in \{0, 1\}$ is the target, and $h$ is a hypothesis that outputs the probability that $y_i = 1$ given $\mathbf{x_i}$. Suppose $h$ is the logistic function:

$$h_{\mathbf{w}}(\mathbf{x_i}) = \frac{1}{1 + e^{-\sum_{j=0}^{n} w_j x_{i,j}}}$$

where $x_{i,0} = 1$, allowing for a constant offset in the summation.

**A. (3 pts.)** Derive a formula for the gradient of the cross-entropy error function with respect to the weights, $\nabla_{\mathbf{w}} J_{CE}$.

**B. (4 pts.)** Consider the data files "wpbc_x_normalized.txt" and "wpbc_yrecur.txt". The first file contains the input data for the Wisconsin Breast Cancer problem, with each row corresponding to an instance (i.e., patient) and each column corresponding to a different feature measured about the patient. The columns of this data matrix have been normalized to have mean zero and standard deviation one. (This can be a helpful trick to use when using gradient descent or similar numerical optimization approaches.) The second file contains 0 or 1 on each line, corresponding to non-recurrence or recurrence of the cancer. Using your results from part A, implement a gradient descent approach that fits a logistic classifier that uses just the first feature and a constant offset. (This first feature is the "nucleus size", which we have seen on several plots in class.) The classifier should thus have the form $h_{\mathbf{w}}(x) = \sigma(w_0 + w_1 x)$. Use your gradient descent code to find the weights $w_0$ and $w_1$ that minimize (as nearly as possible) the cross-entropy error function. Report these weights, report the resulting cross-entropy, and plot the output of the function, $h_{\mathbf{w}}(x)$ as a function of $x$, along with the 0/1 non-/recurrence data. Also, turn in your code.

**C. (3 pts.)** Use your gradient descent approach to fit another logistic classifier, this time using all the features in the input data matrix, plus a constant offset. Report the resulting weights and cross-entropy, and turn in your code. Is this a "better" classifier than the one from part B? Comment. *Hint: For this part especially you may need to be careful about how you choose your initial weights and step-size. I recommend starting the weights at zero. For the step-size, you may use a constant, but experiment until you find a good choice. Too large will lead to oscillations or divergence. Too small will require you to do very many steps until arriving at a satisfaction solution. Experiment until you find a good choice.*

**2. Alternative error function for logistic regression. (10 pts.)**

Consider a logistic regression problem with data set $D = \{(\mathbf{x_i}, y_i)\}$, such that the targets $y_i$ are real values strictly between 0 and 1. One could solve this problem by numerically finding $\mathbf{w}$ that minimizes the typical squared error function $J_{SSQ} = \sum_{i=1}^{m}(y_i - \sigma(\mathbf{w}^T\mathbf{x_i}))^2$. However, consider the following line of reasoning. We want to find weights $\mathbf{w}$ such that for all instances $i$ the we have approximate equality between the target and the hypothesis output: $y_i \approx \sigma(\mathbf{w}^T\mathbf{x_i})$. We can rewrite this as $\sigma^{-1}(y_i) \approx \mathbf{w}^T\mathbf{x_i}$, where $\sigma^{-1}$ is the inverse of the logistic function. We could, then, seek $\mathbf{w}$ that minimizes the alternative squared error function $J_{SSQ2} = \sum_{i=1}^{m}(\sigma^{-1}(y_i) - \mathbf{w}^T\mathbf{x_i})^2$.

**A. (5 pts.)** Does minimizing $J_{SSQ2}$ correspond to maximizing the likelihood of the data under some noise model? If not, explain why not. If so, state the noise model and prove the correspondence of the minimizing $J_{SSQ2}$ and maximizing likelihood under that noise model.

**B. (5 pts.)** Describe a good method for minimizing $J_{SSQ2}$ with respect to $\mathbf{w}$.

**3. Backprop. (10 pts.)**

**A. (4 pts.)** Implement the error backpropagation algorithm (backprop) for computing the gradient of the SSQ of a data set with respect to the weights of an artificial neural network (ANN). You can assume an ANN with a single hidden layer. However, your code should allow for any number of input units and any number of hidden layer units. Choosing initial weights and step-size appropriately, use your code to train a network that approximates the two-input XOR function. (You should create the training data set yourself.) Report the weights obtained, and the output value for each of the four input combinations (00, 01, 10 and 11). Also, turn in your code.

**B. (4 pts.)** As mentioned in class, an ANN with a single hidden layer can approximate any Boolean function, though the number of hidden-layer units required may be exponential in the number of input-layer units. In part B, you are to train a network to approximate the three-input XOR function, also called the "parity" or "odd" function. This function is 1 when the sum of the inputs is odd and 0 when the sum of the inputs is even. Because you may not know ahead of time how many hidden units are required to approximate this function, you should simply try out different choices. Report the minimum number of hidden units required to approximate the function well–meaning, for each input combination, the output is within, say, 0.01 of the correct value. Report the weights of the network, the output value for each of the eight input combinations (000, 001, ..., 111), and turn in your code.

**C. (2 pts.)** Do the same thing as in part B, but for the four-input parity function.