Testing Nonlinear Least Squares and Neural Networks Parameters

Abstract - **The objective was to implement Nonlinear Least Squares (NLS) for GPS localization using various initial values and to evaluate a neural networks performance for plant classification using different learning rates. For NLS the objective function was evaluated using 'lsqnonlin'. The input to 'lsqnonlin' consists of the objective function and three different initial values: [14131454.8, 13301808.7, 17048044.1] (W0), 0.5*W0, and 2*W0. For each unique initial value, a different Cartesian coordinate of the receiver was found. This displays that the evaluated objective function has multiple minimizers and that the initial value has an effect on which minimizer is found. The neural network was run with learning rates of 0.01, 1, and 0.000001 which attained an accuracy of 96%, 0.6667%, and 36% respectively. As displayed by the results, the learning rate has a strong effect on the accuracy of the model in question. A strong initial estimate for NLS problems is important, as it is possible to get vastly different results based on this initial value. Similarly for neural networks, the learning rate has a noticeable effect on the performance of the model. Implementing a method such as ADADELTA is a good idea, as it requires no manual tuning of the learning rate [1].**

INTRODUCTION

There were two primary objectives investigated: the first was to use Nonlinear Least Squares (NLS) in order to locate a receiver based off of GPS locations, and the second was to implement a Two Layer Neural Network to classify plant type based off of data within the fisheriris dataset.

There are cases where problems require optimization that involves multiple independent "readings". For each of these readings, a nonlinear model function exists. These problems can be addressed by implementing NLS formulation. NLS is a form of analysis used to fit a set of observations with a non linear model. For our purposes, NLS will be applied to GPS localization with the goal of locating a receiver. To achieve this, non-weighted NLS will be performed with multiple starting points investigated in order to determine how altering the initial starting parameter affects the end result.

In the brain, neurons are cells that receive input from other neurons and transfer information throughout the brain after reaching a certain activation threshold. In computing, an artificial neural network (ANN) can be constructed using similar principles to investigate underlying relationships in a set of data. Similar to the brain, receptors in a neural network only activate after a certain amount of input has passed through so we can algorithmically classify data and assign it to certain categories. Using the famous fisheriris dataset, the ANN is fed labelled data for training. As more data is inputted to the ANN, it adjusts its weights until it has been fitted appropriately. The learning rate is a hyperparameter used in the training of neural networks that has a small positive value, often in the range of 0.0 and 1.0, and controls how quickly the model adapts to the problem. A smaller learning rate causes smaller changes to each weight update whereas a larger learning rate results in rapid change.

Different initial values will be evaluated to see how changes to this value affect the end Cartesian coordinate. As well, various learning rates will be applied to the ANN to see how increasing and decreasing the learning rate affects the accuracy of the model.

METHODS

In GPS localization problems, the distance from the receiver to the satellite can be determined using the following equation: $gi = (\overline{w}^T \overline{w} - 2\overline{x}_i^T \overline{w} + \overline{x}_i^T \overline{x}_i)^{1/2}$, where x corresponds to GPS data and w corresponds to the mean location of satellites. Distance calculations were stored in a matrix with row counts matching the GPS data, in this case 6. Residual error can be calculated by using the following equation $\overline{r}(\overline{w}) = \overline{g}(\overline{w}) - \overline{y} = \left[g_m(\overline{w}) - y_m\right]$, where g corresponds to the calculated distance to the

satellite, and y corresponds to the actual distance to the satellite. Subtracting these values gives the residual error matrix which we convert into an anonymous function which takes the input 'w' representing our initial estimate. This user defined function can be inserted into the function 'lsqnonlin' along with the initial value vector in order to solve nonlinear least squares curve fitting problems. Lsqnonlin uses the Levenberg-Marquard algorithm which uses the equation $\overline{w_{k+1}} = \overline{w_k} + [J_k^T J_k + \lambda I] J_K^T \overline{r_k}$ in order to solve for the minimizer of the user defined function. The function goes through the user defined function evaluated at each entry of the vector and evaluates its value. The minimum value is returned as output which corresponds to the Cartesian coordinates of the GPS receiver.

The learning rate of a two layer neural network will be investigated by using the famous fisheriris dataset to classify flower types based on the length and width of the flower sepals. To begin, the hidden responses are calculated as $1/(1 + e^{(-xmat * wvecH)})$ where wvecH represents the weights of the hidden layer. A 1 is then appended onto the input layer. Following this, we compute the hidden differential response by defining the activation function as $1/(1 + e^{(-u)})$ and the derivative of the activation function as $activationFunction(u) * (1 - activationFunction(u))$. We then plug the inner product $(xmat * wvecH)$ into the derivative of the activation function to calculate the activation score. This activation score can be used to generate a Jacobian matrix of the hidden layer activations using the equation $J_{2\bar{\phi}} = diag(2\overline{\psi}(2\overline{u}))$. Additionally, the gradient can be calculated using the equation $2D = [I \otimes x]$. These two equations can be combined to calculate the derivative of outputs of the hidden layer following the equation $[J_{2\phi}]\, D$. Following all of these equations, a gradient matrix can be generated by multiplying the derivative outputs of the hidden layer by the weights for layer one. Once completed for all data points, we can differentiate the residual error and scale the created gradient matrix. We can then sum the gradients of each data vector to achieve the net gradient. We use this objective function within the steep fixed function in order to perform optimization. From our steep fixed optimization, a recalculated weight vector is outputted. This optimized weight vector is plugged into the 'annclass' function which computes the response of a simple neural network that has a single hidden layer. It does this by separating the output weights and hidden weights and computing the hidden response. It uses this hidden response to compute the transfer function and then generate a vector of linear response data. The ANN response is a Heaviside step function, meaning it outputs 0 for negative responses, and 1 for positive responses. This Heaviside response represents the predicted labels generated by the ANN. We use this predicted labels vector to calculate both the accuracy and confusion matrix for the model given learning rates of 0.01, 1, and 0.000001.

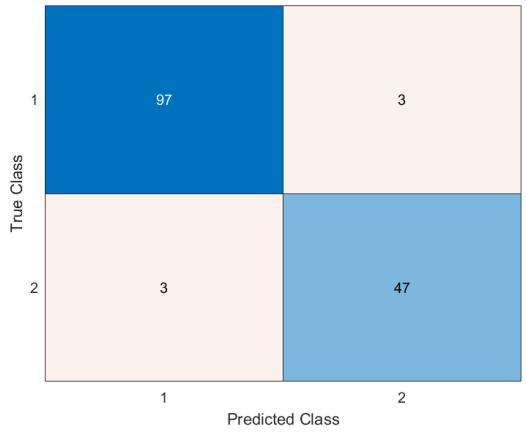
The results of GPS localization were evaluated by observing the different Cartesian coordinates of the receiver produced when the initial estimate was set to three unique values. For the neural network, the results were evaluated by observing the changes in accuracy when three unique learning rates were applied.

RESULTS

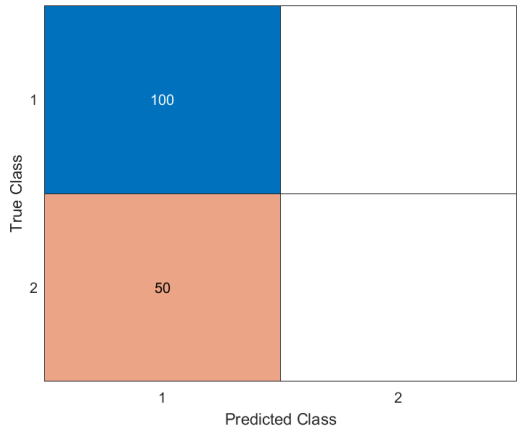
Table 1: The following table contains the initial estimates along with the calculated receiver location as both Cartesian coordinates and Earth Centered Earth-fixed (ECEF) coordinates. The first row represents the initial estimate W0, the second row represent 0.5 * W0, and the third row represents 2 * W0.

Initial Estimate	**Cartesian Coordinates of Receiver**	**Earth centered Earth-fixed (ECEF) Coordinates of Receiver**
[14131454.8,	[14131448.5,	[41.3,

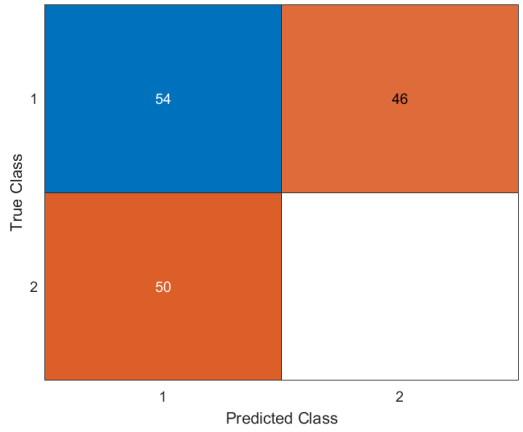
13301808.7, 17048044.1]	13301801.1, 17048042.3]	43.3, 19462775.9]
[7065727.4. 6650904.4, 8524022.0]	[1108024.8, -4346331.7, 4519516.2]	[45.4, -75.7, 107.6]
[28262909.5, 26603617.4, 34096088.1]	[26427712.3, 18394302.0, 38547265.0]	[50.2, 34.8, 43860602.8]



(A)



(B)



(C)

Figure 1: A confusion matrix generated by running a neural network to classify plant types based off of the fisheriris dataset. In all matrices, the top left represents true positive instances, the top right represents false negative instances, the bottom left represents false positive instances, and the bottom right represents true negative instances. (A) The neural network was run with a learning rate of 0.01 (96% accuracy). (B) The neural network was run with a learning rate of 1 (0.6667% accuracy). (C) The neural network was run with a learning rate of 0.000001 (36% accuracy).

DISCUSSION

For the initial problem of using Nonlinear Least Squares (NLS) for GPS receivers, we can observe **Table 1.** and see that changes to the initial estimate drastically change both the Cartesian coordinates and the Earth centered Earth-fixed (ECEF) coordinates for the receiver. When the initial estimate is set to [14131454.8, 13301808.7, 17048044.1] (W0), the calculated Cartesian coordinates were [14131448.5, 13301801.1, 17048042.3]. When the initial estimate was set to 0.5 * W0, the calculated Cartesian coordinates were [1108024.8, -4346331.7, 4519516.2]. When the estimate was set to 2*W0, the calculated Cartesian coordinates were [26427712.3, 18394302.0, 38547265.0]. For the secondary problem of implementing a two layer neural network for supervised learning, we can observe in **Figure 1.** how changes in the learning rates altered the accuracy of our neural network. When the learning rate was set to a value of 0.01, an accuracy of 96% was attained. The confusion matrix shows a true positive value of 97, a false positive value of 3, a false negative value of 3, and a true negative value of 47. When the learning rate was increased to a value of 1, the accuracy dropped to 0.6667% .The confusion matrix shows a true positive value of 100, a false positive value of 50 and a false negative and true negative value of 0. When the learning rate was set to a value of 0.000001 the accuracy was calculated to be 36%. The confusion matrix displayed a true positive value of 54, a false positive value of 50, a false negative value of 46 and a true negative value of 0.

The drastic changes in both Cartesian coordinates and ECEF coordinates when changing initial estimates can be attributed to the objective function having multiple minimizers. Different starting points may be closer to different minimizers, which explains why end results changed when the initial estimate was altered. As ‘lsqnonlin’ is being used, it is solving for the closest minimizer using the Levenberg-Marquard algorithm. This highlights the importance of having a strong and robust method for generating an initial estimate, as results are heavily based on this initial value.

When altering the learning rate of the neural network, drastic changes in accuracy were displayed. Setting the learning rate to a low value of 0.000001 displayed a lower accuracy of 36% compared to an accuracy of 96% when the learning rate was set to 0.01. It is possible that with such a low learning rate, the model may have arrived at a local minimum and become overfit. This may contribute to the lower accuracy displayed. When the learning rate was increased to a higher value of 1, the accuracy dropped drastically to 0.6667%. With such a high learning rate, the model clearly converged too early. This means it was not able to properly fit the relationship between the classes and caused mislabeled data. This is displayed in the confusion matrix **Figure 1. (B)** as we see true positive and false positive have values of 100 and 50 respectively whereas false negative and true negative both have a value of 0. These results display the importance learning rate has on the accuracy and behaviour of a model. A proposed method for determining learning rate would be to implement a per-dimension learning rate method called ADADELTA, which requires no manual tuning of the learning rate [1]. Overall, too small of a learning rate caused the model to overfit and have a lower accuracy, whereas having too high a learning rate caused the model to converge too quickly and produce inaccurate results.

Assigning a strong initial estimate for NLS problems is important as the end results can change drastically based on this value. Therefore, a robust method for generating this value is important. When attempting to assign a learning rate in a neural network, it is clear the importance of choosing an appropriate value as model performance can be drastically altered. ADADELTA is a strong method for generating an adaptive learning rate without manual tuning.

REFERENCES

[1] Zeiler M. ADADELTA: AN ADAPTIVE LEARNING RATE METHOD [Internet]. Available from: https://arxiv.org/pdf/1212.5701.pdf