Donald Trump Tweet Author Analysis

This personal project looks at data from president Trump's Twitter. Donald Trump used to tweet from an iphone while his campaign staff tweeted through his account on Android, so this data can be used to train a neural network to predict who wrote his newer tweets based on a set of features to include how often he used certain words and how long the tweets were.

Load Data

```
% Uncomment this on first run to load in data, takes a while
% opts = detectImportOptions('dataset.xlsx');
% opts.SelectedVariableNames = [8 12:334];
% X = readmatrix('dataset.xlsx',opts);
% opts.SelectedVariableNames = 2;
% Y = readmatrix('dataset.xlsx',opts);
load('X.mat');
load('Y.mat');
s = string(Y);
id = s == "Twitter for iPhone" | s == "Twitter for Android";
s = s(id);
X = X(id,:);
Y = s == "Twitter for iPhone";
```

Mean normalize X

```
non_present = max(X) == 0;
X = X(:,~non_present);
X = (X - repmat(mean(X), size(X,1), 1)) ./ repmat(2*std(X), size(X,1), 1);
X = [ones(size(X,1),1) X];
```

Divide data

```
m = size(X,1);
[trainInd,valInd,testInd] = dividerand(m,.6,.2,.2);
X_train = X(trainInd,:);
Y_train = Y(trainInd,:);
X_val = X(valInd,:);
Y_val = Y(valInd,:);
X_test = X(testInd,:);
Y_test = Y(testInd,:);
```

Logistic Regression - Train theta

Here I will use logistic regression, then I will try a neural network.

```
initial_theta = zeros(size(X_train,2),1);
options = optimoptions(@fminunc,'Algorithm','Quasi-Newton','GradObj', 'on', 'MaxIter', 400);
[theta, cost] = fminunc(@(t)(costFunctionReg(t, X_train, Y_train,1)), initial_theta, options);
```

```
Local minimum found.
```

```
Optimization completed because the size of the gradient is less than
the value of the optimality tolerance.

<stopping criteria details>
```

```

p = predict(theta, X_val);
success = p == Y_val;
success = success(success);
success_rate = length(success)/length(p);
fprintf('Logistic Regression Validation Set Accuracy = %f\n', success_rate);
```

```
Logistic Regression Validation Set Accuracy = 0.850289
```

Neural Network

The neural network is a simple 1-hidden layer neural network.

Initialize Thetas

```
% nn script
hidden_layer_size = 40;
input_layer_size = 308;

initial_Theta1 = randInitializeWeights(input_layer_size, hidden_layer_size);
initial_Theta2 = randInitializeWeights(hidden_layer_size, 1);

% Unroll parameters
initial_nn_params = [initial_Theta1(:) ; initial_Theta2(:)];
```

Train

```
options = optimset('MaxIter', 60);
lambda = 3;

% Create "short hand" for the cost function to be minimized
costFunction = @(p) nnCostFunction(p, input_layer_size, hidden_layer_size, X_train, Y_train, la

% Now, costFunction is a function that takes in only one argument (the
% neural network parameters)
[nn_params, ~] = fmincg(costFunction, initial_nn_params, options);
```

```
Iteration     1 | Cost: 6.980162e-01
Iteration     2 | Cost: 4.951659e-01
Iteration     3 | Cost: 4.478344e-01
Iteration     4 | Cost: 4.135613e-01
Iteration     5 | Cost: 3.823161e-01
Iteration     6 | Cost: 3.513519e-01
Iteration     7 | Cost: 3.427668e-01
Iteration     8 | Cost: 3.306680e-01
Iteration     9 | Cost: 3.296712e-01
Iteration    10 | Cost: 3.237273e-01
Iteration    11 | Cost: 3.203762e-01
Iteration    12 | Cost: 3.155860e-01
Iteration    13 | Cost: 3.138944e-01
Iteration    14 | Cost: 3.133712e-01
Iteration    15 | Cost: 3.102120e-01
Iteration    16 | Cost: 3.082352e-01
Iteration    17 | Cost: 3.047811e-01
Iteration    18 | Cost: 3.017493e-01
Iteration    19 | Cost: 2.973636e-01
Iteration    20 | Cost: 2.968866e-01
Iteration    21 | Cost: 2.952505e-01
Iteration    22 | Cost: 2.940775e-01
Iteration    23 | Cost: 2.935262e-01
Iteration    24 | Cost: 2.929661e-01
Iteration    25 | Cost: 2.921174e-01
Iteration    26 | Cost: 2.920055e-01
Iteration    27 | Cost: 2.893992e-01
Iteration    28 | Cost: 2.861177e-01
Iteration    29 | Cost: 2.852148e-01
Iteration    30 | Cost: 2.846142e-01
Iteration    31 | Cost: 2.845092e-01
Iteration    32 | Cost: 2.842318e-01
Iteration    33 | Cost: 2.838166e-01
Iteration    34 | Cost: 2.836979e-01
Iteration    35 | Cost: 2.824089e-01
Iteration    36 | Cost: 2.809137e-01
Iteration    37 | Cost: 2.763423e-01
Iteration    38 | Cost: 2.739081e-01
Iteration    39 | Cost: 2.737138e-01
Iteration    40 | Cost: 2.694333e-01
Iteration    41 | Cost: 2.581496e-01
Iteration    42 | Cost: 2.551452e-01
Iteration    43 | Cost: 2.516281e-01
Iteration    44 | Cost: 2.514009e-01
Iteration    45 | Cost: 2.510815e-01
Iteration    46 | Cost: 2.497073e-01
Iteration    47 | Cost: 2.491823e-01
Iteration    48 | Cost: 2.471236e-01
Iteration    49 | Cost: 2.463393e-01
Iteration    50 | Cost: 2.451785e-01
Iteration    51 | Cost: 2.450664e-01
Iteration    52 | Cost: 2.436585e-01
Iteration    53 | Cost: 2.429701e-01
Iteration    54 | Cost: 2.426089e-01
Iteration    55 | Cost: 2.402617e-01
Iteration    56 | Cost: 2.346075e-01
Iteration    57 | Cost: 2.329964e-01
Iteration    58 | Cost: 2.254908e-01
Iteration    59 | Cost: 2.252035e-01
Iteration    60 | Cost: 2.240138e-01
```

```
Theta1 = reshape(nn_params(1:hidden_layer_size * (input_layer_size + 1)), hidden_layer_size, (i
Theta2 = reshape(nn_params((1 + (hidden_layer_size * (input_layer_size + 1))):end), 1, (hidden_

pred_val = predict_nn(Theta1, Theta2, X_val);
val_accuracy = mean(double(pred_val == Y_val));
fprintf('Logistic Regression Validation Set Accuracy = %f\n', val_accuracy);
```

```
Logistic Regression Validation Set Accuracy = 0.865171
```