Varun Nagaraj - 50290761

Introduction to Machine Learning

Project 2
Handwritten Character Regression on CEDAR Dataset

Project Synopsis:

The goal of this project is to use machine learning to solve a problem that is to predict the word “and” using Linear regression, Logistic regression and using Neural Networks.

There are three tasks:

1. Train a linear regression model on CEDAR dataset using a gradient descent (GD) solution.
2. Train a logistic regression model on the CEDAR dataset using gradient descent (GD).
3. Train a multilayer perceptron on the CEDAR dataset using Keras.

What is Linear Regression?

Linear regression is a linear approach to modelling the relationship between a scalar response (or dependent variable) and one or more explanatory variables (or independent variables).
Y = X^T (Theta) gives the simplified equation for linear regression.
In the above equation Y is the output generated by the model, X is the design matrix and Theta is the weight matrix or regression coefficients.

What is Logistic Regression?

Logistic regression is a predictive analysis. Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.

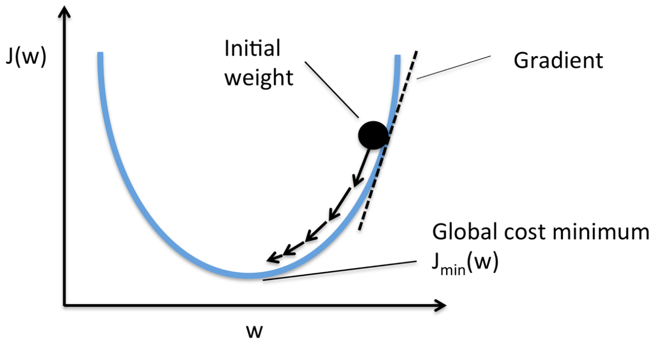
How to implement Linear and Logistic Regression?

The Linear Regression algorithm, the weights are updated incrementally after each iteration (epoch). The magnitude and direction of the weight update is computed by taking a step in the opposite direction of the cost gradient

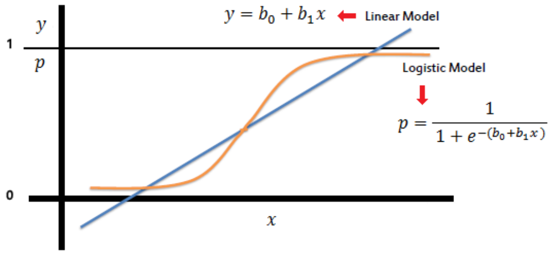
$$\Delta w_j = -\eta \frac{\partial J}{\partial w_j},$$

Here η is the learning rate, which dictates by what value the weights should be updated.

Gradient Descent can be depicted by the below diagram:

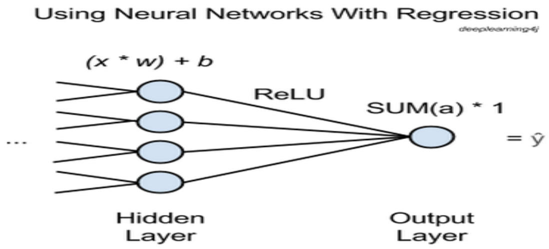


The logistic regression model is implemented in a similar form. The main formula for logistic regression is as shown below.



Neural Network Implementation

Neural Network (NN) is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. In this we use Keras API to implement a perceptron. Below is the representation of how a regression solution is implemented using Neural Networks.



Results

Training the model and changes in accuracy and RMS error value with change in hyper- parameters.

Linear Regression Solution:

Run Number	HOF Subtracted RMS value	HOF Concatenated RMS value	GSC Subtracted RMS value	GSC Concatenated RMS value
1	0.026955832	0.022296236	0.064268956	0.072617301
2	0.029328498	0.025234682	0.069348918	0.078923472
3	0.032139810	0.029712823	0.071922348	0.082103814
4	0.023900109	0.035242190	0.068324091	0.079124719
5	0.028239891	0.021093486	0.062881209	0.075182823

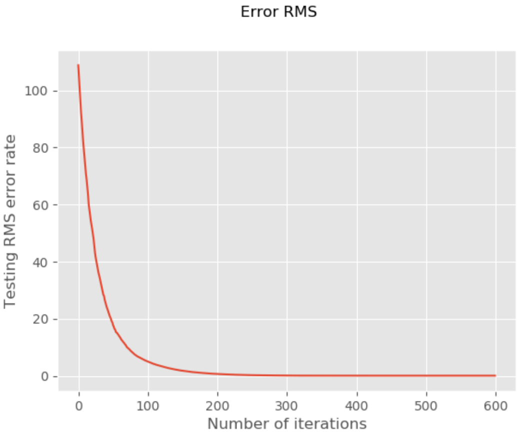
Logistic Regression Solution:

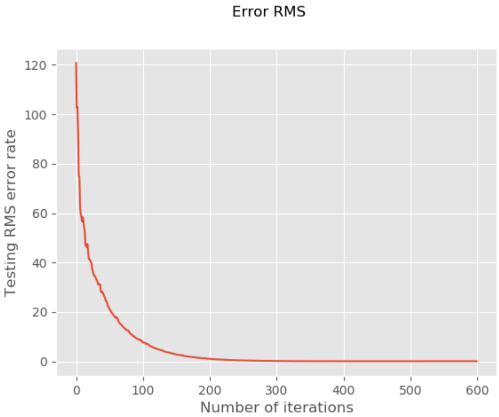
Run Number	HOF Subtracted -RMS value -Accuracy	HOF Concatenated -RMS value -Accuracy	GSC Subtracted -RMS value -Accuracy	GSC Concatenated -RMS value -Accuracy
1	0.419923 58%	0.50632911 50.94%	0.21268956 78.2%	0.3967545 60.33%
2	0.462033 53.79%	0.4556913 54.43%	0.18325834 81.67%	0.3787921 62.97%
3	0.487342 51.26%	0.4775391 52.24%	0.19284021 80.4%	0.3875629 61.25%
4	0.434729 56.52%	0.4812718 51.87%	0.20150322 79.8%	0.3876592 61.23%
5	0.448192 55.2%	0.4482941 55.2%	0.21574924 78.9%	0.3698754 63.11%

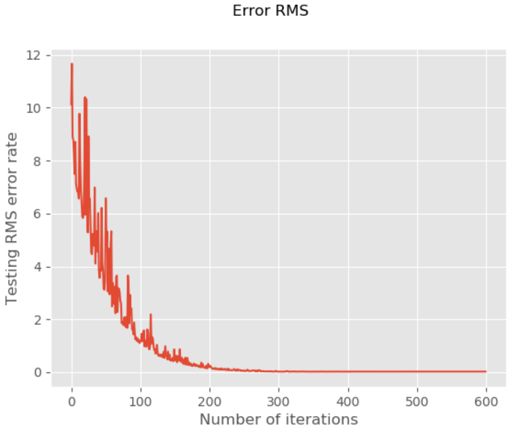
Neural Network Solution:

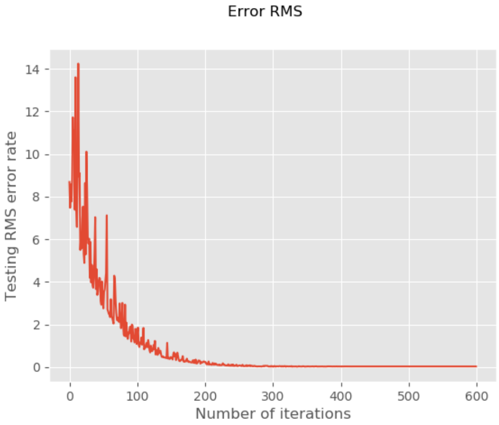
Run Number	HOF Subtracted -RMS value -Accuracy	HOF Concatenated -RMS value -Accuracy	GSC Subtracted -RMS value -Accuracy	GSC Concatenated -RMS value -Accuracy
1	0.107167 87.69%	0.0516 96.84%	0.00760 99.3%	0.01316 99.73%
2	0.07057 93.37%	0.0884 91.79%	0.005690 99.33%	0.008347 99.38%

Linear Regression Graphs: ***top 2 are for GSC (sub and concat) bottom 2 are HOF(sub and concat)***

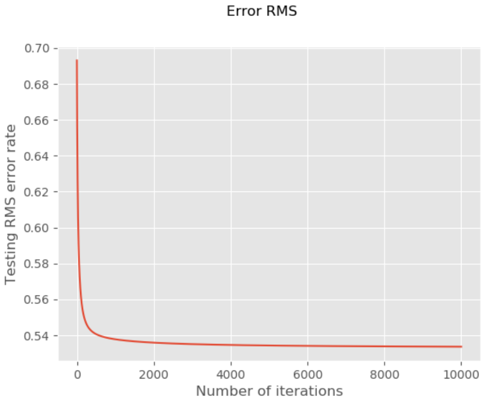


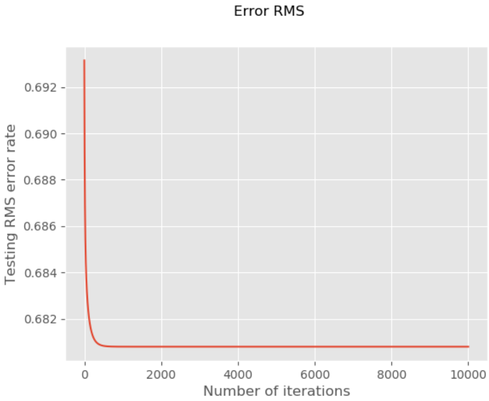


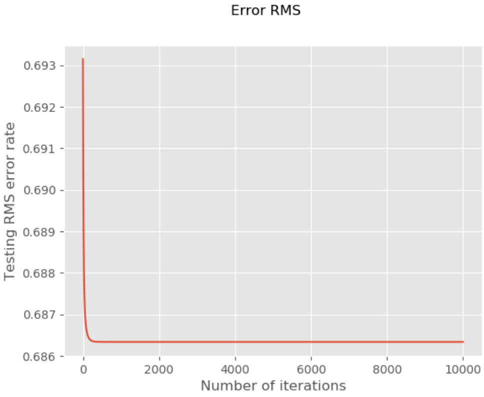


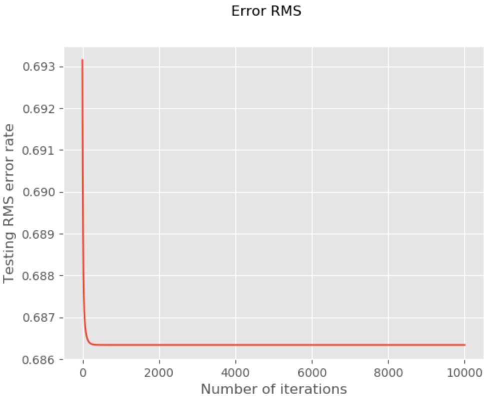


Logistic Regression Graphs: ***top 2 are for GSC (sub and concat) bottom 2 are HOF(sub and concat)***

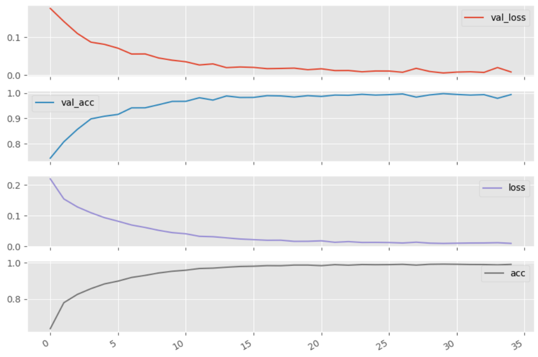


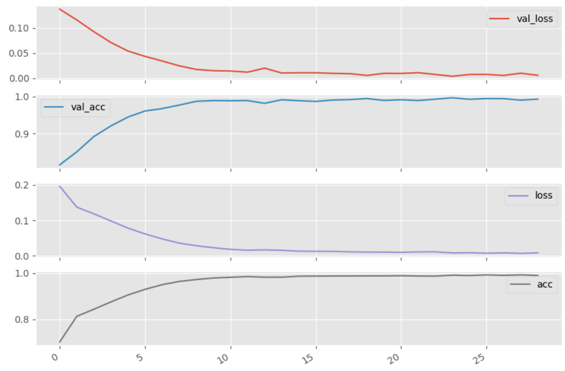


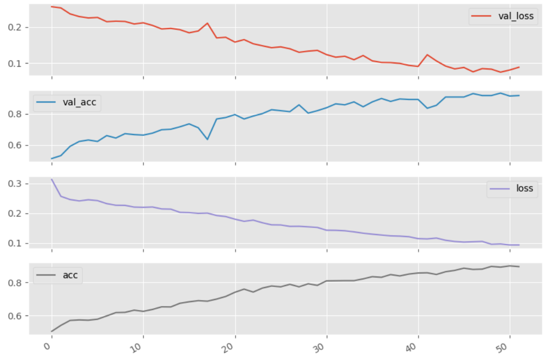


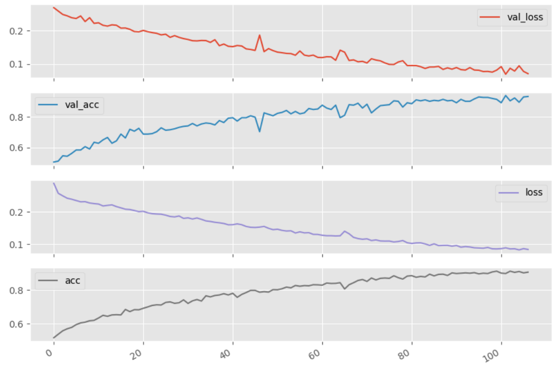


Neural Network Implementation Graphs: ***top 2 are for GSC (sub and concat) bottom 2 are HOF (sub and concat)***









Conclusion:

We can see that with iterations going forward, the error RMS value keeps decreasing and and reaches a point where the error is almost 0. This shows that the models are being trained to predict the values for “and” images correctly. To be noted, it is always better to randomize the selection of the datasets while running the models, so that we will know what the average accuracy is across the entire dataset rather than using the same part of the dataset over and over again.