Programming Assignment 5: Neural Networks

Srihari Akash Chinam
USC ID - 2953497706
Net ID - chinam

Sayan Nanda
USC ID - 2681592859
Net ID - snanda

Implementation

- Language used - Python version 2
- Libraries used - Numpy (arrays and matrices), SciKit-Learn, opencv-python (imported as cv2)

Neural Network

After loading the pgm images (by reading them byte-wise) using file names taken from "downgesture_train.list", it was observed that all pixels had integer values. To adjust for this, weights were initialized randomly between 0 and 1. The neural network constructed contains one input layer, one hidden layer (with 100 neurons) and one output layer. The input layer was of size 960, to take the input of the images of size 32x30. For every neuron, the sigmoid function was used as the activation function. After training using back propagation, the test samples were loaded and tested against the neural network. When tested with multiple initializations, it was seen that the best output of the network came at an accuracy of 77%. To optimize this result, every image's matrix (train and test) was normalized using Z-normalization, which is:

X = (X - mean(image_matrix))/standard_deviation(image_matrix)

After the normalization, the results of the Neural Network improved significantly, giving a best case **accuracy of 97.59%**, that is, **81 correct predictions out of 83** test samples. The predictions for this output are provided in Table 1, with misclassified items highlighted in red.

Table 1 Output for Neural Network's prediction

File Name	Expected Output	Implementation Output
gestures/A/A_down_1.pgm	1	1
gestures/A/A_down_2.pgm	1	1
gestures/A/A_hold_1.pgm	0	0
gestures/A/A_hold_10.pgm	0	0
gestures/A/A_stop_1.pgm	0	0
gestures/A/A_stop_4.pgm	0	0
gestures/A/A_up_1.pgm	0	0
gestures/A/A_up_10.pgm	0	0

gestures/B/B_down_1.pgm	1	1
gestures/B/B_down_2.pgm	1	1
gestures/B/B_hold_1.pgm	0	0
gestures/B/B_hold_2.pgm	0	0
gestures/B/B_stop_1.pgm	0	0
gestures/B/B_stop_2.pgm	0	0
gestures/B/B_up_1.pgm	0	0
gestures/B/B_up_4.pgm	0	0
gestures/C/C_down_1.pgm	1	1
gestures/C/C_down_2.pgm	1	1
gestures/C/C_hold_1.pgm	0	0
gestures/C/C_hold_2.pgm	0	0
gestures/C/C_stop_2.pgm	0	0
gestures/C/C_stop_3.pgm	0	0
gestures/C/C_up_1.pgm	0	0
gestures/D/D_down_1.pgm	1	1
gestures/D/D_down_2.pgm	1	0
gestures/D/D_hold_1.pgm	0	0
gestures/D/D_hold_2.pgm	0	0
gestures/D/D_hold_6.pgm	0	1
gestures/D/D_stop_1.pgm	0	0
gestures/D/D_stop_2.pgm	0	0
gestures/D/D_up_1.pgm	0	0
gestures/D/D_up_3.pgm	0	0
gestures/E/E_down_1.pgm	1	1
gestures/E/E_hold_1.pgm	0	0
gestures/E/E_hold_5.pgm	0	0
gestures/E/E_stop_1.pgm	0	0
gestures/E/E_stop_2.pgm	0	0
gestures/E/E_up_1.pgm	0	0
gestures/E/E_up_2.pgm	0	0
gestures/F/F_down_1.pgm	1	1
gestures/F/F_down_4.pgm	1	1
gestures/F/F_hold_1.pgm	0	0
gestures/F/F_hold_2.pgm	0	0
gestures/F/F_stop_2.pgm	0	0
gestures/F/F_stop_5.pgm	0	0
gestures/G/G_down_2.pgm	1	1
gestures/G/G_down_3.pgm	1	1
gestures/G/G_hold_4.pgm	0	0
gestures/G/G_stop_2.pgm	0	0
gestures/G/G_stop_5.pgm	0	0

gestures/G/G_up_2.pgm	0	0
gestures/G/G_up_5.pgm	0	0
gestures/H/H_down_2.pgm	1	1
gestures/H/H_hold_10.pgm	0	0
gestures/H/H_hold_2.pgm	0	0
gestures/H/H_hold_5.pgm	0	0
gestures/H/H_stop_5.pgm	0	0
gestures/H/H_stop_6.pgm	0	0
gestures/H/H_up_5.pgm	0	0
gestures/I/I_hold_2.pgm	0	0
gestures/I/I_hold_5.pgm	0	0
gestures/I/I_down_3.pgm	1	1
gestures/I/I_stop_5.pgm	0	0
gestures/I/I_stop_6.pgm	0	0
gestures/I/I_up_2.pgm	0	0
gestures/I/I_up_3.pgm	0	0
gestures/J/J_down_5.pgm	1	1
gestures/J/J_down_6.pgm	1	1
gestures/J/J_hold_2.pgm	0	0
gestures/J/J_hold_3.pgm	0	0
gestures/J/J_stop_7.pgm	0	0
gestures/J/J_stop_8.pgm	0	0
gestures/J/J_up_1.pgm	0	0
gestures/J/J_up_2.pgm	0	0
gestures/K/K_down_2.pgm	1	1
gestures/K/K_down_3.pgm	1	1
gestures/K/K_hold_1.pgm	0	0
gestures/K/K_hold_2.pgm	0	0
gestures/K/K_hold_3.pgm	0	0
gestures/K/K_stop_1.pgm	0	0
gestures/K/K_stop_2.pgm	0	0
gestures/K/K_stop_1.pgm	0	0
gestures/K/K_stop_2.pgm	0	0

Next, the Neural Network implementation in sklearn, “MLPClassifier”, was trained using the normalized training data and then tested with the normalized images. After several initializations, the best accuracy obtained for SciKit-Learn’s model was **96.38%**, with **80 images of 83 correctly** classified. The predictions for this output are provided in Table 2, with misclassified items highlighted in red.

Table 2 Output for SciKit-Learn's Neural Network

File Name	Expected Output	Implementation Output
gestures/A/A_down_1.pgm	1	1
gestures/A/A_down_2.pgm	1	1
gestures/A/A_hold_1.pgm	0	0
gestures/A/A_hold_10.pgm	0	0
gestures/A/A_stop_1.pgm	0	0
gestures/A/A_stop_4.pgm	0	0
gestures/A/A_up_1.pgm	0	0
gestures/A/A_up_10.pgm	0	0
gestures/B/B_down_1.pgm	1	1
gestures/B/B_down_2.pgm	1	1
gestures/B/B_hold_1.pgm	0	0
gestures/B/B_hold_2.pgm	0	0
gestures/B/B_stop_1.pgm	0	0
gestures/B/B_stop_2.pgm	0	0
gestures/B/B_up_1.pgm	0	0
gestures/B/B_up_4.pgm	0	0
gestures/C/C_down_1.pgm	1	1
gestures/C/C_down_2.pgm	1	1
gestures/C/C_hold_1.pgm	0	0
gestures/C/C_hold_2.pgm	0	0
gestures/C/C_stop_2.pgm	0	0
gestures/C/C_stop_3.pgm	0	0
gestures/C/C_up_1.pgm	0	0
gestures/D/D_down_1.pgm	1	0
gestures/D/D_down_2.pgm	1	1
gestures/D/D_hold_1.pgm	0	0
gestures/D/D_hold_2.pgm	0	0
gestures/D/D_hold_6.pgm	0	0
gestures/D/D_stop_1.pgm	0	0
gestures/D/D_stop_2.pgm	0	0
gestures/D/D_up_1.pgm	0	0
gestures/D/D_up_3.pgm	0	0
gestures/E/E_down_1.pgm	1	1
gestures/E/E_hold_1.pgm	0	0
gestures/E/E_hold_5.pgm	0	0
gestures/E/E_stop_1.pgm	0	0
gestures/E/E_stop_2.pgm	0	0
gestures/E/E_up_1.pgm	0	0
gestures/E/E_up_2.pgm	0	0
gestures/F/F_down_1.pgm	1	1
gestures/F/F_down_4.pgm	1	1

gestures/F/F_hold_1.pgm	0	0
gestures/F/F_hold_2.pgm	0	0
gestures/F/F_stop_2.pgm	0	0
gestures/F/F_stop_5.pgm	0	0
gestures/G/G_down_2.pgm	1	1
gestures/G/G_down_3.pgm	1	1
gestures/G/G_hold_4.pgm	0	0
gestures/G/G_stop_2.pgm	0	0
gestures/G/G_stop_5.pgm	0	0
gestures/G/G_up_2.pgm	0	1
gestures/G/G_up_5.pgm	0	0
gestures/H/H_down_2.pgm	1	0
gestures/H/H_hold_10.pgm	0	0
gestures/H/H_hold_2.pgm	0	0
gestures/H/H_hold_5.pgm	0	0
gestures/H/H_stop_5.pgm	0	0
gestures/H/H_stop_6.pgm	0	0
gestures/H/H_up_5.pgm	0	0
gestures/I/I_hold_2.pgm	0	0
gestures/I/I_hold_5.pgm	0	0
gestures/I/I_down_3.pgm	1	1
gestures/I/I_stop_5.pgm	0	0
gestures/I/I_stop_6.pgm	0	0
gestures/I/I_up_2.pgm	0	0
gestures/I/I_up_3.pgm	0	0
gestures/J/J_down_5.pgm	1	1
gestures/J/J_down_6.pgm	1	1
gestures/J/J_hold_2.pgm	0	0
gestures/J/J_hold_3.pgm	0	0
gestures/J/J_stop_7.pgm	0	0
gestures/J/J_stop_8.pgm	0	0
gestures/J/J_up_1.pgm	0	0
gestures/J/J_up_2.pgm	0	0
gestures/K/K_down_2.pgm	1	1
gestures/K/K_down_3.pgm	1	1
gestures/K/K_hold_1.pgm	0	0
gestures/K/K_hold_2.pgm	0	0
gestures/K/K_hold_3.pgm	0	0
gestures/K/K_stop_1.pgm	0	0
gestures/K/K_stop_2.pgm	0	0
gestures/K/K_stop_1.pgm	0	0
gestures/K/K_stop_2.pgm	0	0

Software Familiarization

The python library SciKit-Learn offers an implementation for neural networks. The library implementation offers more flexibility by allowing the user to change parameters. A tuple can be passed as the hidden layer size. An element of the tuple represents the number of nodes in the given hidden layer. There is an option to change the solver for the weight optimization. In our implementation we have used a gradient-based optimizer. There are other solvers offered such as those which use quasi-Newton methods.

There are multiple parameters to affect the learning rate, one is to control the learning rate schedule such as adaptive, constant and inverse scaling. The second parameter controls the learning rates initial value. In our implementation, we use a constant learning rate of 0.1. There are also parameters to include tolerance, randomizing the weights at initialization and the maximum number of iterations.

There is a parameter to change the function used in the hidden layers, also called the activation function. In our implementation, we have used a logistic sigmoid function. SciKit-Learn also offers a hyperbolic tan function, rectified linear unit function and an identity function. Some of the possible improvements we could have implemented are, running the neural network multiple times with randomized weights each time, following which we would present the best result.

Applications

Neural networks have found applications in a variety of disciplines. As neural networks are used in signal processing in brain research. They are used for neurological waveform detection and pattern analysis. They are found to be very effective because the assumptions which need to be made with respect to signal and noise are at a minimum [1]. Apart from this specific application, neural networks have found a lot of application in neuroscience. This is because of their ability to represent memory components, self-organizing maps and its use in deep learning.

Neural networks have impacted investment and finance decision making. It is able to detect patterns which evade traditional analytic techniques. There has been research into its application in predicting business failure, debt risk, stock market prediction, assess bond and mortgage risk and predict bankruptcy. [2]

Other applications of neural networks include robotics, control, data processing, filtering, clustering, compression and blind source separation, regression analysis and classification. It is used in medicine (for example diagnosing cancer), finance, machine translation, game-playing, quantum chemistry and many other fields.

Individual Contributions

Sayan Nanda

- Implementation of models
  - Preprocessing Data
  - Normalization techniques
  - SciKit-Learn Implementation
- Documentation
  - Software Familiarization
  - Applications

Srihari Akash Chinam

- Implementation of models
  - Normalization techniques
  - Forward Propagation
  - Backward Propagation
- Documentation
  - Implementation of Neural Network
  - Comparison with SciKit-Learn's results

References

[1] Applications of neural-network (NN) signal processing in brain research http://ieeexplore.ieee.org/document/1642/

[2] Neural Networks in Finance and Investing: Using Artificial Intelligence to Improve Real World Performance https://dl.acm.org/citation.cfm?id=573193