CS 584 - Machine Learning - Assignment Report

Rakesh Adhikesavan

Master's in Data Science, Illinois Institute of Technology, Chicago

Abstract

In this assignment, logistic regression and neural networks was implemented. The performance of each algorithm was evaluated and tested on different datasets, using a 10 fold cross validation method [2]. The algorithms were implemented in Python, from scratch. The performance was compared against scikit-learn's inbuilt MLP classifier. The implementation details along with the results and conclusions are summarized below.

----------*****----------

1. Introduction

The purpose of this assignment is to implement *discriminative learning* algorithms. Discriminative models are a class of models used in machine learning for modeling the dependence of an unobserved variable y on an observed variable x. As opposed to generative models, discriminative models do not generate samples from the joint distribution of x and y. Rather, the algorithm models the conditional probability distribution of $P(y|x)$. For tasks such as classification, that do not require the joint distribution, discriminative models can yield better performance.

For the purpose of this assignment, I chose two popular datasets: The Iris Dataset and Digits data set. First, a 2-class logistic regression algorithm was implemented. The performance was tested on a subset of the Iris Data set. Next, a k-class logistic regression was implemented from scratch and the performance was tested on 3-class Iris dataset.

The k-class logistic regression unit was extended to create a multi-layer perceptron. The performance of this neural network was tested on a subset of the digits dataset. The performance of the classifier was varied by deliberately adjusting the learning rate for the logistic regression algorithm and by adjusting the learning rate, number of units in the hidden layer and momentum for the neural networks algorithm.

All results, comparisons and performance evaluations are summarized below, along with data description and implementation details.

2. Implementation detail

The datasets were imported from Python package sklearn. The two datasets used for this assignment are the iris dataset and the digits dataset.

2.1 Data Description

2.1.1 Iris Dataset

This data sets consists of 3 different types of irises': *Setosa, Versicolour*, and *Virginica*.

Classes	3
Samples per class	50
Samples total	150
Dimensionality	4

Refer to the scatterplot matrix below.

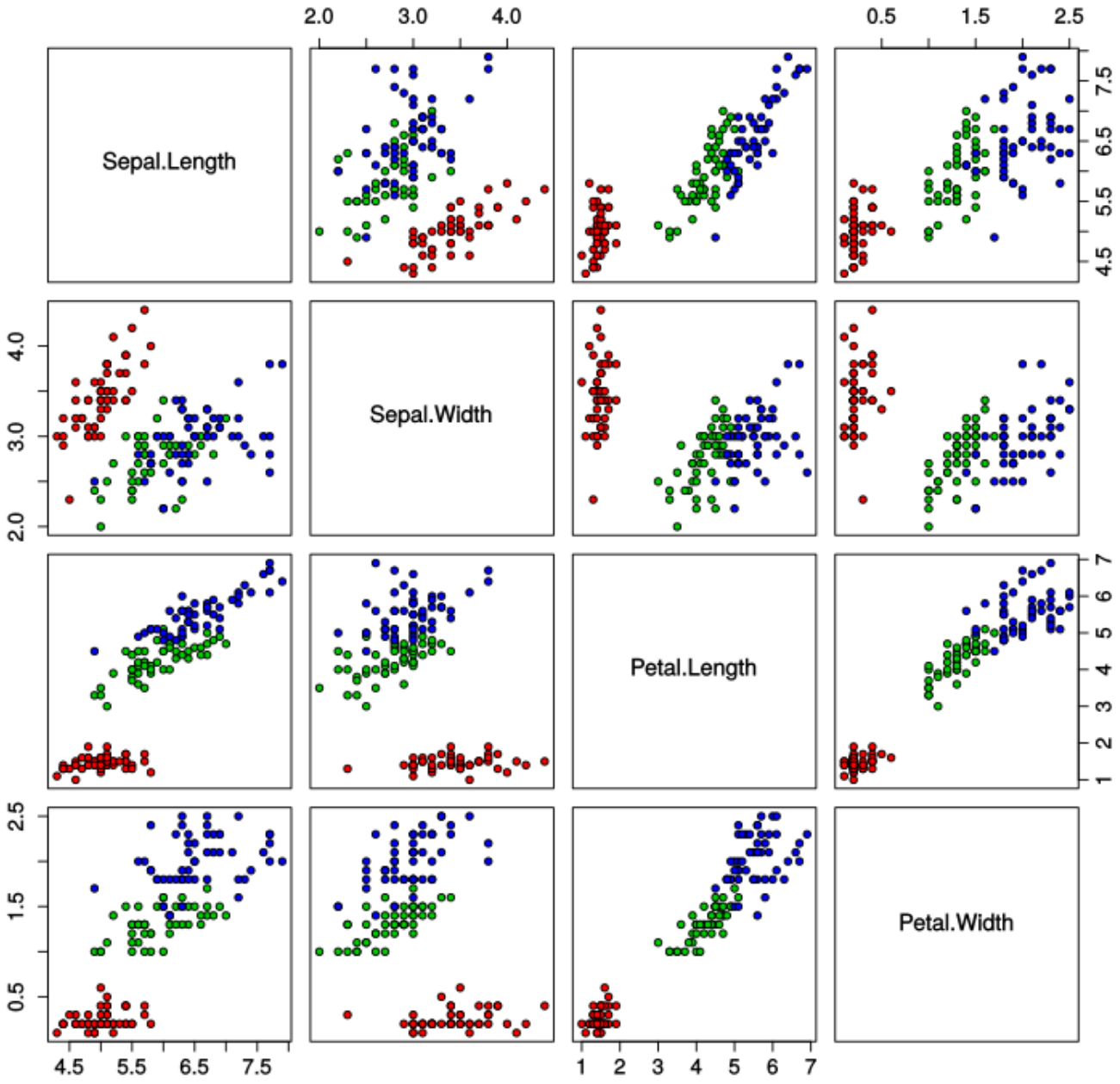


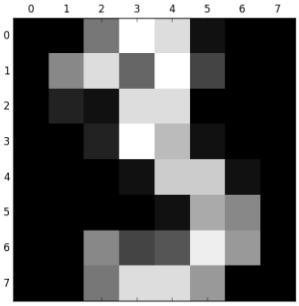
Image: Scatterplot of dataset, red = Setosa, green = Versicolor, blue = Virginica [Source: [3]]

2.1.2 Digits Dataset

Each data point in this dataset is an 8x8 image of a digit. There are a total of 10 classes which are digits 0 – 9. Here is an example visualization of digit 3 from the dataset:

Dataset Description:

Classes	10
Samples per class	~180
Samples total	1797
Dimensionality	64



2.2 Algorithm Description

2.2.1 2-Class Logistic Regression

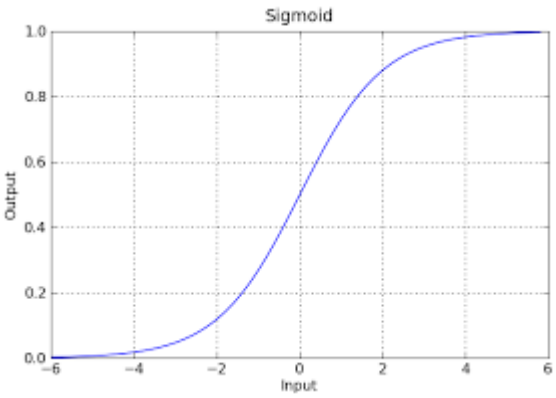
Program Flow:

1. Import the Iris dataset
2. Take a subset of the dataset so that the data includes only two classes and two features
3. Use *ploy.fit_transform* to map the data to higher dimension feature space
4. Shuffle the input data
5. Use 10-fold cross validation to create training and test datasets
6. Train on the train dataset
    a. Start with an initial guess for theta
    b. Using gradient descent iterate till theta converges
    c. The logistic regression hypothesis is defined as

$$h_\theta(x) = g(\theta^T x),$$

Where the function g is the sigmoid function. It is defined as:

$$g(z) = \frac{1}{1 + e^{-z}}.$$



    d. In each iteration adjust theta by subtracting learning rate * gradient of the cost function

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

7. Classify the test data
   Considering a threshold of 0.5 using the sigmoid logistic function, Classification is done by:

$$y = \begin{cases} +1, & \text{if } p(y = +1|\overrightarrow{x}) \geq 0.5 \\ -1, & \text{otherwise} \end{cases}$$

8. Evaluate using the confusion matrix

2.2.2 K-Class Logistic Regression

The Algorithm for K-Class Regression is very similar to the 2-Class, except the Hypothesis is defined by the softmax function instead of the sigmoid function

Program Flow:

1. Import the Iris dataset
2. Use *ploy.fit_transform* to map the data to higher dimension feature space
3. Shuffle the input data
4. Use 10-fold cross validation to create training and test datasets
5. Train on the train dataset
    a. Start with an initial guess for theta
    b. Using gradient descent iterate till theta converges
    c. The logistic regression hypothesis is defined as

$$h_\theta(x) = g(\theta^T x),$$

Where the function g is the softmax function. It is defined as:

$$P(y = j|\mathbf{x}) = \frac{e^{\mathbf{x}^\mathsf{T}\mathbf{w}_j}}{\sum_{k=1}^{K} e^{\mathbf{x}^\mathsf{T}\mathbf{w}_k}}$$

    d. In each iteration adjust theta by subtracting learning rate * gradient of the cost function

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$$

6. Classify the test data
   Considering a threshold of 0.5 using the sigmoid logistic function, Classification is done by:

$$y = \begin{cases} +1, & \text{if } p(y = +1|\overrightarrow{x}) \geq 0.5 \\ -1, & \text{otherwise} \end{cases}$$

   Classification was done by using *One Vs All* method.
7. Evaluate using the confusion matrix

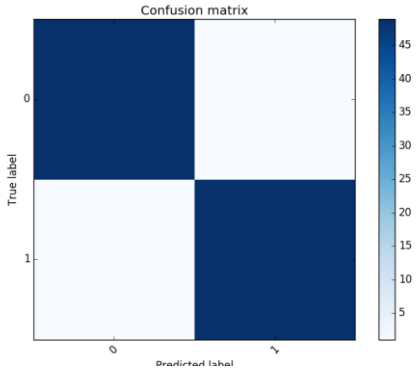
2.2.3 Neural Networks

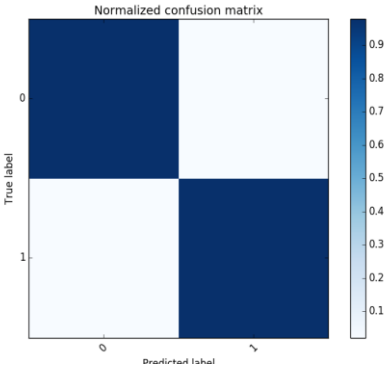
Program Flow:

1. Import the digits dataset
2. Take a subset of the dataset to include only three classes
3. Scale the input data
4. Build a model with n number of nodes in the hidden layer
   Guess random values for weights W and V
5. Train the model based on input data
   Each node is a logistic regression unit, the training is the same as the above algorithm
6. Classify the test data
   The activation of the output layer gives the class of the input
7. Evaluate the predicted data using a confusion matrix

3. Results

2-Class Logistic Regression





accuracy : 0.98
precision : 0.98
recall : 0.98
F-Measure 0.98
Confusion matrix
[[49 1]
[1 49]]
Normalized confusion matrix
[[0.98 0.02]
[0.02 0.98]]

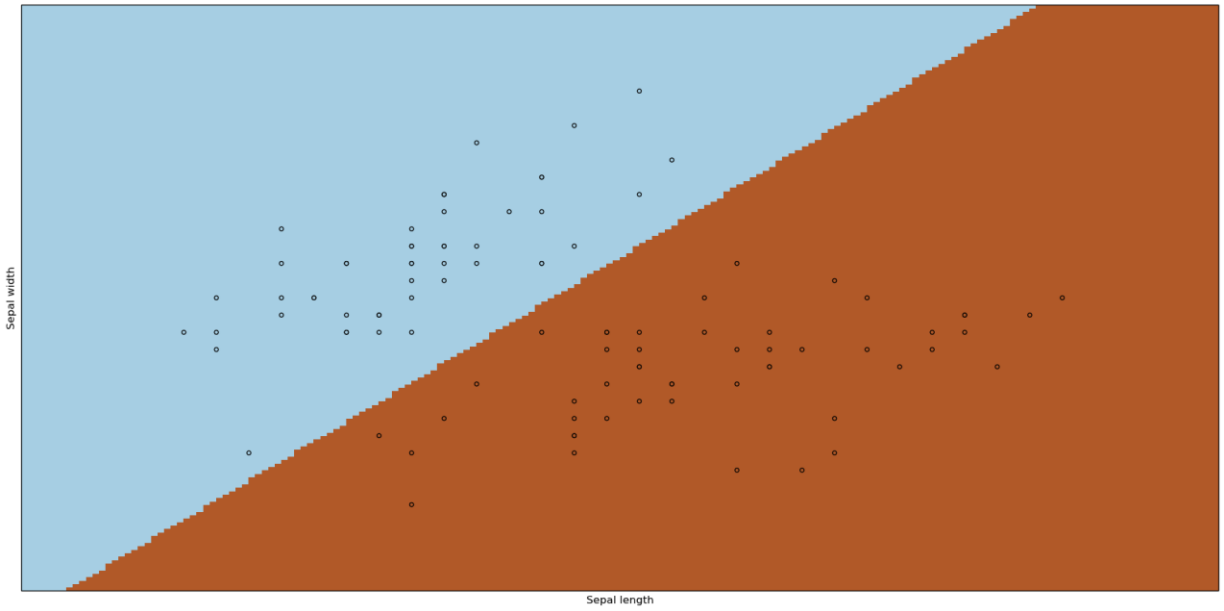
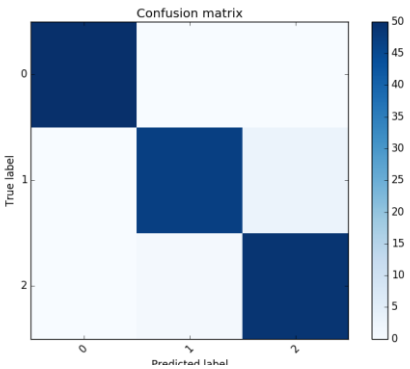
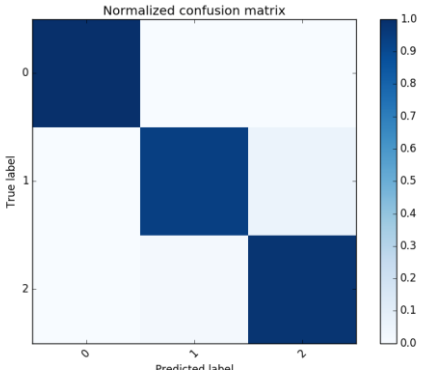


Image: Decision Boundary

K-Class Logistic Regression





Confusion matrix
[[50 0 0]
[0 47 3]
[0 1 49]]
Normalized confusion matrix
[[1. 0. 0.]
[0. 0.94 0.06]
[0. 0.02 0.98]]

Neural Network

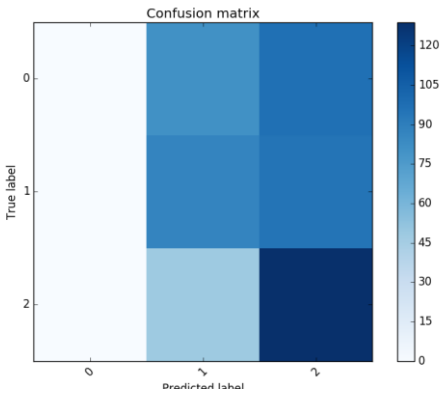
Number of Hidden Layer nodes: 2

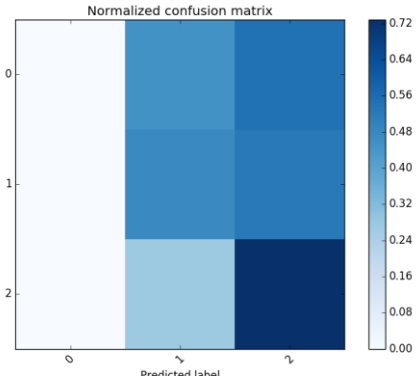
Confusion matrix

[[0 81 97]
 [0 87 95]
 [0 48 129]]

Normalized confusion matrix

[[0. 0.46 0.54]
 [0. 0.48 0.52]
 [0. 0.27 0.73]]





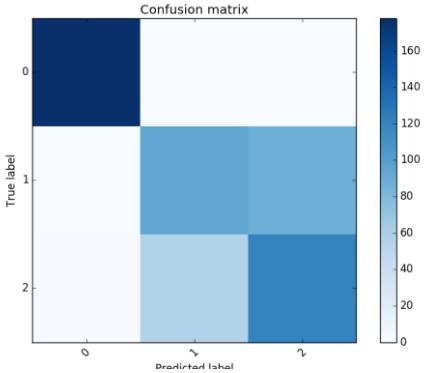
Number of Hidden Layer nodes: 4

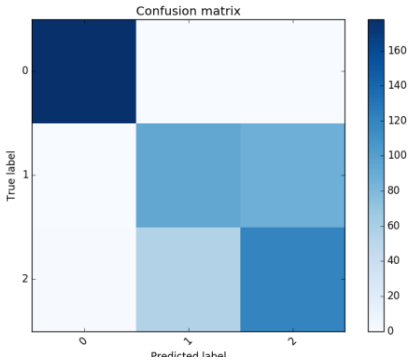
Confusion matrix

[[178 0 0]
 [0 94 88]
 [1 56 120]]

Normalized confusion matrix

[[1. 0. 0.]
 [0. 0.52 0.48]
 [0.01 0.32 0.68]]





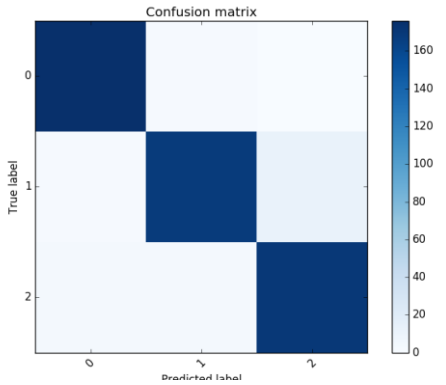
Number of Hidden Layer nodes: 6

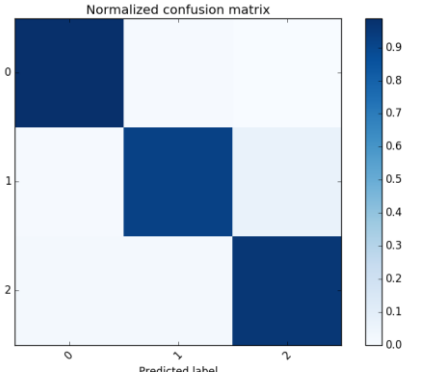
Confusion matrix

[[176 2 0]
 [2 168 12]
 [3 4 170]]

Normalized confusion matrix

[[0.99 0.01 0.]
 [0.01 0.92 0.07]
 [0.02 0.02 0.96]]





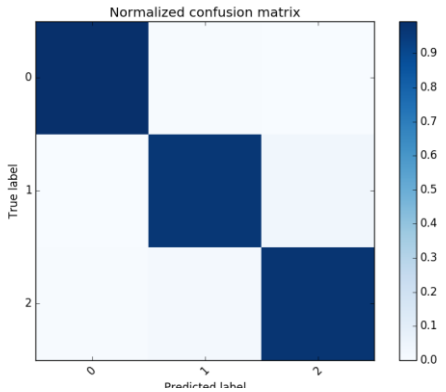
Number of Hidden Layer nodes: 8

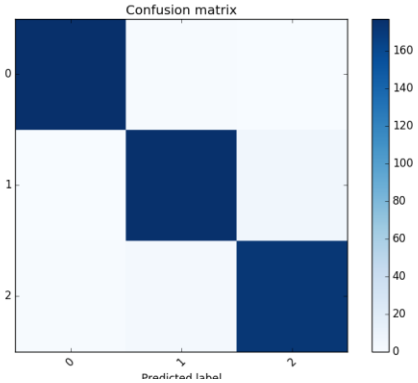
Confusion matrix

[[177 1 0]
 [0 176 6]
 [1 4 172]]

Normalized confusion matrix

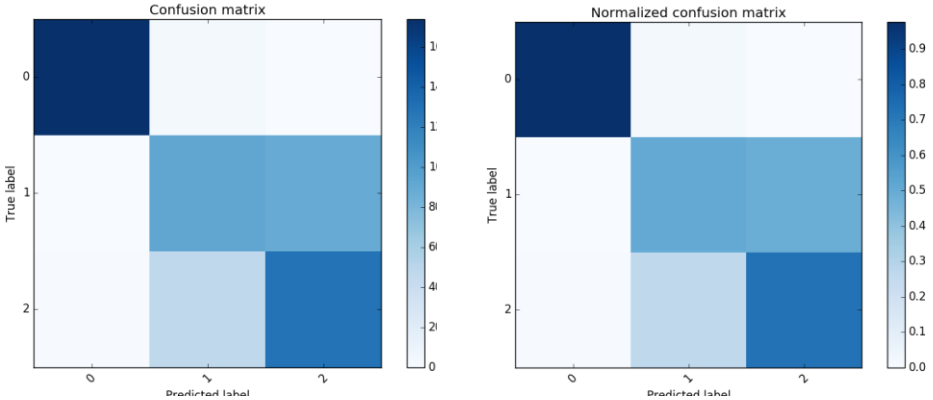
[[0.99 0.01 0.]
 [0. 0.97 0.03]
 [0.01 0.02 0.97]]





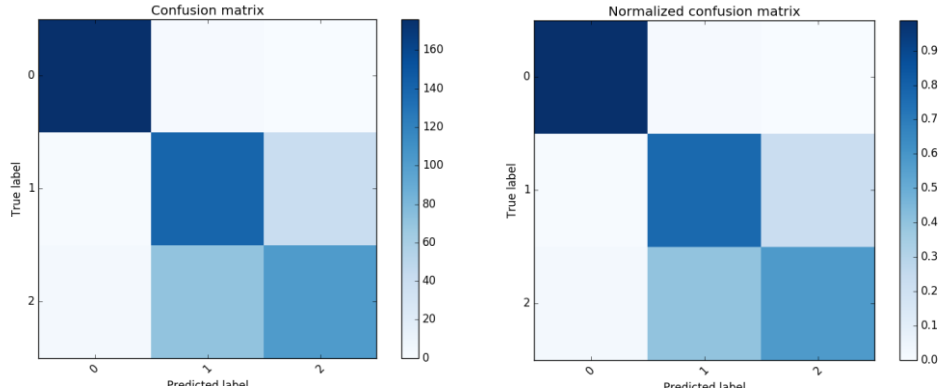
Number of Hidden Layer Nodes: 4, Momentum = 0.02

Confusion matrix, without normalization
[[174 4 0]
[0 93 89]
[1 47 129]]
Normalized confusion matrix
[[0.98 0.02 0.]
[0. 0.51 0.49]
[0.01 0.27 0.73]]



Number of Hidden Layer Nodes: 4, Momentum = 0.04

Confusion matrix, without normalization
[[176 2 0]
[1 140 41]
[3 71 103]]
Normalized confusion matrix
[[0.99 0.01 0.]
[0.01 0.77 0.23]
[0.02 0.4 0.58]]



4. Interpretation

The performance of the classifier improves with increase in number of nodes in the hidden layer. The hidden layer's job is to transform the inputs into something that the output layer can use. The output layer transforms the hidden layer activations into whatever scale the output as to be.

A feed-forward neural network applies a series of functions to the data. In this case it applies the sigmoid function in the hidden layer and the softmax function in the output layer. When we increase the number of nodes in the hidden layer to a number great than the number of nodes in the input layer, it maps the input to higher dimension feature space. When we reduce the number of nodes in the hidden layer to a number lesser than the number of nodes in the input layer, it is equivalent to dimensionality reduction. This is useful to remove dimensions that do not contribute to the output.

Increasing the momentum from 0.2 to 0.4 improved the performance of the classifier. The momentum value prevents large changes in the weight in one iteration.

References

[1] Logistic Regression Wikipedia page https://en.wikipedia.org/wiki/Logistic_regression

[2] k- Fold Cross validation function documentation: http://scikit-learn.org/stable/modules/cross_validation.html

[3] Wikipedia page: Iris Dataset Scatter Plot https://en.wikipedia.org/wiki/Iris_flower_data_set

[4] Iris Dataset: http://scikit-learn.org/stable/auto_examples/datasets/plot_iris_dataset.html

[5] Digits Dataset: http://scikit-learn.org/stable/auto_examples/datasets/plot_digits_last_image.html