Biologically Inspired Computing

Assignment 2

UiO : University of Oslo

Hunter Wilhelm Phillips (*hunterp*)

October 29, 2018

GitHub Repository at

https://github.com/robolux/Biologically_Inspired_Computing

Abstract

This project explores supervised learning with the implementation of a multilayer perceptron (MLP). The task at hand was to analyze data from electromygraphic (EMG) signals to learn eight hand gestures from a robotic prosthetic hand controller. This data set spanned the course of three days and allowed for a large dataset to produce reliable results. The MLP consisted of 1 hidden layer that was tested with 4, 6, 8, and 12 hidden neuron counts. The algorithm was tested with 10, 100, and 1000 possible iterations to test the gamut of possible situations dependent on max iteration count. The results showcased the unpredictability of machine learning with a diverse set of data to analyze. It was found that a node size of 4 provided optimal runtime when coupled with an iteration count to obtain a desired accuracy of greater than 90%. This data concluded in a complete understanding of the MLP implementation and the associated theory behind the results.

1. Execution of Code

User Input for all parameters to solve their choice of conditions pertaining to this situation

The onscreen user input instructions should guide you through the necessary steps to execute.

```
python main.py
```

To run selected test cases (as requested by prompt)

Please note that this can take a while. These datasets are available in the GitHub repository for convenience.

```
python testing.py
```

All data will be available in …/results

2. Results

The following confusion matrices were generated for discussion with the following benchmark being performed to compare runtimes.

Hidden Nodes = 4, Iterations = 10

```
Resulting Confusion Matrix
**********************************
11  0  0  0  1  0  0  0
 0 18  0  0  0  1  0  0
 0  0  9  1  0  0  0  0
 0  0  1 16  0  0  0  0
 0  0  0  0 14  0  0  0
 0  0  0  0  0  9  0  0
 0  0  0  0  0  0 16  0
 1  0  0  0  0  0  0 13


Class Prediction Percentages
**********************************
1 : 91.6666666667%
2 : 100.0%
3 : 90.0%
4 : 94.1176470588%
5 : 93.3333333333%
6 : 90.0%
7 : 100.0%
8 : 100.0%
The average percentage is 94.8897058824%
```

Hidden Nodes = 4, Iterations = 100

```
Resulting Confusion Matrix
**********************************
11  0  0  0  1  0  0  0
 0 18  0  0  0  1  0  0
 0  0  8  0  1  0  0  1
 0  0  1 16  0  0  0  0
 0  0  0  0 14  0  0  0
 0  0  0  0  0  9  0  0
 0  0  0  0  0  1 15  0
 1  0  0  0  0  0  0 13


Class Prediction Percentages
**********************************
1 : 91.6666666667%
2 : 100.0%
3 : 88.8888888889%
4 : 100.0%
5 : 87.5%
6 : 81.8181818182%
7 : 100.0%
8 : 92.8571428571%
The average percentage is 92.8413600289%
```

Hidden Nodes = 4, Iterations = 1000

```
Resulting Confusion Matrix
**********************************
10  0  0  0  2  0  0  0
 0 17  0  0  0  2  0  0
 0  0 10  0  0  0  0  0
 0  1  0 15  0  0  0  1
 0  0  0  0 14  0  0  0
 0  1  0  0  0  8  0  0
 0  0  0  0  0  0 16  0
 1  0  0  0  0  0  0 13


Class Prediction Percentages
**********************************
1 : 90.9090909091%
2 : 89.4736842105%
3 : 100.0%
4 : 100.0%
5 : 87.5%
6 : 80.0%
7 : 100.0%
8 : 92.8571428571%
The average percentage is 92.5924897471%
```

Hidden Nodes = 6, Iterations = 10

```
Resulting Confusion Matrix
**********************************
11  0  0  0  1  0  0  0
 0 18  0  0  0  1  0  0
 0  0  9  1  0  0  0  0
 0  0  0 16  1  0  0  0
 0  0  0  0 14  0  0  0
 0  0  0  0  0  9  0  0
 0  0  0  0  0  2 14  0
 1  0  0  0  0  0  0 13


Class Prediction Percentages
**********************************
1 : 91.6666666667%
2 : 100.0%
3 : 100.0%
4 : 94.1176470588%
5 : 87.5%
6 : 75.0%
7 : 100.0%
8 : 100.0%
The average percentage is 93.5355392157%
```

Hidden Nodes = 6, Iterations = 100

```
Resulting Confusion Matrix
**********************************
11  0  0  0  1  0  0  0
 0 18  0  0  0  1  0  0
 0  0 10  0  0  0  0  0
 0  0  0 16  0  0  0  1
 0  0  0  0 14  0  0  0
 0  0  0  0  0  9  0  0
 0  0  0  0  0  1 15  0
 1  0  0  0  0  0  0 13


Class Prediction Percentages
**********************************
1 : 91.6666666667%
2 : 100.0%
3 : 100.0%
4 : 100.0%
5 : 93.3333333333%
6 : 81.8181818182%
7 : 100.0%
8 : 92.8571428571%
The average percentage is 94.9594155844%
```

Hidden Nodes = 6, Iterations = 1000

```
Resulting Confusion Matrix
**********************************
11  0  0  0  1  0  0  0
 0 18  0  0  0  1  0  0
 0  0  9  0  0  0  1  0
 0  0  0 16  0  0  0  1
 0  0  0  0 14  0  0  0
 0  0  0  0  0  8  1  0
 0  0  0  2  0  2 12  0
 1  0  0  0  0  0  0 13


Class Prediction Percentages
**********************************
1 : 91.6666666667%
2 : 100.0%
3 : 100.0%
4 : 88.8888888889%
5 : 93.3333333333%
6 : 72.7272727273%
7 : 85.7142857143%
8 : 92.8571428571%
The average percentage is 90.6484487734%
```

Hidden Nodes = 8, Iterations = 10

```
Resulting Confusion Matrix
**********************************
11  0  0  0  1  0  0  0
 0 16  0  0  0  3  0  0
 0  0  9  1  0  0  0  0
 0  1  0 15  1  0  0  0
 0  0  0  0 14  0  0  0
 0  0  0  0  0  9  0  0
 0  0  0  1  0  2 13  0
 1  0  0  0  0  0  0 13


Class Prediction Percentages
**********************************
1 : 91.6666666667%
2 : 94.1176470588%
3 : 100.0%
4 : 88.2352941176%
5 : 87.5%
6 : 64.2857142857%
7 : 100.0%
8 : 100.0%
The average percentage is 90.7256652661%
```

Hidden Nodes = 8, Iterations = 100

```
Resulting Confusion Matrix
**********************************
 0  0  0  0 10  0  0  2
 0 18  0  0  0  1  0  0
 0  0  0  1  2  0  1  6
 0  0  0 16  1  0  0  0
 0  0  0  0 14  0  0  0
 0  0  0  0  0  9  0  0
 0  0  0  0  0  2 14  0
 0  0  0  0  0  0  0 14


Class Prediction Percentages
**********************************
1 : 0%
2 : 100.0%
3 : 0%
4 : 94.1176470588%
5 : 51.8518518519%
6 : 75.0%
7 : 93.3333333333%
8 : 63.6363636364%
The average percentage is 59.742399485%
```

Hidden Nodes = 8, Iterations = 1000

```
Resulting Confusion Matrix
**********************************
11  0  0  0  1  0  0  0
 0 18  0  0  0  1  0  0
 0  0  9  0  0  0  1  0
 0  0  0 15  1  0  0  1
 0  0  0  0 14  0  0  0
 0  0  0  0  0  9  0  0
 0  0  0  0  0  1 15  0
 1  0  0  0  0  0  0 13


Class Prediction Percentages
**********************************
1 : 91.6666666667%
2 : 100.0%
3 : 100.0%
4 : 100.0%
5 : 87.5%
6 : 81.8181818182%
7 : 93.75%
8 : 92.8571428571%
The average percentage is 93.4489989177%
```

Hidden Nodes = 12, Iterations = 10

```
Resulting Confusion Matrix
**********************************
11  0  0  0  1  0  0  0
 0 17  0  0  0  2  0  0
 3  0  0  4  0  0  1  2
 0  0  0 16  1  0  0  0
 0  0  0  0 14  0  0  0
 0  0  0  0  0  9  0  0
 0  0  0  0  0  1 15  0
 1  0  0  0  0  0  0 13


Class Prediction Percentages
**********************************
1 : 73.3333333333%
2 : 100.0%
3 : 0%
4 : 80.0%
5 : 87.5%
6 : 75.0%
7 : 93.75%
8 : 86.6666666667%
The average percentage is 74.53125%
```

Hidden Nodes = 12, Iterations = 100

```
Resulting Confusion Matrix
**********************************
11  0  0  1  0  0  0  0
 0 16  0  0  0  3  0  0
 2  0  0  5  0  0  0  3
 0  0  0 16  1  0  0  0
 0  0  0  0 14  0  0  0
 0  0  0  0  0  9  0  0
 0  0  0  0  0  1 15  0
 1  0  0  0  0  0  0 13


Class Prediction Percentages
**********************************
1 : 78.5714285714%
2 : 100.0%
3 : 0%
4 : 72.7272727273%
5 : 93.3333333333%
6 : 69.2307692308%
7 : 100.0%
8 : 81.25%
The average percentage is 74.3891004829%
```

Hidden Nodes = 12, Iterations = 1000

```
Resulting Confusion Matrix
**********************************
11  0  0  0  1  0  0  0
 0 18  0  0  0  1  0  0
 0  0  9  0  0  0  1  0
 0  0  0 16  1  0  0  0
 0  0  0  0 14  0  0  0
 0  0  0  0  0  9  0  0
 0  0  0  0  0  1 15  0
 1  0  0  0  0  0  0 13


Class Prediction Percentages
**********************************
1 : 91.6666666667%
2 : 100.0%
3 : 100.0%
4 : 100.0%
5 : 87.5%
6 : 81.8181818182%
7 : 93.75%
8 : 100.0%
The average percentage is 94.3418560606%
```

Benchmark Comparisons between varying parameters

Case	Computation Time (s)
n = 4, I = 10	49.2578539848
n = 4, I = 100	95.499423027
n = 4, I = 1000	213.475446224
n = 6, I = 10	128.268749952
n = 6, I = 100	88.4659719467
n = 6, I = 1000	293.713326931
n = 8, I = 10	93.4126780033
n = 8, I = 100	111.951656818
n = 8, I = 1000	366.953434944
n = 12, I = 10	236.344048977
n = 12, I = 100	256.656649113
n = 12, I = 1000	512.682374001

3. Discussion

The data from the results section can be initially summarized as follows:
With a node count of 4, the overall accuracy was approximately: 95% for 10 iterations, 93% for 100 iterations, and 93% for 1000 iterations.
With a node count of 6, the overall accuracy was approximately: 94% for 10 iterations, 95% for 100 iterations, and 91% for 1000 iterations.
With a node count of 8, the overall accuracy was approximately: 91% for 10 iterations, 60% for 100 iterations, and 93% for 1000 iterations.
With a node count of 12, the overall accuracy was approximately: 75% for 10 iterations, 74% for 100 iterations, and 94% for 1000 iterations.

It is important to note that when these tests were run multiple times, the results varied due to the unpredictable nature of the MLP. This is just one example of the results that can be obtained and extensive testing needs to be completed to fully characterize the algorithmic process.

It can be seen that with a node count of 4 & 6, the MLP has a steady increase in accuracy with the coupled iteration count to a realistic degree. This same trend is seen with a node count of 8,

although it does not reach the same overall accuracy. With a node count of 8 it also appears to have a bad classifying trend for the 100-iteration round on two EMG signal types. This is an expected side effect though since the classifier can dig into valleys unexpectedly. The trend breaks when the node count reaches 12 and the results are all over the place. The best results of 95% accuracy are within the node counts of 4 & 6 and it shows how unpredictable this algorithm can be by the node count of 8 holding the lowest of 60%.

In the confusion tables the most commonly mislabeled data types were 3 and 8. This occurred in almost all tests and shows that they must be easily mistaken for each other by the algorithm.

The benchmarks show that the computation time is greatly increased when the iteration count gets large. This is mainly dependent on the for-loop nature of code, ballooning upon exponential gain. This also plays apart in the conclusion of this project as runtime is critical in code execution.

4. Conclusion

Through this assignment, the facets of understanding a multilayer perceptron network were discovered. The MLP was successfully implemented and the results analyzed to form a basis for the conclusion. It appears that having more nodes does not necessarily mean a better result is obtained. What most likely happens is that the management of too many nodes coupled with large iteration sizes allows for divergence of certain solution classification in particular cases. When computational time is taken into consideration it would make the most sense from the results found to select a node size of 4. With this choice in mind, the amount of iterations that are desired to be completed would be decided based on amount of program runtime available at the user's disposal. It is interesting to note though that on certain runs of the program this conclusion would have been altered since this algorithm tends to have an unpredictable behavior. This is common within machine learning algorithms, leading to trial and error being the best solution at advancing knowledge on a particular problem. Through continued experience with these types of algorithms an individual is able to make an educated guess on input parameters to reach the desired classification result. The assignment concludes in a more complete understanding of machine learning problems and the associated algorithmic development to solve them.