Fall 2021 COSC 3P71 Artificial Intelligence: Assignment 3

Instructor: B. Ombuki-Berman
Course Coordinator: Tyler Crane
Teaching Assistants: Alanna McNulty, Nicholas Aksamit, Liam McDevitt

Available Date: November 17, 2021
Due Date: Friday December 3rd, 2021@NOON

Option A – Parity Bit Implementation

Every byte of data that is stored in a system memory contains 8 bits of data, each bit is either a zero or a one. It is possible to count up the number of zeros or ones in a byte. For example, the byte 11011010 has 5 ones and 3 zeros. The byte 00101000 has 2 ones and 6 zeros. Hence, some bytes will have an even number of ones, and some will have an odd number. When parity checking is enabled, each time a byte is written into memory, a parity generator/checker (aka logic circuit) examines the byte and determines whether the data byte had an even or odd number of ones. If it had an even number of ones, the ninth (parity) bit is set to a one, otherwise it is set to a zero. The outcome is that no matter how many ones there were in the original eight data bits, there are an odd number of ones when all the 9 bits are looked at all together. This is called odd parity. On the other hand, if the byte had an odd number of ones, and the parity bit is set to a one, the system would be checking for even parity. (Note: the standard in PC memory is odd parity) **Goal:** Train a standard feed-forward neural network to act as a parity checking system.

Task: Implement a standard feed forward neural network and the backpropagation training algorithm from scratch. Demonstrate your implementation by training a network to act as a parity checking system.

The Dataset

The network will implement a 4-bit parity checking system (1/2 a byte, for example, 0101), generating an odd parity bit as the output. Thus, each training example will consist of 4 input values, with each value either zero or one, and a single target value, also either zero or one. This means there are $2^4 = 16$ possible training examples, and the network is expected to memorize the correct output for each one. You will need to generate your own training data as described above. Below is a sample of the training examples.

Input Pattern	# of 1s	Expected Output	Total Number of 1s in Input and Output
0 0 0 0	0	1	1
0 1 0 1	2	1	3
1 0 0 0	1	0	1
1 0 1 1	3	0	3

As you can see above, this dataset will require 4 input nodes in your neural network and a single output node.

Implementation Details

- The implementation should support training a standard 3-layer feed-forward neural network with backpropagation. At minimum it should support an input layer, a single hidden layer, and an output layer. You are not required to support multiple hidden layers.

- The implementation should use the Logistic function as the activation function for the nodes in the hidden and output layers and Mean Squared Error as the loss function. The input nodes can simply output the data passed to the network.
- The implementation should support changing the learning rate, number of input nodes, number of hidden nodes, and number of output nodes. It is acceptable if a recompile is required to change parameters.
- It is up to you how your program acquires the dataset described above. Some options include writing a function to generate the data, typing the data into .csv files and having your program read those files, or even typing the data into some data structure within your program.

Expected Output

When run, the program should train a standard feed forward neural network for some number of epochs (also referred to as iterations) using the backpropagation algorithm on the parity bit dataset. It is up to you to decide on the learning rate, hidden layer size, and number of epochs/iterations. As a starting point consider using a hidden layer size of 8 and 10000 epochs/iterations. The input layer size and output layer size are of course set to fit our dataset.

While training, the program should periodically output the iteration number, and the current Mean Squared Error (for example this could be output every 200 epochs/iterations). After training is complete (i.e., the chosen number of epochs/iterations is performed) the program should output the number of hidden nodes used, the learning rate and the final Mean Square Error. Finally, the program should output each training example, along with the network's output value for that example and the expected output.

Your electronic submission should include:

- Your source code
- A compiled copy of the program where appropriate (For example .class files for java or .py files for python)
- Instructions for running and compiling your program and changing parameters
- Sample output from running your program

Option B – Analysis and Report using WEKA

WEKA is a free to use, open-source software written in Java that provides various models and algorithms for the use of data processing, implementing machine learning algorithms and data visualization. Several popular machine learning algorithms are available to use and require minimal implementation work to create a model for many machine learning problems. Your task for this option is to implement a neural network in WEKA for performing data classification on two popular machine learning classification problems. You will then perform a detailed comparison and analysis of the impact that the parameters of a neural network have on the performance of the model.

The Datasets

There are two data sets that you will use for this option, the Iris data set and the Wisconsin Breast Cancer dataset. The data files can be found at the following links:

- Breast Cancer Wisconsin Data Set
  - https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Original%29
- Iris Data Set
  - https://archive.ics.uci.edu/ml/datasets/Iris

The iris data set contains data regarding three different classifications of iris plant. There are four attributes about each plant described, each pertaining to different measurements such as petal length or sepal width. The data set contains 150 data vectors, 50 for each class. There are three possible output classes, Iris-setosa, Iris-versicolor, and Iris-virginica. Inside the data folder, you will use the iris,data file.

The Wisconsin breast cancer data set contains information pertaining to different cell nuclei and breast cancer clusters, with the goal to be able to use these attributes to determine whether a tumour was a malignant or a benign tumour. This data set had 10 attributes, each one containing integer data classifying an attribute on a scale from one to ten, and 699 total data vectors. There are two possible output classes, benign (stored as a 2), and malignant (stored as a 4). Inside the data folder, you will use the breast-cancer-wisconsin.data file.

Getting Setup with Weka

There are a few steps required to get set up with WEKA and to begin running your experiments. The entire process will be covered in tutorial, but a summary will be included here if you missed it.

1. Download and install WEKA from https://www.cs.waikato.ac.nz/ml/weka/
2. Obtain the data files from the UCI repository
   a. Change the file extension to .csv instead of .data
   b. Add a new line at the top of the file and name each column/attribute of the data
3. Open WEKA and click the 'Explorer' button.
4. Open your first data file using open file (you may need to manually tell it to look for csv files).
   a. Any transformations you need to do should be done before moving to the classify section.
5. In the classify section, select the Multilayer Perceptron classifier.
   a. Set your parameters to your desired values.
   b. Ensure that the class attribute is selected correctly.
   c. When running experiments, make sure to change the seed parameter for each run.
6. Click start and wait for the run to complete.
   a. It may take a couple minutes, the bird in the bottom right shows whether its still running.

You are welcome to use an open-source library other than WEKA to perform your experiments if you prefer, but WEKA is the library that will be covered in tutorial and that the TAs will be familiar with when you have questions.

The Analysis

Run your experiments and collect performance data for each unique parameter set tested. At the end of each run, a lot of information is available for analysis, and it is up to you to decide which information is useful to

keep. Each parameter set should use a minimum of 15 runs to ensure statistical accuracy. It is up to you what experiments you perform, but at a minimum your experiments should modify and analyze:

- The learning rate
- The number of hidden nodes (a bit tricky to change, see tutorial video)
- The momentum factor
- The number of epochs to train for
- The number of folds used in cross validation
    - Also compare to using a percentage split approach

The Report

The results are to be handed in via a technical report using an IEEE style file format with the following headings: (including an abstract)

1) Abstract, Introduction & Problem definition

- What is the goal of this work? Outline of rest of paper, Etc.
- Clear definition of the problem and data being used, applicability of neural networks for this problem.
- Why is the work you are doing important?

2) Background (Neural networks and Supervised Learning).

This section will need to include some Math to be properly/clearly explained.

- Paragraph describing feed forward networks in general, and another describing the learning rules used.
- Section for each of the chosen learning rules. Each section should explain weight update, as well as how learning is accomplished.
- What is similar/different between the networks? A small pseudocode algorithm may be useful.
- After reading this section the reader should be clear as to how the learning rules work. Explain generalization techniques, problems, etc. here as well.

3) Methods of Analysis

- What are you going to analyze?
- How are you going to analyze it?
- This part will probably have a separate subsection for each of the analytical methods used. Each subsection should be about 2-3 paragraphs (in most cases), explaining exactly how the method is used (try not to say the words neural network in this section – i.e. Just explain the method, worry about the neural net in the next section). These subsections are therefore going to be explaining statistical stuff (no need to explain average, std. deviation, and other straight forwards). Choose whatever you like to make your point clear, possible things include:
    - Comparison of means
    - t-tests/tests for normality/Mann-Whitney U tests
    - Confidence intervals
    - ANOVA/Kruskall-Wallis
    - Contingency table analysis

4) Results and Discussion

- Here you will outline the parameters used and show plots, etc. of your results.
- Also, explain (using the methods of analysis) what the results lead you to conclude.
- Devote a good amount of time and space here.
- Ensure that the effect of the learning rule, number of hidden nodes, etc…. are all tested.

5) Conclusion

- Summarize what has been done in your paper and what experiments you performed.
- Summarize the conclusions you made throughout your results and discussions.

6) References

- Include the references you used for work, including book (s) and research articles.

You must use the IEEE Conference style found at
http://www.ieee.org/conferences_events/conferences/publishing/templates.html

The report must be written in a clear and concise manner.

Your submission for this option will only need to include your final report as either a doc or pdf file.

Hand in and submission notes:

- Submission will be done through Sakai.
- Unity is not allowed for this assignment by any students. We recommend using Java, Python, or C#. Your source code should be presented in a format that can be easily compiled. All assignments must include detailed instructions for the marker to compile and run the submission. Failure to provide adequate explanation and documentation may result in a submission not being graded.
- Please note that the virtual COMMONS is available to all students at Brock. If you are unsure of whether or not the markers will be able to test your submission you can test it on the virtual COMMONS. Machines in the virtual COMMONS have IDEs for Java, Python, and C#.
- Any questions/concerns etc., regarding the grading of any assignment MUST be raised within 7 days of graded assignment hand-back. In this case, please send your concerns/questions to Course Coordinator Tyler at: "tcrane2@brocku.ca". To better serve you, please don't send multiple queries on the same topic to Tyler, Professor and other TAs. Tyler will be the point of contact for any such queries and the Professor will receive them all at once from Tyler.
- We will be creating a FAQ for this class and adding some of your questions there (while maintaining your privacy). In this case, please send all your email questions to Tyler at "tcrane2@brocku.ca". Tyler will try his best to respond to all your questions as soon as possible, so please wait 48 hours before you contact course instructor if Tyler has not responded. Note: Tyler will be working closely with the course instructor, and she will have access to all your questions.