CSC426-01 Machine Learning
Final Project
Due: May 16, 2021 by 11:59PM

Late Policy for this Final Project: If submitted on May 17, which is the day after the due date, a 10% late penalty will apply. No submissions will be accepted after May 17. If submitted after May 17, a zero will be assigned as the grade.

Final Project Title: Neural Networks Implementation and Analysis

1 Project Description

In this project you will get hands-on experience implementing a multilayer neural network learner from scratch using the backpropagation pseudocode provided in class. You will also get hands-on experience investigating hidden layer representations. You will also gain hands-on experience working in a team to create deliverables containing source code, documentation, statistical results files, plots of results, and written reports with analysis of results. You will tie together many of the things you've learned during the semester in order to:

- implement the backpropagation algorithm from scratch to implement multilayer neural network learning,
- gain experience analyzing hidden layer representations, and
- work productively in a team to produce professional well-documented software deliverables and reports.

2 Teams

Team Assignments will be emailed to students on May 8, 2021.

3 Tasks

3.1 T1: Implement the backpropagation neural network learning algorithm from scratch.

Implement the backpropagation neural network learning algorithm from scratch using the programming language(s) of your choice from the following set: Python, Java, C, C++. There cannot be any software dependencies for your system that are not part of the standard installation on the TCNJ HPC system. Provide documentation and in particular, a README file explaining how to run the software from the command line.

3.2 T2: Run your backpropagation learning algorithm for a specific Learning Problem (LP) and Create data files during learning

Run your backpropagation learning algorithm on the following LP.

LP: Use a network architecture of 8 x 3 x 8, or in other words, 8 inputs, 3 hidden units, and 8 output units. Learn the target function $f(\vec{x}) = \vec{x}$, where $\vec{x}$ is a vector containing seven 0's and a single 1. The function to be learned is the identity function so the network must learn to reproduce the eight inputs at the corresponding eight output units. Use:

$D = \{((1,0,0,0,0,0,0,0),(1,0,0,0,0,0,0,0)),$
$((0,1,0,0,0,0,0,0),(0,1,0,0,0,0,0,0)),$
$((0,0,1,0,0,0,0,0),(0,0,1,0,0,0,0,0)),$
$((0,0,0,1,0,0,0,0),(0,0,0,1,0,0,0,0)),$
$((0,0,0,0,1,0,0,0),(0,0,0,0,1,0,0,0)),$
$((0,0,0,0,0,1,0,0),(0,0,0,0,0,1,0,0)),$
$((0,0,0,0,0,0,1,0),(0,0,0,0,0,0,1,0)),$
$((0,0,0,0,0,0,0,1),(0,0,0,0,0,0,0,1))\}$

as your set of training data. Set initial weights to random values in the interval (-0.1,0.1), use learning rate $\eta = 0.3$, and stop learning after 5,000 epochs, or in other words after 5,000 iterations of the outer loop of the algorithm (i.e., after 5,000 iterations through each of the eight training examples).

Add appropriate record-keeping to your source code so that you can record and analyze the progress of learning during the 5,000 epochs. Specifically, your code must create and write two types of data files:

1. SSE (Sum of Squared Errors) file (you will produce one file of this type), and

2. Hidden Unit Encoding files (you will produce eight files of this type).

One SumOfSquaredErrors file should be produced. The one SumOfSquaredErrors file should be a comma-separated values (csv) file with 5,000 lines, one line for each completed epoch of learning. There should be eight columns: SSE_output_unit1, SSE_output_unit2, ..., SSE_output_unit8. On each line you should put the value of the sum of squared errors for each of the output units over the eight training examples for the epoch of learning that was just completed. Format your numbers in the file so that they're truncated after a reasonable number of decimal places.

Eight HiddenUnitEncoding files should be produced, one for each of your input values. You can name them HiddenUnitEncoding_10000000.csv, HiddenUnitEncoding_01000000.csv, HiddenUnitEncoding_00100000.csv, ..., HiddenUnitEncoding_00000001.csv. The HiddenUnitEncoding files should be csv files with 5,000 lines, one line for each completed epoch of learning. There should be three columns: HiddenUnit1Encoding, HiddenUnit2Encoding, HiddenUnit3Encoding. On each line you should put the three values emitted by the three hidden units in your network for the relevant input value for the epoch of learning that was just completed. So, for example, HiddenUnitEncoding_10000000.csv should contain the three values emitted by the three hidden units in your network for the input $\vec{x} = (1, 0, 0, 0, 0, 0, 0, 0)$ after each of the 5,000 epochs of learning.

3.3 T3: Create graphs of your results

3.3.1 T3.1: Graph Sum of Squared Errors for each output unit

Create a single graph: "Sum of Squared Errors for each output unit". The y-axis should measure the sum of squared errors. The x-axis should measure the epoch number. There should be eight plotted lines on the graph, one plotted line for each the SSE for each of the eight output units of your network. Make sure to make each of the eight plotted lines use different

colors or different styles (e.g., long dashed, short dashed, solid, dotted, dash-dotted, etc.) so that they can be distinguished from each other. This graph should be easy to produce by using the data from the SumOfSquaredErrors file produced in Task 2.

3.3.2 T3.2: Graph Hidden Unit Encodings for each of the eight inputs

Create eight graphs: "Hidden Unit Encoding for input 10000000", "Hidden Unit Encoding for input 01000000", ..., "Hidden Unit Encoding for input 00000001". The y-axis should measure the values emitted by the three hidden units in your network for the relevant input value. The x-axis should measure the epoch number. There should be three plotted lines on the graph, one plotted line for each of the three hidden units of your network. Make sure to make each of the three plotted lines use different colors or different styles so that they can be distinguished from each other. These eight graphs should be easy to produce by using the data from the eight Hidden Unit Encoding files produced in Task 2. So, for example, the file HiddenUnitEncoding_10000000.csv should have the data to easily produce the "Hidden Unit Encoding for input 10000000" graph.

3.4 T4: Analysis

Inspect the three hidden unit values in your eight Hidden Unit Files produced in Task 2. In particular, look at the three values on the last line of the file, after the 5,000th epoch of learning. Inspect the three hidden values for each of your eight inputs (i.e., the last lines of each of your eight Hidden Unit Files). Construct a single summary HiddenRepresentationsFile that has eight lines in total. You can write code to produce this file or you can create it manually. Each line should contain one of the eight input values (e.g., 10000000), the three hidden values from the last line of the corresponding Hidden Unit File, and the output value, which is always identical to the input value since the identity function was learned. One way to think about the contents of this file is that the hidden values are an encoding that the machine learned for each input and that the machine now can encode inputs into the hidden value representations and decode the hidden representations to produce outputs. Round all your hidden values to the nearest integer. What observations can you make about the machine's hidden value encodings of the input values?

Prepare a written report of your observations and analysis regarding the hidden value encodings learned by the machine.

4 Deliverables

4.1 D1

All source code neatly documented with a README.txt file saying what is what and where everything is and giving clear usage instructions.

4.2 D2

Your one Sum of Squared Errors file and your eight Hidden Unit Encodings files that were created in T2.

4.3 D3

Your nine graphs that were created in T3.

4.4 D4

Your HiddenRepresentationsFile and written analysis from T4.

4.5 D5

Responses to the following questions:

- What was easy about this assignment?
- What was challenging about this assignment, or parts that you couldn't get working correctly?
- What did you like about this assignment?
- What did you dislike about this assignment?
- How did your team function, including details such as what each team member contributed, how the team communicated with each other, and how team software development & design was accomplished?
- What did you learn from this assignment?

4.6 D6

A main README file in plain text documenting all of your submission, explaining what's what and where it's located. The README file must also document all build instructions and command-line execution instructions for your software.

5 Submission Format

Please submit your entire package as a tar.gz file via file upload to Canvas.