Neural Network

Jongoh (Andy) Jeong / December 9, 2019

- This neural network consists of an input, hidden and output layer with different number of nodes. The number of hidden nodes is generally in between the number of input and output nodes, and for the Iris data 4 is chosen. The number of epochs and the learning rate that yield one of the best performances are empirically found to be 150 epochs and rate of 0.1, with macro-averaged accuracy of 0.987.

Build

Makefile contains options for train/test only, or both. Type in `make all` to run both one after another, or `make train` and `make test` separately.

Makefile:

```
train:
	g++ train.cpp -std=c++11 -o train.exe
	./train.exe

test:
	g++ test.cpp -std=c++11 -o test.exe
	./test.exe

all:
	g++ train.cpp -std=c++11 -o train.exe
	./train.exe
	g++ test.cpp -std=c++11 -o test.exe
	./test.exe

debug_train:
	g++ -std=c++11 -g -o trainDebug.exe train.cpp

debug_test:
	g++ -std=c++11 -g -o testDebug.exe neuraltest.cpp

clean:
	rm -f *.exe *.stackdump *.out
```

Dataset: Fisher Iris

- The experimented dataset is the Fisher Iris dataset, which is often used in machine learning for its simplicity in data and yielding good enough performance. This dataset contains class labels for the Iris plant (3), along with its attributes (4). Since there are more than 2 class labels, one hot-encoded vector notation is employed to represent the output -- for example, Setosa == [0 0 1], Versicolour == [0 1 0], Virginica == [1 0 0]. The train-test split for this 150-sample dataset is performed so that there are 100 examples for training and 50 for testing. To work with the neural network we have at hand, the values

have been normalized between 0 and 1 with precision of 3 digits after decimal point, taking into account minimium and maximum of each attribute values. The initial weights have been randomly drawn from a Gaussian distribution.

Source: https://archive.ics.uci.edu/ml/datasets/Iris (UCI Machine Learning Repository)

File Descriptions by Extension Format

- **.init**: this file is used to initialize a neural network of specified input, hidden and output nodes. The first line consists of [num_input, num_hidden, num_output] nodes in integer format. Then each of the following 'num_hidden' number of lines contains weights from the input to hidden nodes, and each of the following 'num_output' number of lines contains weights from the hidden to output nodes.
- **.train**: this file contains the data for training examples. The first line consists of [num_examples, num_input, num_output] nodes in integer format. The following lines contain 'num_input' number of inputs and 'num_output' number of output values (one hot-encoded vector) in each line.
- **.test**: same format as *.train*
- **.trained**: this file is similar to *.init* but with adjusted weights from backpropagation during training stage.
- **.result**: this file is the final output containing [True Positive, False Negative, False Positive, True Negative] with micro-averaged metrics for [accuracy, precision, recall, f1 score] in each of 'num_output' lines. The following lines contains macro-averaged statistics for the aforementioned (four) metrics.