Neural Networks

Bijoy Singh (120050087)
Sai Kiran Mudulkar(120050068)
Manik Dhar(120050006)

Lab Question

1. Let M be the number of distinct graphemes. Allocate ceiling(logM) bits for each grapheme.

2. Let N be the number of distinct phonemes. Allocate ceiling(logN) bits for each phoneme.

3. Your training data has EQUAL no. of graphemes and phonemes on either side of each item of the parallel corpus.

4. If L is the length of the grapheme sequence (correspondingly phoneme sequence), concatenate L bit vectors to form the G-P training data; If the size of the parallel corpus is K, you will have K vectors, each of size ML on the grapheme side, and of size NL on the phoneme side.

5. The G-to-P feed forward n/w will have an i/p layer of size ML, and o/p layer of size NL; similarly for P-to-G.

6. You will have to find by TRIAL AND ERROR the appropriate size of the hidden layer. Normally one hidden layer is fine, but the number of neurons in it has to be determined. Start with, say, average of ML and NL.

7. Do the usual 5-fold cross validation. Observe accuracy.

Perceptron Model

We implemented the perceptron model in the general form.

- **θ** was a part of the weights with constant input of -1
- Both the models for deciding the value of the output were allowed(step threshold and sigmoidal threshold)

Also we made the schema general enough to use directly as well in a Neural Network

Perceptron Training

Perceptron training for problems just one perceptron uses the PTA algorithm.

It takes a truth table modifies it to create appropriate classes and inverts 0 class.

It now trains using the PTA algorithm which is guaranteed to converge if the function is separable.

Perceptron Performance

A perceptron can solve the problems as one would expect:

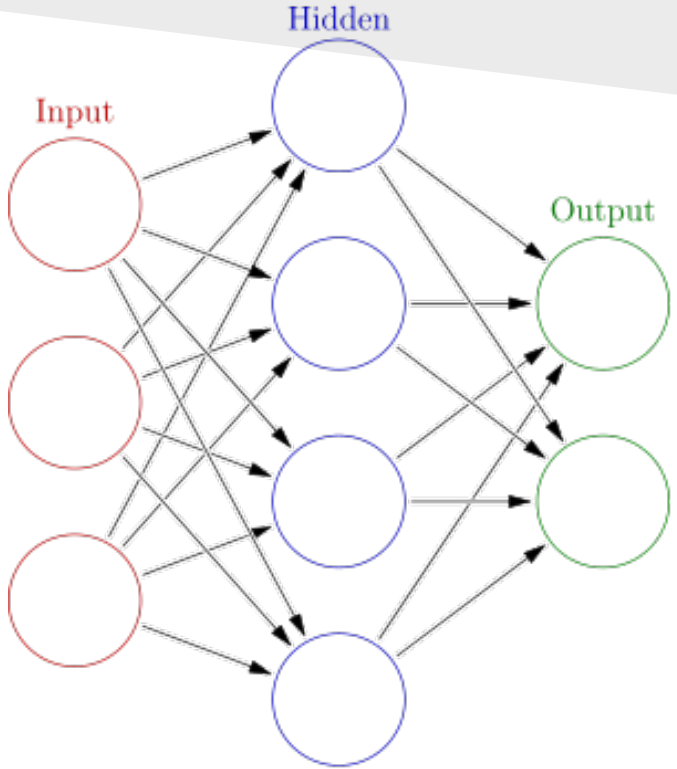
- AND
- OR
- MAJORITY

but cannot to

- PALINDROME (if it could it would be able to solve XOR, which we know it cannot)
- XOR

Neural Networks

We now used our existing infrastructure of perceptrons and using the sigmoid threshold neuron model, we used the to make a neural network



Neural Network

The user can specify

- The number of inputs, number of neurons on each row, the number of outputs, η(for Δw_{ij}) and the accuracy anticipated.
- This way the user can control the learning rate(through η) and the final result accuracy.

Neural Network Training

- We use the standard back propagation algorithm with the parameters η and accuracy(ε).
- For the smaller boolean circuits, we have trained the network until the ε becomes very small(0.1) this can be modified by the user
- The algorithm stops when the training is complete

Neural Network Performance

The neural network was tested on the following circuits and it performed as expected

3 input PALINDROME

4 input PALINDROME

XOR

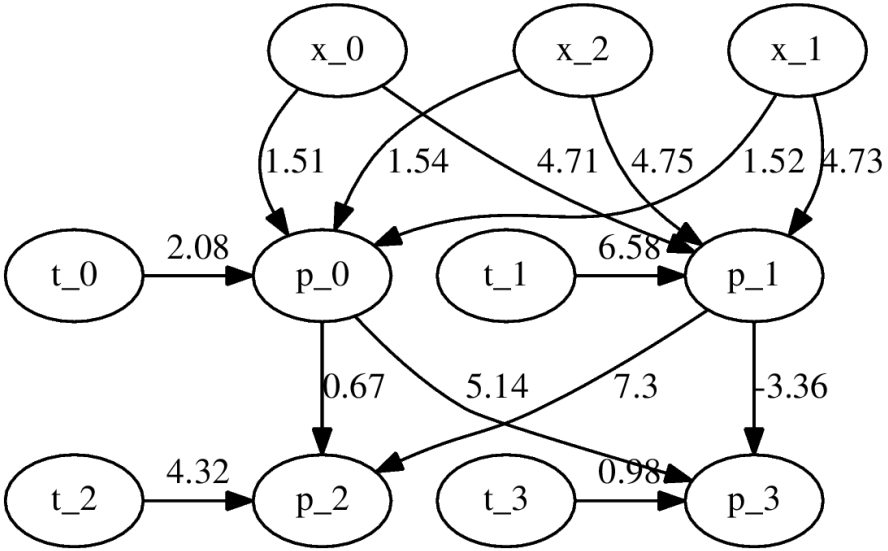
FULL ADDER

Visual Results of Neural Training

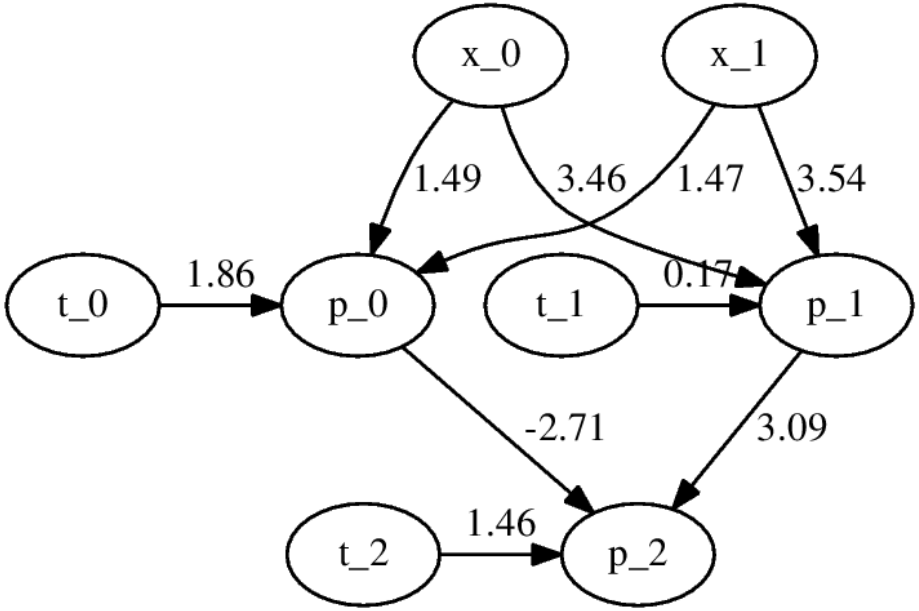
Now the results of these neural networks are a bit difficult to visualise from text. Hence we made a module to allow for UI output for the neurons

The outputs of the various neural networks are displayed

Visual Results of Neural Training

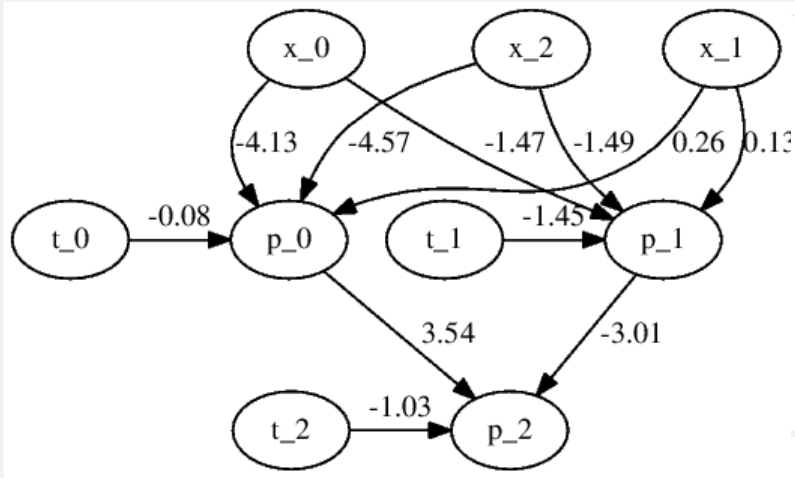


FULL ADDER

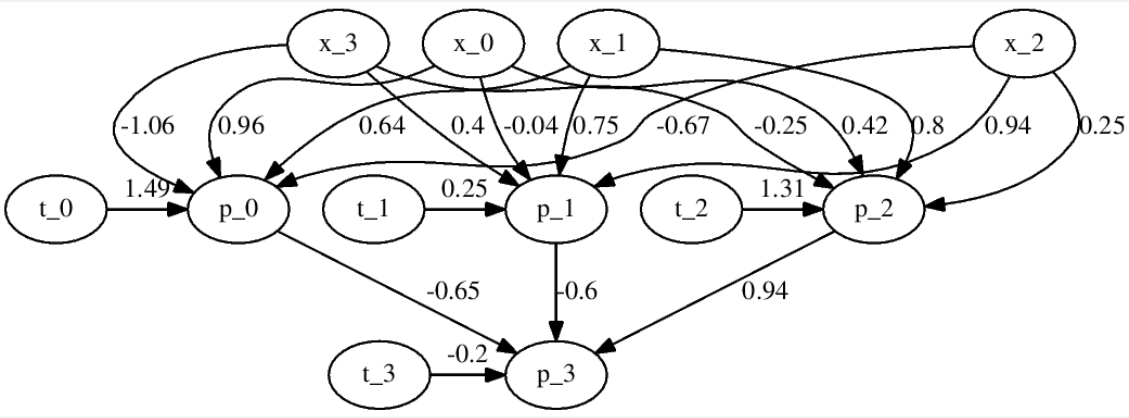


XOR

Visual Results of Neural Training



3 PALINDROME



4 PALINDROME

Grapheme To Phoneme using NN

- We encoded the graphemes and phonemes using bit sequences by giving each grapheme and a phoneme a bit representation as given in the question
- For padding we used 0000...
- For testing we used this bit sequence

Training Adjustment

- As the neural network is large we cannot expect it to train over the entire data till convergence(which is not even known if it will happen),
- We iterate over a large cycles to let it train the data. The user can specify this using a command line argument

Neural Network Performance

The neural network performance is mixed* **(we'll analyse this later in this presentation and see why this answer isn't the best measure what our system could achieve)**

- Measures to look at:
- Word Length
- Test Cases
- Training Duration
- Character Accuracy
- Bit Accuracy

THE RESULTS

	Test Case 1	Test Case 2	Test Case 3	Test Case 4	Tes
Test Case	Word Length 4 Test cases 10	Word Length 4 Test cases 10	Word Length 6 Test cases 100	Word Length 8 Test cases 1000	Word Length 10 Test cases 10000
Training Duration	10000 iterations **1m 45.074s**	100000 iterations **9m40.104s**	20000 iterations **~10 hours**	20000 iterations **5 days 22hours!!!!!** *(Yes we actually ran that on a server for 6 days just to see what happens)*	1000 iterations **115m7.984s**
Accuracy	70.0%	82.5%	32.66%	54.72%	23.9986%
Word Efficiency	97.1428%	98.5714%	83.11%	89.6719%	73.463%

Performance Issues and Improvements

The performance on smaller data being so high indicate that the mechanism has the potential of performing well,

However but as the amount of data becomes bigger, the amount of time to train(to the same level) increases exponentially.

Hence the increased requirement in time but reduced accuracy

How we can improve the accuracy

1) Increase the time to train (exponential)
2) Use more sparse numbering of the input grapheme bits
3) Dont use padding and work with only a standard size

Thank You