Texas Tech University

ECE 5342 - Deep Learning, Dr. Sunanda Mitra and Harshit Parmar

Alexandre S. Silva, Rishi Das, Jud Wizee

1. The Perceptron Algorithm

PERCEPTRON ALGORITHM (200 max. No of Iterations)						
	AND[0. 0. 0. 1.]	OR [0. 1. 1. 1.]	NAND [1. 1. 1. 0.]	NOR [1. 0. 0. 0.]	XOR [1. 1. 0. 0.]	
LEARNING RATE	Convergence Iterations	Convergence Iterations	Convergence Iterations	Convergence Iterations	Convergence Iterations	Average Iterations per Gates
1.00E-04	6	4	6	4	0	4
1.00E-03	6	4	6	4	0	4
1.00E-02	6	4	6	4	0	4
1.00E-01	6	4	4	4	0	3.6
1.00E+00	6	4	6	4	0	4
Average Iterations per Learning Rate	6	4	5.6	4	0	3.92

From the table above, it is obvious that the XOR gate do not converge. Therefore, XOR gate cannot be implemented using this approach.

Reason:

The output of a perceptron is a linear combination of its inputs. Hence, it can classify input-output space only if you can fit a line between the two or more classes. However, this is not true with the XOR problem, which is non-linearly separable.

	X	Y	Class
Desired i/o pair 1	0	0	0
Desired i/o pair 2	0	1	1
Desired i/o pair 3	1	0	1
Desired i/o pair 4	1	1	0

Form the equations:

$$
\begin{aligned}
0 \times w_1 + 0 \times w_2 + w_0 \leq 0 &\iff w_0 \leq 0, \\
0 \times w_1 + 1 \times w_2 + w_0 > 0 &\iff w_0 > -w_2, \\
1 \times w_1 + 0 \times w_2 + w_0 > 0 &\iff w_0 > -w_1, \\
1 \times w_1 + 1 \times w_2 + w_0 \leq 0 &\iff w_0 \leq -w_1 - w_2.
\end{aligned}
$$

If Wo is greater than –W2 AND –W1 then the fourth equation is not realizable since it is self-contradictory. Therefore, XOR cannot be predicted using the linear learner perceptron.

2.a. 2-bit logic gates.

5 different neural networks (NN) were trained to predict the logic gate AND, OR, NAND, NOR, XOR. By using backpropagation and gradient descent to minimize cost of a mean square loss function and to update the NN's weights, the NN.

The results can be seen below:

Neural network (10000 max. No of Iterations)						
	AND[0. 0. 0. 1.]	OR [0. 1. 1. 1.]	NAND [1. 1. 1. 0.]	NOR [1. 0. 0. 0.]	XOR [1. 1. 0. 0.]	
Hidden layer and nodes	Convergence Iterations	Convergence Iterations	Convergence Iterations	Convergence Iterations	Convergence Iterations	Average Iterations per Gates
2 and 4	28	87	42	595	192	188.8
4 nodes	15	18	14	19	No convergence	16.5
2,3	35	36	41	46	90	49.6
8 nodes	12	12	12	13	No convergence	12.25
1,3	137	171	69	90	452	100
Av. Ite per number of nodes	45.4	64.8	35.6	152.6	105	73.43

The neural network was tested with different number of hidden layers and different number of units within each layer. The results were validated by matching prediction and target, with prediction being rounded for the matching process. Printing the cost function during iterations, along with a measure of accuracy, was also used to verify that the NN was converging. With one layer of nodes, it was not possible to converge to a correct XOR prediction. However, with the addition of another hidden layer, within a few iterations the NN converges to the correct prediction.

2.a. BONUS CREDIT

The previous NN required a modification to predict this function. Instead of outputting at its last layer the two probabilities of 0 or 1 – which requires normalizing the sigmoid output to sum up to one – both of this NN's outputs can have 0's or 1's. The network details can be seen on the next page:

NN ==========

no. inputs (layer 1): 2

layer 2: 32 units

layer 3: 64 units

layer 4: 32 units

output layer (5): 2

learnable weights: 4224

max epochs: 10,000,000

learning rate(rho): 0.3

And the convergence data:

converged: True

no. of iterations to converge: 17

max iterations set: 10,000,000

learning rate: 0.3

matches with target: True

prediction:

[[0. 0.]

[1. 0.]

[1. 0.]

[0. 1.]]

2. b and c. Classification of Iris dataset

We were required to train a model to classify Fisher's iris dataset, where the input data were 4 measurement and the output were classification of 3 categories of flowers. We used the same cost function and we defined our initial weight biases with number of features, hidden units and number of outputs. We used the same forward and back propagation functions (which were written for arbitrary numbers of layers/units) to train and predict functions. Selecting random train samples with parameters, the algorithm is trained based on a slice of data for train and the

remaining data were used for testing purposes. The iris flower classifications were setosa, versicolor, and virginica. Finally, we trained the NN and fed train_package variable to a prediction function. A learning rate of 0.1 was used for the purpose with a maximum 5000 epochs.

In our observations, one additional small layer has not required more iterations to train than with one layer.

2b data

train/test samples: 50/100
* NN **********
  no. inputs (layer 1): 4
  layer 2: 2 units
  output layer (3): 3
  learnable weights: 14
  max epochs: 5,000
  learning rate(rho): 0.1
  Start/final Cost: 6.059387/0.723772
  Train start/final accuracy(50 train samples): 2.00/96.00
  Test accuracy(100 test samples): 85.00%
  Time Taken: 5.620442867279053 seconds

train/test samples: 75/75
* NN **********
  no. inputs (layer 1): 4
  layer 2: 2 units
  output layer (3): 3
  learnable weights: 14
  max epochs: 5,000
  learning rate(rho): 0.1
  Start/final Cost: 9.394372/1.823107
  Train start/final accuracy(75 train samples): 1.33/74.67
  Test accuracy(75 test samples): 57.33%
  Time Taken: 5.262930870056152 seconds

train/test samples: 100/50
* NN **********
  no. inputs (layer 1): 4
  layer 2: 2 units
  output layer (3): 3
  learnable weights: 14
  max epochs: 5,000
  learning rate(rho): 0.1
  Start/final Cost: 12.217134/0.923285
  Train start/final accuracy(100 train samples): 0.00/96.00
  Test accuracy(50 test samples): 98.00%

Time Taken: 5.195034503936768 seconds

train/test samples: 125/25
* NN **********
no. inputs (layer 1): 4
layer 2: 2 units
output layer (3): 3
learnable weights: 14
max epochs: 5,000
learning rate(rho): 0.1
Start/final Cost: 15.304454/0.921326
Train start/final accuracy(125 train samples): 1.60/96.80
Test accuracy(25 test samples): 84.00%
Time Taken: 5.28982400894165 seconds

2c data

train/test samples: 50/100
* NN **********
no. inputs (layer 1): 4
layer 2: 2 units
layer 3: 4 units
output layer (4): 3
learnable weights: 28
max epochs: 5,000
learning rate(rho): 0.1
Start/final Cost: 5.789398/0.008458
Train start/final accuracy(50 train samples): 32.00/100.00
Test accuracy(100 test samples): 86.00%
Time Taken: 5.363662958145142 seconds

train/test samples: 75/75
* NN **********
no. inputs (layer 1): 4
layer 2: 2 units
layer 3: 4 units
output layer (4): 3
learnable weights: 28
max epochs: 5,000
learning rate(rho): 0.1
Start/final Cost: 8.685021/0.322521
Train start/final accuracy(75 train samples): 33.33/98.67
Test accuracy(75 test samples): 94.67%
Time Taken: 5.396573543548584 seconds

train/test samples: 100/50
* NN **********
no. inputs (layer 1): 4

layer 2: 2 units
layer 3: 4 units
output layer (4): 3
learnable weights: 28
max epochs: 5,000
learning rate(rho): 0.1
Start/final Cost: 11.665744/0.308544
Train start/final accuracy(100 train samples): 32.00/99.00
Test accuracy(50 test samples): 94.00%
Time Taken: 5.481348752975464 seconds

train/test samples: 125/25
* NN **********
no. inputs (layer 1): 4
layer 2: 2 units
layer 3: 4 units
output layer (4): 3
learnable weights: 28
max epochs: 5,000
learning rate(rho): 0.1
Start/final Cost: 14.502462/0.348447
Train start/final accuracy(125 train samples): 33.60/99.20
Test accuracy(25 test samples): 84.00%
Time Taken: 5.712730646133423 seconds