DL Lab 1: Back-Propagation

Wilbert
wilbert.phen@gmail.com
NCTU CS Department

October 11, 2019

1 Introduction

Backpropagation algorithms are a family of methods that are usually used to efficiently train neural networks following a gradient-based optimization algorithm that exploits the chain rule. The main feature of backpropaggation is its iterative, recursive and efficient method for calculation the weights updates to improve the network until it is able to perform the task for which is is being trained. It is closely related to the Gauss-Newton algorithm.

Backpropagation requires the derivatives of activation functions to be known at network design time. Automatic differentiation is a technique that can automatically and analytically provide the derivatives to the training algorithm. In the context of learning, backpropagation is commonly used by the gradient descent optimization algorithm to adjust the weight of neurons by calculating the gradient of the loss function; backpropagation computes the gradient(s), whereas (stochastic) gradient descent uses the gradients for training the model (via optimization).

2 Experiment Setups

2.1 Sigmoid Functions

A sigmoid function is a mathematical function having a characteristic "S"-shaped curve or sigmoid curve. A standard choice for a sigmoid function is the logistic function shown in the first figure and defined by the formula.

$$S(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{e^x + 1} \tag{1}$$

2.2 Neural Network

Artificial neural networks (ANN) or connectionist systems are computing systems that are inspired by, but not identical to, biological neural networks that constitute animal brains. Such systems "learn" to perform tasks by considering examples, generally without being programmed with task-specific rules. For example, in image recognition, they might learn to identify images that contain cats by analyzing example images that have been manually labeled as "cat" or "no cat" and using the results to identify cats in other images. They do this without any prior knowledge of cats, for example, that they have fur, tails, whiskers and cat-like faces. Instead, they automatically generate identifying characteristics from the examples that they process.

An ANN is based on a collection of connected units or nodes called artificial neurons, which loosely model the neurons in a biological brain. Each connection, like the synapses in a biological brain, can transmit a signal to other neurons. An artificial neuron that receives a signal then processes it and can signal neurons connected to it.

In ANN implementations, the "signal" at a connection is a real number, and the output of each neuron is computed by some non-linear function of the sum of its inputs. The connections are called edges.

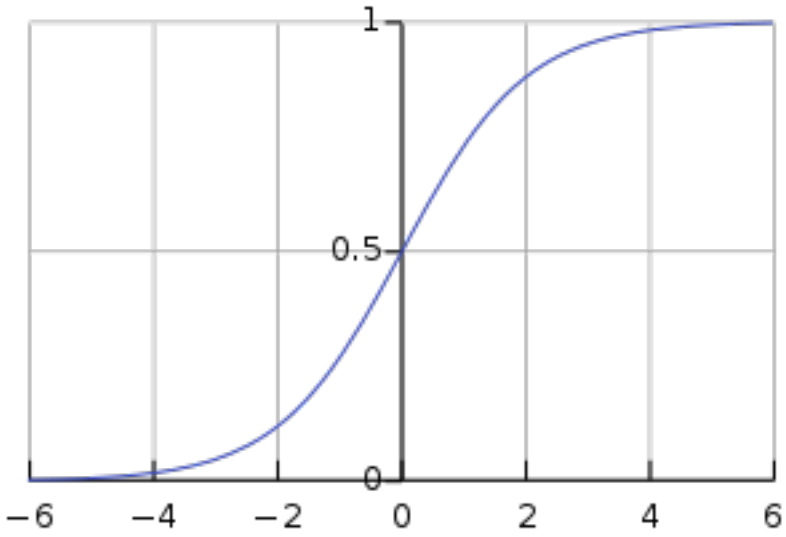

Figure 1: Sigmoid Function Graph

Neurons and edges typically have a weight that adjusts as learning proceeds. The weight increases or decreases the strength of the signal at a connection. Neurons may have a threshold such that a signal is sent only if the aggregate signal crosses that threshold. Typically, neurons are aggregated into layers. Different layers may perform different transformations on their inputs. Signals travel from the first layer (the input layer), to the last layer (the output layer), possibly after traversing the layers multiple times.

The original goal of the ANN approach was to solve problems in the same way that a human brain would. However, over time, attention moved to performing specific tasks, leading to deviations from biology. ANNs have been used on a variety of tasks, including computer vision, speech recognition, machine translation, social network filtering, playing board and video games, medical diagnosis and even in activities that have traditionally been considered as reserved to humans, like painting.

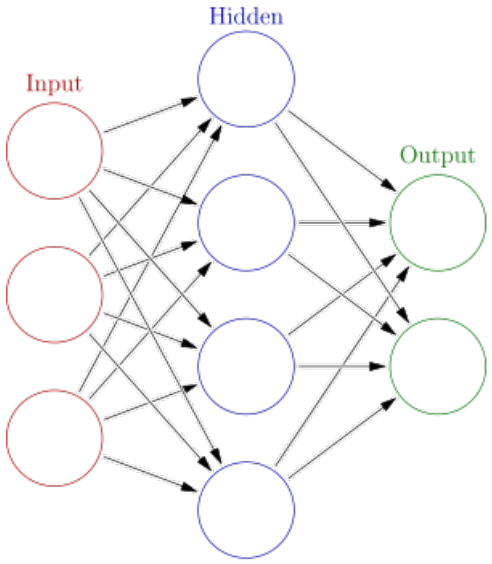

Figure 2: An artificial neural network is an interconnected group of nodes

2.3 Backpropagation

On a single layered network, derivation of single layered network is much easier. The gradient descent method involves calculating the derivative of the loss function with respect to the weights of the network. This is normally done using backpropagation. Assuming one output neuron, the squared error function is

$$E = L(t, y)$$

where
E is the loss for the output y and the target value t,
t is the target output for a training sample, and
y is the actual output of the output neuron.

Finding the derivative of the error Calculating the partial derivative of the error with respect to a weight w_{ij} is done using the chain rule twice:

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial o_j}\frac{\partial o_j}{\partial w_{ij}} = \frac{\partial E}{\partial o_j}\frac{\partial o_j}{\partial net_j}\frac{\partial net_j}{\partial w_{ij}}$$

3 Results of Testing

3.1 Graph of the result

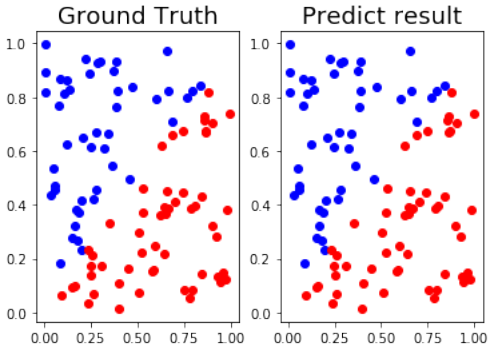

Figure 3: Result for Linear Model

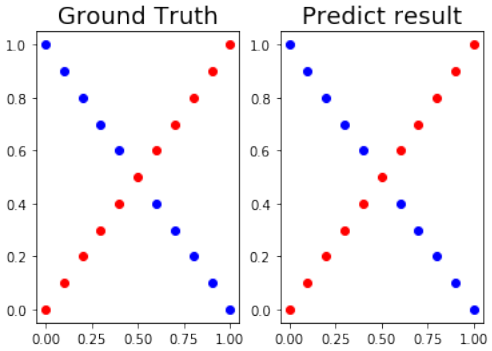

Figure 4: Result for XOR Model

3.2 Loss Results

```
-----------------------------------------------------
Linear Model has started
-----------------------------------------------------
Epoch 5000 loss: 0.0037818146047146777
```

```
Epoch 10000 loss: 0.001558447987060981
Epoch 15000 loss: 0.0008758128700111306
Epoch 20000 loss: 0.0005693068981069266
Epoch 25000 loss: 0.00040380531765245786
Epoch 30000 loss: 0.0003038856192793466
Epoch 35000 loss: 0.00023872732158024425
Epoch 40000 loss: 0.0001937349194023958
Epoch 45000 loss: 0.00016125831376040758
Epoch 50000 loss: 0.0001369715175833124
Epoch 55000 loss: 0.00011827763750285421
Epoch 60000 loss: 0.00010354018244280873
Linear Model Performance is satisfactory
```

Above is the result for Linear Model

```
------------------------------------------------------
XOR Model has started
------------------------------------------------------
Epoch 5000 loss: 0.0005635906918288244
Epoch 10000 loss: 0.00017885907522759828
Epoch 15000 loss: 0.00010068474071034245
XOR Model Performance is satisfactory
```

Above is the result for XOR Model

3.3 Prediction Results

```
linear test loss :  9.998650667526666e-05
linear test accuracy : 100.00%

 linear test result :
 [[1.59874280e-05]
 [9.99997616e-01]
 [4.31640750e-06]
 [4.97630860e-04]
 [1.16766819e-05]
 [3.95854445e-06]
 [3.11504112e-06]
 [2.59215700e-06]
 [2.49154996e-06]
 [9.99999314e-01]
 [9.92519378e-01]
 [2.76201833e-06]
 [9.99999382e-01]
 [2.60861225e-06]
 [2.47244166e-06]
 [4.58996364e-06]
 [9.99999383e-01]
 [2.39847915e-06]
 [2.45193153e-06]
 [9.99999362e-01]
 [2.68239981e-06]
 [2.75800076e-06]
 [9.99999381e-01]
 [3.20796695e-06]
 [9.99998963e-01]
 [9.99999407e-01]
 [2.59725283e-06]
 [2.83704416e-06]
 [9.99999393e-01]
```

[9.99998984e-01]
[2.90209478e-05]
[9.99998747e-01]
[9.99999351e-01]
[3.27658508e-06]
[9.43864606e-01]
[9.59051055e-01]
[3.17704325e-06]
[2.45104506e-06]
[2.44802058e-06]
[2.03350185e-03]
[1.02962447e-04]
[9.99999369e-01]
[2.60275046e-06]
[9.99999382e-01]
[4.86426911e-06]
[9.99999380e-01]
[9.99998757e-01]
[9.99999403e-01]
[9.99999323e-01]
[2.43478012e-05]
[2.61118200e-06]
[9.99982282e-01]
[4.73947261e-05]
[1.12771283e-05]
[9.99999401e-01]
[2.40226697e-06]
[9.99999404e-01]
[9.99999391e-01]
[3.08721776e-06]
[5.92323717e-03]
[9.99999045e-01]
[3.73498323e-06]
[9.99999161e-01]
[2.47351296e-06]
[9.99999171e-01]
[9.99999392e-01]
[3.09767450e-06]
[9.99999408e-01]
[9.99999402e-01]
[9.99999394e-01]
[9.99999396e-01]
[2.75798603e-06]
[9.99999391e-01]
[8.58729747e-06]
[9.99999407e-01]
[7.11901675e-02]
[9.99999016e-01]
[3.24751246e-06]
[9.99999382e-01]
[2.79208045e-06]
[2.46433336e-06]
[9.99278124e-01]
[9.99988681e-01]
[2.39029525e-06]
[2.87425828e-06]
[2.85284439e-06]
[9.99999115e-01]

```
 [2.61240139e-06]
 [2.40866782e-06]
 [2.78702970e-06]
 [9.99999362e-01]
 [9.99999340e-01]
 [2.66812841e-06]
 [9.99999400e-01]
 [9.99999372e-01]
 [9.97471713e-01]
 [2.80299725e-06]
 [2.60840882e-06]
 [9.99494195e-01]
 [2.79360151e-06]]

XOR test loss :  9.994957400957609e-05
XOR test accuracy : 100.00%

 XOR test result :
 [[0.00544366]
 [0.99993656]
 [0.00720054]
 [0.99993344]
 [0.00931375]
 [0.9999239 ]
 [0.01150479]
 [0.99985182]
 [0.01319998]
 [0.97690564]
 [0.01369826]
 [0.01265334]
 [0.97746585]
 [0.01047016]
 [0.99966354]
 [0.00804613]
 [0.99976592]
 [0.0060936 ]
 [0.99976423]
 [0.0048382 ]
 [0.99974897]]
```