Assignment: Machine Learning with a Neural Network

October 20, 2020

1 Introduction

As a C programmer (and an engineer), you will frequently be asked to use your specialist skills to contribute functional code as a smaller part of a larger project. In this assignment, you will write code to implement components and functions of a larger C program that implements a simple, artificial neural network (ANN), a kind of machine learning system.

Note that, you do not need any in-depth knowledge of ANNs or any other machine learning technique to complete this assignment.

2 Background

Artificial neural networks are kind of model often used in artificial intelligence for allowing machines to *learn how to solve problems*. They work by taking data about a problem, and training until they can solve it accurately. Originally, they were developed to imitate how the human brain solves problems, but since then ANNs of many shapes and sizes have been developed, including very large ones (so-called *deep learning* models).

A conceptual diagram of the ANN you will be working with is shown in Fig. 1.

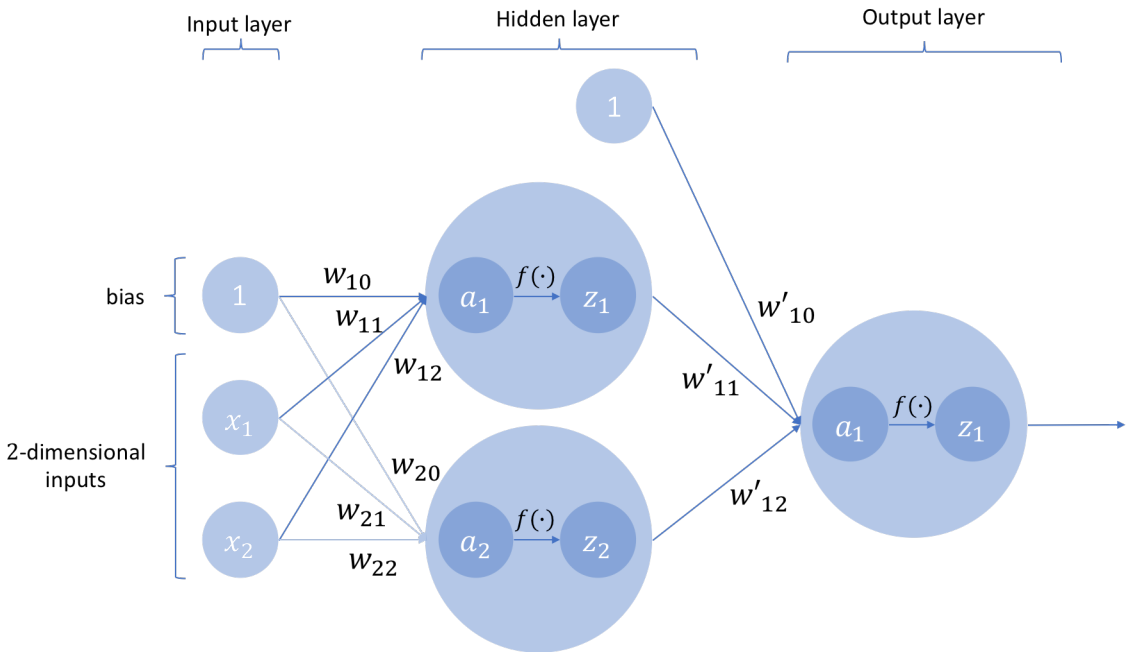

Figure 1: Single hidden layer neural network.

The ANN works by propagating signals from left to right. Each big circle represents a "neuron" and takes a set of numbers as inputs (marked by the arrows going into the neurons in the diagram), performs some

simple mathematical operations on them, and produces an output (marked by the arrows going out of the neurons). This output may be fed as the input to another neuron, or it may form the output of the whole network.

The network in Fig. 1 is designed to learn to solve a famous machine learning problem called the *exclusive or* ("XOR") *problem*. The aim is to get the network to learn how an XOR logic gate works. To do this, the network takes two inputs and produces one output. The inputs can take the values one or zero (*i.e., true* or *false*), and the network should output one (*i.e., true*) if either of the two inputs are true, and zero (*i.e., false*) if (i) both of the inputs are false or (ii) both of the inputs are true. The truth table for all possible inputs are shown in Table 1.

Input 1	Input 2	Output
0	0	0
0	1	1
1	0	1
1	1	0

Table 1: Truth table of XOR logical operator.

The neural network program should learn to predict the correct output, given each of these possible inputs.

3 Preparation

To complete the assignment, you will need the following files:

- `in.csv`
- `out.csv`
- `ann.c`
- `libann.h`
- `libann.c`
- `k0000000.c`

Download these from KEATS and save them to your computer.

Use the following commands to compile the program:

```
$ gcc libann.c k0000000.c ann.c -o ann
$ ./ann
```

You should find that it compiles without errors.[1]

If you have completed this correctly, you should see the following output in your terminal:

```
Pat     Input1  Input2  Target1 Output1
0       0       0       0       0.000000
1       0       0       0       0.000000
2       0       0       0       0.000000
3       0       0       0       0.000000
```

Important In the following, you will be asked to implement functions in `k0000000.c` to create a working ANN. When completing the tasks, be sure to *only implement code at the locations indicated in the tasks*. Do *not* change the number or type of parameters used in the functions nor the type of their return values.

[1] Your compiler may issue warnings about unused variables. These can be safely ignored.

Do *not* modify any file, other than `k0000000.c`. You are encouraged to comment your code to aid understanding for the markers.

Complete the following exercises.

4 Tasks

Complete the following tasks.

1. The first task is to write a function to read in the data required for training the network, or report an error if the data cannot be found.

   To complete this part of the assignment, find and modify the following lines in the template file:

   ```
   /* --------------- Begin Answer to Task 1 Here ------------ */
   return 0;
   /* --------------- End Answer to Task 1 Here -------------- */
   ```

   Implement code that reads the training data from the files `in.csv` and `out.csv` and stores them in the arrays `trIn` and `trOut`.

   The first index of the array `trIn` should correspond to the index of the data point (*i.e.*, row of `XorIn.csv`) and the second should correspond to the index of the input (*i.e.*, *Input 1* or *Input 2*).

   The function should not take any inputs. It should return *one* and exit immediately if any errors are encountered when reading the files, or *zero* otherwise.

   [5 marks]

2. The next task is to implement the basic building blocks of the neural network: the neurons.

   To complete this part of the assignment, find and modify the following lines in the template file:

   ```
   /* --------------- Begin Answer to Task 2 Here ------------ */
   return 0;
   /* --------------- End Answer to Task 2 Here -------------- */
   ```

   Implement the function `neuron()` to compute the output of a neuron z_j from its inputs x_i, using the following steps.

   Step 1. Each of the inputs x_i should be multiplied by a weighting factor w_{ji} and added together according to the equation

   $$a_j = \sum_{i=1}^{d} x_i w_{ji} + w_{j0} \tag{1}$$

   where w_{j0} is a constant (called the *bias* term) and d is the number of inputs.

   Step 2. The resultant value a_j should be passed through another function (called the *activation function*) to compute the output

   $$z_j = f(a_j). \tag{2}$$

   As the activation function, use the equation

   $$f(a) = \frac{1}{(1 + e^{-a})}. \tag{3}$$

   This is called the *sigmoid function*.

The template code `template.c` contains an empty definition of a function `neuron()` to compute the output of a neuron from its inputs. It takes the following parameters: (i) the number d of inputs `num_in`, (ii) an array `input[num_in]` that contains the actual input values x_i, (iii) an array `weight[num_in]` that contains the current weights for the neuron w_{ij}, and (iv) `bias` that represents the bias term w_{j0}. The return value should be the output of the neuron. Complete the implementation of the function so that it computes and returns the output of a neuron from these input parameters, following *Step 1.* and *Step 2.*

[5 marks]

3. The final task is to write code to train the neural network on the data.

   To complete this part of the assignment, find and modify the following lines in the template file:

```
/* ---------------- Begin Answer to Task 3 Here ------------ */

error = update_network(DeltaWeightIH, DeltaWeightHO,
            DeltaBiasIH, DeltaBiasHO,
          WeightIH, WeightHO, biasIH, biasHO,
          DeltaO, DeltaH, SumDOW, eta, alpha, index[0]);

/* ---------------- End Answer to Task 3 Here -------------- */
```

   Neural networks of the kind implemented here learn by iteration. At each step, the training data is fed to the network, and the network updates itself by (i) trying to predict the output data from the input data, and (ii) correcting the errors it makes. It can take many iterations before a good prediction is found. In the template code, training is performed iteratively in the function `learn()`. This function has been partially implemented for you.

   In the template code, the function `update_network()` is used to update the network. This function takes as input (i) the current network state as parameters (*i.e.,* weights, bias terms, *etc*) and learning parameters (*i.e.,* `eta` and `alpha`). Its last argument `p` is the index of the training pattern (*i.e.,* row of `trIn`) that the network should use for testing its predictions. Learning usually works best if this index is randomly shuffled after each training step (*i.e.,* the network is given the data in random order). The template code provides the function `shuffle_index()` to compute a set of indices that can be used to access the training data array in random order.

   Complete the implementation of the `learn()`, so that it (i) initialises `error` to zero, (ii) calls `update_network()` for each row of `trIn` in random order, and (iii) repeats this process for a thousand iterations.

   [5 marks]

If you have implemented all of these tasks correctly, you should find that the neural network learns to (approximately) predict the correct output when given the inputs listed in Table 1.

5 Submission

To submit your assignment, you need to upload your code to KEATS. To do this, follow the below instructions.

First, change the file name of `k0000000.c` to your k-number. For example, if your k-number is k1234567, the file name of your assignment should be `k1234567.c`. Next, submit this file to KEATS by following the *Assignment 1* link.

Important The code you submit will be assessed based on its compatibility with the specifications, including whether it runs with the template code provided. *If it does not run, it may be awarded zero marks.*

Completed assignments should be submitted to KEATS by 5pm (UK time) on November 5, 2020.

This assignment is worth 15% of the module mark.