Computational Physics III
Machine Learning
WS-18/19

Distribution: 22/11/18, Due: 28/11/18

Overview

The aim of this tutorial is to perform a step-by-step computation to understand the concept of *Backpropagation*. For setting up a simple problem we will use a neural network with two inputs (i_1, i_2), two hidden neurons (h_1, h_2) and two output neurons (o_1, o_2). Additionally we assume that the hidden and output neurons will include a bias, b, with bias weights b_1 and b_2 respectively.

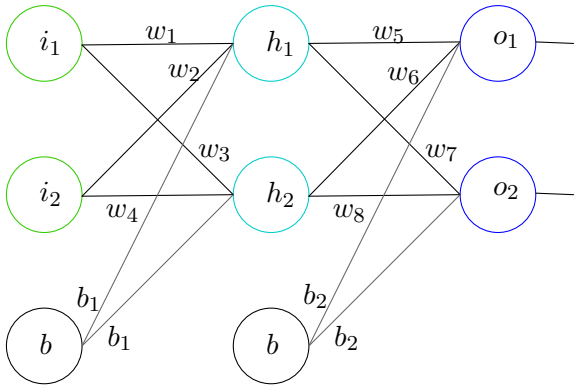

Figure 1: Schematic of the Neural Network

The initial weights, the biases, and training inputs/outputs are listed in the table below:

w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	b	b_1	b_2	i_1	i_2	o_1	o_2
0.15	0.20	0.25	0.30	0.40	0.45	0.50	0.55	1.00	0.35	0.60	0.05	0.10	0.01	0.99

Backpropagation is used to optimize the weights (ignore optimizing the bias weights for this exercise and keep them as they are given in the table) so that the neural network can learn how to correctly map arbitrary inputs to outputs. In order to grasp the concept we will work with a single training set: given inputs 0.05 and 0.10, we want the neural network to output 0.01 and 0.99.

In the following we deconstruct one iteration in the optimization process as the Forward Feed/Pass and the Backward Feed/Pass.

The Forward Feed/Pass

For the forward feed (i.e. going in the direction of input $\rightarrow$ output), compute using the given weights, biases and the training inputs the prediction of the given (cf. Fig. 1) neural network. To go about this:

1. Compute the total net input to each hidden layer neuron (i.e. net_{h1} and net_{h2}). Then, evaluate for each hidden neuron the activation function value (i.e. out_{h1} and out_{h2}) using the obtained net inputs. For this exercise we use the logistic function as the activation function,

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

2. Repeat the process with the output layer neurons using the output (out_{h1} and out_{h2}) from the hidden layer neurons as inputs. For uniformity, call net input for the output neurons as net_{o1} and net_{o2}, and the activation function output for these neurons as out_{o1} and out_{o2}.

3. Calculate the error for each output neuron (i.e. E_{o1} and E_{o2}) using the squared error function and sum them to get the total error:

$$E_{total} = \Sigma \frac{1}{2} \left(target - output\right)^2 = E_{o1} + E_{o2} \tag{2}$$

   Note: The 'target' output is the output we want the neural network to output eventually, i.e. 0.01 for o_1 and 0.99 for o_2.

The Backward Feed/Pass

The aim of backpropagation algorithm is to update (optimize) each of the weights in the network so that they bring the actual output closer to the target output, thereby minimizing the error for each output neuron and the network as a whole. To initiate this:

1. (Say) we start by finding out how much a change in w_5 affects the total error, i.e. find out $\frac{\partial E_{total}}{\partial w_5}$ (aka the gradient with respect to w_5). In order to compute this gradient apply the chain rule as:

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} \frac{\partial out_{o1}}{\partial net_{o1}} \frac{\partial net_{o1}}{\partial w_5} \tag{3}$$

   where $\frac{\partial E_{total}}{\partial out_{o1}}$: How much does the total error change with respect to the output? $\frac{\partial out_{o1}}{\partial net_{o1}}$: How much does the output of o_1 change with respect to its total net input? and $\frac{\partial net_{o1}}{\partial w_5}$: How much does the total net input of o_1 change with respect to w_5?

2. To decrease the error, we subtract the gradient value from the current weight (multiplied by some learning rate, η, which we here set to 0.5):

$$w_5^+ = w_5 - \eta \times \frac{\partial E_{total}}{\partial w_5} \tag{4}$$

3. Repeat the above process to obtain the new weights w_6, w_7, and w_8.

   Note 1: One needs to use the relevant gradient (cf. Eq. (3)) in computing the new weights.
   Note 2: Use the original weights and not the updated weights while continuing the backpropagation algorithm until the next iteration of forward feed phase.

4. Continue the backward feed/pass by calculating the new weights for w_1, w_2, w_3, and w_4. Similar to step 1, one needs to compute the relevant gradients for the input-hidden layer weights, for example,

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} \frac{\partial out_{h1}}{\partial net_{h1}} \frac{\partial net_{h1}}{\partial w_1} \tag{5}$$

This computation is performed slightly different to account for the fact that the output of each hidden layer neuron contributes to the output (and therefore the error) of multiple output neurons, i.e.,

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}}. \tag{6}$$

Where one can use $\frac{\partial E_{o1}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial net_{o1}} \times \frac{\partial net_{o1}}{\partial out_{h1}}$ and $\frac{\partial E_{o1}}{\partial net_{o1}} = \frac{\partial E_{o1}}{\partial out_{o1}} \times \frac{\partial out_{o1}}{\partial net_{o1}}$. Similar expressions are used in computing $\frac{\partial E_{o2}}{\partial out_{h1}}$.

5. Once the gradient (cf. Eq. (5)) is obtained for weight w_1, use equation similar to Eq. (4) to obtain w_1^+ and repeat the process to obtain new weights for w_2, w_3, and w_4.

The above steps constitute one iteration of optimization (Forward Feed and Backpropagation). Check that, indeed, the optimization helped in decreasing the error by doing a Forward Feed with new weights. Though, the decrease might not seem much but repeating the iteration some thousand times, for example, the errors would go down significantly. After this, when we forward feed the initial inputs, the two output neurons should generate the targeted outputs.

Points: Computing new weights correctly carries 3-points each and showing that the newly obtained weights indeed brings down the error after one iteration carries 6-points.