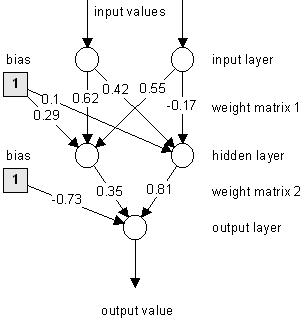
**Neural Network take-home assignment 1:**

**Prepared by Haihan**

Your goal is to implement a neural network and the stochastic gradient descent algorithm from scratch in Python. Your neural network will resemble the following:



It will have two inputs, the activation function will be **tanh(x)** for all neurons, and it will have one output. Note we have regular weights and *bias* weights as well (there was some confusion in class regarding the bias and bias weights, but you can see how the constant bias value of 1 is absorbed into the bias weights). Initially the weights and bias weights are uniformly randomly initialized on the interval [-0.5,0.5].

Note the ‘input’ nodes are passthrough and do not apply any activation functions to the numeric inputs. The number of neurons in the hidden layer *will be variable i.e. you can specify any number of hidden layer neurons*. Your neural network will be trained on the following dataset:

|  |  |  |
| --- | --- | --- |
| **X1** | **X2** | **Out** |
| -0.5 | -0.5 | -0.5 |
| 0.5 | -0.5 | 0.5 |
| -0.5 | 0.5 | 0.5 |
| 0.5 | 0.5 | -0.5 |

Your loss function will be the Mean Square Error (MSE) loss as described in lecture, and the optimizer will be regular Stochastic Gradient Descent (i.e. batch size = 1).

**Task 1: Implement the neural network above with 2,4 and 6 hidden layer neurons. Set the learning rate α to be 0.1. Convergence is reached when your average MSE for the whole dataset is less than 0.05. Plot and compare your results.**

**Task 2: Implement the neural network above with 2,4 and 6 hidden layer neurons now with momentum (1-st order lowpass). Set the learning rate α to be 0.1 and momentum parameter γ to be 0.9. Convergence is reached when your average MSE for the whole dataset is less than 0.05. Plot and compare your results.**

You are free to push the code and results to your public git repo.