CS457 - Computational Intelligence

**Project 2**

Investigate the use of the Multi-Layer Perceptron (MLP, Sections 4.2.1 and 4.4.1 in textbook) to approximate (a regression problem) the following function:

, are real numbers

You may use any implementation of the MLP.

**What to do:**

1. Set up two sets of data, one for training and one for testing.
2. Use the training set to compute the weights of the network.
3. Evaluate the approximation accuracy (the root-mean squared error) of the network on the test data.
4. Use a variable number of hidden layers (1-2), each with a variable number of neurons (0-19).
5. Investigate how the network’s performance is affected by varying the number of hidden layers and the number of hidden neurons in each layer. It is not necessary to try all 20 x 20 combinations of neurons in the two hidden layers; try only some of them.

**Hint**: If you use your own implementation:

* Use two input neurons: one for *x* and one for *y*.
* For the hidden neurons, use bipolar or unipolar sigmoid activation functions.
* Use a single output neuron, with the identity (linear) activation function (read Sections 3.2.3 and 4.4.1 from textbook).
* Do not forget to add a fixed input (-1) to each layer

If you choose Weka, have a look at:

<https://www.youtube.com/watch?v=Gw5s3yYVQAE>

<https://stackoverflow.com/questions/31604266/weka-multi-perceptron-with-multiple-hidden-layers>

<https://machinelearningmastery.com/use-regression-machine-learning-algorithms-weka/> (the MLP part)

**What to upload in a compressed folder (Canvas):**

* Describe which implementation you used and, if it is your code, attach it
* The approximation accuracy for the different MLP architectures used
* Your conclusions about how the number of hidden layers and neurons influences the approximation error

**Hint:** You can use WolframAlpha Computational Intelligence tool (<https://www.wolframalpha.com/>) to plot the function.