Neural networks homework assignment with hands-on activities

Saaketh Desai and Alejandro Strachan
Purdue University

The following problems will help you understand methods to pre-process data before training a neural network on the data, understand the architecture of a neural network, as well as quantify common pitfalls such as overfitting or underfitting your data. Before starting with the assignment make sure you go over the accompanying lecture and hands-on tutorial available at: https://nanohub.org/resources/34287.

For the assignments below, you will work with the *Neural Network Regression to predict material properties* notebook in the following tool: https://nanohub.org/tools/mseml

Problem 1. Before building and training a neural network we need data. Open the notebook "Querying databases, Organizing and Plotting Data" in the *mseml* tool. Study the notebook and execute the code cells (Shift+Enter). Modify the second code cell to obtain the Young's modulus of Al. Include a screen capture with the result.

Problem 2. Which atoms in the list have a Poisson ratio of 0.26?

Problem 3. From the plot of Young's modulus vs. melting temperature colored by crystal structure answer the following questions. What is the element with the highest melting temperature and what is its crystal structure? What element is the stiffest and what is its crystal structure.

Problem 4. Now open and carefully go over the notebook "Neural Network Regression to predict material properties". How many inputs do we pass to the neural network? What is the 7th input to the neural network and what is the minimum and maximum value of this input? What is the minimum and maximum value of this input after normalizing the data (Section 2 of the notebook)? Hint: the command `print(train_values[:,6])` will print the values of the 7th column. You can add the functions `min()` and `max()`.

Problem 5. The default network in the learning module is a two-layer network. Compute the number of parameters in this network by computing the total number of weights and biases in the network. Compare your result with the output of the model.summary() command.

Problem 6. What is the objective function used to optimize the neural network weights? Write down the equation of this objective function and compute its value for the training and testing sets. Also compute the Mean Squared Error (MSE) on the testing set.
Hint: Use the equation on slide 9 of the lecture with "test_predictions" as the array of predictions and "test_labels" as the array of ground truth data. To compute the MSE, you need to subtract the arrays and square each element in the array. We can do this in python simply using:
`(test_predictions - test_labels)**2`.
To sum all the elements in this array, we can use the `np.sum()` command, passing the array to the function. To divide the array 'A' by a constant 'c', we can just use `A/c`.

Problem 7. Fit a neural network with three layers such that the first hidden layer has 32 neurons, the second hidden layer has 64 neurons, and the third hidden layer has 128 neurons. Compute the mean absolute errors on the training and testing sets. How does it compare to the two-layer network? Include snapshots of the code cell with the network setup and plot of the training showing mean absolute error vs. epoch.