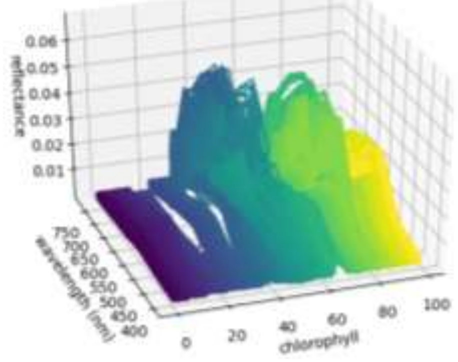
Zachary Burnett – 2020-08-16

Data and Objective

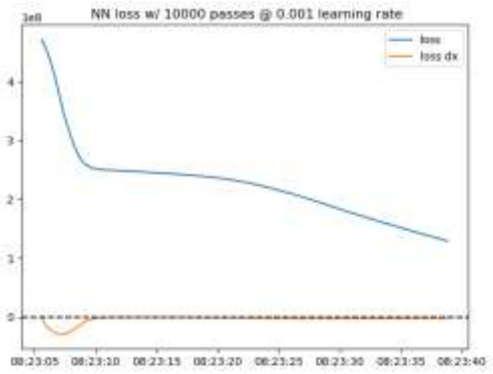
The objective of this task is to build a model to predict the chlorophyll index of several sites in `data/testing.csv`, given 16 reflectances per-site, as well as training data in `data/training.csv` and validation data in `data/validation.csv`. I then plotted a series of wavelength readings sampled from `data/training.csv` (figure left), sorted by chlorophyll value, showing the pattern to which the model will fit predicted chlorophyll. I figured that a good rough comparison metric would involve comparing the RMSE of the reference data provided in `data/validation.csv` (predicted vs in-situ measurements) with standard deviation of the measurements.



I chose to use the Python PyTorch library because, from what I have read, it provides a more forgiving, debuggable, and Pythonic interface.

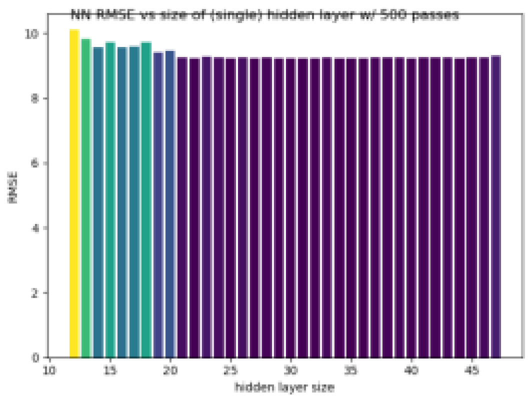
Model Fitting

Using `requires_grad=True`, I optimized weights with a learning rate of 1e-6, fitting them to the training data over 10000 iterations. I chose a hidden layer size of 2/3 of the input layer size plus the output layer size, following a rule-of-thumb that seemed to make sense. With 16 inputs, this determined a hidden layer with size of 12. The first iteration of this script is `models/1_autograd.py`. After optimizing weights over 10000 iterations on this run, the RMSE of predicted data vs reference data, around 8.5 over several runs, was within 20% of the standard deviation of in-situ measurements, 9.38. Results from this model are stored in the "Chl" column of `outputs/1_autograd/testing.csv`.



Then, using PyTorch's `torch.nn` module, I then replaced the manual gradient following with building a model with three explicit layers, the built-in Adam optimizer, and a much more liberal initial learning rate (`models/2_nn.py`). The plot (figure left) shows sums of mean squared error (from predicted to reference) over time, using the previously determined hidden layer size (12).

Optimizing Hidden Layer Size



Since this methodology returns a concrete metric (RMSE) that can be minimized, this means that the size of the hidden layer can also be optimized. In `models/3_nn_optimH.py`, I iterated over several different hidden layer sizes, with a reduced number of passes, and plotted their RMSE values (figure left). This plot shows an apparent decrease in RMSE as the hidden layer size increases past the input size (16). From this run, the hidden layer size with the lowest RMSE (9.23) is 44.

From these results, there does not appear to be any single hidden layer size that results in a minimum RMSE, so I will use the results from the initial neural network module run (above) to submit.

Results and Further Work

Along with this document, I submitted a filled `testing.csv` from the initial `torch.nn` run (with hidden layer size of 12). More work can be done to optimize multiple models per configuration, as the RMSE appears to vary over runs of the same values. This might make the data and make it easier to find a true critical point between the RMSE and the hidden layer size. Additionally, several other factors can be optimized programmatically, including the number of iterations and the learning rate.