Read Me:

Rolling Map:

The purpose of this script is to introduce the splitting of a feature into strips of Hcontent and mass.

Separation of Data:

There are two nested loops, M (mass) and H (Hcontent). The M loop splits up the data into multiples of 81 (each mass has 81 Hcontent points associated with it in the Z=0.02 grid). The H loop splits the data in 6 groups, each trained entirely separately, and chosen from experience (not too many groups, not too few etc..). Overlap is used here.

Prepare for Training:

First take the log of all columns excluding the mass column. Then the data is scaled between 0 and 1 and shuffled to prepare for input into the model. An extra column (‘radius2’) is introduced to aid inverse scaling afterwards. The weighting of all data points is initial set to 1.

Training:

The data is first put through small models with large learning rates to learn the general characteristics of the data. Next, the absolute error of each data point is calculated, and added onto the weighting of each data point respectively (with a power of 0.2 found by trial and error which ensures the model does not disregard the initially accurate data). The data weighting is designed to help learn the more ‘troublesome’ data.

Next, the model uses a very low learning rate, potentially high epoch number and the calculated weightings to carry out the bulk of the training.

Results:

For each loop over H, a fresh DataFrame ‘results’ stores the outputs and other relevant data which is concatenated onto a larger DataFrame ‘answers’. The overlap is removed prior to this. The scaling is undone. Further predictions are made on an array with 1000 points to provide information on the model behaviour between the data points.

Plots:

A plot is produced of the loss metrics of the model to show the learning process. All the predictions are on the next graph. Next, predictions are made on the grid points and the percentage errors calculated. This is used to make a heatmap.

Predictions

The point of this script is to carry out custom predictions. However, this is tough as the pre-processing must be replicated exactly. Importantly, if wishing to train on many masses then predict on one, then the scaling must be noted. I am sure that there must be many more efficient ways to do this whole process; but I didn’t find them! Also, this only works on a single mass choice for now.

Pre-processing:

The data is separated into the 6 groups, the X0, X1 ... X5. Also, the feature scaling of the training must be replicated, so the relevant parts of the dataset (Y0 ... Y5) are produced for scaling purposes. MinMaxScaler scales everything 0 to 1. Dataset\_pred allows for as much interpolation as desired.

Predicting:

The relevant models are loaded from the locations set by the training script. Reverse scaling is carried out. The outputs of the 6 models are concatenated.

Note there is a well-documented keras bug where the model does not load and predict properly (<https://github.com/keras-team/keras/issues/4875> for example) but should give the exact same results as the training script (I checked the models are identical down to the individual model weights).

In theory, if a certain feature of a star with a mass M is desired then the training script can be used to learn this feature (at any number of masses). The predictions script can provide predictions at any Hcontent increment without retraining, or other masses nearby.

Comments:

In theory, it should be possible to extend the model to predict lots of features at once, by adding many more columns to the ‘results’ DataFrame. I am unsure of the effect this would have on accuracy. Similarly, if it were considered useful to predict many masses at once, this may be possible.

Z=0.02 or otherwise refers to metallicity.

The behaviour of the model in-between data points is relatively untested and the impact of overfitting may lead to unphysical predictions if not careful.

Extrapolation seems to be relatively effective in Mass, but tests extending beyond the Hcontent range suggests poor results.

Because of the pre-processing and the creation of a large dataframe to predict from, the prediction process is currently slow.