Abstract

Here, we present a modern machine learning architectures, fully connected and feedforward neural network called Icy. Icy has an ability to learn the relationship between the data and parameters upon which it is dependent, which can both determined by the user then predict the data. Mathematically, Icy performs regression in N-dimensional space, where N is the number of parameters relevant to the data. In this project, we used Icy to predict spectra in order to increase the number of models in a pre-computed grid of radiative transfer models for dusty evolved stars, GRAMS (citation). Icy can predict the spectrum with certain range of wavelength under 5 different physical parameters. We will §2 Introduce the structure of Icy, §3 Test the performance of Icy and §4 Test the accuracy of its results.

§1 Introduction

[Comment to add to paper: The difference in flux between an interpolated spectrum and one calculated with PHOENIX is usually less than 1%, but increases for lower temperatures and can reach up to 10% for the coolest stars in the grid.]

The asymptotic giant branch (AGB) stars is one of the most important role to the life cycle of dust in the interstellar medium(ISM). These stars form large amounts of dust in their circumstellar shells. This dust is recycled to the ISM. AGB mass loss therefore significantly affects the chemical evolution of galaxies. Up to 75% of this dust is contributed by the dustiest 5% of the population (Riebel et al. 2012, Srinivasan et al. 2016). It is therefore important to identify these dusty stars and also constrain their dust-production rates.

The Grid of RSG and AGB ModelS (GRAMS)[1] is a pre-computed grid of models used to measure the mass-loss return from evolved stars. The grid consists of radiative transfer models computed using the 2Dust[2] code. The underlying AGB/red supergiant (RSG) model photospheres are from Aringer et al. (2009) and have effective temperatures between 2600 and 4000 K, and luminosities from ~2000L to 40000L. The dust shell is constructed for five value of inner radius Rin (1.5,3,4.5,7 and 12 Rstar) and twenty-six values of optical depth at 11.3 um, ranging from 0.001 to 4. Totally it produced ~12000 models. These model can be used to fit the spectral energy distribution (SED) and compare with the observation source. For instance, the GRAMS grid has been used to fit the SEDs of ~30,000 asymptotic giant branch (AGB) and red supergiant (RSG) stars in the LMC. We have used this grid to estimate the mass-loss return from evolved stars in the LMC (Riebel et al. 2012) and the SMC (Srinivasan et al. 2016). However, the current GRAMS grid does not have adequate coverage at high optical depths. GRAMS contains models for only 7 values of optical depth larger or equal to 1, making it difficult to place strong constraints on the dust content of the dustiest AGB stars. To have a accurate dust-production rates of these stars, more models at the higher optical depth are needed to extend the current GRAMS grid.

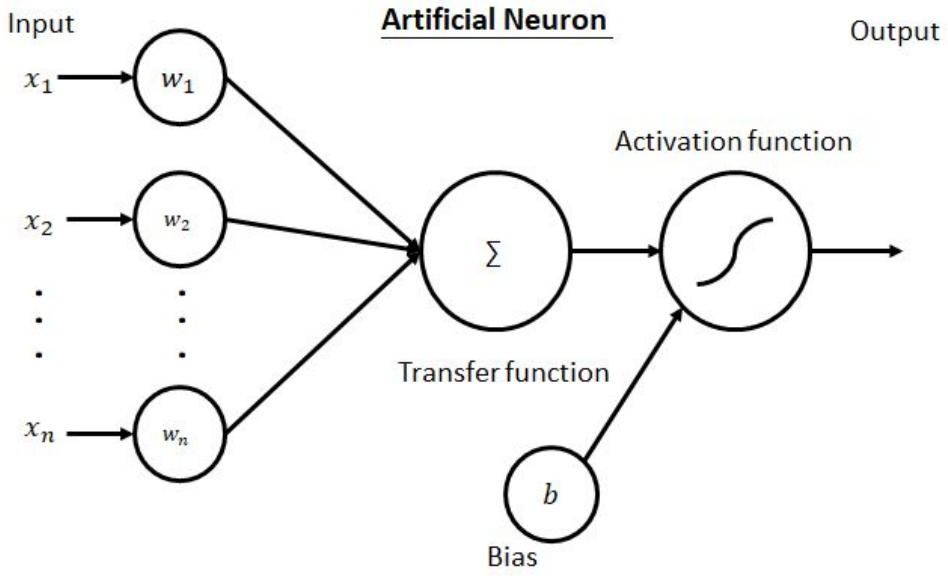
One of the possible way to extend the grid is to construct new models by interpolating the spectra of existing models in the grid. In this project, we use the popular machine learning technique of neural networks to extend the GRAMS grid.

---

[1] https://www.aanda.org/articles/aa/full_html/2011/08/aa17033-11/aa17033-11.html
[2] https://arxiv.org/pdf/astro-ph/0212523.pdf

(machine learning history notes:source 1.https://arxiv.org/pdf/1411.5039.pdf)

The artificial neural network is a computational model inspired by the working principle of human brains.[3] It simulates the biological neural networks in human brain. In an artificial neural network, each neuron, the basic unit of neural network, will have certain one or more input and output connections, each consisting of a neuron or an array of neurons. We call each array of neurons a layer. Inside the neurons will have an activation function inside. It received data from the input. The input can be the features or other neurons input signal. It will first sum up all the data with weight in the transfer function and add bias constant then put the processed data into the activation function. The output value can be the signal to next layer or the final output value.[4] The following figure shows the working concept of a single neuron.



One of the simple type of neural network is the fully-connected and feedforward network. For a feedforward network, the output of the neuron must be directed either to the output features or to the next layer neuron. The word "fully-connected" means that the input number of each neurons are the data source features or previous layer neuron number and the output number are the output label or the next layer neuron number.

For workflow of neural network, it input the features to the neural network and it will output the predicted answer. It can be train then perform well and comparable with the true answer. Therefore it can provides a high accuracy and help us to extend the current GRAMS model using neural network. We will talk more in the §2.

So, we developed a neural network named Icy to extend our current GRAMS grid.

[3] https://link.springer.com/chapter/10.1007/978-1-4615-0377-4_5 or http://www.cs.cornell.edu/courses/cs4700/2011fa/lectures/13_Ann.pdf (need find the earliest)
[4] http://pakacademicsearch.com/pdf-files/com/507/24-28%20Vol%203,%20No%201%20(2013).pdf

§2 Architecture of Icy

Icy is a three-layer fully-connected and feedforward neural network (In deep learning , we dont count the input layer as one layer) . The following table showed the detail Structure of Icy and detail of each layer.

Layer	neuron Input number	neuron Output number	Activation function	Bias
Input Layer	Number M By users	4*M	Sigmoid	True
Hidden Layer 1	4*M	Number N By users	Sigmoid	True
Hidden Layer 2	N	N	Sigmoid	True
Output Layer	N	N	Linear	True

In GRAMS, All of the spectrum data depended on 6 parameters/features. Inner Radius(Rin), Effective Temperature (Teff), Mass (Mass) , Carbon-Oxygen Ratio(C2O), optical depth at 11.3 um (tau) and logg (logg). Since most of the data using Mass=2Ms, GRAMS mainly depended on 5 parameters(Rin,Teff,C2O,tau,logg). As an output, GRAMS spectrum gave out 130 data points with range of wavelength from 0.2339 to 188.3 um. So we selected M,N=5,130.

Icy learnt the relation between the 5 parameters and 130 spectrum output in the training process. At this paper , we selected 11753 models from total 12243 models with Mass=2 for training and testing use in GRAMS model.

Before the training process, All the parameters also called feature and the spectrum value (called label) will being pre-processed. The label being neutral log process because the range of the value wide from order from 10^{-42} to 10^{0} . The log process help to avoid the situation of low absolute error but high error percentage at the small value region. Then, the label will being pre-processed by following expression:

$$new\,feature = \frac{0.8*(feature - Min(all\,feature))}{(Max(all\,feature) - Min(all\,feature))} + 0.1$$

Where Min(all feature) and Max(all feature) is the minimum and maximum of parameter set. That made the new feature value bounded between 0.1 and 0.9. This helped Icy converge to solution faster in the training process. We then randomly pick 95% models for training and 5% for testing from those 11753 data.

The training process were similar to least square fit process. We first define the error function. We were using the square error as the error equation at this project. It can be express as:

$$Error = \sum_{n}^{Number\ of\ training\ data} \sum_{m}^{Number\ of\ data\ points\ in\ each\ spectrum} (h_{mn} - y_{mn})^2$$

Where y_{mn} is the log spectrum value from GRAMS model (called label in machine learning) and h_{mn} is the predicted value from Icy's output layer. Since the last layer was linear activation function , the mathematical expression of $h_{mn} = (W^{[3]})^T * a^{[2]} + b^{[3]}$. W , b were the weight and bias parameter and the number 3 donate the meaning of third layer. It was a 20*130 matrix. The a^2 was the value forwarded from previous layer (Hidden Layer 2) . Since the previous layer use the Sigmoid function, the mathematical expression of $a^{[2]}$ will be $a^{[2]} = \frac{1}{1+e^{-z^{[2]}}}$ and $z^{[2]} = (W^{[2]})^T * a^{[1]} + b^{[2]}$.

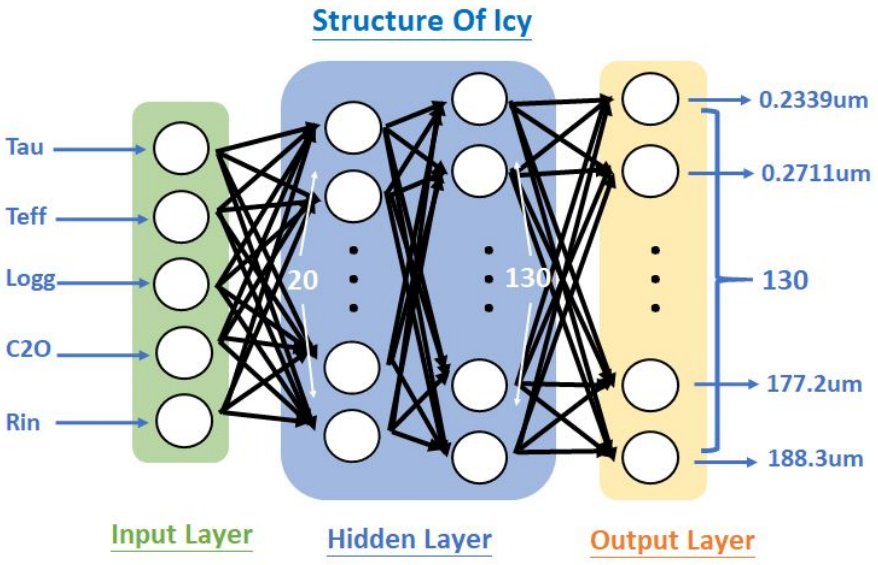
All the layers will have weight and bias parameters and those value were randomly chosen by the numpy random library[5] before training process. By finding the correct value of those two types of parameter, icy can predict the spectrum value with high accuracy. Therefore it become a optimization problem of finding the minimum value from derivative between Error function and all W,b. In this ____, we use popular optimization technique from machine learning called ADAM[6]. It is one of the method from stochastic gradient descent to find the minimum value. ADAM helps the training process converge to the local minimum in the region of the Error equation faster and lower local minimum value.

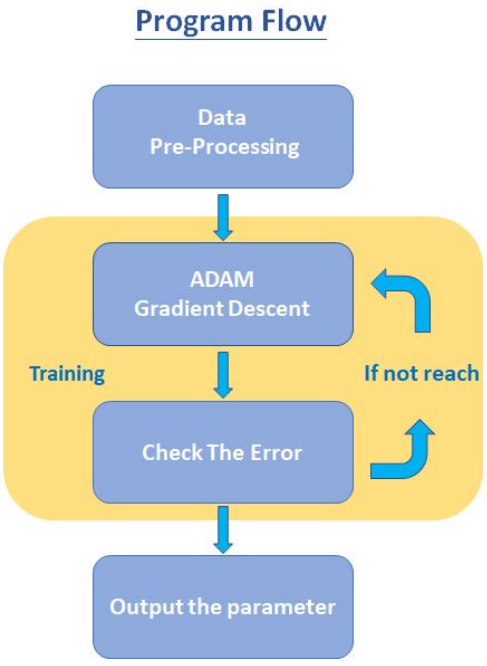
The training process will keep updating the parameter to adjust the network until the error reach the value set by user. This value affect the computation time since high accuracy take more time to converge. more detail will be discussed at §3 . In this paper, we called that value ϖ and define a function to to determine ϖ which can express as:

$$value\ \varpi = e^{\frac{\sqrt{Error}}{Number\ of\ data\ points\ in\ each\ spectrum \times Number\ of\ training\ data}} - 1$$

We choose $\varpi = 4 \times 10^{-6}$ as the final product and use it for error analysis at §3. The ϖ is the average value of root mean square distance between the data point from true value. When the error reach this value, the whole training process finished then saved the parameter value , training and testing data as hdf5 and pkl file for user predicting value. The following figures show the flow chart of training process and more detail of Icy structure.

[5] https://docs.scipy.org/doc/numpy/index.html
[6] https://arxiv.org/pdf/1412.6980.pdf





§3 Performamce Test

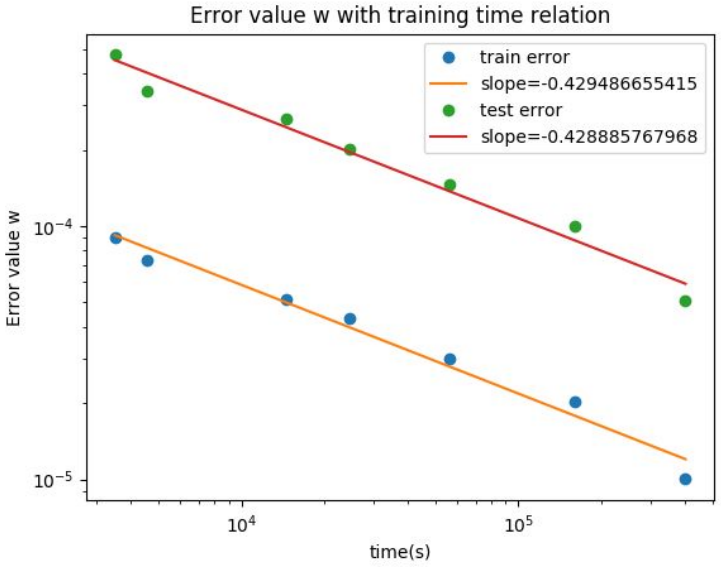
We used the new modern python deep learning library Pytorch[7] to build up the neural network in this paper. Pytorch provide the advantage of imperative programming and easy gpu speed up by just modifying a few lines of code. Icy provide both cpu and gpu version for user.(It may change to a option to turn on /off). Also, Icy support mini batch training which is training the model using limited

[7] https://github.com/pytorch/pytorch

number of training data each. It is a training method to speed up the training process when using cpu. We also made a time performance test of Icy with difference value of ϖ. In this test, we were using the following setup:

CPU	Intel(R) Core(TM) i7-3720QM CPU @ 2.60GHz
RAM	16 G DDR3
Environment	Python 2.7.12 with Anaconda custom (64-bit) using GCC 4.4.7
Pytorch Version	0.1.12
CUDA	8.0
optimizer	ADAM with betas=(0.9, 0.99)
Learning rate	0.005
GPU Speed up	Off
Mini Batch	Off

Here were the result:



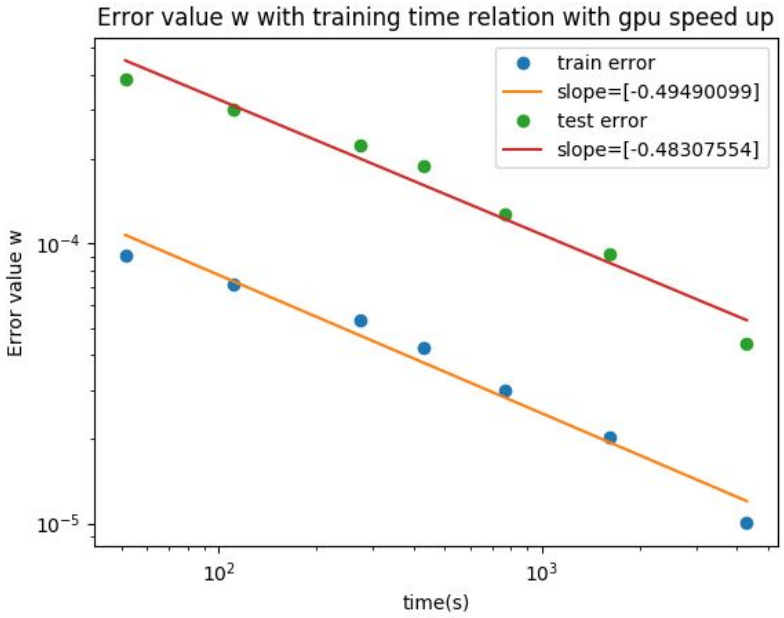
In this test, we plotted two data set ,the training data error and testing data error. The training data error is the value ϖ input by the user. The testing data error is the error between testing data set and predicted value from Icy under that value ϖ which testing data did not join the training process. It is a reference to monitor the overfit problem when training. Total 7 data points of ϖ from 9e-5 to 1e-5 tested. We can see the testing error followed the decrease trend of training value ϖ when the training time is increase. That meant Icy were well trained and not overfitting at those ϖ. The Error decrease relation was linear in log space.

This test also showed a problem of long training time at low value ϖ when even using 4 Cores cpu. The importance of gpu speed up raised for reaching the low value ϖ. So, we also done the same performance test using gpu speed up in the following setup:

CPU	AMD Ryzen(TM) R5 1600 CPU @ 3.2GHz
RAM	16 G DDR4 RAM
GPU	NVIDIA Geforce 1050 Ti 4GB 128bit GDDR5
Environment	Python 2.7.12 with Anaconda custom (64-bit) using GCC 4.4.7
Pytorch Version	0.1.12
CUDA	8.0
optimizer	ADAM with betas=(0.9, 0.99)
Learning rate	0.005
GPU Speed up	ON
Mini Batch	Off

Here were the result of computational time analysis:

ϖ	9×10^{-5} (11.12%)	7×10^{-5} (8.65%)	5×10^{-5} (6.18%)	4×10^{-5} (4.944%)	3×10^{-5} (3.70%)	2×10^{-5} (2.47%)	1×10^{-5} (1.24%)
CPU time	3492s	4588s	14540s	24462s	56526s	159172s	400606s
GPU time	51s	111s	274s	429s	767s	1622s	4296s
speed up	67.96x	41.17x	53x	56x	73x	98x	93x

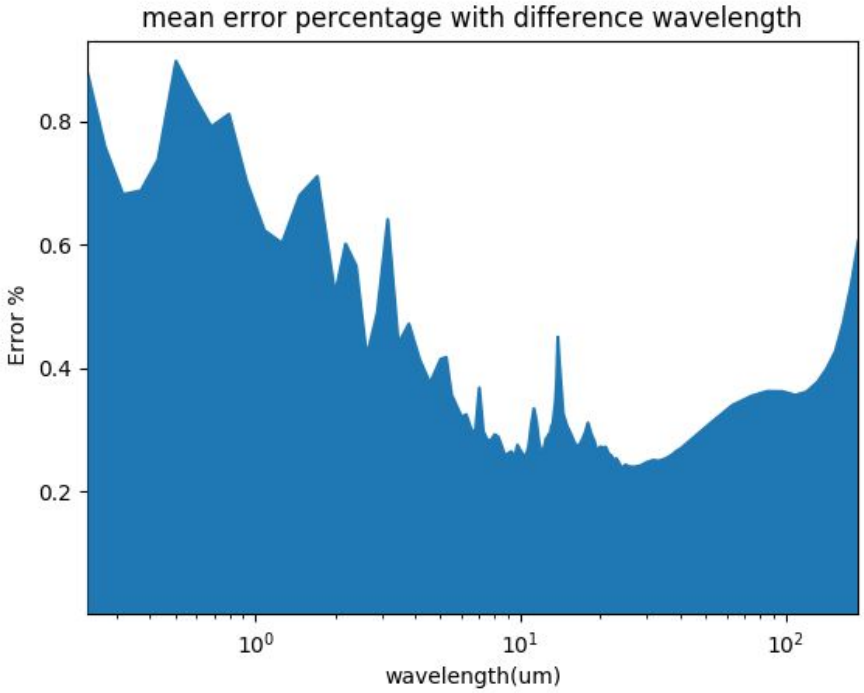


That showed the time decreasing trend also happen and linear relation in log space while using gpu speed up. The result was reproducible. The speed up of gpu between 41 to near 100 times. Also the speed up was more faster when ϖ is small. As a short conclusion, gpu training is strongly recommended while high accuracy is needed.

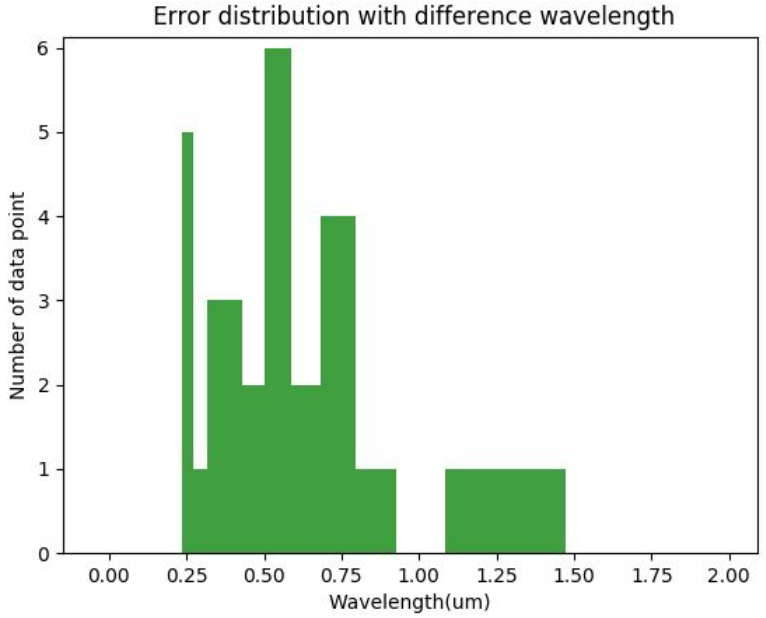
§4 Accuracy Performance And Discussion

§4.1 Error Analysis

Icy had a very high accaury performamce after training process with value $\varpi = 4 \times 10^{-6}$ and testing set error = 2.32×10^{-5} . It still obeyed the linear relation in log space which showed at §2 . we will use the testing data set for the following error analysis. The average error percentage between the prediction and testing data set was 0.434% which from the GRAMS model. A plot of mean error of difference wavelength at SED also showed more detailed error distribution .



To ensure the whole data is highly reliable, we selected the data points with maximum 5% error percentage or even higher for analysis. For those data points with 5% or even higher error percentage, we called those extreme data points. 29 out of total 61100(Number of testing sample * 130) extreme data points found which was about 0.05% of data. The maximum of error was about 12.7%. A histogram of extreme data points versus difference wavelength at SED also showed detailed distribution .



All of the extreme data points were at the short wavelength. It would discuss later.

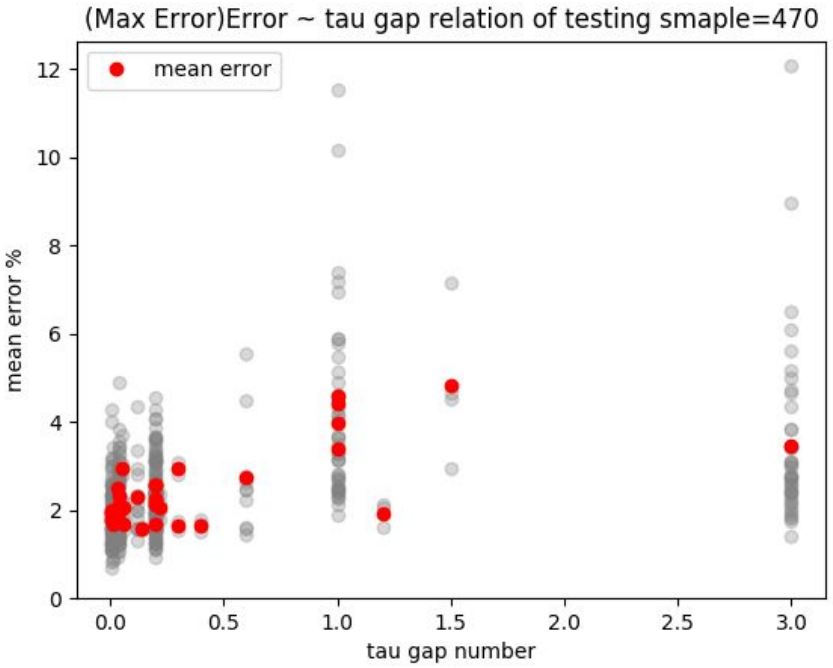
Since Icy predict the whole spectrum under 5 parameters, we also using the whole spectrum as unit for analysis. We selected the maximum error of data point in the spectrum and look at the relation

between the parameter and error. We called that maximum error point. Total 18 out of 470 spectrums found extreme data points from the maximum error point. About 3.8% of the whole testing data set.

All of the extreme point in testing set shared a property that their optical depth is equal or larger than 1. It can be explained by GRAMS model data. Since the GRAMS model have intensive value of optical depth below 1 but alienated after 1. The following table showed the total optical depth data from GRAMS

Value Distance	Optical depth Value
>=0.001~0.002	0.001,0.002,0.004,0.006,0.008
>=0.01~0.02	0.01,0.02,0.04,0.06,0.08
>=0.1	0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9
>=0.5	1,1.5,2,2.5,3,3.5,4

All the extreme data exist at the widest value distance 0.5. We can also plot a error percentage versus value distance plot to ensure this. For each maximum error data point , we search the two nearest optical depth value in front and below that data point from the training set.



The gray dots were the maximum error data point selected from each spectrum. The red dots were the mean error of all gray dots at the same value distance. All of the extreme data points had a value distance/ tau gap number were larger than 0.5 The value distance distance larger than 3 mean they could not find two near data points. Those data points were the outside the tau that training set had. We called that lied on the edge or extrapolating value. The extrapolating value perform good compare with lagrangian interpolation but still had a larger error. Since It is an 5 dimensional space, that mean

some data points would lied at the “corner” which is extrapolating at all five parameter. Because of that, a plot of edge number versus error also showed the relation of edge-error distribution.

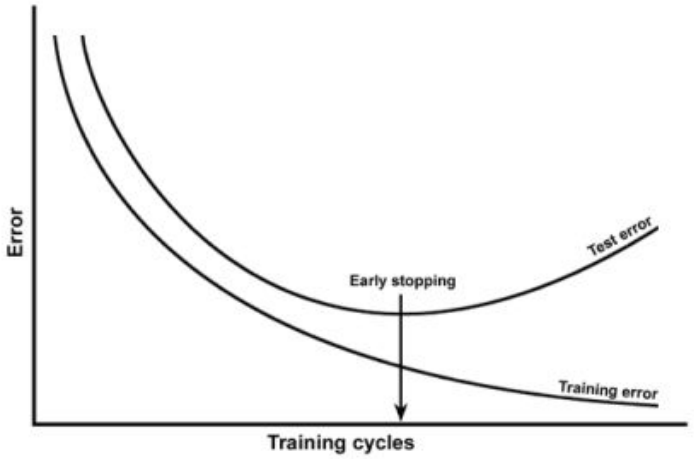
edge number 0 told that this data point not lied on the edge. It showed that the error will have a linear increase relation when the edge number increase. Also most of the data lied on at least one edge. To keep the predicted data have a low error percentage, prevent predicting the data point lie on multiple edge at the parameter space.

§ Discussion

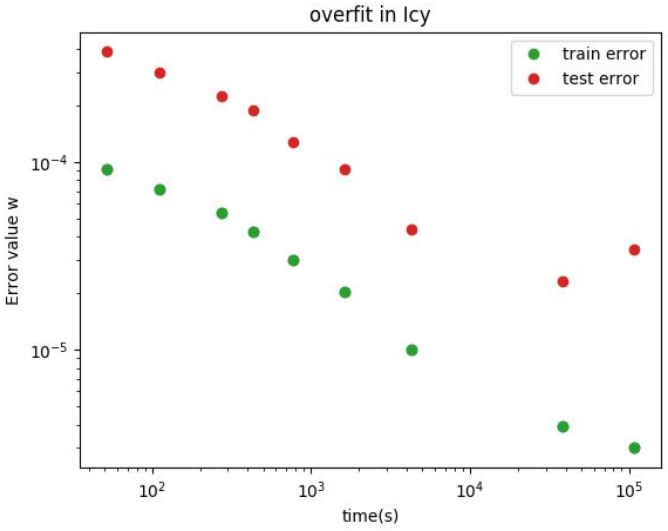
Limit of Icy: Overfit

1. As the training time increase, the error of the training data kept decrease and trend to 0 but it is not true for the testing data. As more training cycles lower the ϖ value , the test error just reach a minimum then the error increase again . That called overfit phenomenon. So finding a right training cycle or ϖ value let the model perform the best. The following figure showed the concept of overfit in machine learning.[8]

[8] http://www.sciencedirect.com/science/article/pii/S0893608098000100



We found that, the best value ϖ can get in this model is 4×10^{-6}. We also tried the $\varpi = 3 \times 10^{-6}$ but the test error increase to 3.44×10^{-5} which using 108529s. The following plot in log space showed the issue.



So, the best value will between the $\varpi = 4 \times 10^{-6}$ and 3×10^{-6}.

2. Another factor would cause the overfitting is the model complexity. The model complexity meant how complicated the structure of a neural network. For instance, the number of neurons and the hidden layers. Just like the training time, increase the number of neurons and layers will not always decrease the error but too simple structure will also cause the underfit problem. Choosing a right model complexity also a quite important factor for the accuracy of Icy. Nonetheless, it is not an easy problem. Not only the trying different structure took times,but also more complicated structure took more time to train. The four layer structure using here already took about 12 hours of GPU time to train the best value setting. It was a very time consuming. So, the model presenting here is between the accuracy and time.

(Still under testing, trying to show a table of difference architecture of model will have difference accuracy it can get and the current one is the best model)

Conclusion

The new developed neural network Icy is high reliable with maximum of 12% error and average error of 0.434%.