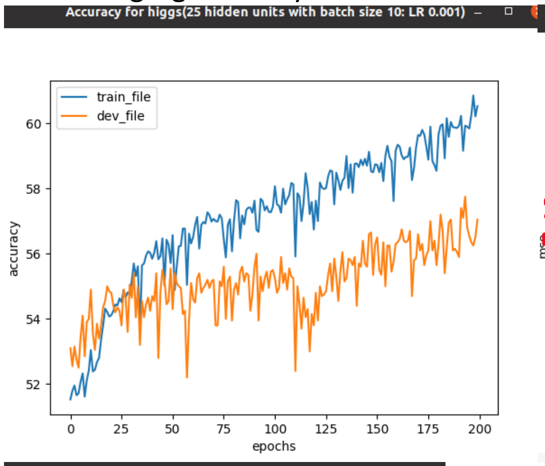
Report

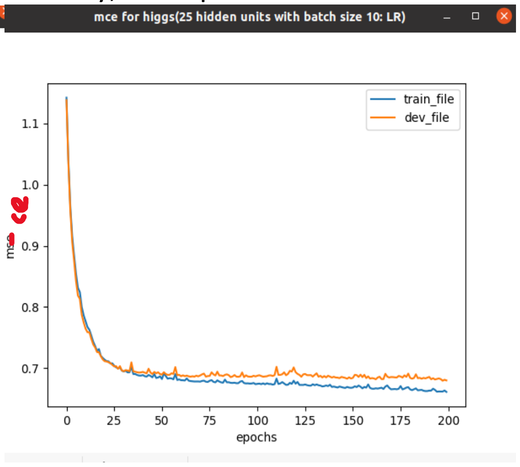
Abstract:
This report is about the results of experimenting 2 data sets, higgs and htru2. In the experiment, a neural network with only 1 hidden layer and arbitrary hidden units was used to make binary classification. The activation function of hidden layer is “tanh” and function of output layer is “softmax”. Since the neural network has only 1 hidden layer, the resulting accuracy might not be very high with some parameters (hidden units, learning rate, epoch). However, in this report, you can find most figures of mean of cross entropy error function was really descending with respect to epochs, which is a good direction.

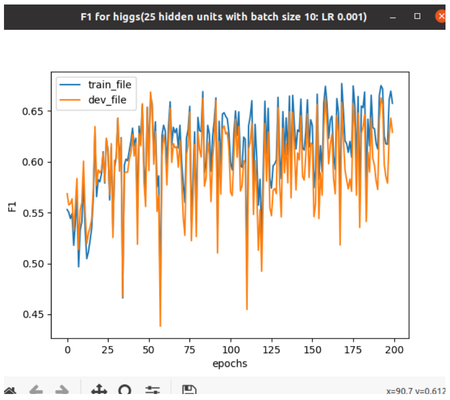
Higgs:

For all experiment done for higgs, I used batch size 10 with varying other parameters.

Firstly, I did a basic try with 25 hidden units, 200 epochs, learning rate 0.001. The reason I choose a relatively low learning rate, 0.001, is I tried use 1, 0.1, 0.01 and the accuracy was fluctuating significantly. Back to the first basic try, the 3 plots are shown below:

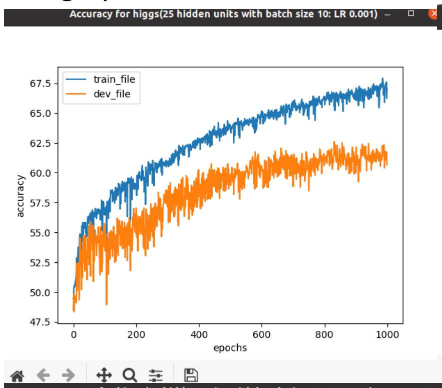


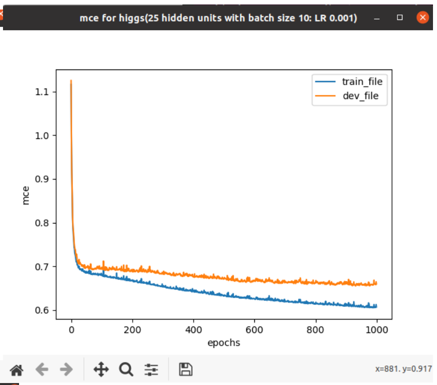


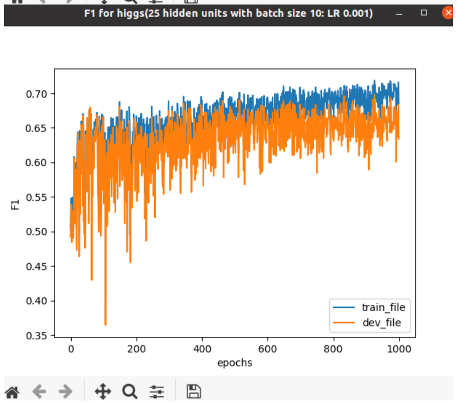


Compared to the pervious naïve trials, which are not shown graphically here, it at least had reasonable shapes with the parameters. As we can see, the accuracy of train file and error of train file are mostly better than development file ‘s accuracy and error. One possible reason is that with this many hidden units, the model already starts overfitting with provided dataset. It can be partially solved by adding “regularization term” in Error function with further implementation. However, either the error or accuracy of development file is generally increasing in the trend.

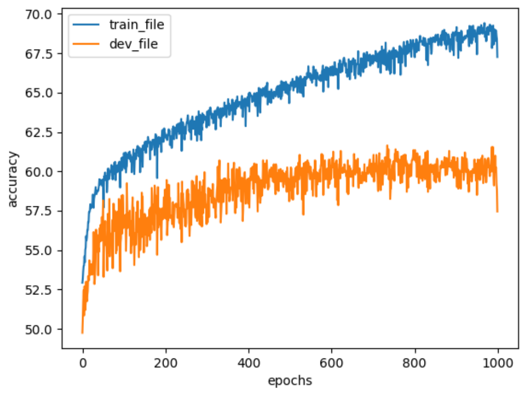
Thus, I expected that if I could give it more epochs to train, it would result in better outcome. Therefore, I trained it with 1000 epochs with same number of hidden units and learning rate. The graphs are shown below:

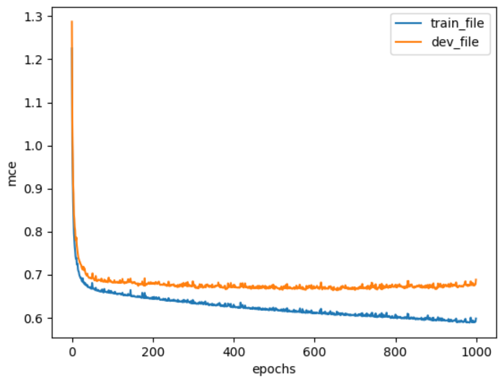


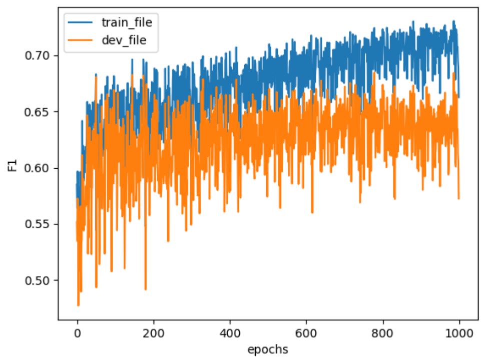




As we can see, the accuracy further increases and the mean error further decreases with more epochs, but the trend is still increasing for accuracy and decreasing for error. Because of limitation of my computer and time, I could not do 1000 more epochs to see an even better result. Instead of that, I tried to increase the number of nodes, from 25 to 35. The graphs are shown below:







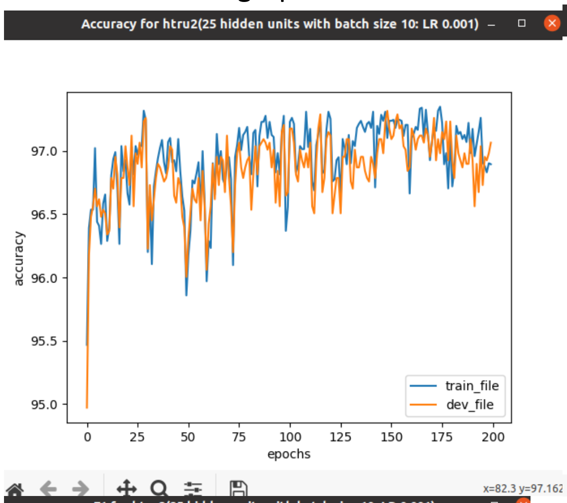
In the 3 graphs above, the difference of accuracy between train file and development file is larger, without significantly increasing the result. Then we can conclude, the effect of overfitting becomes worse, so the accuracy of development file is even decreasing compared to the model with 25 nodes. **Thus, based on the trials I did for this data set, the better parameters should be 1000 epochs, 25 hidden units, learning rate 0.001, batch size 10.**

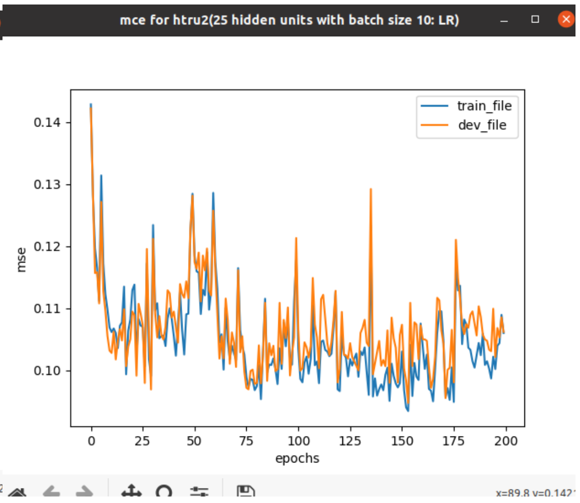
In summary, even I did 1000 epochs, the accuracy of my model was still potentially increasing. I expect that one could still get better result by increasing epochs. I think I could not simply change hidden units and learning rate to get better result, because increasing learning rate would cause significant fluctuation as I tried before and decreasing learning rate would require even more epochs to get present accuracy which would take much longer time. By increasing hidden units is also not a solution because it already started overfitting when I increased the number to 35. However, I think the direction of my model is good, because the error was truly descending which is meaning of gradient descent. For the f1 scores, the model can easily reach the place greater than 0.6 which is fair enough.

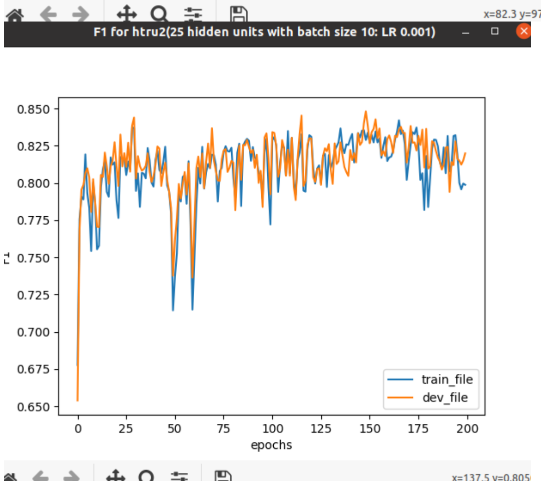
Htru2:

Similarly, for data sets htru2, I also chose 10 for batch size.

Like before, I also ran a basic trial for this data set, with 200 epochs, leaning rate 0.001, 25 hidden units. The graphs are shown below:

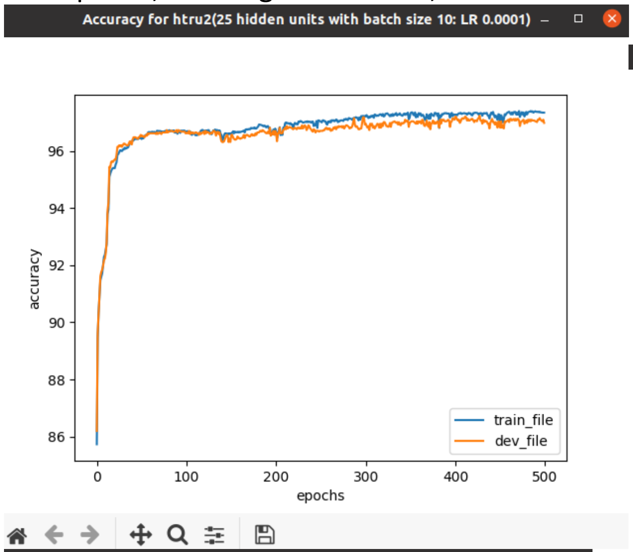


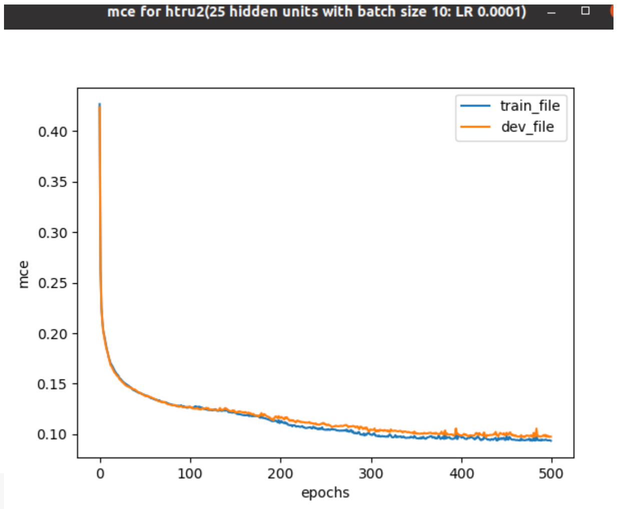


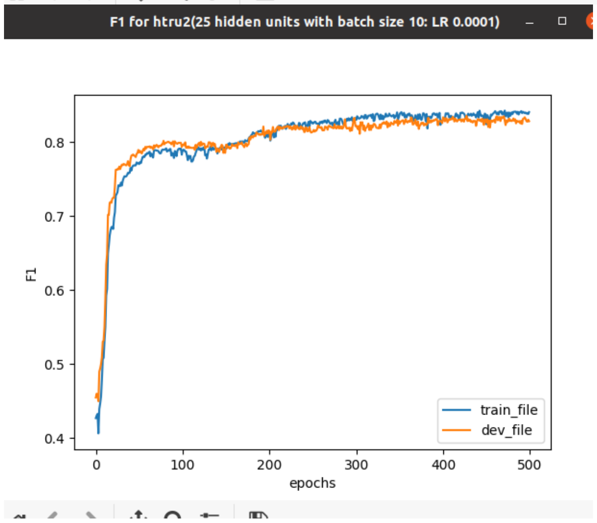


From the fluctuations of accuracy and mean error, we can see the learning rate must be set too high for this data set because the model kept missing local minimum. Thus we do the trial with

500 epochs, learning rate 0.0001, 25 hidden units.

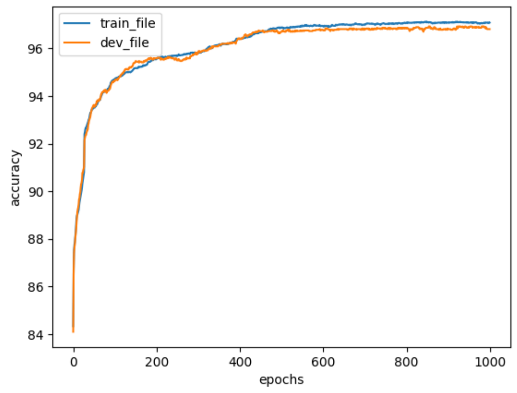


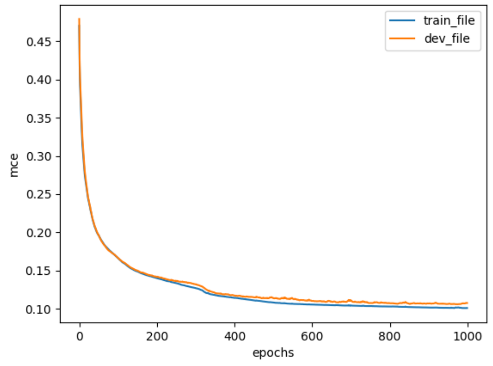


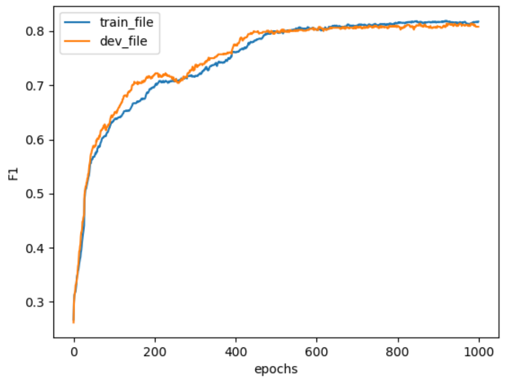


As the content shown in graphs, with decreasing in learning rate, the curve became smoother, meaning the model was doing more effective gradient decent. Even though we still can see some fluctuations at the tail of curve, which means the model probably was missing the minimum by a small amount. Thus, it is worth to try even small learning rate. Since the graph did not present a very serious overfitting problem, difference between train curve and dev_curve is too high, we could try to increase the hidden units at the same time.

Then I use the model with 1000 epochs, 0.00001 learning rate, 35 hidden units. Graphs are shown below:







With updating the parameters, the model's curves became much smoother, which means it was correctly descending, but the overfitting problem was not enlarged by increasing the node. Although I expect one can achieve more accurate model by increasing epochs with such small learning rate, due to capacity of my computer, the best model I could come up is this one. **Thus, the recommended parameter, based on these trials, are epochs 1000, learning rate 0.00001, hidden units 35, batch size 10.**

In summary, the quality of the model can still be potentially increased by increasing epochs. However, I think the model is already fair enough with high f1 scores.

Notice:
There are more graphs with different parameters in the folder named "pic_for_mlp"