Name: Phuc H. Lam

CSC246 Report

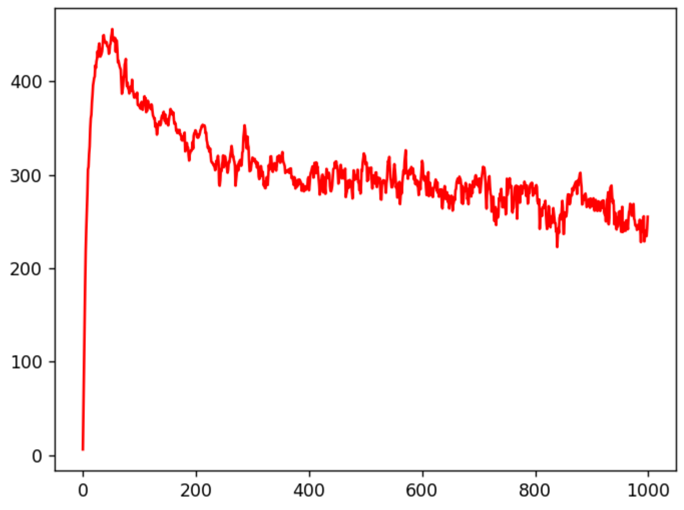
1. Results and recommended parameters

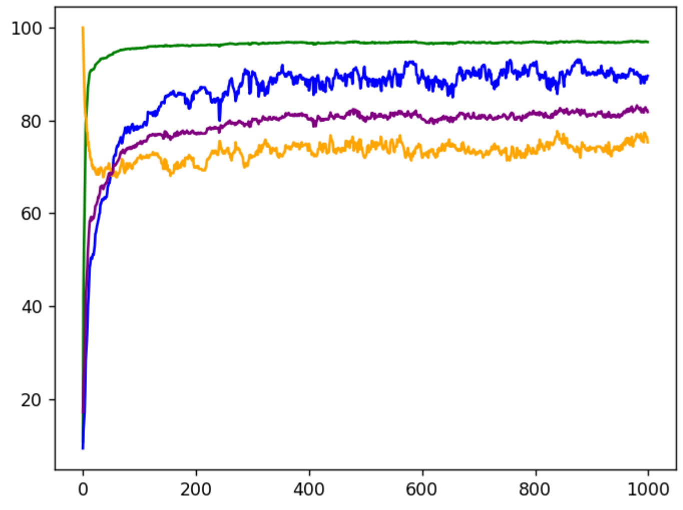
In the below, "red" shows the cross entropy loss in each epochs, "green" shows accuracy, "blue" shows precisions, "orange" shows recalls, and "purple" shows F1 scores. Here are a few examples.

a) epoch $= 1000$, hidden units $= 100$, batch size $= 256$, learning rate $= 0.02$

Run time: 45.94 seconds.

Here, the accuracy hits a plateau of $\approx 97\%$, and the F_1 score a plateau of $\approx 84\%$. This means we would not try to change the epoch; instead, we modify the other parameters.

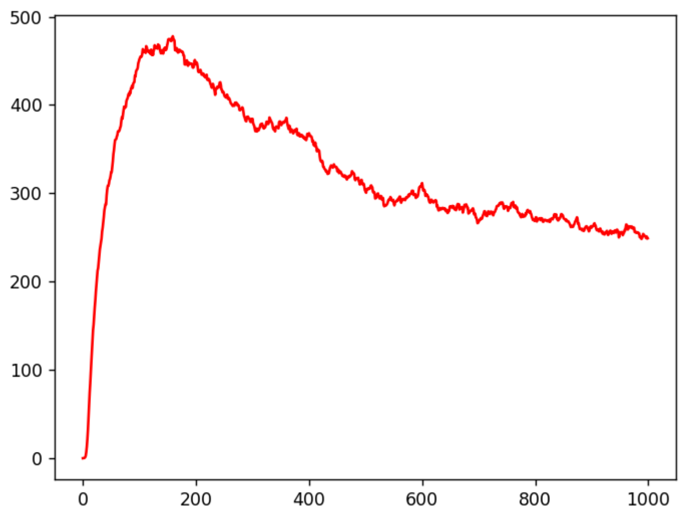


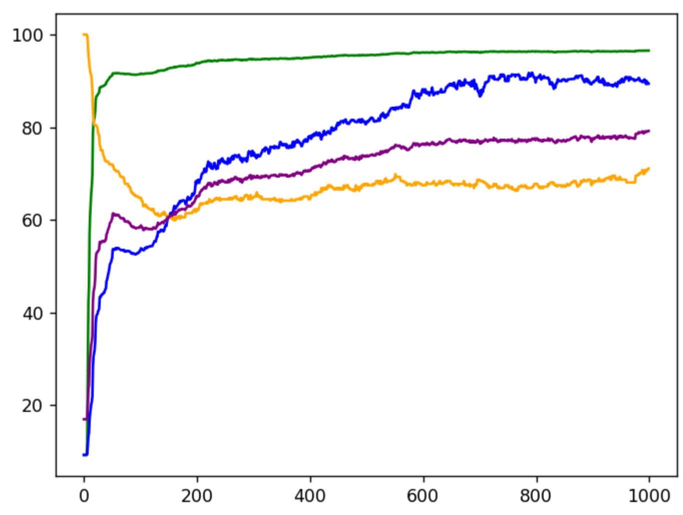


b) epoch $= 1000$, hidden units $= 100$, batch size $= 256$, learning rate $= 0.01$

Run time: 46.39 seconds.

Here, the accuracy hits a plateau of $\approx 97\%$, and the F_1 score a plateau of $\approx 79\%$.

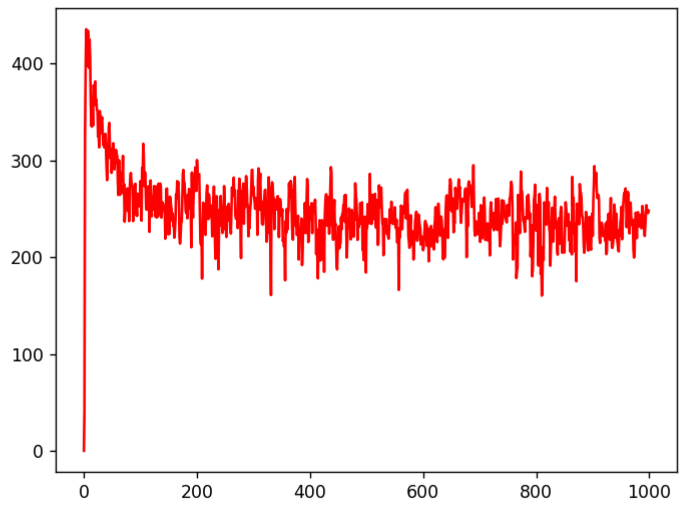


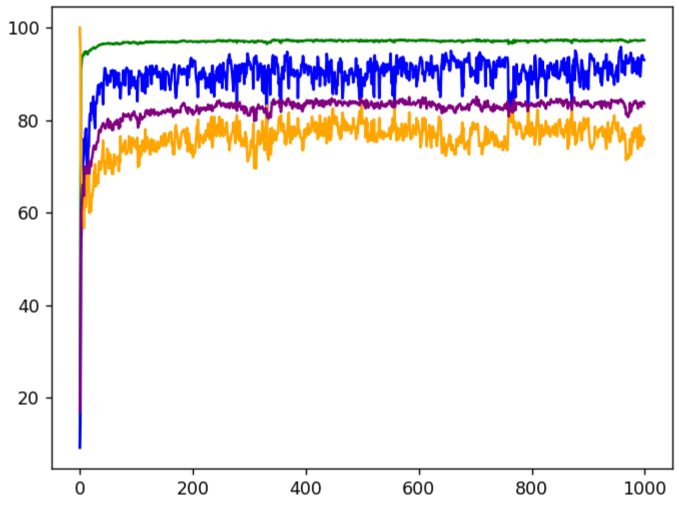


c) epoch $= 1000$, hidden units $= 100$, batch size $= 256$, learning rate $= 0.1$

Run time: 41.14 seconds

Here, the accuracy hits a plateau of $\approx 97\%$, and the F_1 score a plateau of 84%.

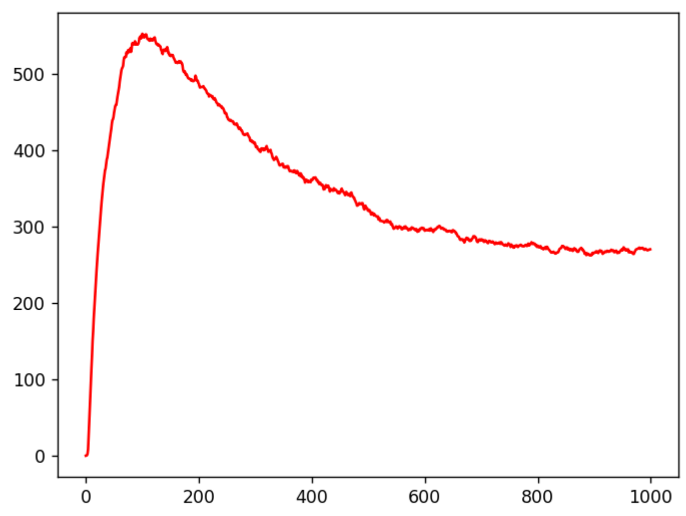


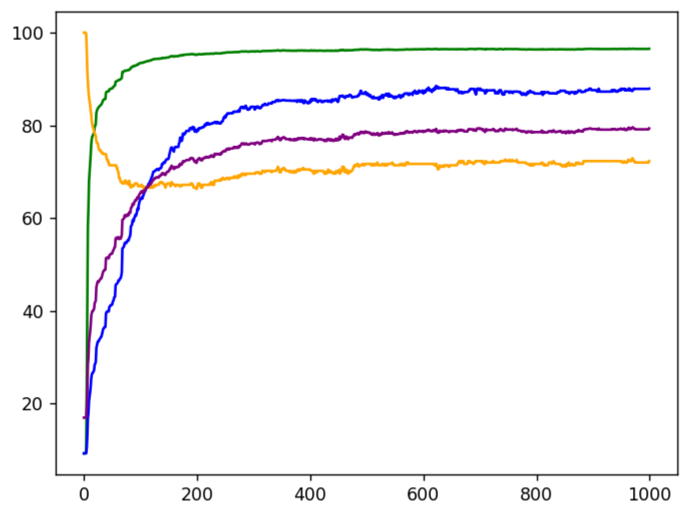


d) epoch $= 1000$, hidden units $= 100$, batch size $= 512$, learning rate $= 0.01$

Run time: 47.90 seconds.

Here, the accuracy hits a plateau of $\approx 97\%$, and the F_1 score a plateau of $\approx 80\%$. From b) and c), it seems like the 0.02 learning rate works better.

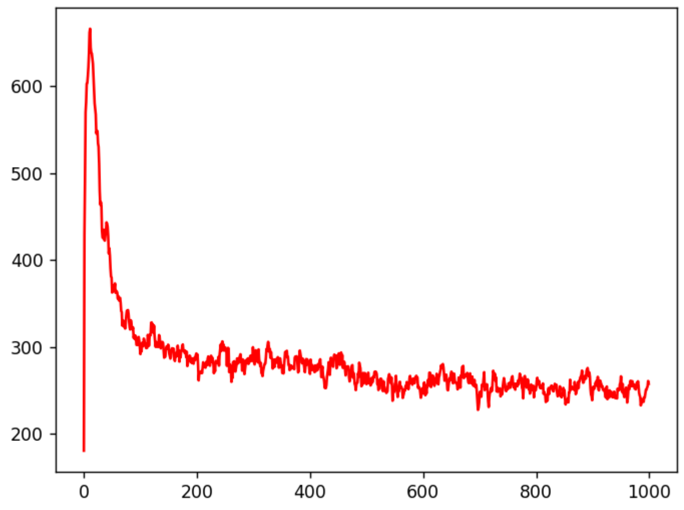


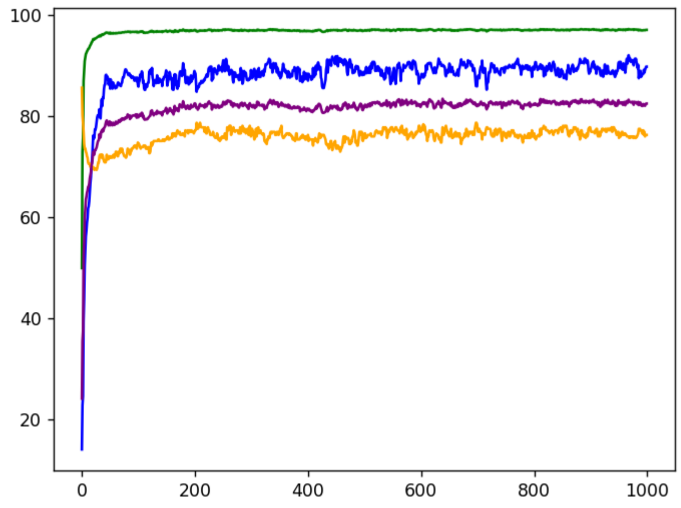


e) epoch $= 1000$, hidden units $= 300$, batch size $= 512$, learning rate $= 0.02$

Run time: 81.70 seconds

Here, the accuracy hits a plateau of $\approx 97\%$, and the F_1 score a plateau of $\approx 82\%$.

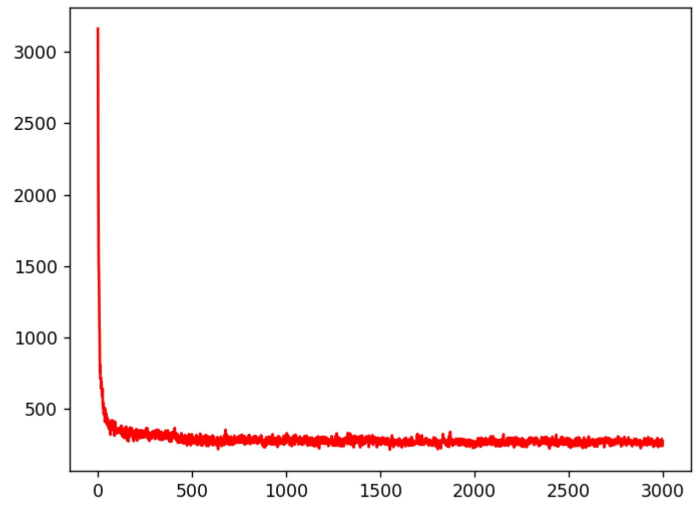


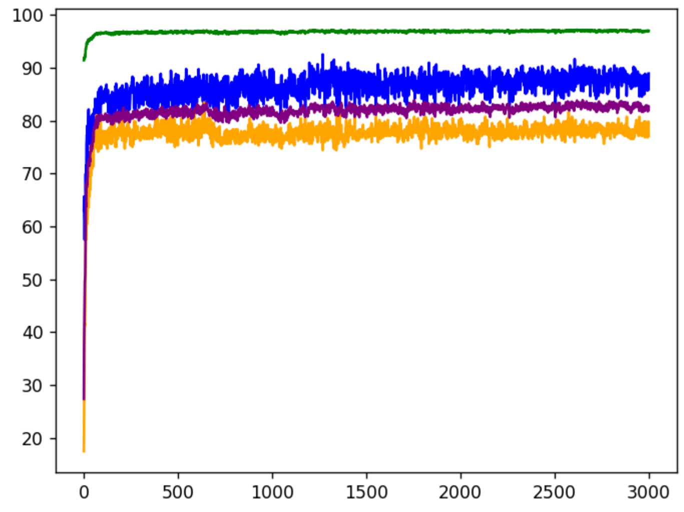


f) (Trying something extreme here) epoch $= 3000$, hidden units $= 1000$, batch size $= 512$, learning rate $= 0.02$

Run time: ≈ 6 minutes. (This is the first one that I run, and without the timing yet, so this is just an approximation of the actual run time)

Here, the accuracy hits a plateau of $\approx 97\%$, and the F_1 score a plateau of $\approx 84\%$.

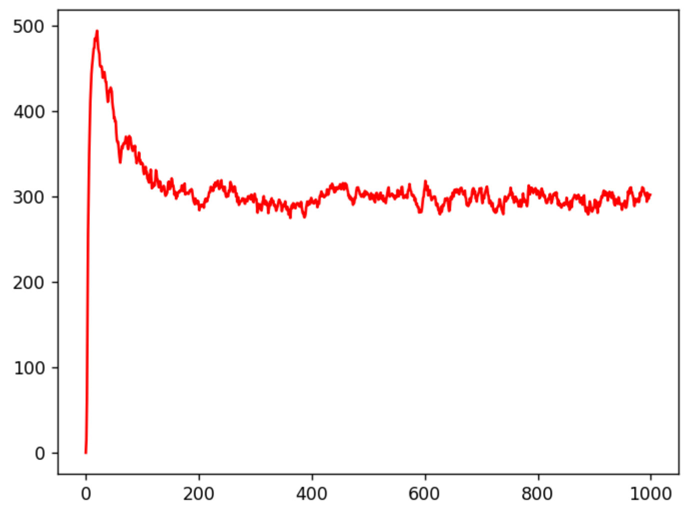


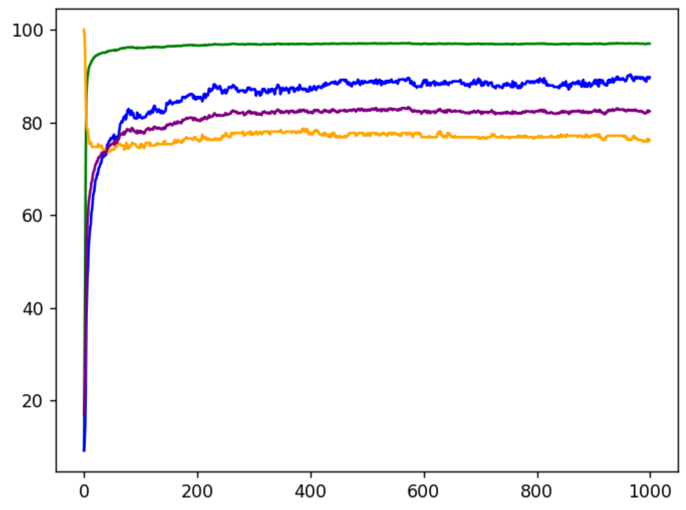


Thus, it seems like drastically changing the number of hidden units does not do much difference to either accuracy or F_1 scores (other than that the loss function decreases much better). Moreover, more hidden units seems to be more computationally inefficient. This means that the number of hidden units can be kept around a few hundreds. (I suggest it could be around 500). Also, the accuracy and F_1 scores hit a plateau after some time, so increasing the number of epochs would not make a significant difference. So we keep it at ≈ 1000. From the first 3 examples, the learning rate can be kept around 0.01. Finally, although we did not try any other batch sizes than 256 and 512, from the above, a batch size of 512 can be considered reasonable. Here are the results.

Run time: 111.83 seconds.

Here, the accuracy hits a plateau of $\approx 97\%$, and the F_1 score a plateau of $\approx 82\%$





2. Instructions to execute program

To execute the program, we only modify the file **Implemented.py**. I made notes for the users at lines 15, 25, and 35 in the code. First, change the name of the training file at line 16 and the development file at line 26. Then, modify the parameters from line 36 to 41. (If the batch size is too large, users will receive a warning and the program returns NotImplementedError).

3. What I learn

The hardest part, for me, is to write the backpropagation method for the MLP class. Debugging this part and keep track of the matrices' sizes took me a few days (probably even more). Also, I could not use the **linearSmoke** and **xorSmoke** files. This is because the data points in **linearSmoke** has 3 features and 1 output, whereas those in **xorSmoke** has 2 features and 1 output. I could not resolve this problem, since my code is contingent on the data points in the training set and development set having the same number of features.

Modifying the parameters is also time-consuming, and different combinations (even the recommended ones above) is not assured to be the best ones. However, we should also take computation time into account when training the models. This is why even if a larger number of hidden units gives better accuracy and F_1 scores, we could not make them too large.