Report: Practice #3

ITE4053, **Deep Learning Methods and Applications.**
2016025305, **Jihun Kim<jihunkim@hanyang.ac.kr>.**

Python + NumPy implementation of binary classifiers using 2-layered network.

Run

Requirements
- Python 3.6 or later
- NumPy 1.17 or later

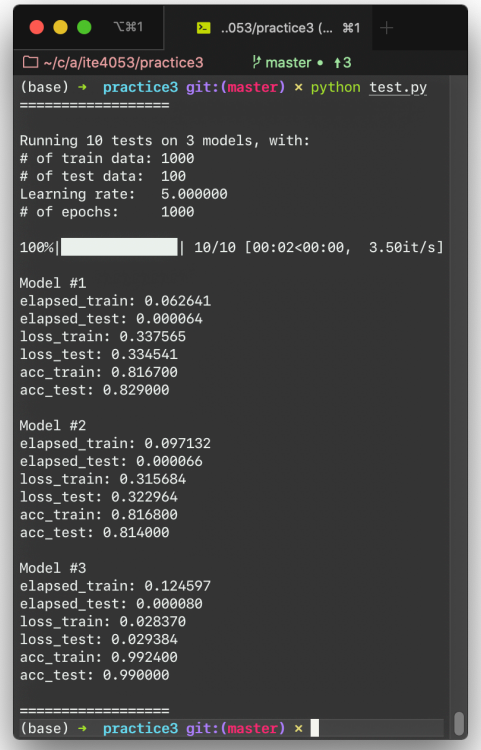
```
$ cd /path/to/repo/practice3
$ python task1.py   # To run test on specific task
$ python test.py    # To run test on every tasks
```

Results

Results below are mean of 10 tests per model,
trained and tested on hyper-parameters of:

- Number of train data: 1,000
- Number of test data: 100
- Number of epochs: 1,000
- Learning rate: 5.0
- Same train/test data among models per test

		Model #1	Model #2	Model #3
Accuracy	Train Set	81.67%	81.68%	99.24%
	Test Set	82.90%	81.40%	99.00%
Loss	Train Set	0.33	0.31	0.02
	Test Set	0.33	0.32	0.02
Elapsed Time	Training	62.64ms	97.13ms	124.59ms
	Inference	0.06ms	0.06ms	0.08ms



Conclusion

As the model being complex, its accuracy and loss generally improves because it is capable of making more complex decision boundary. In the same time, training and inference time increases and accuracy could be worse if training is not sufficient, due to increase of parameters that need to be learnt. Using profiler, I've found an interesting fact that sigmoid activation function consumes approximately 25% of total runtime, while other operations like dot product doesn't. It seems that using sigmoid as activation function largely affects performance.