NeuroNetwork Implementation

Brief:

實現手刻 Feedforward & Back propagation 的流程以及損失函數 cross-entropy，並且比較了三種不同的初始權重的用法。

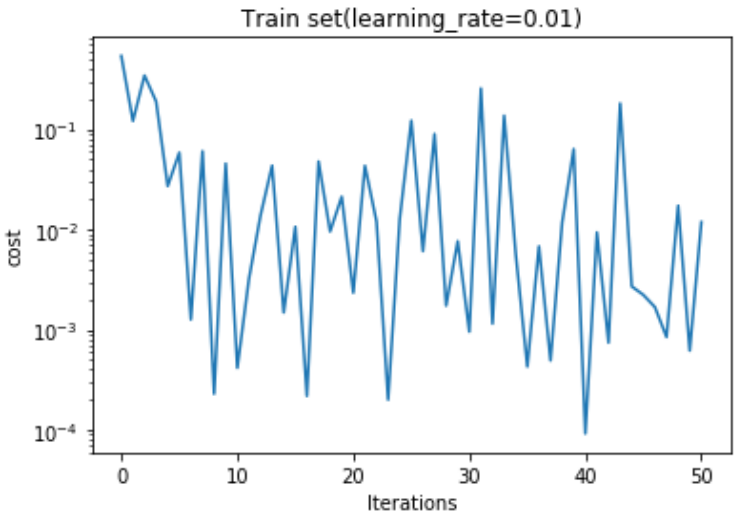
Results:

1.

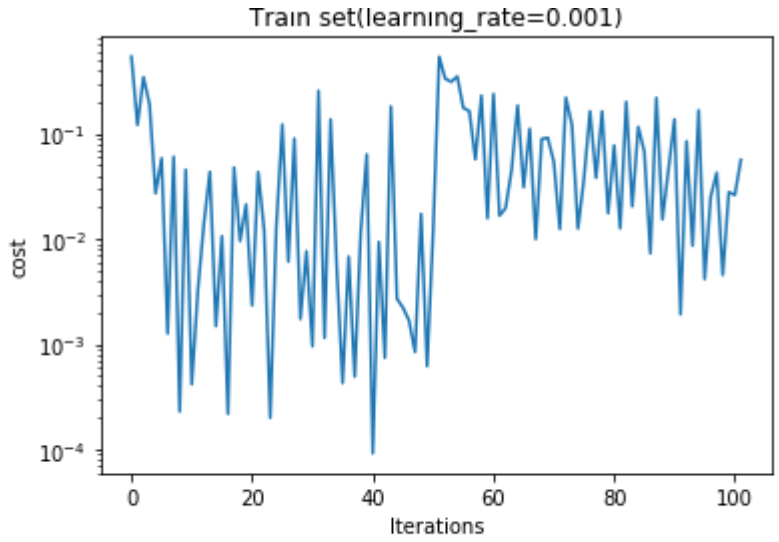
activate function: sigmoid & relu

initial weight: random_ initialize

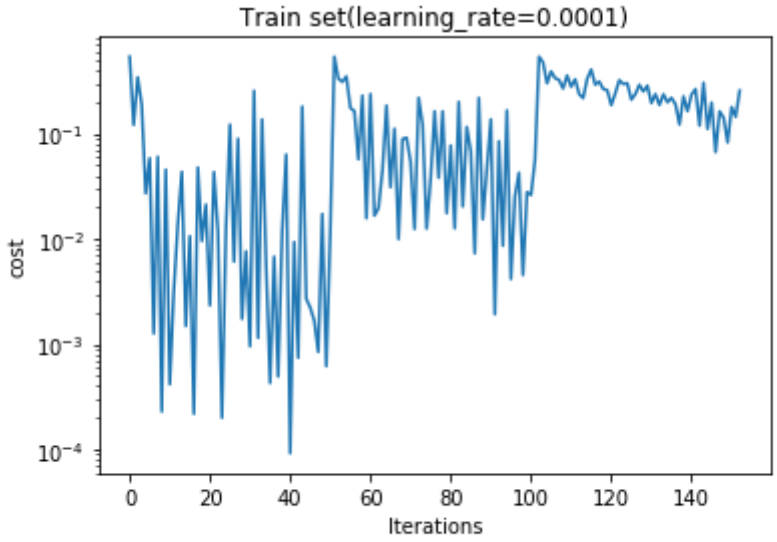
learning rate: 0.01 / 0.001 / 0.0001




```
test set的正確率(%): 94.45
```




```
test set的正確率(%): 91.16
```



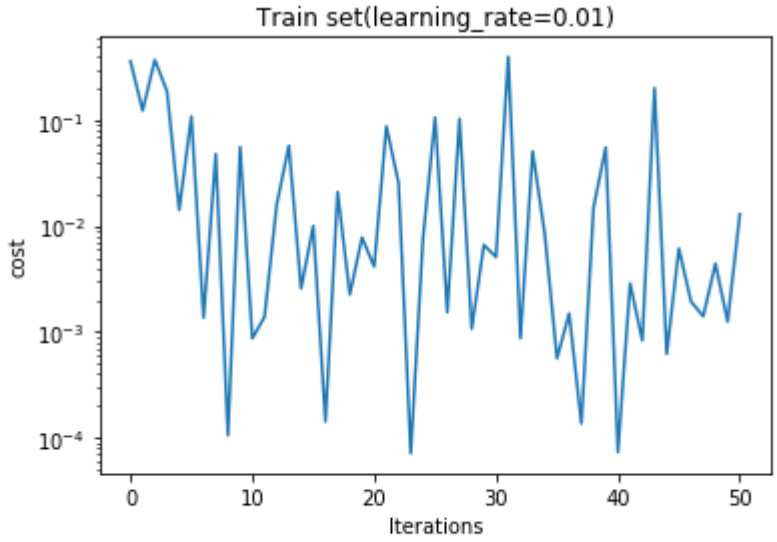

```
test set的正確率(%): 78.35
```

2.

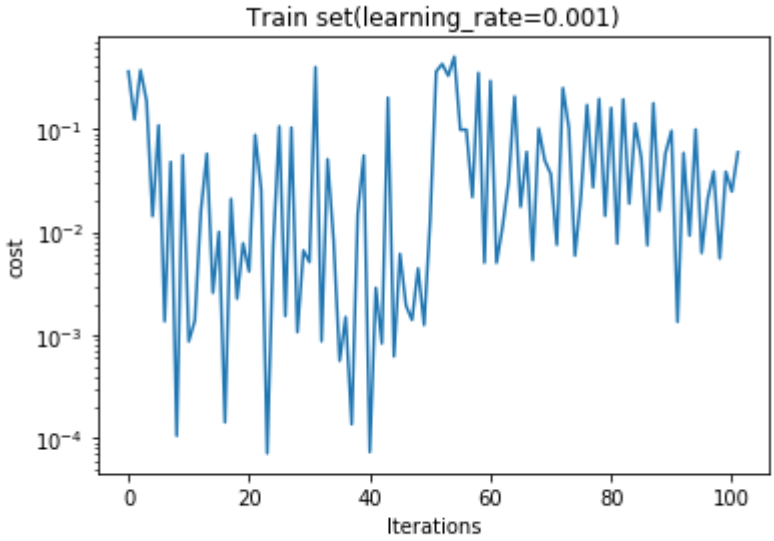
activate function = sigmoid & relu

initial weight = Xavier_ initialize

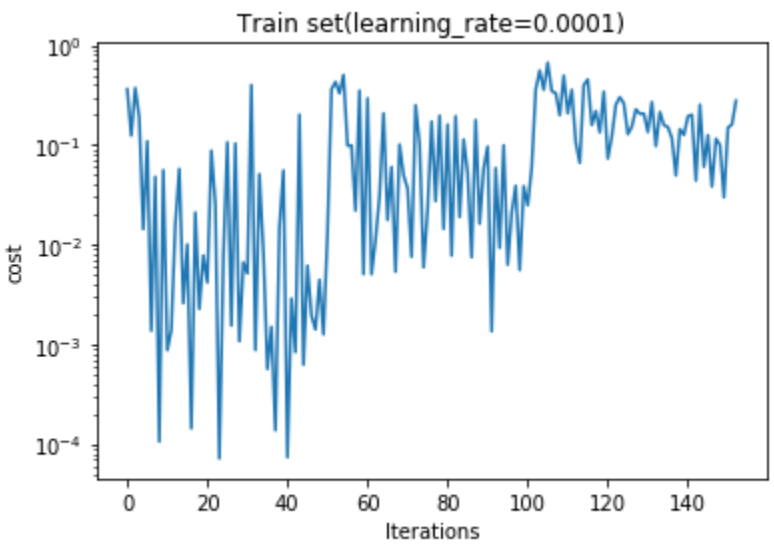
learning rate = 0.01 / 0.001 / 0.0001




```
test set的正確率(%): 94.3
```




```
test set的正確率(%): 91.31
```



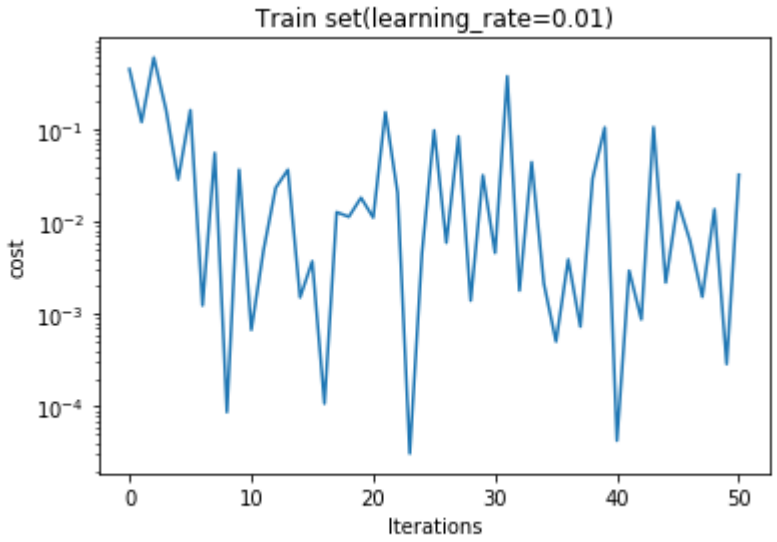

```
test set的正确率(%): 79.07
```

3.

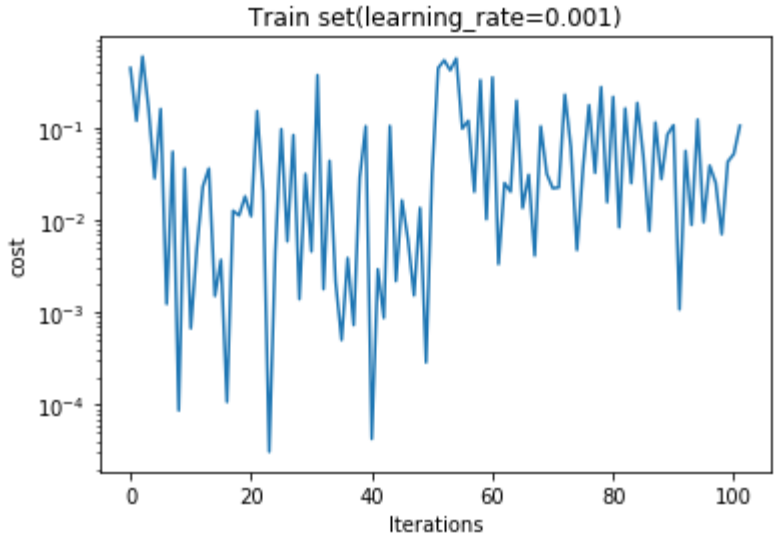
activate function = sigmoid & relu

initial weight = He_ initialize

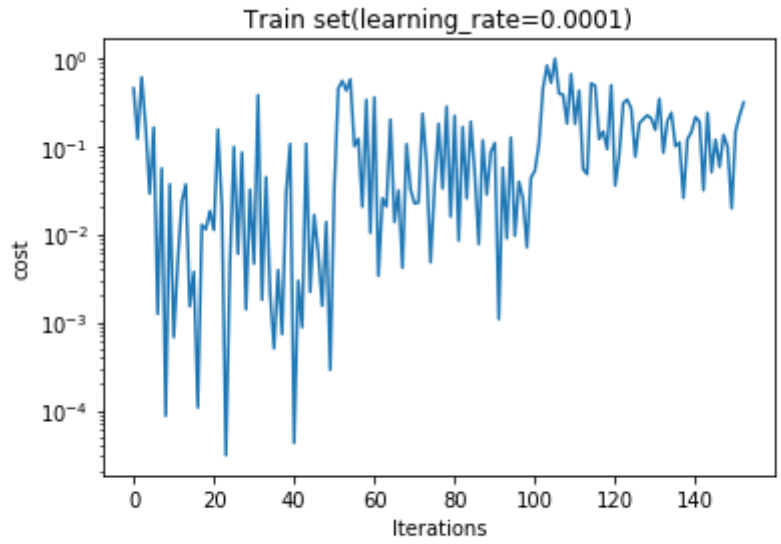
learning rate = 0.01 / 0.001 / 0.0001




```
test set的正確率(%): 93.88
```




```
test set的正確率(%): 90.14
```




```
test set的正確率(%): 74.98
```

Conclusion:

因為想要更深入了解每一層每一個部分的操作流程，所以不打算使用 keras/tensorflow 等套件，只利用 python 實現每一個部分，雖然還有一點瑕疵，沒有像套件的結果這麼完美，但是成就感蠻大的，也學到更多。