プログラミング課題 5

03-170312
航空宇宙工学科
新幡 駿

2018 年 1 月 9 日

1　フィードフォワードニューラルネットワーク

1.1　実装

損失関数の勾配の計算には誤差逆伝搬法を用いる. またデータ読み込み処理は [1] を用いた. また実装は全体的に [1] にあった 2 層のネットワークを N 層に設定できるように拡張した. また隠れ層の edge の数は層間で変えてない (ともに 50).

1.2　結果

Operating System:	Windows 10 Home 64-bit
Processor:	Intel(R) Core(TM) i5-4690K CPU @ 3.50GHz (4 CPUs), ~3.5GHz
Memory:	8192MB RAM

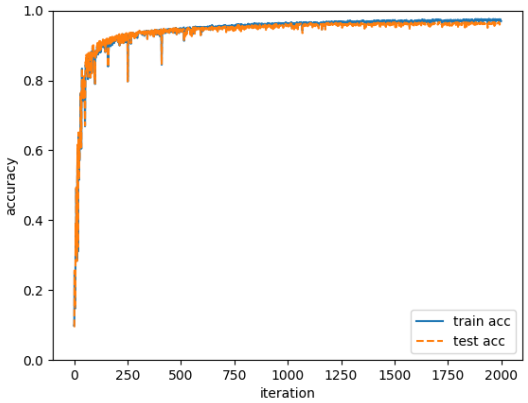
表 1: 実行環境

1.2.1　N=2 のとき

プログラムの learning rate を 0.1 とした.

正答率	0.9611
処理時間	839[sec]

表 2: N=2 の結果

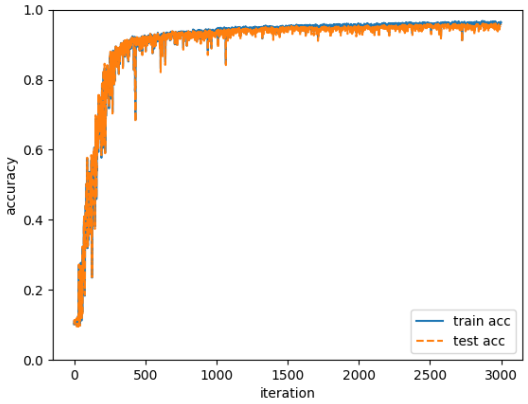


1.2.2 N=3 のとき

プログラムにおける learing rate を 1 とした.learing rate を 0.1 とすると正答率があがらなかった. 局所解に陥ったように思える.

正答率	0.9552
処理時間	1448[sec]

表 3: N=3 の結果



2 K Nearest Neighbor

[2] を参考にしてプログラムを作成した. k=6 とすると

正答率	0.96770
処理時間	2600[sec]

表 4: kNN の結果

3 考察

fNN ではなく kNN でも高い正答率が出せることがわかった. 今回のレポートでは kNN の性能のほうが fNN よりもよいが, 層の数や learning rate などのパラメーターを変化させると正答率が変わるので一概に kNN の方が優れているとは言えない. また前述の通り fNN は局所解に陥ることがある.

参考文献

[1] 斎藤康毅, deep-learning-from-scratch, https://github.com/oreilly-japan/deep-learning-from-scratch

[2] Steven Traversi,How to Get 97% on MNIST with KNN https://steven.codes/blog/ml/how-to-get-97-percent-on-MNIST-with-KNN/