**ロボットインテリジェンス レポート課題A**

170308 劉宇湖

1. 概要

MNISTの手書き数字のデータセットを用いてフィードフォワード型ニューラルネットとBP学習を行うプログラムを作った。このプログラムでは、28x28画素の濃淡画像を識別することができる。

このプログラムを用いて、各パラメータを調整するとどのような結果が出てくるかを調べた。

1. 前提条件

・データセットはMNISTの手書き数字のデータセットを用いる。（訓練データが50000セット、試験データが10000セット）

・用いた言語はpython2.7で、これをjupyter notebook上で動かした。

・試験データに0%、12.5%、25%のノイズが入った場合での、より効率のいい学習方法を探ることを今回の目的とする。

・各試行において、その都度効率の良い学習率にチューニングしてからグラフを描画している。（学習率は明示する。）

・各試行、学習回数を10回、ミニバッチサイズを10としている。これ試行錯誤の末に落ち着いた数字である。

・活性化関数、隠れ層のニューロン数、隠れ層の数を調査対象とする。

・活性化関数は隠れ層が1層の状態で

1. sigmoidを用いる場合
2. 隠れ層にsigmoid、出力層にsoftmaxを用いる場合
3. 隠れ層にReLU、出力層にsoftmaxを用いる場合

　の3つを調査する。

・1つの隠れ層のニューロン数は10～140まで、10刻みで調査する。

・隠れ層を2層使う場合は、2つの層は同じニューロン数にする。

・グラフの横軸は隠れ層のニューロン数、縦軸は試験データでの成功数で、青、橙、緑の線がそれぞれノイズ0%、12.5%、25%の場合である。

・グラフの描画にあたり、10回の学習結果のうち最も正解数が多い回の結果をプロットしている。

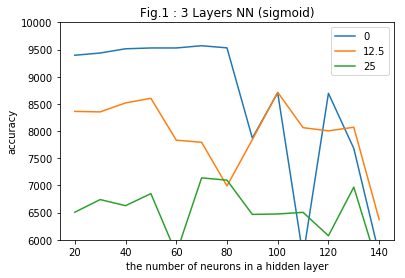
・ニューラルネットのプログラムの骨格は「ニューラルネットと深層学習」

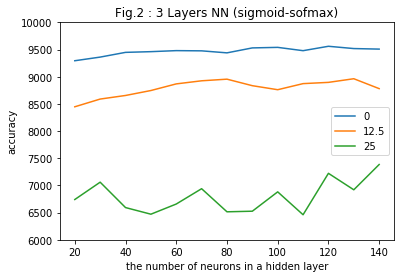
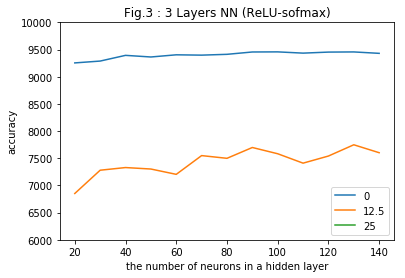
（http://nnadl-ja.github.io/nnadl\_site\_ja/index.html）

をもとに、自分なりに改編を加えて作ってみました。

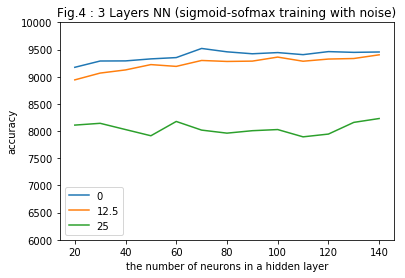
３．試行結果

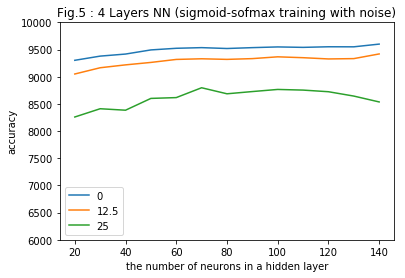
1. 活性化関数をすべてsigmoidにした場合（学習率：3.0）



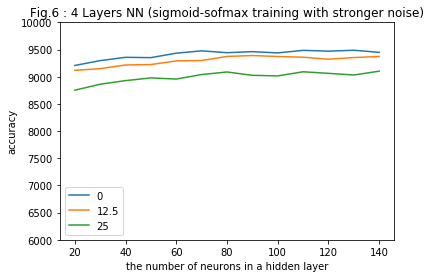
1. 隠れ層をsigmoidで出力層をsoftmaxにした場合（学習率：3.0）
2. 隠れ層をReLUで出力層をsoftmaxにした場合（学習率：0.2）

（※ノイズ25%の時は正解が6000より少なかったため、グラフには出てきていない）

1. ②で訓練データに10%のノイズを入れた場合（学習率：3.0）
2. ④で隠れ層を2層にした場合（学習率：1.0）



1. ⑤でノイズを20%にした場合



~おまけ~ 隠れ層がない場合（学習率：0.1）

・訓練データのノイズが0%で、試験データのノイズが0%、12.5%、25%の時それぞれ識別数は約9000、6000、4000であった。

・訓練データのノイズが20%で、試験データのノイズが0%、12.5%、25%の時それぞれ識別数は約8800、8700、8200であった。

４．考察

・①より、3層NNで活性化関数をsigmoidのみにした場合、ノイズが0%の試験データについて、隠れ層のニューロン数が低いうちは識別率が95%前後と高い数値を出しているが、80を超えたあたりから一気に落ち込んでいる。これは過学習による結果だと考えられる。

ノイズが乗っている試験データについては、ニューロン数ごとの識別率のばらつきが激しいが、ノイズが多いほど識別率が下がる傾向がみられた。

　　・①と②を比較すると、出力関数がsigmoidからsoftmaxになることにより、過学習による影響が今回の実験でのニューロン数では見られなくなり、また、ノイズが乗った試験データの結果が安定するようになった。これは、自然対数の特徴により出力の値がより際立って出てくるためと思われる。

　　・②と③を比較すると、隠れ層の活性化関数がsigmoidからReLUになってもノイズが乗っていない試験データの識別率はあまり変化が見られないが、ノイズが乗っている試験データの識別率が大きく落ち込んだ。Sigmoidに比べてReLUはグラフが滑らかでなく出力が極端になるため、ノイズの影響に左右されやすいためと考えられる。最近では隠れ層の活性化関数はsigmoidではなくReLUを用いる、という風習があるようなので、②より③の方がいい結果が出てくると思ったが、そうでもなかったので意外であった。もしかしたら訓練データにノイズを乗せて学習させることで改善がみられるかもしれないが、時間がなかったのでこの仮説は検証できなかった。ReLUについては今後の学習の課題として残る。

　　・②と④を比較すると、訓練データに10%のノイズを入れることにより、ある程度のノイズ耐性が得られることが見て取れる。

　　・④と⑤を比較すると、隠れ層を2層に増やすことにより全体的な精度向上とノイズ耐性の向上が見て取れる。

　　・⑤と⑥を比較すると、訓練データのノイズを10%から20%に増やすことにより更なるノイズ耐性の向上が見て取れる。25%のノイズが乗っている試験データを90%の精度で識別できるのはなかなか上出来だと思う。

　　・~おまけ~ を見ると、隠れ層がないニューラルネットでも意外にそこそこ識別性能があることが分かった。すごい。

５．参考文献

・「ニューラルネットと深層学習」（<http://nnadl-ja.github.io/nnadl_site_ja/index.html>）

　プログラムの作成の骨格、データセットのダウンロードに使用

・「誤差逆伝播法のノート」（<https://qiita.com/Ugo-Nama/items/04814a13c9ea84978a4c>）

　BP法についてここで学習した

・「活性化関数のまとめ（ステップ、シグモイド、ReLU、ソフトマックス、恒等関数）」

　（<https://qiita.com/namitop/items/d3d5091c7d0ab669195f>）

　様々な活性化関数をここから学んだ

６．感想

座学では身につかない力がこのレポートの実践で身についたように思えます。ただ、いきなり自分でプログラムを書こうとしてもあまりにもとっかかりがなさ過ぎて、必死にネットで調べるなどした末、ほとんどネットに乗っているプログラム丸写しのようなコードができてしまいました。プログラムの丸写しはだめだ、と思い、まずはネットにあった他人のコードをじっくり読むところから始めたので、そのあと自分で書く際もそれに引きずられて、ほぼ同じようなものが出来上がりました。Numpyを使ってより高速化を実現しようとしましたが、元のプログラムより高速なものを実装することはかないませんでした。

ただ、プログラムを全部理解した証として、割と詳しくコメントを入れたので、それでお許しください……

オリジナルの自分の要素としては、データセットの読み込みはインポートした際に自動で行われるようにし、活性化関数を複数用意して更にそれにオーバーフロー対策をしたり、ノイズを入れたり、隠れ層の数を操作したり、結果を見やすくグラフに描画したりなどしました。隠れ層が大きくなればなるほど計算が遅くなっていき、グラフ１枚の描画で1時間ずつほどかかりました。

色々調べていくにあたり、めちゃめちゃ便利そうなライブラリがいっぱいあったので、今後いじって遊んでみたいと思いました。

　授業に関しては、ぼく個人としては、今学期の中で一番楽しい授業でした。とても話が分かりやすく、いい授業でした。好きです。