第 1 章

方法

手書きの数字のデータセットである MNIST データセットを題材にして、実際にフィードフォーワード・ニューラルネットワークと畳み込みニューラルネットワークの構築を行なった。

簡単なフィードフォーワード・ニューラルネットワークを実装したのち、活性化関数、コスト関数、最適化アルゴリズム、隠れ層の数などの部品を組み換えてプログラムを実行させ、それらがどのように性能に影響を与えるかを研究した。 同様に簡単な畳み込みニューラルネットワークを実装し、畳み込みフィルターのサイズやプーリングカーネルの種類を変化させてその結果を比較した。さらに LeNet-5 を実際に構築して性能を確認した。最後に出力層に向かって、畳み込みフィルターと隠れ層のユニット数を増加させたときと減少させたときで性能を評価し、どちらが応用に向いているのかを判定した。

1.1 MNIST データセット

MNIST データセット（Mixed National Institute of Standards and Technology database）[**?**] は 60000 個の訓練データと 10000 個のテストデータからなる、手書きの数字のデータセットである。それぞれのデータは手書き数字の画像と、表している数字の正解のラベルからなる。各画像は 28×28 ピクセルで、各ピクセルは 0（白）から 255（黒）までの値でその明度を表している。MNIST データセットは、最初の 60000 個が訓練セット、後ろの 10000 個がテストセットにはじめから分かれている。このデータセットは非常によく使われており、新しい分類アルゴリズムが登場するとまず MNIST データセットで性能を測るので、機械学習の初心者が必ず触れる題材であると言われている。図 1.1 は、MNIST データセットの訓練セットからランダムに取り出された 50 個のデータである。各画像の右下には正解のラベルの値も付してある。

1.2 フィードフォーワード・ニューラルネットワークによる学習

1.2.1 ベースとなるフィードフォーワード・ニューラルネットワークの構築

活性化関数や隠れ層の数などを変化させたときにニューラルネットワークの性能にどのような違いが生じるかを試したかったので、できるだけシンプルな、隠れ層が 1 層のフィードフォーワード・ニューラルネットワークのプログラムを作成した。機械学習のためのオープンソースライブラリである Keras を用いてソースコードを書いた。

まず MNIST データセットの各画像データはサイズが 28×28 の 2 次元の配列であるが、これを 784 次元のベクトルに変換した。各ピクセル値は 0 から 255 までの整数であるが、これを 0 から 1 までの値

図 1.1　MNIST データセットの 50 個のデータ

に変換した。またデータのラベルに対応したワンホットベクトルを作成しておいた。次に学習をさせるニューラルネットワークを構築した。各ユニットが次の層のすべてのユニットに影響する全結合型ニューラルネットワークとし、隠れ層は 1 層でそのユニット数は 100 としておいた。入力層は 784 ユニット、出力層は 10 個の分類をするので 10 ユニットである。図にすると図 1.2 のようになる。入力層から隠れ層へ

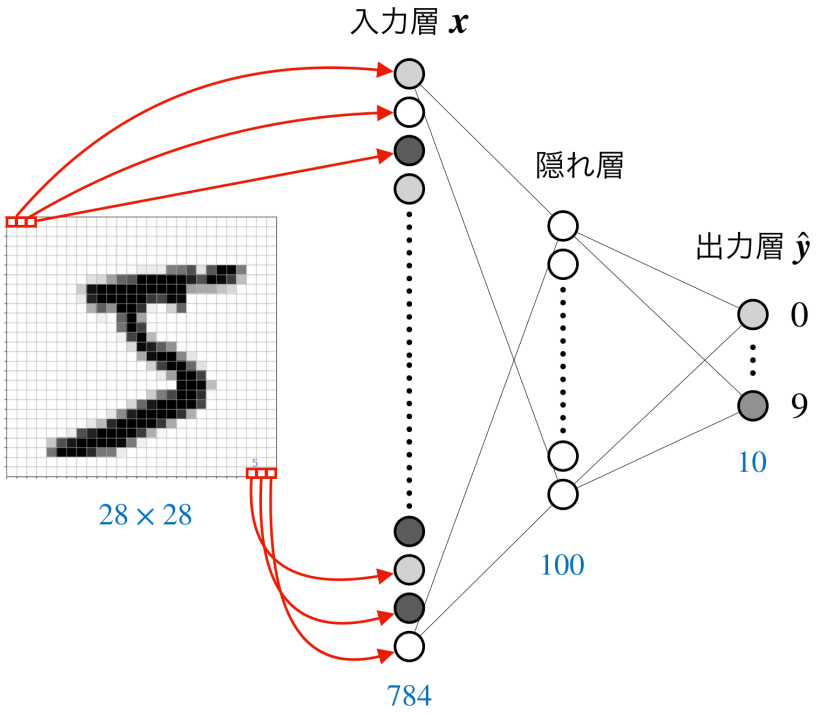

図 1.2　最初に構築したシンプルなニューラルネットワーク

の活性化関数は ReLU を、隠れ層から出力層への活性化関数はソフトマックス関数とした。コスト関数は交差エントロピーを採用し、出力とラベルのワンホットベクトルからこれを計算した。誤差逆伝搬によりコスト関数の勾配から、入力層と隠れ層、隠れ層と出力層の間の重みパラメータおよび隠れ層、出力層

の各ユニットについてのバイアスのパラメータの更新を行なった。ただし最適化アルゴリズムは確率的勾配降下法とし、ミニバッチのサイズは 1000 としておいた。この処理を $60000/1000 = 60$ 回繰り返した。ただしミニバッチは確率的に選ばれるため、この反復によって 60000 個の訓練データすべてが用いられるわけではない。次に 10000 個のテストデータを用いて予測を行い、正解率を算出した。ここまでの反復を 1 エポックという。エポック数は 100 として、合計で $100 \times 60 = 6000$ 回のパラメータ更新が行われるようにした。初期設定を箇条書きにまとめると次のようになる。これ以降ではこれらを少しずつ変化させて性能を比較していった。

- 隠れ層：層の数は 1 でユニット数は 100
- 入力層から隠れ層への活性化関数：ReLU
- 隠れ層から出力層への活性化関数：ソフトマックス関数
- コスト関数：交差エントロピー
- 最適化アルゴリズム：確率的勾配降下法（SGD）
- ミニバッチのサイズ：1000
- エポック数：100

プログラムのソースコードとそのより詳しい説明は付録**??**の**??**に示した。

1.2.2　活性化関数の比較

活性化関数を ReLU からシグモイド関数に変えてプログラムを実行し性能を比較した。

1.2.3　コスト関数の比較

コスト関数を交差エントロピーから平均二乗誤差に変えてプログラムを実行し性能を比較した。

1.2.4　最適化アルゴリズムの比較

最適化アルゴリズムを確率的勾配降下法から Adam に変えて性能を比較した。

1.2.5　エポック数、バッチサイズの変更

エポック数を 10, 1000 に、バッチサイズを 100, 10000, 100000 にそれぞれ変更したときの結果を表示させ、それらを比較した。

1.2.6　隠れ層を 2 層に増やした場合の性能評価

隠れ層を 1 層追加して、 2 層にして性能を比較した。ただしユニット数をいくつにすれば良いのかわからなかったため、第 1 層目の隠れ層のユニット数は 100 のまま、追加した第 2 層目の隠れ層のユニット数を 10, 25, 50, 100, 200 のように変化させたときのものを出力した。逆に第 2 の隠れ層のユニット数を 100 にしたまま、第 1 の隠れ層のユニット数を同じように変化させて比較した。なお追加した第 2 層目の隠れ層の活性化関数は第 1 層目と同じく ReLU とした。

1.2.7　隠れ層を 3 層以上に増やした場合の性能評価

節 1.2.6 を実行することによって、出力層に向かうにつれてユニット数を少なくしてゆく「漏斗型」のフィードフォーワード・ニューラルネットワークが、精度良く数字を予測することを経験的に発見した。そこでネットワークをより深くするとき、どのように細くしてゆくのがいいかを検証した。具体的には a をある自然数、$n_{\text{classes}} = 10$ をクラスの数、隠れ層の数を N としたとき（入力層と出力層も合わせれば全体で $(N+2)$ 層のネットワークになる）、隠れ層のユニット数を順方向に $10 \times a^N,\ 10 \times a^{N-1},\ \ldots,\ 10 \times a^2,\ 10 \times a$ のように、指数関数的に減らしていくことを考えた（図 1.3）。ただし入力の次元が 784 であるから第 1 隠れ層のユニット数がそれよりも小さくなるように、$n_{\text{classes}} = 10$ と a の値を決めたのち $10 \times a^N < 784$ を満たす最大の隠れ層数 N について検証した。そのような数の組み合わせは表 1.1 のように 7 通りあるので、それらを全て試した。ただし隠れ層の活性化関数はすべて ReLU とした。またソースコードの上で隠れ層を増やすのには反復文を用いて手間を省く工夫をした。具体的には付録**??**のソースコード**??**のように変更した。

比較のため、隠れユニット数を順方向に $10 \times a^1,\ 10 \times a^2,\ \ldots,\ 10 \times a^N$ のように増加させていった「逆漏斗型」のニューラルネットワークの性能を $a = 2,\ N = 6$ の場合に測定した。それとは別に、ユニット数が 210 で一定で層数 6 の「寸胴型」ニューラルネットワーク（総隠れユニット数は $210 \times 6 = \sum_{k=1}^{6} 2^k = 1260$ で同じになる）についても性能を測定した。

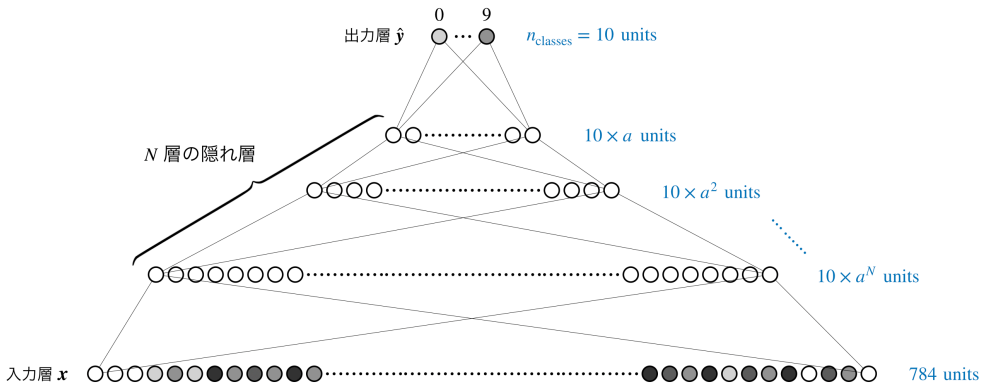

図 1.3　隠れユニット数が指数的に減少するニューラルネットワーク

表 1.1　a と N の組み合わせ

a	2	3	4	5	6	7	8
N	6	3	3	2	2	2	2

1.2.8　ドロップアウトの効果

ドロップアウトは畳み込みニューラルネットワークで用いられる手法であるが、その中でも出力層に近くにある、部分的にフィードフォーワード・ニューラルネットワークとみなせる部分で用いられるため、節 1.2.1 のニューラルネットワークを用いて検証を行った。

節 1.2.1 のベースとなるニューラルネットワークの隠れ層と出力層の間にドロップアウト層を挿入し、ドロップアウトの割合を $p = 0$（ドロップアウトをしない）, 0.25, 0.5, 0.75 のように変化させたときの

出力をみた。ただし収束を早めるため最適化アルゴリズムは Adam を採用し、エポック数、ミニバッチのサイズはドロップアウト率に合わせて変化させた。

1.3　畳み込みニューラルネットワークによる学習

1.3.1　ベースとなる畳み込みニューラルネットワークの構築

隠れ層を畳み込み層、プーリング層などに変化させたときに畳み込みニューラルネットワークの性能にどのような違いが生じるかを試したかったので、できるだけシンプルな、隠れ層が 1 層の畳み込みニューラルネットワークのプログラムを作成した。エポック数は 20, ミニバッチのサイズは 1000 とした。

まず MNIST データセットの各画像データをそのまま扱うため、これを $60000 \times 28 \times 28$ の配列に変換した。各ピクセル値は 0 から 255 までの整数であるが、これを 0 から 1 までの値に変換した。またデータのラベルに対応したワンホットベクトルを作成しておいた。次に学習をさせるニューラルネットワークを構築した。各ユニットが次の層のすべてのユニットに影響する全結合型ニューラルネットワークとし、隠れ層は 1 枚のフィルターを持つ 1 層の畳み込み層とし、そのフィルターのサイズは 3×3 としておいた。畳み込み層の出力サイズが入力と変わらないように、適宜ゼロパディングを行っておいた。入力層から隠れ層への活性化関数は ReLU を用いた。この隠れ層の出力のサイズは 1000（ミニバッチのサイズ）$\times 1$（フィルターの数）$\times 28 \times 28$（画像のサイズ）の 4 次元であるが、次の出力層に入れるため、これを $1000 \times (1 \cdot 28 \cdot 28)$ の 2 次元に変換する必要があった（Flatten 層）。入力のデータが 1 枚のときの、この畳み込みニューラルネットワークの概念図を図 1.4 に示す。隠れ層から出力層への活性化関数はソフ

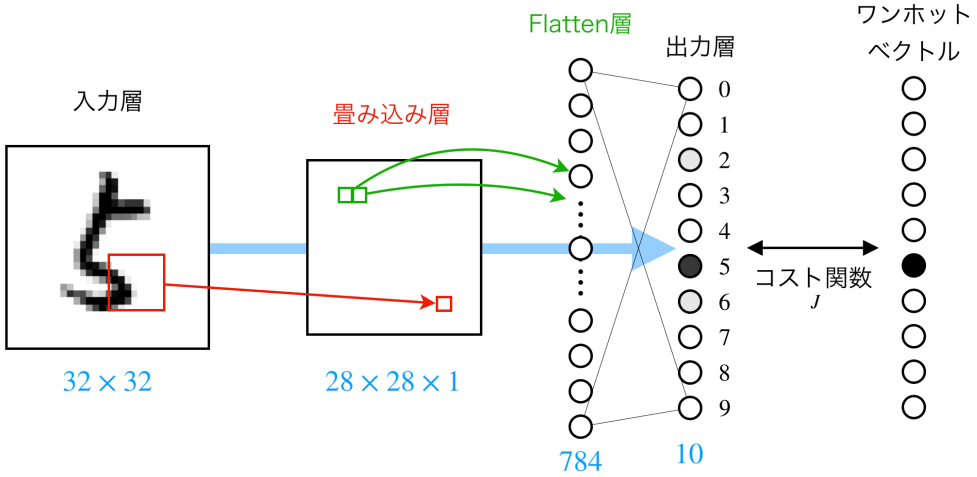

図 1.4　畳み込み層が一つの畳み込みニューラルネットワーク

トマックス関数とした。最適化アルゴリズムは確率的勾配降下法とし、エポック数は 20 として、合計で $20 \times 60 = 1200$ 回のパラメータ更新が行われるようにした。ここでパラメータはフィルターの $3 \times 3 = 9$ 個の値と、全体に加える 1 つのバイアスである。初期設定を箇条書きにまとめると次のようになる。これ以降ではこれらを少しずつ変化させて性能を比較していった。

- 隠れ層：畳み込み層
  - フィルターのサイズ：3×3
  - フィルターの数：1

- 入力層から隠れ層への活性化関数：ReLU
- 隠れ層から出力層への活性化関数：ソフトマックス関数
- コスト関数：交差エントロピー
- 最適化アルゴリズム：確率的勾配降下法（SGD）
- ミニバッチのサイズ：1000
- エポック数：20

プログラムのソースコードとそのより詳しい説明は付録**??**の**??**に示した。

1.3.2　フィルターのサイズと枚数の比較

3×3 のフィルター数を 1 枚ずつ 6 枚まで増やし、学習の結果を出力させた。またフィルター数は 1 枚に固定して、そのサイズを 4×4, 5×5, 6×6 と変化させてその結果を表示させた。

1.3.3　プーリング層の追加

次に畳み込み層の代わりに最大プーリング層を通して実行結果をみた。プーリングカーネルのサイズは 2×2 とした。また平均プーリングを行ってどのような差異が現れるか確認した。最大プーリングカーネルのサイズを 3×3, 4×4 のように変化させたとき実行結果がどのように変化するかを試した。

1.3.4　LeNet-5 の実装

節**??**で説明したアーキテクチャ LeNet-5 を実装する。層の名前は図**??**に準ずるものとする。入力層にゼロパディングを施して 32×32 の画像にし、5×5 のフィルター 6 枚を持つ畳み込み層 C1 を構築した。それに 2×2 のプーリングカーネルで最大プーリングを施した（S2)。それぞれの出力画像に対して 16 枚の 5×5 のフィルターで畳み込みを行い（C3)、さらにその各出力画像について 2×2 のプーリングカーネルで最大プーリングを施した（S4)。この時点で出力のサイズは $5 \times 5 \times 16$ であるがこれを 120 次元のベクトルに展開した（C5)。次に 84 のユニットを持つ隠れ層 F6 を通してから出力層とした。最後の 3 つの層は全結合型のフィードフォーワード・ニューラルネットワークとみなせる。

これを実装したあと、畳み込み層 C5 と F6 の間、F6 と出力層の間、そのどちらにも $p = 0.25$ のドロップアウトを挿入した場合について性能を確認した。

1.3.5　畳み込みニューラルネットワークのアーキテクチャ

畳み込みニューラルネットワークにおいて層の数増やしていくのがいいのか、減らしていくのがいいのかを実験して検証した。節 1.3.4 の LeNet-5 のネットワークを用いて、畳み込みのフィルターを入力層から順に 16, 32, 64 とし、出力層の手前の層のユニット数を 128 にしたネットワーク A と、それを逆にしたネットワーク B について実験を行なった（図 1.5)。エポック数は 15, ミニバッチのサイズは 1000 とした。

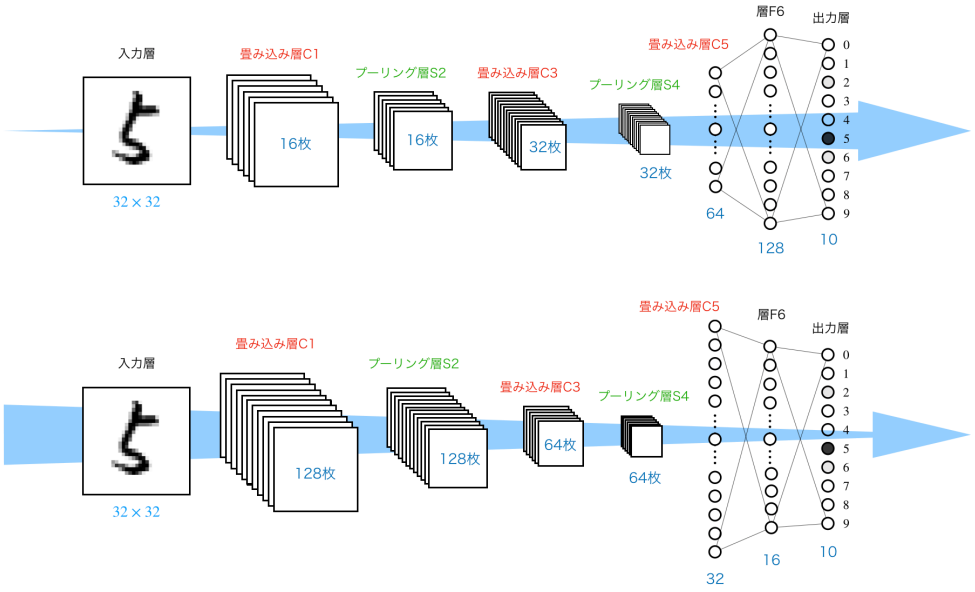

図 1.5　上図がネットワーク A、下図がネットワーク B