株価ローソク足チャート画像を用いた畳み込みニューラルネットワークによる株価変動予測

池　田　欽　一

概 要

本研究では、近年飛躍的な進化を遂げたニューラルネットワーク技術（ディープラーニング）の株式投資へ応用可能性というテーマで分析を行っている。本論文では、ディープラーニングの中でも画像や動画解析で様々な成果を挙げている畳み込みニューラルネットワークを用い、株価のローソク足画像による「上がる」、「下がる」という株価変動予測の方法の提案、およびシミュレーション実験による結果をまとめている。

1　はじめに

本研究では、ディープラーニングを用いた株価の変動予測について分析を実施する。ディープラーニングは、神経細胞を数理モデル化したユニットによるネットワーク、いわゆるニューラルネットワークの表現や学習能力を高めたものである。本論文ではディープラーニングの中でも人間の視覚をモデルとして、画像、動画認識、囲碁や将棋の推論手法などへの応用が進んでおり、人工知能技術の大幅な進展に貢献している畳み込みニューラルネットワーク（CNN, Convolutional Neural Network）を用いることとする。CNN は畳み込みニューラルネットワークとも呼ばれ、画像データが入力される入力層に近い部分にフィルタを用いた畳み込み計算を実施することからこのように呼ばれている。以前筆者らが本論文と同様に株価予測に用いた、ユニットを層状に並べただけのフィードフォワードニューラルネットに比べると [1]、入力が 2 次元画像（複数枚やカラー画像も可能）であるので、2 次元的な関係性を学習に取り入れることが可能となる点が CNN を用いる利点となる。株価は時系列として取得することが容易であるため、フィードフォワードによる分析の方がシンプルなシステムとなるが、CNN を用いることにより、例えば、ある期間の価格の分布と時間経過など、より多くの情報をシステムへ与えることが可能となる。

ディープラーニングでは大量のデータ、いわゆるビッグデータによる学習が特徴である。経済データについては、月や四半期、さらには年単位のデータのみしか入手ができないものも多く、ディープラーニングの適用は困難が予想されるが、株価データは 1 日単位（日足）はもちろんのこと、1 分単位、あるいはティックといわれる取引ごとのデータ（銘柄によっては 1 日に数万件の取引）も入手が可能であり、ディープラーニングによる予測と相性が良いと考えられる。大量のデータによる学習により、システムは学習したデータに近い場合だけでなく、経験則に基づかない予測、学習したデータに近い変動であってもわずかな差により結果が違う状況で、適切な予測ができる可能性がある。この能力により、様々な市場参加者の意思決定の結果により複雑なシステムにより価格が決まる株式の予測へディープラーニングが有効であると考えられる。株価は正規分布やベキ分布に従うランダムな現象としてモデル化されることが多いが、ディープラーニングは非線形性計算を伴った、多数のフィルタ畳み込みやユニットを組み合わせたものであるので、複雑な非線形システムの学習、予測に強く、

株価などランダム性の強い現象と考えられている対象を、決定論的な非線形現象として分析が可能であると考えられる。

以下、2. ではディープラーニング手法、特に本研究で用いる CNN について述べ、3. ではディープラーニングを株価予測へ応用するための実装方法や用いる株価データ、パラメータ設定について述べる。4. ではシミュレーション結果とその考察を示し、5. でまとめとする。

2 ディープラーニングとは

ディープラーニング（深層学習）とは、神経細胞（ニューロン）を数理モデル化してネットワーク状に接続したニューラルネットワークを複雑化したり、層を多数化したものと言える。ディープラーニングには様々なモデルが研究されているが、本論文では、視覚野の構造をモデル化し、主に画像や動画解析に応用が進んでいる CNN を用いる [3]。CNN は、局所的な特徴に反応する単純細胞とそれら特徴量の位置の部分的なずれに反応する複雑細胞によりモデル化されているネオコグニトロン [3] を起源としている。単純細胞は、画像データとフィルタの畳み込み層（Convolution）を用いてフィルタに類似した特徴を抽出ことで実現され、複雑細胞は周辺の細胞（ユニット）の平均値や最大値を求める Pooling 層によりモデル化されている。畳み込み層では、学習により最適化されていくフィルタに類似する入力データの局所的パターンに反応する。つまり右上がりの太い直線など、学習したフィルタと同様な特徴を入力データから見つけ出す能力を有している。Pooling 層は畳み込み層により得られた特徴の分布を一定の範囲で集約する働きがあり、特徴量の出現位置にあいまい性を持たせる働きをしている。畳み込み層と Pooling 層を交互に多数接続することにより文字認識などへ応用がなされていた。

複数の畳み込み層と Pooling 層により抽出された特徴は、フィードフォワード型のネットワーク層に送られ、非線形変換の後に、出力層にて所属するカテゴリや数値などの出力がなされることとなる。フィードフォワードネットワーク [4, 5, 6] は、データの入力側の 1 つ隣の層の各ユニット（脳細胞を数理モデル化したニューロン）からの出力を集計する。各ニューロンは 1 つ前の層のニューロンの出力値に接続強度であるウェイトをかけて集計したものにバイアス値を足し、活性化関数といわれる関数により変換して出力側に順次送られていく。

ネオコグニトロンでは競合学習による自己組織的学習が行われていたが、LeCun らによる LeNet でフィードフォワードニューラルネットワークと同様な BackPropagation による誤差伝搬による誤差の逆伝搬学習方法が確立され、畳み込み層と Pooling 層を多数積み重ねたネットワークの学習も可能となった [2]。元々は文字認識分野において高い認識能力が示されていたが、画像のカテゴリ判別、人体や物体の検出など様々な分野において応用研究が進んでいる。

2.1 畳み込み層

本研究では、入力データには、1 分単位の株価（1 分足株価）の始値（1 分間の最初の株価）、終値（1 分間の最後の株価）、高値（1 分間で最も高い株価）、安値（1 分間で最も安い株価）を数分間分用いたローソク足の画像を用いる。ローソク足の詳細は後述する。ローソク図の画像の画素サイズを $W \times W$ とし、$x(i,j), i = 0, 1, \cdots, W-1, j = 0, 1, \cdots, W-1$ で表すこととする。この画像の部分的な特徴を抽出するために、入力画像よりも画素数の小さい画像をフィルタとして用いるが、このフィルタのサイズを $H \times H$ とし、$h(p,q), p = 0, 1, \cdots, H-1, q = 0, 1, \cdots, H-1$ で表す。このフィルタ

$h(p,q)$ を用いた画像 $x(i,j)$ の畳み込み $u(i,j)$ は以下の式のように表される [2, 3]。

$$u(i,j) = \sum_{p=0}^{H-1} \sum_{q=0}^{H-1} h(p,q)x(i+p, j+q). \tag{1}$$

x(0,0)	x(0,1)	x(0,2)	x(0,3)
x(1,0)	x(1,1)	x(1,2)	x(1,3)
x(2,0)	x(2,1)	x(2,2)	x(2,3)
x(3,0)	x(3,1)	x(3,2)	x(3,3)

図 1: 入力画像例 $x(i,j)$

h(0,0)	h(0,1)
h(1,0)	h(1,1)

図 2: フィルタ例 $h(k,l)$

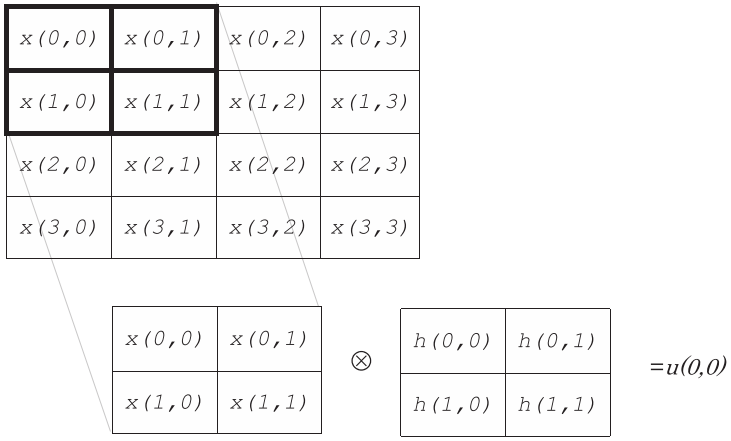

図 3: 畳み込みの模式図

この畳み込み演算を模式図により考えてみる。便宜上、ここでは画像サイズを $W = 4$(図 1)、フィルタサイズを $H = 2$(図 2) としておく。畳み込み演算は、図 3 のようにフィルタ $h(k,l)$ と同じサイズに切り取った画像 $x(i,j)$ の類似度を数値により求める演算であると言える。なお、本研究では、元々の画像は 256 階調のグレースケールで作成するが、$x(i,j)$ は $-1 \sim 1$ の間の値へ標準化した後入力する。フィルタと切り取った画像の積和演算であるので、$0 \sim 255$ の数値で表されるグレースケールであれば、フィルタに近い画像である場合には、畳み込み $u(i,j)$ は大きな数値となり、類似度が低い場合には小さな値となる。$-1 \sim 1$ に標準化した入力の場合には、類似度が高い場合には大きな数値となり、類似度が低い場合には 0 に近い値、濃淡を反転した画像とフィルタの類似度が高い場合には絶対値の大きな負の数となる。

この畳み込みを複数のフィルタを用いて入力画像全体に適用し、各フィルタのバイアスを加える演算をし、演算結果に出力関数（活性化関数）を適用した値を求めるネットワークが CNN の畳み込み

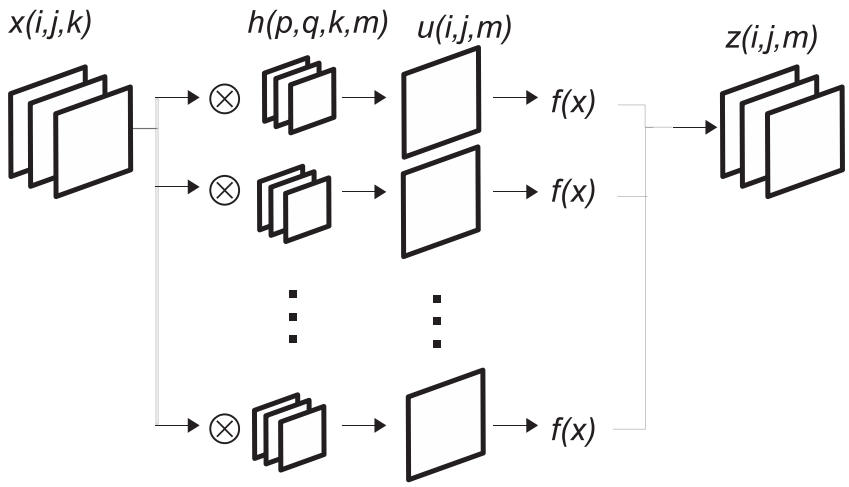

図 4: 畳み込み層

層となる。畳み込みを入力全体に適用する時、$u(0,0), u(0,1), u(0,2)$ は図 3 の画像切り取りの太枠を s ずつ右、あるいは下にずらして計算していくので、同じフィルタに類似の画像パターンが存在するかどうかを画像全体について計算していくこととなる。s はストライドと呼ばれ、1 回の畳み込み計算ごとにどれだけ切り取る画像の位置をずらすかを表す数値となる。本研究では $s = 1$ としておく。$u(0,3)$ を計算するためには、切り取る画像情報が不足するのでそのままでは計算できないが、不足する画素に 0 や周辺画素の数値の平均を補って計算するパディングという手法もあるが、本研究では、パディングは用いないこととする。この場合、元の画像サイズ $x(i,j)$ に比べ、畳み込み $u(i,j)$ はフィルタサイズ奇数の場合 $2\lfloor H/2 \rfloor$、偶数の場合 $2\lfloor H/2 \rfloor$-1 だけ小さいサイズとなる。$\lfloor\ \rfloor$ は床関数である。

フィルタは最初はランダムな数値で作成されるが、ネットワークの出力と望ましい出力との誤差関数の勾配を用いて、望ましい出力に近くなるように修正（学習）されていくことにより、出力計算に有効な部分的な画像を抽出することが可能となってくる [2]。カラー画像の場合には、R,G,B の 3 原色に分解し、3 枚の画像を 3 チャネルとして入力し、R,G,B 画像それぞれにフィルタを適用し、結果の畳み込みの同じ位置の値の総和を取ることにより処理することができる。画素サイズを $W \times W$ でチャネル数が K である入力画像を $x(i,j,k), i = 0,1,\cdots,W-1, j = 0,1,\cdots,W-1, k = 1,2,\cdots,K$ とし、入力画像の K のチャネルごとに、それぞれ M 枚のサイズ $H \times H$ のフィルタを $h(p,q,k,m), p = 0,1,\cdots,H-1, q = 0,1,\cdots,H-1, k = 1,2,\cdots,K, m = 1,2,\cdots,M$、各フィルタごとのバイアスを $b(i,j,k,m)$ で表すとき、畳み込み層の内部状態は次の式により表される。

$$u(i,j,m) = \sum_{k=1}^{K} \sum_{p=0}^{H-1} \sum_{q=0}^{H-1} h(p,q,k,m) x(i+k, j+l, k) + b(i,j,k,m). \tag{2}$$

ただし、バイアスはフィルタの適用位置に依存せず $b(i,j,k,m) = b(k,m)$ とされるケースが多い。この内部状態 $u(i,j,m)$ に以下のように活性化関数 $f(\,)$ を適用することにより、畳み込み層の出力 $z(i,j,m)$ が求められる。

$$z(i,j,m) = f(u(i,j,m)). \tag{3}$$

活性化関数には、以下のようなシグモイド関数、双曲線正接関数、 Relu（Rectified Linear Unit, Rectifier）関数（正規化線形関数）がよく用いられる。

シグモイド関数：

$$f(x) = \frac{1}{1 + \mathrm{e}^{-x}}. \tag{4}$$

双曲線正接関数:

$$f(x) = (\tanh(x)). \tag{5}$$

Relu 関数：

$$f(x) = \max(x, 0). \tag{6}$$

畳み込み層の出力はフィルタ枚数 M と同数の多チャネルデータとなり、それぞれ出力値は元の入力画像についてフィルタと類似の特徴の分布状況を表す画像と考えることもできる。各画像のサイズは、前述の通り、パディングを実施しない場合には元の画像サイズより小さなサイズとなる。

2.2 Pooling 層

Pooling 層は、通常畳み込み層（複数の畳み込み層を連続して用いることもある）の次に設置され、畳み込み層のフィルタにより表される特徴の存在する位置の、あいまいな検出を可能とすることができるものである。これにより特徴の位置の変化にある程度柔軟な対応が可能となり、特に画像分析に有効である。画像への Pooling の適用は、画像の正方領域 $N \times N$ を考え、その中の画素値の最大値や平均値を求める演算である。Pooling 演算の領域は、通常重複のないように適用されるので、出力サイズは入力画像の $1/N \times 1/N$ となる。$N = 2$ の場合、縦横のサイズは 1/2 画素数は 1/4 となり、この層以降の畳み込み層や後述するフィードフォワード層の計算コストの低減も可能となる。入力が多チャネルの場合にはチャネルごとに適用されるので、チャネル数を K とすると、(W, W, K) のサイズの入力からは、$(W/N, W/N, K)$ の出力が得られる。

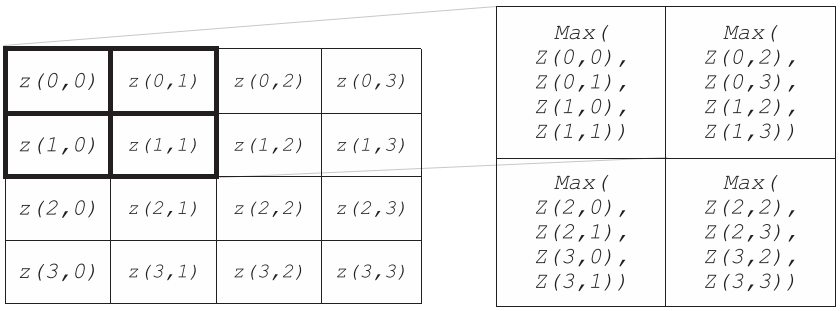

図 5: MAX Pooling の模式図

図 5 には 2×2 領域の最大値（MAX）Pooling の例を示している。4×4 の入力が、2×2 の出力へ変換されている。この演算により畳み込み層で求められた $z(0,0), z(0,1), z(1,0), z(1,1)$ のいずれかに適用フィルタとの類似度の高い値が存在すると、Pooling 演算後の出力も大きな値となる。これにより特徴の存在する場所をあいまいとなり、ある程度の広がりを持った画像領域に畳み込み層のフィルタに類似の特徴があるかどうか検索できることになる。

Pooling 層の後に出力を標準化する正規化層を用いることもある [4]。

2.3 全結合層

CNN の畳み込み層、Pooling 層を経て（複数回適用の場合もある）、入力データは全結合層へとつながり、最終的に出力層において、入力データの所属カテゴリや対応する数値が出力される。全結合層とは、以下のようにニューロンを層状に並べたネットワークである。これは従来のフィードフォワードニューラルネットワークと同様なものである。

2.3.1 ニューロン

ニューロンとは神経細胞を次のようにモデル化したものである。

$$u_j = \sum_{i=1}^{I} w_{ji} x_i + b_j, \tag{7}$$

$$z_j = f(u_j). \tag{8}$$

ここで、u_j は第 j ニューロンの内部状態を表し、$x_i (i = 1, 2, \ldots, I)$ であらわされるニューロンへの入力（1 つ前の層のニューロンの出力値）、w_{ji} は 1 つ前の層の第 i ニューロンから第 j ニューロンへの接続の強さを表すウェイト、b_j で表されるバイアスから計算される。z_j はニューロンの出力値で、内部状態に出力関数（活性化関数）$f(\,)$ を適用した値で求められる。ただし、全結合層の最初の層への入力データは、通常 Pooling 層からの出力画像の画素値をすべてベクトルとして並べたものである（複数チャネルの場合には、チャネルごとにベクトル化したものを並べたベクトルとなる）。例えば、Pooling 層の出力サイズが $T \times T$、K チャネルである場合、全結合層への入力データは T^2K 次元のベクトルとなり、最初の層の必要ユニット数も同数となる。活性化関数 $f(\,)$ には、CNN の畳み込み層と同様に、シグモイド関数や双曲線正接関数、Relu 関数がよく用いられる。

2.3.2 フィードフォワードニューラルネットワーク

上記のようなニューロンをまとめた層をネットワークの入力側から出力へ、入力層、任意の数の中間層、出力層と並べ、各層は 1 つ入力側の層の全ニューロンからのみ入力を受け取る。このようなネットワークをフィードフォワードニューラルネットワークと呼ぶ。第 $l, l = 1, 2, \cdots, L$ 層の内部状態 $\boldsymbol{u}^l = (u_1^l, u_2^l, \ldots, u_j^l, \ldots, u_{J^l}^l)'$ は次のように表される。

$$\boldsymbol{u}^l = W^l \boldsymbol{z}^{l-1} + \boldsymbol{b}^l. \tag{9}$$

ここで、$\boldsymbol{z}^{\boldsymbol{l-1}} = (z_1^{l-1}, z_2^{l-1}, \ldots, z_i^{l-1}, \ldots, z_{I^l}^{l-1})'$ は 1 つ入力層側の第 $l-1$ 層のニューロンの出力からなる第 l 層への入力ベクトルで、$\boldsymbol{b}^{\boldsymbol{l}}$ は第 l 層の各ニューロンのバイアス値を列ベクトルで表したものである。W は次のような第 $l-1$ 層と l 層のニューロン間のウェイト行列である。

$$W^l = \begin{pmatrix} w_{11}^l & w_{12}^l & \cdots & w_{1I^l}^l \\ w_{21}^l & w_{22}^l & \cdots & w_{2I^l}^l \\ \vdots & \vdots & \ddots & \vdots \\ w_{J^l1}^l & w_{J^l2}^l & \cdots & w_{J^lI^l}^l \end{pmatrix}. \tag{10}$$

第 l 層の出力 $\boldsymbol{z}^{\boldsymbol{l}} = (z_1^l, z_2^l, \ldots, z_j^l, \ldots, z_{J^l}^l)'$ は次のように表される。

$$\boldsymbol{z}^l = \boldsymbol{f}(\boldsymbol{u}^l). \tag{11}$$

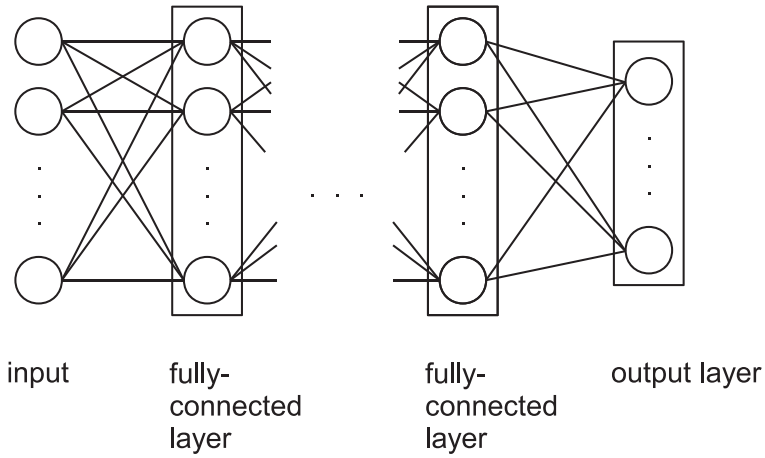

図 6: 全結合層

ここで、$\boldsymbol{f}(\boldsymbol{u})$ は活性化関数ベクトルで次のようなものである。

$$\boldsymbol{f}(\boldsymbol{u}^l) = (f(u_1^l), f(u_2^l), \ldots, f(u_{J^l}^l))'. \tag{12}$$

ただし、予測対象がカテゴリである場合、出力層のユニット数はカテゴリと同数とし、活性化関数ではなく、以下の softmax 関数により、各ユニットの出力値が 0 ～ 1 の値、総計が 1 となるように変形される。これは各カテゴリに所属する確率を表すと考えることができる。

$$z_j^L = \frac{\exp(u_j^L)}{\sum_{k=1}^{J^L} \exp(u_k^L)}. \tag{13}$$

全結合層の最も入力側にデータが与えられ、出力層へ入力データが順次渡され、出力層で CNN からのシステム全体の出力が得られる。教師データが連続値である場合は出力層のユニットは 1 つとして、学習後のこの 1 ユニットの出力を予測値として扱うことが一般的である。カテゴリデータのクラス分類の場合、出力層のユニット数をグループ（クラス）の数と同一とし、各ユニットを各グループに割り当て、学習後に入力データを与え、最も出力の大きいユニットに割り当てられたグループに所属すると予測することが一般的である。

2.4 畳み込みニューラルネットワーク（CNN）

入力画像は畳み込み層（複数の場合もあり）、プーリング、（正規化層でデータの標準化をする場合もある）を通常は数度経て、その後画素値の 1 つを 1 つの入力値として全結合層への入力値とし、全結合層を数層経過したのち出力層にて出力を得る。カテゴリ分類の場合、最終の出力層にはカテゴリの数と同数のニューロンユニットが配置され、出力の最も大きいユニットに対応するカテゴリへ分類されると予測をする。出力層のユニットの出力は式 (13) の softmax 関数により求められ、これらの出力値と実際のカテゴリ（教師データ）との間のクロスエントロピー誤差を計算し、この誤差を減ずるように誤差を入力側へ伝搬しながら、全結合層のウェイト、バイアス、畳み込み層のフィルタ値、

バイアスを更新して学習を進める [2]。クロスエントロピーは、学習サンプル $n, n = 1, 2, \ldots, N$ の実際の所属カテゴリが C_n である場合、以下の式により求められる [3]。

$$E(\theta) = -\sum_{n=1}^{N} \log z_{C_n}. \tag{14}$$

ここで、z_{C_n} は学習サンプルの実際の所属カテゴリに対応する出力層のユニットの softmax 関数適用後の出力値であり、θ はネットワークのパラメータ、つまり、全結合層のウェイト、バイアス、畳み込み層のフィルタ値、バイアスである。カテゴリ判別ではなく、数値を学習、予測する場合には、誤差関数として、望ましい出力（教師）とネットワークの出力の二乗誤差が用いられる。

本論文では、畳み込み層のフィルタ、バイアス、および、全結合層のウェイト、バイアスの学習には、Adam(Adaptive moment estimation) を用いる。Adam とは、少量のメモリで一次勾配のみにより適用可能な効率的な確率的最適化手法で、パラメータごとに勾配の平均、分散の推定値を利用する手法で、多くのケースにおいて良好な学習が得られることが示されている [7]。各パラメータの勾配の平均、分散を推定したものを利用し、勾配が疎になる場合や非定常に強い性質を持っている [8]。

3 CNN の株価変動予測、学習への応用

株価予測への CNN 応用では、1 分毎の株価四本値（始値、安値、高値、終値）よりローソク足チャートを作成し、これを入力画像として、出力層では数分後に株価が「上がる」か、「下がる」（変わらないを含む）かを予測することとする。始値とは、観測期間（本研究では 1 分間）の中で、最初に確定した価格で、終値は最後に確定した価格、安値、高値はそれぞれ期間中の最も安い価格、高い価格である。ローソク足は、江戸時代に米相場の買い場、売り場予測に利用されたことで有名であるが、特定のパターンが現れたかどうかにより、移動平均などと組み合わせたり、トレンドの方向や転換の推定が行われる。株価は、株価が上がると考える市場参加者（トレーダー）が多いと需要の高まりで実際にも価格が上がり、下がると考えるトレーダーが多いと実際に株価も下がるという性質がある。よって、ローソク足を参考に、トレンド、底値（株価下降から上昇への転換点）、天井値（株価上昇から下降への転換点）を推定するトレーダーが多い場合には、実際の株価も同様に底値、天井値となる可能性が大きくなる。つまり、ローソク足を参考に取引するトレーダーの比率が多いほど、ローソク足による予測は正確性が上昇することとなる。以下では、ローソク足の概要、および実際のシステム構築方法、パラメータの設定についてまとめている。

3.1 用いる株価データとローソク足

ディープラーニングの学習の基本的考え方は大量のデータによる学習を用いることであるので、株価データは年足や月足よりも日足、さらにはティックデータである方がデータ分量という点では適切であると考えられる。ティックデータであるとサンプル発生期間が一定でなく、確定的な期間先の予測という目的には不適合と考えられるので、本研究で用いる株価データは、1 分足データとし、これから作成したローソク足チャートを CNN の入力画像データとする。また、特定の個別銘柄株では個別のイベント（決算予測発表や新製品の発表など）により、通常とは異なる変動が発生する可能性があるので、本研究では東証一部上場 225 銘柄の平均指標である日経平均株価を対象としてもちいる。株価の変動は実際に取引が成立した際に起こるので、日経平均は変動のしやすさという点でもディープラーニングによる予測に適していると考えられる。個別銘柄を対象とすることも可能であるが、よ

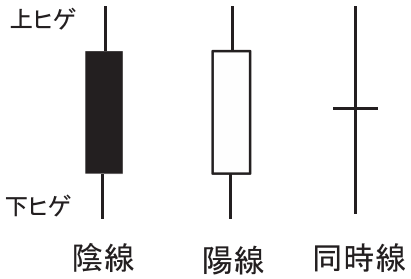

図 7: ローソク足

図 8: ローソク足チャート

り取引量の多い（流動性の高い）銘柄の株価データを用いることが必要である。教師データはタイムステップ T_{pred} 先に、1 分足終値基準で「上がる」(1 であらわす) か、または「下がる」(0 で表し、終値が変化しない場合も含む) のカテゴリデータを用いることとする。

図 7 にはローソク足を示している。上辺の位置を高値、下辺の高さを安値とした四角形を書き、始値より終値が低い（期間中に株価が下がった）場合には黒く塗りつぶされ陰線と呼ばれる。始値より終値が高い（期間中に株価が上がった）場合には、白く塗りつぶされ陽線と呼ばれる。始値と終値が同じである場合には四角が上下に潰れた線となり、同時線と呼ばれる。さらに、四角の上辺から高値まで直線を描き（上ヒゲ）、下辺から安値までにも直線を描く（下ヒゲ）。これにより、期間中の 4 本値と株価が上がったのか、下がったのかを、一目で視覚的に確認できることになる。

このローソク足を時系列的に並べた図をローソク足チャートという。図 8 には、本論文で用いるローソク足チャートの例を示している。入力画像では xy 軸は省略し、ローソク足と背景を区別できるように、背景色は白と黒の中間のグレー色としている。この図では 20 分間のローソク足チャートを示しており、1 分間のローソク足を 20 分間分並べ、20 分間の高値と安値により画像の y 軸の範囲を決めている。この画像を CNN の入力とすることにより、人間のトレーダーがローソク足チャートを参照し投資行動を決めることと同様な学習をすることができることになる。東京証券取引所では、朝 9 時から 11 時 30 分の前場と 12 時 30 分から 15 時までの後場があるが、前場、後場それぞれで、150 分分のデータが観測できる。本論文では、1 つの画像に 20 分のデータを用い、最大 5 分先までの予測を実施するので、前場、後場それぞれ、126 の入力画像と教師データ（1 から 5 分後に上がるか下がるか）が作成できる。2017 年 8 月 1 日から 2018 年 1 月 26 日の間の市場営業日の 120 日分を用い、30,240 個の画像を作成した。この画像の 90%を CNN のパラメータ最適化に用いる学習データとし、残り 10%を学習には用いず、学習していないデータをどの程度正確に予測できるかの汎用性検証（テスト）データとして用いることとする。10%のテストデータは、実験ごとに全サンプルの中から毎回ランダムに選択することとする。

3.2 実装方法

本研究の入力データであるローソク足チャートは、日経平均の1分足の始値、終値、高値、安値から、Microsoft Excelの株価チャートグラフとしてVBAにより全期間分自動的に作成し、52ピクセル × 52ピクセルの画像として出力した。この1枚の画像当たり2,704画素を1行に並べたものを、全画像数の30,240行分準備し、CNNへの入力に用いた。図8にあるような元々の画像は、255段階のグレースケールで、陽線を白(0)、背景色をグレー（127)、陰線と上下のヒゲを黒（255）で表現しているが、CNNへの入力の際には陽線を-1、背景を0、陰線を1となるように標準化したデータを用いる。

CNNシステムの構築には、Chainerフレームワーク（ライブラリ）を用いることとした[7]。Chainerとは、ネットワークを柔軟に表現できるフレームワークで、ディープラーニングの計算では、計算結果のみでなく計算過程や傾きを保存しておき、ウェイトの修正量計算に利用していることが特徴である。さらにChainerではGPUを用いた並列計算が容易に実施可能で、本研究のシミュレーションでは、CPUのみによる計算のおよそ10倍の計算速度となり、並列化は大量のデータ（ビッグデータ）を使い大量の計算を実施するディープラーニングシミュレーションでは必須の条件と考えられる。また、Chainer以外にもディープラーニングのフレームワークは多数あるが、Chainerは単純なネットワークであっても、複雑なネットワークであっても柔軟に設計ができることや、ネットワーク設定を別途準備するなどの必要もなくプログラミング言語Pythonのみで記述が可能で、開発のしやすさも特徴である[7]。

なお、本研究ではPython、Chainer、およびGPU並列計算用のライブラリであるNVIDIA社のCUDAライブラリをMicrosoft Windows10へインストールしたシステム上でシミュレーションを実施した。

3.3 CNNの設定

用いるCNNでは、入力されるデータは2次元画像で、本研究では、4.1、4.2で示したようなローソク足チャート画像である。ローソク足チャート画像1枚の入力に対し、出力層のユニット数は、設定したタイムステップ後に株価が「上がる」か「下がる」（変わらないを含む）かによる2カテゴリの予測とするため2ユニットとし、どちらのユニットの出力が大きいかによりカテゴリ（上がるか、下がるか）の変動予測を行う。具体的には、1つ目のユニット（0番ユニット）の出力の方がもう一方のユニットの出力より大きければ、株価予測は「下がる（変わらない含む）」、2つ目のユニット（1番ユニット）の出力の方が大きければ「上がる」という予測となる。また、本シミュレーションでは株価変動の上下の判断に、ローソク足チャート画像の最後の時刻（右端のローソク足）から、1分後から5分後までの1分足終値が「上がる」か、「下がる」かを予測することとする（何分後の変動を予測するかを$T_{step} = 1 \sim 5$で表すこととする)。一般的に言えば株価が1～5分後に変わらないということもよく発生するが、本研究では255種類の株価の平均である日経平均を予測対象としているので、変わらない値となることは稀にしか発生せず、変わらないを「下がる」（あるいは「上がる」）というカテゴリに含めても大きな影響はないと考えられる。個別銘柄の場合には、変わらないということも、多々発生するので、何らかの対処が必要となると考えられる。

入力画像は、まず、1つ目の畳み込み層で5×5のフィルタ20枚により処理され、パディングは用いていないため、52×52の画像が48×48の画像が20枚出力され、これら出力は2×2の最大値プーリングにより24×24の画像20枚となる。2つ目の畳み込み層では、この24×24の画像20枚（20チャネル）の入力を5×5のフィルタ50枚のフィルタにより処理により20×20の画像50枚の出力と

p	0.1	0.3	0.5	0.7	0.9	1.0
train acc	0.97	1.00	1.00	1.00	1.00	1.00
test acc	0.58	0.61	0.59	0.61	0.61	0.60
profit	4478	5825	4948	5909	6102	5728

表 1: ユニット使用比率による結果への影響

なり、再度の最大値プーリングにより、10×10 サイズの 50 枚の出力を得る。この出力は画素に分解され個々に全結合層の入力となるので、全結合層の入力データの次元は 5,000（$= 10 \times 10 \times 50$）次元となる。全結合層は第一層 5,000 ユニット、第二層 2,000、最終の出力層 2 ユニットという構成で設定した。畳み込み層、全結合層の活性化関数には式 (4) のシグモイド関数を用いた。

また、層の数や畳み込み層のフィルタ枚数、全結合層のユニット数を増やすと、より複雑な入出力関係を学習することができるが、増やしすぎると、畳み込み層のフィルタ、閾値や全結合層のウェイト、閾値の最適化に用いる学習データ（training data）への最適化が進みすぎ、学習に用いていない検証用データ（test data）の予測精度が下がってくる現象（過学習、overfitting）が発生することが知られている。本研究では、各入力データごとに学習の際に確率的に選択したユニットのみ使用することにより、学習の際にデータごとに最適化に用いられるユニットを少数に限定し、過剰な学習を防ぐことができるドロップアウト [3, 10] を導入することとする。ドロップアウトは畳み込み層にも導入可能であるが、本研究では全結合層部分のみに適用し、使用するユニットの比率を $p, 0 \leq p < 1$ と表すこととする。学習後の検証の際のテストデータを用いた予測では、すべてのユニットを用いて出力を計算するが、ドロップアウト使用時に比べて、ユニット出力の合計値が平均 $1/p$ 倍となるので、出力値を p 倍して、出力値の合計の平均値が同程度となるような調整を実施必要がある。

4 シミュレーション結果

以下では提案手法の予測精度について、ドロップアウト率の変化と予測のタイムステップ T_{pred} の違いによる影響をシミュレーションにより検証することとする。

ここでは予測の正確性を検証するため以下の予測精度（Accuracy、acc と標記）を用いる。

$$\text{acc} = (\text{上がると予想して正解した数} + \text{下がると予測して正解した数})/\text{全データ数}. \tag{15}$$

この式で全データ数とは学習データ、あるいはテストデータのサンプル数である。なお、前述のように変わらない場合は「下がる」に含めている。

4.1 ドロップアウト率 p の影響

まずは、ドロップアウトの際のユニット使用率 p を変更することによる、テストデータの正確性（Accuracy）の変化について検証を実施する。何分後の予測をするかのタイムステップは $T_{pred} = 3$ で固定し、ユニット使用比率は $p = 0.1, 0.3, 0.5, 0.7, 0.9, 1.0$ の 6 ケースについてシミュレーションを実施した。$p = 1.0$ ではすべてのユニットによる学習が実施され、ドロップアウト未使用ということになる。

図 9 には、$p = 0.5$ の場合の、学習データに対する式 (14) で示されるクロスエントロピー誤差（loss と標記）の学習回数（epoch）による変化を示している。これによると学習が進むことによりクロス

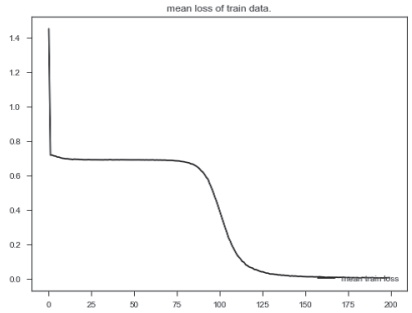

図 9: $p = 0.1$

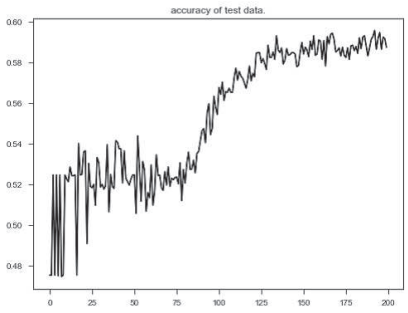

図 10: $p = 0.3$

エントロピー誤差は減少を続けており、学習データ（ローソク足チャートとその後の株価の上がり下がり）をシステムが学習できていることを示している。他の p の値を用いた際もほぼ同様の学習曲線となったので、図は省略している。ただし、ユニット使用率が低いほど、誤差の減少率が緩やかとなっており、収束に時間がかかる結果となった。

また、図 10 には予測精度（acc）の epoch に対する変化を示している。学習開始時には予測精度は約 52%で、これはテストデータの「上がる」、「下がる（変わらないを含む）」の比率 48 : 52 に近い値で、学習初期にはフィルタやウェイト最適化がまったくされておらず、どのような画像が入力されても「上がる」か「下がる」のいずれかのみの予測を繰り返すことによる影響があると考えられる。このことにより、初期段階では 48%か 52%に近い値が出ており、学習回数（epoch）が増えることにより、60%に近い値まで上昇している。こちらの図からは、学習データについて学習が進むと、学習に用いていないデータ（テストデータ）についても変動しながらではあるが、予測精度が上昇していることが確認できる。つまり、各時刻での変動パターン画像とその後の「上がる」「下がる」には関連性があり、60%弱の精度での予測が可能であることを示している。

表 1 には、ユニット使用率（p）を変化させたときの、学習 200epoch 後の、学習データの判別精度（train acc）、テストデータの予測精度（acc）、テストデータの予測に従い株式の取引をした際の損益（profit）をまとめている。train acc と test acc は少数点第三位にて四捨五入、profit については小数点第一位にて四捨五入している。

学習後に学習データをどの程度正しく分類できるかを表す判別精度については、ユニット使用率 p が $0.3, 0.5, 0.7, 0.9, 1.0$ の場合には 1.00 となっており、ドロップアウトによるユニット使用制限を実施しても全学習データ 27,216 サンプルについて入力ローソク足チャート画像とその後の「上がる」「下がる」の入出力関係をすべて学習できていることを示している。全結合層のユニットの 9 割のユニットに使用制限を与える $p = 0.1$ でも、0.97 とほぼ入出力の関係性を学習できていることを示している。テストデータの acc については、p を変化させても、今回の CNN の設定ではあまり大きな変化はなく、58 ～ 61 の値となっている。使用率 0.9 のケースでやや低くなっているが、おおよそ 60%前後となり、大きな違いは見受けられなかった。これらの結果により、本研究のシステム設定においては、ドロップアウトによる学習の収束性、テストデータの予測精度（汎化性能）には大きな違いが見られなかったが、これは全結合層のユニット数が 2000 であり、この 10%の 200 ユニットでも十分な十分な自由度があるということが理由であると考えられる。

損益 (profit) については、次のような取引により得た利益、あるいは損失について示している。

- （「上がる」予測の場合）CNN による予測が「上がる」であった場合には、ローソク足チャートの右端の終値で購入し、予測タイムステップ（T_{pred}）後に手仕舞い（購入した株をその時点

の株価で売却）をする。

- （「下がる」予測の場合）CNN による予測が「下がる」であった場合には、ローソク足チャートの右端の終値で空売り（株を借りて売却）し、予測タイムステップ（T_{pred}）後に反対売買（売却した株をその時点の株価で購入し返却）をする。

本シミュレーションでは $T_{pred} = 3$ としているので、「上がる」、「下がる」の予測により株の購入、あるいは空売り後、3 分後に決済することとなる。表 1 より、p が 0.1、0.5 のケースで 5,000 円弱となっているが、その他のケースでは 6,000 円前後の値となってフィルタ値やウェイトのランダムな初期値による収束性の違いによる程度と考えられ、p の違いによる大きな違いはないと考えられる。6,000 円という値のおおよそ年間の利益を計算してみると、テストデータのサンプルは 3,024 なので、1 回の取引当たりの平均利益は 6000/3024=1.98 円となる。前場、後場それぞれ 150 分間（昼休みは不連続と扱う）の取引可能回数は、ローソク足チャート作成に開始直後の 20 分分必要で、3 分後の予測なので、前場、後場それぞれで 128 回取引が可能となる。よって 1 日の取引可能回数は 256、1 年間の市場営業日を約 250 日とすると、1 年間の利益は $6000/3024 \times 256 \times 250 = 126,984$ 円となる。本シミュレーションで用いた 2017 年 8 月 1 日から 2018 年 1 月 26 日の 1 分足の終値の平均値は 21,557 円で、3 分間保有を継続するので、最大 64,671 円（$= 21557 \times 3$）の投資額が必要である。この必要投資額によって計算した年間の利益率は単利計算でおよそ 196%となる。ただし、本研究では株式市場で直接取引のできない日経平均株価を対象としており、さらには、これらの取引はローソク足チャート作成と同時に予測ができ、ローソク足チャートの右端の終わり値で瞬時に株式を購入、あるいは売却を決定し約定できると仮定し、株価取引の手数料を考慮していないので、あくまで仮想的なシミュレーションでの値となる。とくに、年間およそ 64,000 回の売買が必要となり手数料体系にもよるが取引手数料はかなりの額になると思われる。

4.2 予測のタイムステップ T_{pred} の違いによる影響

ここでは、予測のタイムステップ T_{pred}、つまり、ローソク図チャートに用いた株価（終値）から何分後の終値を予測対象とするかの違いによる予測精度、損益額の比較を行う。$T_{pred} = 1, 2, 3, 4, 5$ の 5 つのケースを仮定し、学習時のドロップアウトにおけるユニット使用率は、$p = 0.3$ で固定しておく。

図 11 から 14 には、予測タイムステップ $T_{pred} = 3$ のケースの学習データのクロスエントロピー誤差、学習データの判別制度（その時点での学習データの分類精度）、テストデータのクロスエントロピー誤差、テストデータの予測精度の epoch=200 までの推移を示している。学習データの誤差の推移は、ドロップアウトのユニット使用率で見たものと同様に、学習回数（epoch）が増えるにつれ減少しており、CNN の学習が進んでいることが分かる。学習データの判別制度を見ると、学習の初期ではほぼランダムな判別の 0.5 をわずかに超えた程度であるが、100epoch 辺りから急激な上昇をしており、200epoch では 1.0 へ収束していることが分かる。この 2 つの図により、200epoch で学習データの入力ローソク足チャートの画像から 3 分後の株価の「上がる」、または「下がる」を適切に学習できていることが確認できる。

次に、テストデータの誤差を見ると、学習データとは逆に 100epoch 辺りから上昇をしている。ただし、テストデータに関しては誤差が増加したとしても、「上がる」、または「下がる」のどちらの出力が大きいかによる分類が目的であるので、ある程度の誤差上昇があったとしても、分類予測の精度が上昇していれば問題はない。ただし、誤差の面からみると、学習データによる最適化が、かならずしもテストデータの最適化と一致していないことを示している。テストデータの予測精度もドロップアウトの検証の際と同様に、当初は「上がる」「下がる」の割合によって 48 ～ 52% で推移し

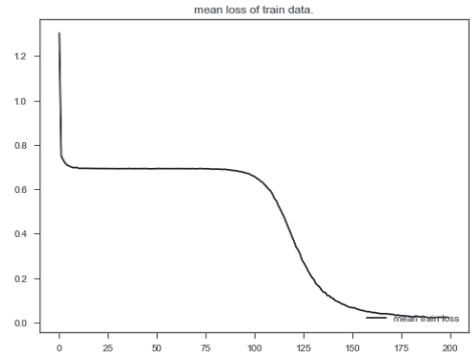

図 11: 学習データの誤差

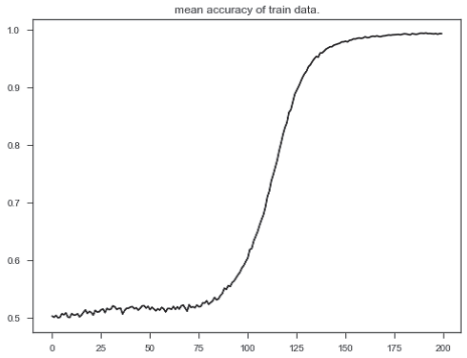

図 12: 学習データの判別精度

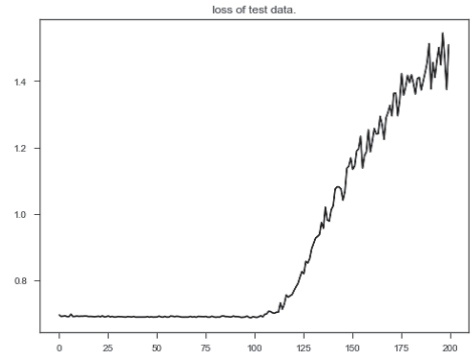

図 13: テストデータの誤差

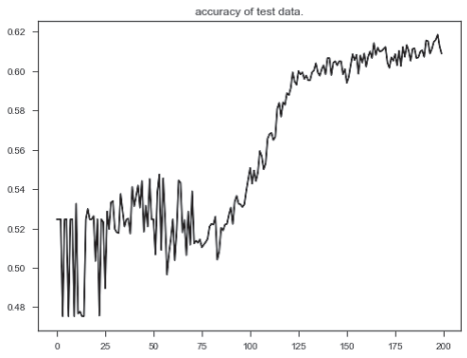

図 14: テストデータの予測精度

ているが、100epoch 辺りから上昇を始め、200epoch では 61%程度で変動しており、誤差は上昇をしているが、予測精度については学習データとともに上昇をしていることが分かる。この予測精度と学習データの判別精度の図より、学習データとシステムの最適化に用いていないテストデータには、入力（つまりローソク足チャート）、出力（「上がる」、「下がる」）の関連性に共通する点があり、ある時点での学習がそれ以外の時点での予測に有効であることが示されている。

図 15 から 18 には、それぞれ予測タイムステップが $T_{pred} = 1, 2, 4, 5$ のケースの学習データの判別精度の epoch に対する変化を示している。これらの図によると $T_{pred} = 3$ のケースと同様に学習データの判別制度は epoch に応じて増加し、どのケースでも 200epoch でおよそ 1.0 に近い値を示している。ただし、$T_{pred} = 1$ については、精度の上昇が他のケースよりも遅く、最終的に予測精度が 1 に満たない値で終了しており、入出力関係がより複雑であることを示唆している（epoch 数を増やすと判別制度が 1 に収束する可能性はある）。学習データに対する誤差は学習が進むにつれ減少していくが、$T_{pred} = 1, 2, 4, 5$ のケースは $T_{pred} = 3$ のケースと大きな違は見られなかったので図は省略している。

図 19 から 22 には、それぞれ予測タイムステップが $T_{pred} = 1, 2, 4, 5$ のケースのテストデータの予測精度の epoch に対する変化を示している。$T_{pred} = 1$ では、200epoch の学習後であっても予測精度はおよそ 0.5、つまりランダムに「上がる」、「下がる」を予測する行動と同じ予測精度となっている。上記のように 200epoch では学習データについてはほぼ学習が収束しているので、入力のローソク足チャートとその 1 分後の「上がる」「下がる」には、違う時点の入出力関係に傾向的なものがほ

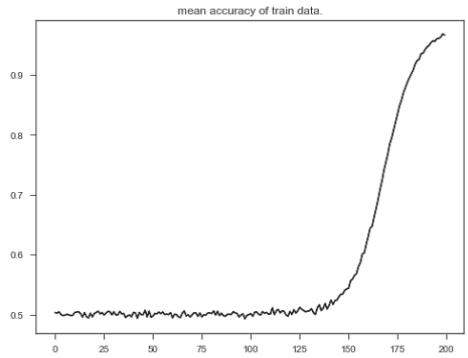

図 15: 学習データ判別精度（$T_{pred} = 1$）

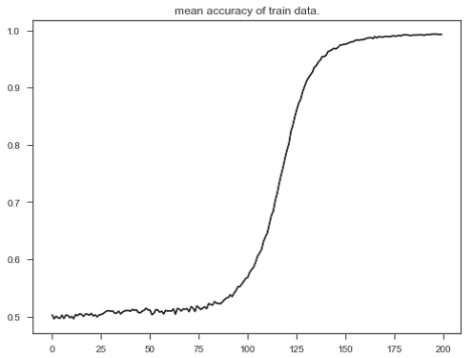

図 16: 学習データ判別精度（$T_{pred} = 2$）

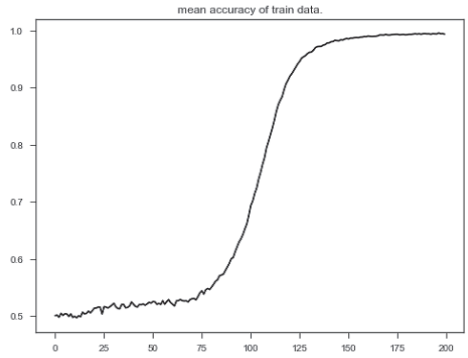

図 17: 学習データ判別精度（$T_{pred} = 4$）

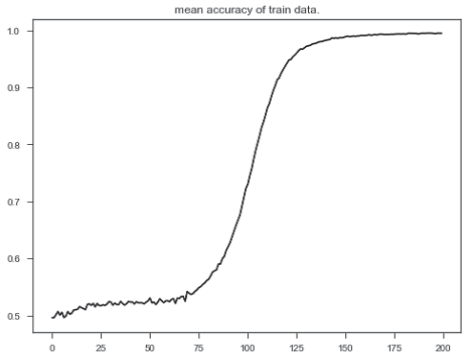

図 18: 学習データ判別精度（$T_{pred} = 5$）

ぼなく、学習に用いたデータ以外への予測の適用は難しいと考えられる。これは CNN システムが学習データに特化していることを示している。$T_{pred} = 2, 4, 5$ については、$T_{pred} = 3$ と同様に epoch が増加するにつれて予測精度が上昇をしている。ただし、$T_{pred} = 2$ については、初期の 48 ∼ 52 近辺での変動が他のケースと比べ長く続き、160epoch 辺りを境に変動しながらではあるが下降傾向を見ることができる。これは、$T_{pred} = 1$ のケースと同様に学習データの入出力関係とテストデータの入出力関係の類似度が他のケースに比べるとやや低く、学習データへの過学習が発生していると考えられる。$T_{pred} = 4, 5$ は $T_{pred} = 3$ と同様 200epoch までは過学習の傾向はみられず、変動しながらではあるが上昇傾向にあるので、学習データの入出力関係とテストデータの入出力関係にある程度の共通点があり、実際の株式取引の意思決定に有効である可能性を示唆している。

表 2 には、$T_{pred} = 1, 2, 3, 4, 5$ のそれぞれの予測タイムステップについての 200epoch 後の、学習データの判別制度、テストデータの予測精度、テストデータについての損益額と年間の投資利益率の推計値を示している。学習データの判別制度（train acc）についてはどの T_{pred} でもほぼ 1 になっており、学習データの入出力関係が適切に学習できていることが分かる。テストデータの予測精度（test acc）については、$T_{pred} = 1$ については「上がる」「下がる」のサンプルの比率と同様にランダムに売買を実行したときの 50%とほぼ同じ値となり、テストデータについての適切な予測が実施できていないことを示している。$T_{pred} = 2, 3, 4, 5$ については 0.57 ∼ 0.63 の値となっており、ランダムな予測よりも良い精度を示している。また、タイムステップが長くなるほど予測精度が向上しているので、ロー

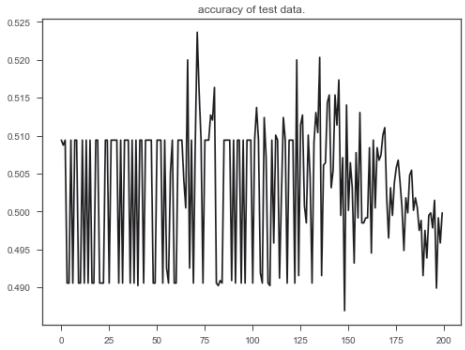

図 19: テストデータ予測精度（$T_{pred} = 1$）

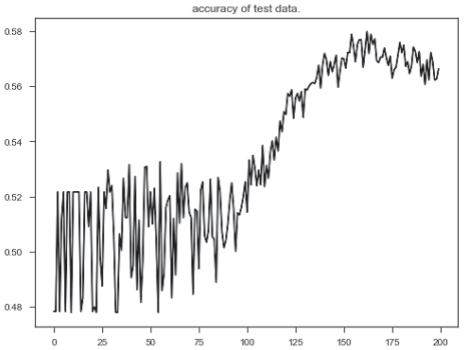

図 20: テストデータ予測精度（$T_{pred} = 2$）

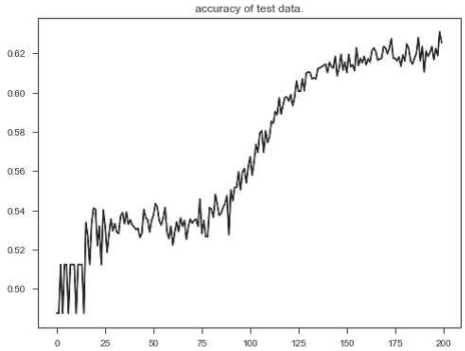

図 21: テストデータ予測精度（$T_{pred} = 4$）

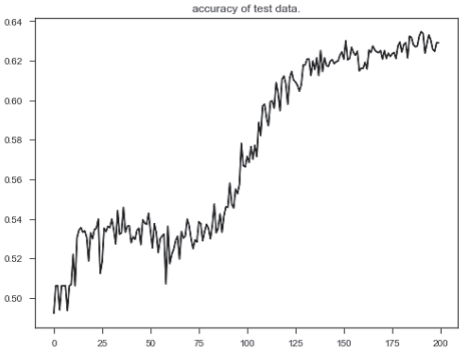

図 22: テストデータ予測精度（$T_{pred} = 5$）

ソク足チャートに使った株価の直後の「上がる」、「下がる」よりも、数分後の「上がる」、「下がる」の方が学習データの傾向とテストデータの傾向に類似性が高いと考えられる。これは、1 分足チャートを見てから、実際の人間や取引システムが反応するまでに数分を要し、1 分後では対応が間に合っておらずランダムな要因により変動が発生しているということが原因ということかもしれない。

つぎに、テストデータ 3,024 サンプルについて、ドロップアウトの検証と同様に「上がる」「下がる」の予測に従い株式の購入、あるいは空売りをして、T_{pred} 後に決済する取引を実行した場合の損益額（profit）と、システムに従った取引を 1 年間繰り返した際の収益率（Rate of profit）の推計値を見ていく。1 年間の推計はドロップアウトの検証の際と同様に、テストデータの 1 サンプル当たりの平均損益額（=損益額/3024）を求め、これに 1 日の可能な取引回数（実際は全タイムステップで異なるが、256 回で統一とする）と、1 年間のおよその市場営業日（250 日とする）をかけた額を、必要な投資額で割った値である。つまり、1 回当たりおよそ 21,557 円（用いた株価の平均終値）の投資行動を、1 年間でおおよそ 64,000 回繰り返した場合の株価収益率の推計値である。$T_{pred} = 1$ では、損益額は-60、年間の推計損益率は-5.9%である。これはランダムな投資行動とほぼ同じ予測精度であったことから、投資の勝ち、負けを繰り返しこのような額、率になったと考えられる。このことからも $T_{pred} = 1$ での予測は困難であることが分かる。$T_{pred} = 2, 3, 4, 5$ については、損益額は T_{pred} が長くなるに従い増えている。これは T_{pred} が長くなることにより予測精度が向上していることも原因ではあるが、1 回の取引で T_{pred} の期間株式の保有（あるいは空売りの継続）をしているので、保有期

T_{pred}	1	2	3	4	5
train acc	0.99	1.00	1.00	1.00	1.00
test acc	0.49	0.57	0.61	0.63	0.63
profit	-60	2795	5825	7726	8801
Rate of profit	-5.9%	91.5%	190.6.6%	189.6%	172.8%

表 2: T_{pred} の違いによる結果への影響

間が長くなることにより株価変動額の平均が大きくなることも理由である。1 年間の推定利益率は、$T_{pred} = 3$ で最も高い 190.6.1% を示している。この率は、1 回当たり 21,557 円の投資（ただし T_{pred} 期間保有するので、最大 3 倍の 64,671 円の投資額が必要）により、1 年間で 123,280 円の利益となることを示している。投資の保有期間を考慮しないのであれば、$T_{pred} = 5$ が最も 1 年間の推計利益が大きくなり、186,265 円となる。ただし、ドロップアウトの変化の影響の際にも述べたように、瞬時の予測、売買が必要であることと、$T_{pred} = 2, 3, 4, 5$ の 1 回の投資行動当たりの平均損益額は、それぞれ $0.92, 1.93, 2.55, 2.91$ となっており、取引の手数料を考えると実際に得られる利益はこの推計値よりも小さくなると考えられる。

上記の学習データの学習状況、テストデータの予測精度、損益額、推計年間収益率より、CNN により学習したシステムを用いると、学習の用いていない期間の予測がある程度可能となり、1 分足株価であれば、5 分先まででは予測タイムステップが長いほど予測精度が向上し、1 年間の収益率でみると 3 期先（3 分後）の予測が効率的であることが分かった。

5 むすび

本研究では、近年研究が急速に進んできたディープラーニング、特に畳み込みニューラルネットワークの株式投資への応用について提案、およびシミュレーションによる有効性の検証を実施した。具体的なシステムとしては、1 分足株価のローソク足チャート図を入力とし、1 ～ 5 分後の「上り」、「下がり」の変動予測を行った。今回の日経平均株価の 2017 年 8 月 1 日から 2018 年 1 月 26 日の 1 分足データにおいては、1 分後の予測は困難であったが、2 分から 5 分の予測精度と損益額、収益率はランダムな取引と比較するとよい結果となった。今後は個別銘柄への適用、勝敗率の向上、よりよいハイパーパラメータの設定値など様々な面からの検証を続けていきたい。

謝辞

本研究は JSPS 科研費 JP17K01267 の助成を受けたものであり、ここに感謝の意を表します。

参考文献

[1] 池田欽一, 林田実, ディープラーニングの株価予測への応用, 北九州市立大学商経論集,52(1–4), 13–26, 201, https://kitakyu.repo.nii.ac.jp/?action=repository_uri&item_id=552&file_id=22&file_no=1.

[2] Yann Lecun and Léon Bottou and Yoshua Bengio and Patrick Haffner, “Gradient-based learning applied to document recognition”, Proceedings of the IEEE, 2278–2324,1998.

[3] 岡谷貴之, 深層学習, 講談社,2015.

[4] Rosenblatt F.,“The Perceptron: A probabilistic model for information storage and organization in the brain”,Psychological Review, 65, 386–408, 1958.

[5] Minsky M. and Papert S.,Perceptrons: An Introduction to Computational Geometry. MIT Press, MA,1969.

[6] Rumelhart D. E. , Hinton G. E. and Williams R. J. ,“Learning internal representations by error propagation. Parallel Distributed Processing”, 1, MIT Press, MA, 318-362, 1986.

[7] Diederik P. K. , Jimmy L. B., “ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION”, Conference paper at ICLR 2015,1-15,2015.

[8] “A Powerful, Flexible, and Intuitive Framework of Neural Networks”,`http://chainer.org/`.

[9] 神嶌敏弘（編）, 麻生英樹, 安田宗樹　他（著）, 深層学習, 近代科学社, 2015.

[10] Srivastava N. , Hinton G. E. , Krizhevsky A. , Stutskever I. , and Salakhutdinov R, “Dropout: A simple way to prevent neural network from overfitting”, Journal of Machine Learning Research, 15(Jun), 1929–1958, 2014.