卒業論文

株価予測に使われる機械学習モデルの比較検証

ウォード　空

名古屋大学情報学部自然情報学科複雑システム系

学籍番号 101810045

2022年（令和4年）2月14日提出

株価予測に使われるモデル比較検証

ウォード　空

概要

株価は非線形な関係性を持つとされ、昔から予測が困難と考えられてきた時系列データである。しかし、機械学習分野の発展と、それに伴う非線形な関係性を学習できるモデルの開発によって、以前よりも高精度の予測が可能になってきている。そこで、本研究では近年の株価予測の研究で使われている機械学習モデルを網羅的に調査し、その予測精度を比較した。この結果、長・短期記憶型ニューラルネットワーク（LSTM）が最も良い精度を示した。続いて勾配ブースティング決定木が良い精度を示し、サポートベクトルマシンが最も低い精度を示した。なお、一般的にニューラルネットワークは隠れ層を増やすことで精度が上がるとされているが、本研究ではスタッキングによって隠れ層を増やすことよりも、隠れ層のベクトルの次元を増やす方が良い精度が得られることがわかった。また、特徴量重要度とモデル精度の比較により、精度の高いモデルは株式市場の格言「出来高は株価にの燃料である」[1]を反映していることがわかった。

Contents

1　序論

株式市場は現在、我々の経済に大きな影響力を持っており、多大な資金がやり取りされる場所である。株式の値段を予測し、適切な時に買い、適切な時に売ることによって一部の投資家は巨大な富を築きあげてきた。その為、以前から株式市場の変動を数理的に予測するために以前から研究者や投資家は株価の変動を予測するシステムを開発し、実装することに注目してきた。しかしながら、株価の変動は非線形的でノイズが非常に多く、予測が非常に困難とされてきた[2]。有名な効率的市場仮説(Fama *et al.* 1965[3])は、株価がランダムウォークに従うとし、予測によって得られる利益は、継続的に市場平均を上回ることがないことを主張している。しかし近年は、このように株価が予測できないという主張に反した結果を報告している研究が多くあるため、説得力を失いつつある[4]。このような研究の中に、成熟市場と新興市場の効率性の違いを測定するものや、株式市場の有効な予測モデルを構築しようとするものがある。

株価予測に対するアプローチは主に二通りある。一つはファンダメンタル分析であり、これは企業の経済活動を分析し、その株価を評価する分析手法である。もう一つはテクニカル分析であり、これは企業の過去の株価を分析し、その株価を評価する分析手法である。前者はテキストデータなどの整備されていないデータを用いることが多いのに対し、後者はそれぞれの取引所が発表する、定量的で整ったデータに基づいており[5]、近年の株価予測の研究の多くはテクニカル分析を用いている。

テクニカル分析で使われる予測システムを構成するモデルは主に二種類に分類することができる。一つは統計モデルであり、これにはARIMAやGARCHなどのモデルが含まれる。もう一つは機械学習モデルであり、これにはニューラルネットワークなどが含まれる。

近年テクニカル分析で使われる機械学習モデルには様々なものがあり、勾配ブースティング決定木を使ったもの[6],[7]や、サポートベクターマシンを使ったもの[8]、ニューラルネットワークを使ったもの[9],[10],[11]などがある。この分野では、ニュースのテキストデータやテクニカル指標を使って多くの特徴量を作成し、これを様々なベースモデルを合体させた複雑なモデルに入力として与え、予測を行い、精度を競うという傾向がある。しかし、ほとんどの場合論文によって入力として使っているデータが互いに異なっており、論文のコードを公にしていないために追試が困難になっている。そのため、モデル間の優劣が明確になっていないのが現状である。

そこで本研究では、これら複雑なモデルを構成するベースモデルとして勾配ブースティング決定木、サポートベクターマシン、回帰型ニューラルネットワーク(LSTM)を取り上げた。これらを同じデータで訓練、モデル選択、テストすることによってパフォーマンスを比較する。これによって株価予測に適したモデルを求め、

モデル間の精度の違いについて考察する。

本論文ではまず2章で比較方法の定式化と使用したデータについて述べる。 3, 4, 5章では比較したモデルについてその構造とデータへのフィッティングについてそれぞれ述べ、6章でそれらのモデルの精度の比較と結果の考察をする。7章では論文のまとめと、展望について述べる。

2　問題の定式化

ここではデータをどこから取ってきたのかと、どのようにして加工し、モデルを学習し、比較するのかについて述べる。

2.1　予測方法の定式化

本研究は短期的な予測に焦点を当て、次の日に株価が「上がる」または「上がらない」のニクラス分類問題に帰着させた。「上がる」クラスをC_1,「上がらない」クラスをC_2とした。また、それぞれのモデルのスコープは50日とした。これによって短期的な関係性のみを学習させることにした。

2.2　データについて

データはデータはテック企業の取引が盛んであるアメリカのNASDAQ取引所からのアップル社の日次取引データを使用した。データは2010/1/14-2021/9/10の期間であり、csv file形式で、最初の5行は以下の図2.1のようになる。

	Date	Open	High	Low	Close	Adj Close	Volume	Name
0	2010-01-04	7.622500	7.660714	7.585000	7.643214	6.562591	493729600	AAPL
1	2010-01-05	7.664286	7.699643	7.616071	7.656429	6.573935	601904800	AAPL
2	2010-01-06	7.656429	7.686786	7.526786	7.534643	6.469369	552160000	AAPL
3	2010-01-07	7.562500	7.571429	7.466071	7.520714	6.457407	477131200	AAPL
4	2010-01-08	7.510714	7.571429	7.466429	7.570714	6.500339	447610800	AAPL

図. 2.1

ここでカラムの意味はそれぞれ

- Date - そのデータが記録された日にち
- Open - マーケットオープン時の株価（始値）
- High - その株価の一日の最高値（高値）
- Low - その株価の一日の最低値（安値）
- Close - マーケットクローズ時の株価（終値）
- Adj Close - 株式分割などを考慮するように終値を調整したもの(調整後終値)
- Volume - その一日に成立した売買の数量（出来高）

- Name - 株式の名前

となっている。

2.3 特徴量生成

データに与える特徴量は主に以下の三つの特徴量の対数を取ったものを使用した。

- Volume
- Volatility = High − Low
- Adj Close return = 今日のAdj Close ÷ 昨日のAdj Close

Volatilityは株価の安定性の指標になっており、Adj Close return（調整後終値の収益率）は株価の前日からの変化の指標になっている。
この三つに絞った理由は、これらは株式市場でよく使われる基本的な指標データであり、まず簡単なデータを使って精度を測りたかったからである。
ニクラスのラベルは調整後終値の収益率を、その平均より少し大きい値を閾値として設定し、閾値より大きい値を1、閾値より小さい値を0としてC_1, C_2のラベルを1, 0とした。これによって得られたラベルを1日未来にずらすことによって予測するデータとした。
勾配ブースティング決定木とSVMは回帰型ニューラルネットワークとは異なり、データを入力した順番で時間を認識しない。したがって時間を認識させるために以下のような日にちデータを特徴量として加えた。

- day of week - 曜日データ
- day of month - 月始からの日数
- day of year % 50 - 年始からの日数を50で割った余り

なお、それぞれのモデルの特徴から、ニューラルネットワークと勾配ブースティングの入力データは(-1, 1)の間の値をとるようにスケールし、サポートベクターマシンの入力データは平均0, 標準偏差1となるようにスケールした。

2.4 比較の定式化

2.1で述べたそれぞれのモデルが選択される過程は次の通りである。まず、これら特徴量からなるデータを訓練データ、検証データ、テストデータに分けた。次に、訓練データで訓練したモデルを、検証データに対して推論させ、評価指標によってモデ

ルの選択を行った。最後に選択したモデルをテストデータに対して推論させ、評価指標(accuracy)によって比較した。accuracyの定義は以下の通りである。

$$\text{accuracy} = (\text{真陽性率} \ + \ \text{真陰性率}) \ / \ 2 \tag{2.1}$$

3　勾配ブースティング決定木

本節では、非常に単純でありながら非線形な関係性をモデリングできる勾配ブースティング決定木モデルの特性を述べ、近年著しい成果を挙げているlightGBMと呼ばれる勾配ブースティング決定木アルゴリズムを使った株価のモデリングを行う。

3.1　モデルの系譜学

機械学習の文脈においては、一つの最良のモデルを探すのではなく、複数のモデルを組み合わせて予測を行う方法(アンサンブル)がしばしば用いられる。これは1962年に”The Combination of Forecasts”[12]が発表されて以来あらゆる予測問題の主力手法となっている[13]。
広く使われるアンサンブル手法にはバギングやブースティングがある。決定木のアンサンブルにおいて時系列データをより良くモデリングするのは後者であるため[13]、ここでは後者について述べる。ブースティングでは順次訓練されるモデルでアンサンブルが構成され、構成する各モデルはその直前のモデルが生成した誤差の修正に焦点を当てて学習する、モデル訓練技法である。

アルゴリズムとして決定木を実装した初期のものにはCART(Breiman *et al.*, 1984)やID3(Quinlan, 1986)などがあるが、これらは一本の決定木を作成するアルゴリズムであり、そのままでは表現力に乏しい。しかし、先述のブースティング手法を適用した勾配ブースティング決定木アルゴリズムはその初期のアルゴリズムAdaBoost(1999)[14]が発表されてから20年以上経った今でも、最も強力なモデルの一つになっている。実際、ここ数年で時系列用の機械学習手法と比べて現代的な勾配ブースティング決定木アルゴリズムがはるかに大きな成果を挙げていることが、kaggleのコンテストや産業界の機械学習会議などにおいて多々報告されている[13]。

3.2　モデルの概要

3.2.1　決定木について

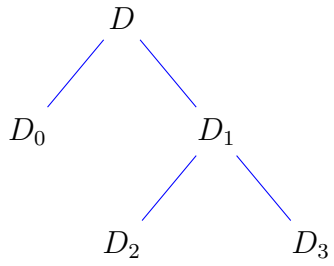

図. 3.1: 決定木

決定木は、人間の非線形的な決断方法を模倣した形のモデルである。図3.1に見られるように、決定木学習は与えられた訓練データDを、各ノードにおいて特徴ベクトルの要素を使ってD_0, D_1に分割することで進められる。分割されたデータ集合D_0, D_1は、それぞれ子ノードへ送られる。子ノードは送られてきたデータ集合を分割する。このように、分割が再起的に繰り返されて決定木が成長していく。分割後の葉ノードにあるデータ集合がある条件を満した時点で分割をやめる。分割をやめた葉ノードのデータ集合の残差の平均値を、予測値として付与する。これによって決定木は入力空間を多次元の直方体領域に区分する。

決定木の分割アルゴリズム　ここで、具体的に決定木がどのように分割しているかを、簡単な分割アルゴリズムであるCART訓練アルゴリズムを使って説明する。また、分割は二分割とする。

CARTアルゴリズムの手順[15]

1. 木の根ノードから始める(j = 1)。
2. 不純度（誤差関数）を最も小さくする分割点s^*をデータ集合Dの中の分割点候補sの中から選ぶ。
3. ノード(j = 1)におけるデータをs^*を使って二つのノード(j = 2,j = 3)に分割する。
4. 末端のノード(j = 2, j = 3)それぞれに対してステップ1-3を木の成長終了条件が満たされるまで適用する。

なお、上のアルゴリズムの手順で述べた不純度の指標として良く使われるものに、負の交差エントロピー誤差関数がある。葉ノードを$\tau = 1, \ldots |T|$と番号づけ、データ集合D内においてクラスC_1に割り当てられるデータ点の集合の割合を$p_{\tau 1}$、クラスC_2に

割り当てられるデータ点の集合の割合を$p_{\tau 2}(=1-p_{\tau 1})$とすると、交差エントロピー誤差関数は

$$Q_\tau(T) = p_{\tau 1}\log(p_{\tau 1}) + (1-p_{\tau 1})\log(1-p_{\tau 1}) \tag{3.1}$$

と表される。交差エントロピーはノード確率に対する感度が高いため、決定木の成長のために誤分類率よりも優れた尺度である[16]。

決定木の特性　決定木は与えたデータに過剰適合しやすい性質を持っているため、その対策として決定木の木の形を制限するようなハイパーパラメータを与える。代表的なものに木の深さを制御するものや、ノードを分割するために必要なサンプル数の下限を設定するものなどがある。また、決定木はデータ集合の細部に対して非常に敏感であり、実際には訓練データのわずかな変化により大きく異なる分割結果が得られることが知られている[16]。

3.2.2　勾配ブースティング決定木について

基本的な勾配ブースティング決定木であるAdaBoost[14]のアルゴリズムの手順は以下の通りである。
ここで、データ集合Dは二値の目標変数$l \in \{1, 0\}$を伴う入力ベクトル$\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$であるとする。
AdaBoostの手順[16]

1. $n = 1, \ldots, N$のデータの重み係数$\{w_n\}$を、$w_n^{(1)} = 1/N$に初期化する

2. $m = 1, \ldots, M$について以下を繰り返す:
   (a)分類器$y_m(\mathbf{x})$を、次の重み付けされた誤差関数を最小化するように訓練データにフィットさせる。

$$J_m = \sum_{n=1}^{N} w_n^{(m)} I(y_m(\mathbf{x}_n) \neq l_n) \tag{3.2}$$

   ここで、

$$I(y_m(\mathbf{x}_n) \neq l_n) \tag{3.3}$$

   は指示関数であり、$y_m(\mathbf{x}_n) \neq t_n$の時に1, それ以外(すなわち正く判定した時)は0である。
   (b)次の値を計算する

$$\epsilon_m = \frac{J_m}{\sum_{n=1}^{N} w_n^{(m)}} \tag{3.4}$$

これは指示関数3.3のデータ点の重み$w_n^{(m)}$による加重平均であり、各分類器$y_m(\mathbf{x})$の誤差率の尺度である。これを用いて、次の量を求める。

$$\alpha_m = \log\left\{\frac{1-\epsilon_m}{\epsilon_m}\right\} \tag{3.5}$$

これは誤差率ϵ_mに反比例するため、α_mの値は精度が大きな分類器では大きく、精度が小さな分類器では小さくなる。
(c)データ点の重み係数を以下の式で更新する

$$w_n^{(m+1)} = w_n^{(m)} \exp\{\alpha_m I(y_m(\mathbf{x}_n) \neq l_n)\} \tag{3.6}$$

正く分類されたときは3.3の値が0なので$w_n^{(m+1)} = w_n^{(m)}$となって重みは変わらない。誤って分類されたときは3.5のα_mが指示関数に掛かっているので、高精度な分類器であるほど誤分類したデータ点の重みは大きくなるよう更新される。

3. 以下の式で、最終モデルの予測を構成する。

$$Y_M(\mathbf{x}) = h\left(\sum_{m=1}^{M} \alpha_m y_m(\mathbf{x}) - \beta\right) \tag{3.7}$$

ここでhは閾値を超えると1、越えないと0を返すステップ関数であり、βは閾値である。ここでは$y_m(\mathbf{x})$に3.5のα_mが掛かっているため、より正確な分類器の予測値がより大きく出力に影響することがわかる。

lightGBM　lightGBMは2017年に発表された勾配ブースティングアルゴリズムである。その効率と、他の勾配ブースティング決定木アルゴリズムと比べても遜色ない精度から、このアルゴリズムをデータに当てはめることにする。下記の二つの特徴から、lightGBMは大きなデータセットに対しての学習で、他の手法より優位であり、そのため近年ではkaggleなどのデータ分析コンペでよく使われるようになっている。

Leaf-wise tree growth
ほとんどの勾配ブースティング決定木アルゴリズムはLevel-wise tree growth(深さ優先分割)を採用しているのに対して、lightGBMはLeaf-wise tree growthという手法で葉を増やしている。
Leaf-wiseはLevel-wiseより低い誤差を生み出す傾向があるものの、データが小さい時、過剰適合しやすくなっている。

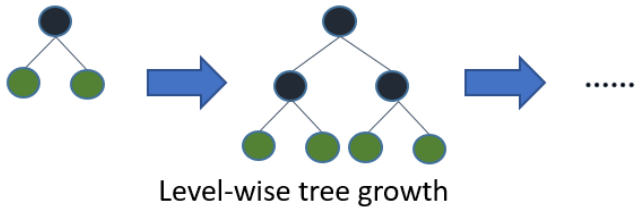

図. 3.2: [17]

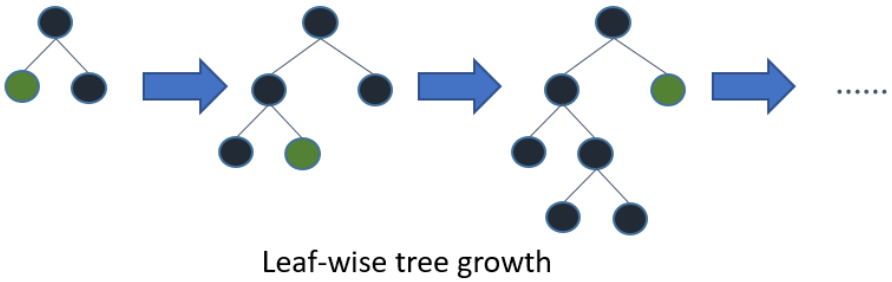

図. 3.3: [17]

Histogram-based

多くの勾配ブースティング決定木アルゴリズムは決定木のノードが行う分割方法にpre-sortedアルゴリズムを採用しているのに対して、lightGBMではHistogram-basedアルゴリズムを採用している。

pre-sortedアルゴリズムは事前にソートされた特徴量データの全ての分割点を数え上げて最適な分割点を探していく。これに対し、Histogram-basedアルゴリズムは連続な特徴量データを離散的にすることで得られるデータ区間を元に、分割点を探していく。

以上の点から分かるように、pre-sortedアルゴリズムではより正確な分割点を見つけることができるのに対し、Histogram-basedアルゴリズムではより少ないメモリ使用量で、より速く分割点を求めることができる。

3.3　データへのフィッティング

lightGBMには非常に多くのハイパーパラメータがあり、これらを一つ一つチューニングしていくのは困難であるため、訓練とバリデーションデータを使い、ハイパーパラメータ探索アルゴリズムであるOptuna [18]を使ってチューニングを行った。また、lightGBMはモデル検証において閾値で処理していない、連続な値を返すので、accuracyを使ってモデル評価をすることができない。そこで二クラス分類タスクにおいて、連続な値を返すモデルの評価にしばしば使われるArea Under Receiver Operator Characteristic(AUC)を使った。Receiver Operator Characteristic(ROC)とは、閾値を変えていった時の偽陽性率を横軸、真陽性率を縦軸に取ったグラフのことで(図3.4)、Area Under Receiver Operator Characteristicはこのカーブ下の面積を指す。以下の図で、これは青線の下の面積に対応している。

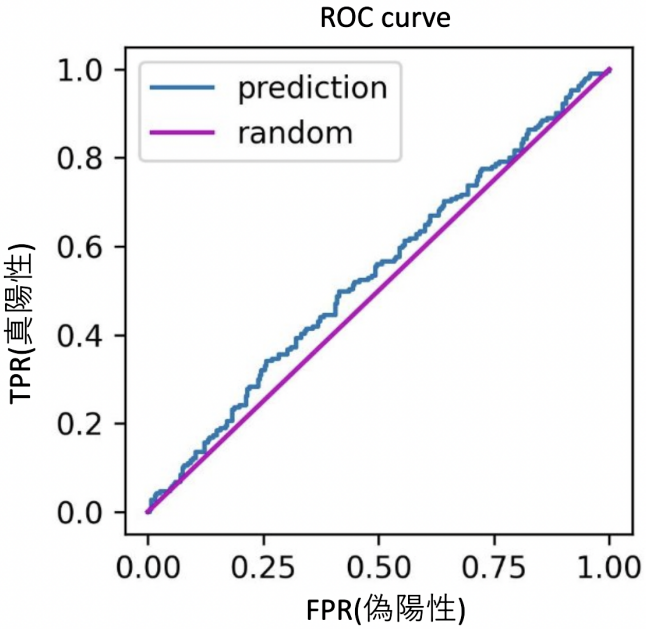

図. 3.4: ROCカーブ

OptunaにはこのAUCを最大化させるようにモデル選択をさせた。lightGBMはOptunaの毎試行ごとにもハイパーパラメータ探索を行っているため、試行回数は5回とした。l1,l2正則化係数はともに$[1e-8, 1.0]$の対数スケール空間から、葉の数は$[2, 112]$の一様空間、bagging_fraction, feature_fractionはともに$[0.4, 1.0]$の一様空間、bagging_freqは$[0, 10]$の一様空間、min_child_samplesは$[5, 100]$の一様空間からサンプルする様にした。この結

果、以下のようなパラメータの組み合わせが選ばれた。

'lambda_l1' : 0.000296,

'lambda_l2' : 0.981,

'num_leaves' : 19,

'feature_fraction' : 0.44072512405923314,

'bagging_fraction' : 0.7486595391434323,

'bagging_freq' : 5,

'min_child_samples' : 74

選択されたパラメータを使ってテストデータを予測すると、accuracyは0.532であった。
また、選ばれたパラメータを使ってテストデータの最初の50日について予測をした結果、以下のようになった。

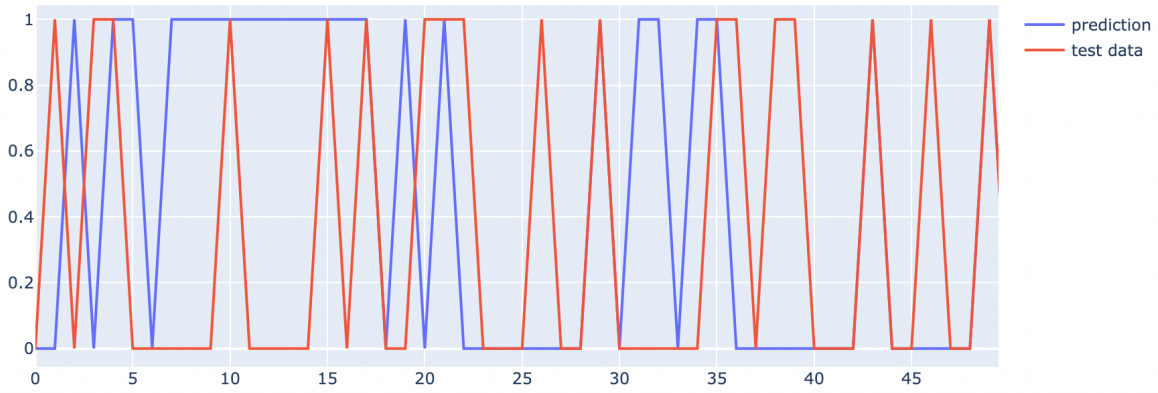

図. 3.5

4　サポートベクターマシン

近年ではニューラルネットワークの活躍によりあまり耳にすることがなくなっているサポートベクトルマシンであるが、サポートベクトルマシンは理論的に学習の汎化性能を求めた結果生み出されたものであり、実用面でも幅広く使われている。本節ではサポートベクトルマシンを使って株価のモデリングを行う。

4.1　モデルの系譜学

サポートベクトルマシンはOptimal Separating Hyperplane[19]を元に開発された機械学習モデルである。このモデルは初めは二クラス分類モデルであったが、その後多クラス分類や回帰にも拡張された。ディープラーニングの発展によってニューラルネットが広く使われるようになる前は、様々な分野で、最先端の機械学習アルゴリズムとして使われていた。

4.2　モデルの概要

分類問題において、一般にはクラスを正確に分類できる解は多数存在し得る。このように訓練データを分類する解が複数存在する場合、汎化誤差が最も小さくなるような解を求めることが望ましい。サポートベクターマシン(SVM)はマージンという概念を用いて、このような解を求めようとする手法である。マージンとは、訓練データと分類境界の最短距離であり、SVMはマージンを最大化する分類境界を求める。

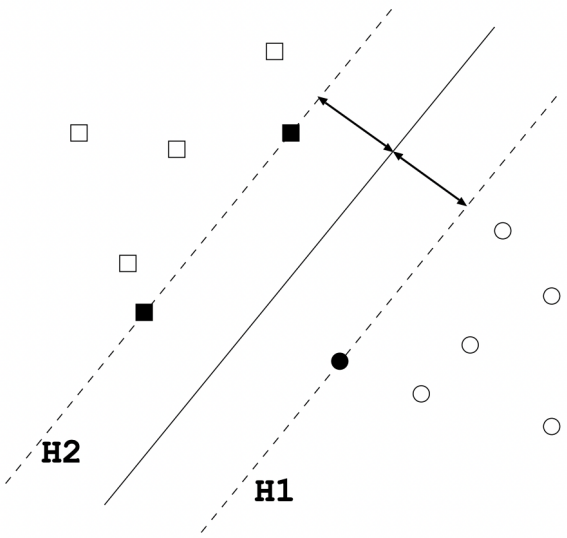

図. 4.1: 実線と点線の間がマージン、●・■はサポートベクター[20]

線形分離するSVM　ここで線形な識別関数を持つSVMについて考える。このとき訓練データは線形分離可能とする。識別関数は

$$y(\mathbf{x}) = sign(\mathbf{w}^{\mathrm{T}}\mathbf{x} - h) \tag{4.1}$$

と表せる。ここで$sign$は引数の符号を返す符号関数である。訓練データ点$\mathbf{x}_i$に対応するラベルを$l_i \in \{-1, 1\}$とすると、H1 : $\mathbf{w}^{\mathrm{T}}\mathbf{x}_i - h = 1$と H2 : $\mathbf{w}^{\mathrm{T}}\mathbf{x}_i - h = -1$の2枚の超平面で訓練サンプルが完全に分離されている様なパラメータが存在する。識別平面$y(\mathbf{x})$とこれらの超平面との距離がマージンであり、これを最大化する$\mathbf{w}, b$を求める最適化問題を解くと、線形識別関数を構成するための$\mathbf{w}, b$は上の図のH1、H2上に存在するデータ点にのみ依存することがわかる。この様なデータ点$\mathbf{x}_i$のことを「サポートベクター」と呼ぶ。図からもわかるように、これは元々の訓練データに比べて少ない。SVMはこのサポートベクターを用いて線形識別関数のパラメータを決定し、それを用いて予測を行う。このように、訓練データ点を予測にも用いる手法を、カーネル法と呼ぶ。カーネル法では、学習は高速に行うことができる反面、テスト点に対する予測には時間がかかる傾向がある[16]。

非線形分離への拡張　元の特徴ベクトル$\mathbf{x}$を非線形の写像によって変換し、その空間で線形分離を行うことで、先に示した線形分離するSVMを非線形へと拡張することができる。これは識別関数4.1に、

$$y(\mathbf{x}) = sign(\phi(\mathbf{w})^{\mathrm{T}}\mathbf{x} - h) \tag{4.2}$$

のように固定された特徴空間変換関数ϕを導入することで達成できる。なお、二つの入力ベクトルを特徴空間変換関数によって変換したものの内積はカーネル関数として知られ、

$$k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^{\mathrm{T}}\phi(\mathbf{x}') \tag{4.3}$$

と表される。
カーネルには様々なものがあるが、しばしば計算が容易なものが選ばれる。代表的なものにGauss(RBF)カーネル

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(\frac{-||\mathbf{x} - \mathbf{x}'||^2}{\sigma^2}\right) \tag{4.4}$$

がある。

4.3　データへのフィッティング

アルゴリズムはsklearn[21]のものを使った。主なハイパーパラメータにはC(正則化項の係数の逆数)、カーネルの種類があり、訓練とバリデーションデータを使い、

Optuna[18]によってこの二つのハイパーパラメータを探索した。Cは$[0.01, 100]$の間の対数スケールの空間からサンプルし、カーネルはRBFカーネルとsigmoidカーネルのどちらかをサンプルした。このとき、探索のためにaccuracyを用い、これを最大化するようにした。試行回数は100回とした。
この結果、以下のパラメータが選択された。

$$\text{'kernel'} : \text{'rbf'}$$

$$\text{'C'} : 62.0$$

選択されたパラメータを使ってテストデータを予測すると、accuracyは0.511であった。下の図4.2はテストデータの50日分とその予測値をプロットしたものである。

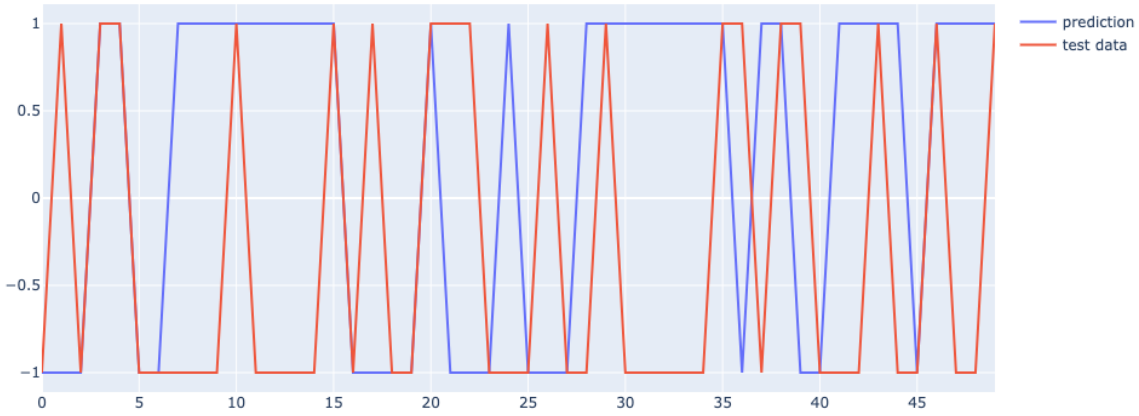

図. 4.2

5　ニューラルネットワーク

本章では2009年以降、機械学習の分野において再び脚光を浴びているニューラルネットワークを取り上げる。まず最も単純なフィードフォワード型から回帰型へと拡張し、そこから2017年以降、株価の予測研究において主流となっている長・短期記憶型ニューラルネット(LSTM)と呼ばれるモデルを使ったモデリングを行う。

5.1　モデルの系譜学

「ニューラルネットワーク」という言葉は、生体システムにおける情報処理を数学的に表現しようという試みにその起源がある[16]。そのためニューラルネットワークとは、人間の神経細胞をグラフのエッジとノードとしてモデリングした”ニューロン”の構造体（ネットワーク）であるものの、ここでは統計的パターン認識のためのモデルとしてのニューラルネットワークを扱う。ニューラルネットワークの発展には主に三つの波があり、最初は1960〜70年代、続いて1980〜90年台、そして2006年に始まり、現在も続いている波がある。現在の波では、「ディープラーニング」と呼ばれる、近年の機械学習アルゴリズムおよびハードウェアの進歩によって可能になった、隠れ層を増やしたニューラルネットワークが主流である[22]。

5.2　モデルの概要

5.2.1　フィードフォワード型について

フィードフォワードネットワーク　ここでは、ニューラルネットワークのもっとも単純な構造であるフィードフォワードネットワークについて書いていく。

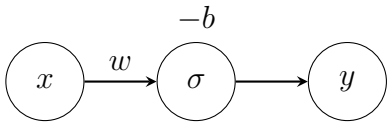

図. 5.1: 単純パーセプトロン

上の図 1 ではニューラルネットワークの最も単純な例である単純パーセプトロンを示している。ここではxを入力、wを重み、bを閾値、σを非線形活性化関数、出力をyとしている。入力値は一つのノードからエッジを介して次のノードに移る際に、エッジの重みを掛けられ、閾値が引かれて、続いて非線形活性化関数に通されて出力される。このネットワークの関数は、

$$y = \sigma(wx - b) \tag{5.1}$$

と表される。また、ここで$w_1 = w,\ w_0 = -b,\ x_0 = 1$とすると、これは

$$y = \sigma(wx + w_0 x_0) = \sigma\left(\sum_{j\in\{0,1\}} w_j x_j\right) \tag{5.2}$$

と表される。
なお、

$$a = \sum_{j\in\{0,1\}} w_j x_j \tag{5.3}$$

は活性と呼ばれる。

単純パーセプトロンの非線形活性化関数は以下の図2のようなステップ関数となっており、人間のニューロンの活性化を模倣していることがわかる。

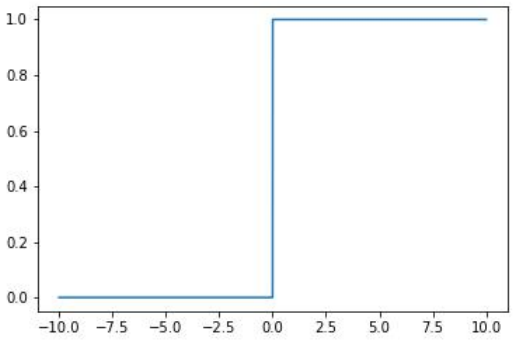

図. 5.2: ステップ関数

これに対し、近年機械学習の文脈で用いられるニューラルネットワークでは図3のような連続な非線形活性化関数が用いられている。

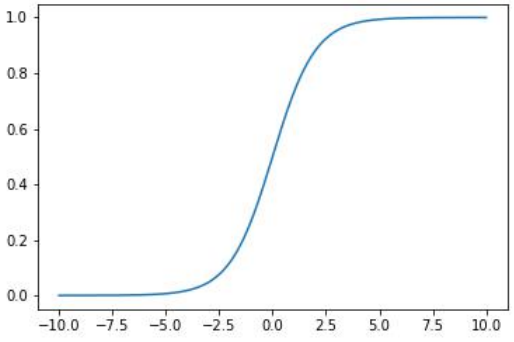

図. 5.3: ロジスティックシグモイド関数

このことは、ニューラルネットワークによる関数はネットワークパラメータに関して微分可能であることを意味しており、この性質がネットワークの訓練において中心的な役割を果たしている。図5.1にあるようなニューロン構造をいくつもつなげたものが図5.5で示すような一般的なフィードフォワードネットワークである。

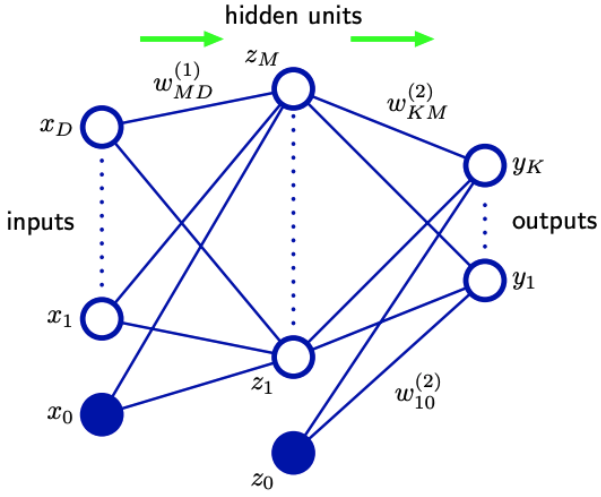

図. 5.4: フィードフォワードネットワーク（Bishop, 2006, p.228)[16]

先ほどの単純パーセプトロンにおける重み、閾値はここではそれぞれ重みパラメータ、バイアスパラメータに相当する。また、式5.2と同様に、ここでバイアスパラメータは、値が$x_0 = 1, z_0 = 1$と固定された入力変数x_0, z_0を追加することで、重みパラメータの集合の中に含めている。これによって、ネットワーク全体の関数は

$$y_k(\mathbf{x}, \mathbf{w}) = \sigma\left(\sum_{j=0}^{M} w_{kj}^{(2)} h\left(\sum_{i=0}^{D} w_{ji}^{(1)} x_i\right)\right) \tag{5.4}$$

となる。なお、hは隠れ層の活性化関数で、σは出力層の活性化関数である。ここでは隠れ層が一層であるため、一般的には 1 隠れ層ネットワークなどと呼ばれるが、この隠れ層を何層にも増やすことでニューラルネットワークを「ディープ」にすることができ、これが近年至る所で耳にする「ディープラーニング」である。

パラメータ推定法　ニューラルネットワークの学習は、出力層の活性化関数が返す値と損失関数との差を最小にする最適化問題として定式化される。
二クラス分類問題には、しばしば出力層の活性化関数にロジスティックシグモイド関数を使い、単一の出力を持つネットワークが利用される。また、損失関数は交差エントロピー関数がよく使われるため、ここでもそれに倣う。なお、ここでもクラスC_1, C_2はそれぞれ次の日に株価が上がる、上がらないという事象に対応している。

ロジスティックシグモイド関数は

$$\sigma(a) = \frac{1}{1 + \exp(-a)} \tag{5.5}$$

と表され、これは図5.3のシグモイド関数と同じである。したがって、日にちtにおける観測データを$\mathbf{x}_t$、ネットワークの重みパラメータを$\mathbf{w}$とすると、シグモイド関数の値域からネットワークの出力は $0 \leq y(\mathbf{x}_t, \mathbf{w}) \leq 1$ となり、これは $p(C_1|\mathbf{x}_t)$ に対応すると考えることができる。また、これによって $1 - y(\mathbf{x}_t, \mathbf{w})$ は $p(C_2|\mathbf{x}_t)$ に対応づられる。

以上より、入力$\mathbf{x}_t$と正解ラベル $l \in \{1, 0\}$ が与えられたときの目標の条件付き分布（尤度関数）は

$$p(l|\mathbf{x}_t, \mathbf{w}) = y(\mathbf{x}_t, \mathbf{w})^l \{1 - y(\mathbf{x}_t, \mathbf{w})\}^{1-l} \tag{5.6}$$

となり、負の対数をとると、

$$E(\mathbf{w}) = -\{l \log(y(\mathbf{x}_t, \mathbf{w})) + (1 - l) \log(1 - y(\mathbf{x}_t, \mathbf{w}))\} \tag{5.7}$$

という形の交差エントロピー誤差関数になる。目標は式5.5の最大化であるので、交差エントロピー誤差関数を最小化することによってこれを達成する。重み空間内で$\mathbf{w}$から$\mathbf{w} + \delta\mathbf{w}$へ少しだけ移動すると、誤差関数の変化は$\delta\mathbf{w}^{\mathrm{T}} \nabla E(\mathbf{w})$であり、したがって$E(\mathbf{w})$ が最小値となる$\mathbf{w}$では$\nabla E(\mathbf{w}) = 0$が成り立つ。しかし、ニューラルネットにおいて方程式$\nabla E(\mathbf{w}) = 0$を解析的に解くことはほとんど不可能なので、$\nabla E(\mathbf{w})$の値が小さくなる方向$(-\nabla E(\mathbf{w}))$へ重み$\mathbf{w}$を重み空間内で

$$\mathbf{w}^{\tau+1} = \mathbf{w}^{\tau} - \eta \nabla E(\mathbf{w}^{\tau}) \tag{5.8}$$

という形で動かす。ここでτは反復ステップ数を表している。また、$\eta > 0$は学習率パラメータとして知られている。現在広く使われている勾配降下アルゴリズムはこのアルゴリズムの拡張となっている。なお、ニューラルネットワークの複雑な構造から、$\nabla E(\mathbf{w})$を効率よく評価する必要がある。これは誤差逆伝播法と呼ばれる手法によって可能である。誤差逆伝播法は、各ユニットの誤差を、層の間を出力から入力の方向へと伝播する。誤差を伝播させるときに、微分の連鎖法則を用いて各ユニットの誤差を計算する。

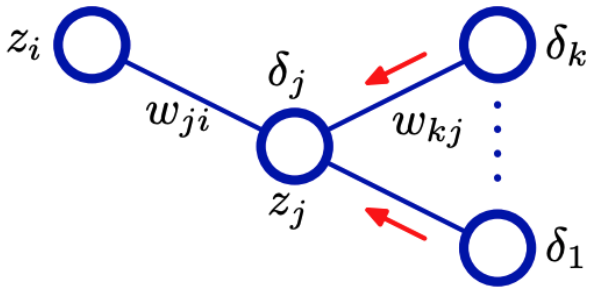

図. 5.5: 誤差逆伝播(Bishop, 2006, p.244)[16]

活性化関数をh、活性をaとしたとき、図5.5のようなネットワークに対する一般的な誤差逆伝播法のアルゴリズムは以下の通りである[16]。

1. 入力ベクトル$\mathbf{x}_n$をネットワークに入れ、ネットワーク上を準伝播させ、全ての隠れユニットと出力ユニットの出力を求める。
2. 全ての出力ユニットの誤差δ_kを評価する。
3. 逆伝播公式$\delta_j = h'(a_j)\sum_k w_{kj}\delta_k$を使って$\delta$を逆伝播させ、ネットワークの全ての隠れユニットの$\delta_j$を得る。
4. δ_jを使って$\frac{\partial E}{\partial w_{ji}}$を評価する。

何層もあるニューラルネットワークではこのアルゴリズムがその全体に適用され、$\nabla E(\mathbf{w})$が求められる。このの$\nabla E(\mathbf{w})$を5.8に代入することで勾配降下を行う。この二つのステップが何度も繰り返されることでネットワークのパラメータが学習される。

5.2.2 回帰型について

系列ベースのタスク、言語、時系列予測、時系列分類において、最もよく成功したモデルの多くは回帰型ニューラルネットワーク(RNN)と呼ばれるアーキテクチャ群に含まれる。
本小節ではRNNと、その拡張であるLSTMについて書いていく。

NNをRNNに拡張する　前小節の1隠れ層フィードフォワードニューラルネットワーク(FFNN)は、出力層が1ユニットの場合、下の図のようになる。

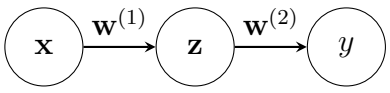

図. 5.6: 1隠れ層、1出力ユニットのFFNN

入力を系列データとし、その長さをIとする。I回に分けて系列データの値を入力として取り込み、その都度隠れ層に同じパラメータ$\mathbf{W}_z^{(1)}$を適用し、その後に出力層から値が出力されるモデルを考える。これが回帰型ニューラルネットワーク(RNN)と呼ばれるモデルである。これは下の図のように表される。

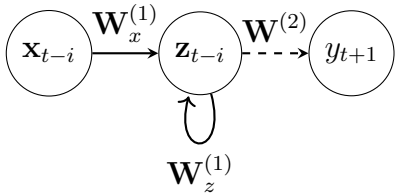

図. 5.7: 1隠れ層、一出力ユニットのRNN

これはまた

$$\begin{aligned} y_{t+1} &= \sigma^{(2)}(\mathbf{W}_z^{(2)}\mathbf{z}_t) \\ \mathbf{z}_{t-i} &= \sigma^{(1)}(\mathbf{W}_z^{(1)}\mathbf{z}_{t-i-1} + \mathbf{W}_z^{(1)}\mathbf{x}_{t-i}), \quad i \in \{I-1, ..., 0\} \end{aligned} \tag{5.9}$$

と表される。なお、重さパラメータに組み込んだバイアスパラメータのために$x_0 = 1 \in \mathbf{x}_{t-i}, z_0 = 1 \in \mathbf{z}_{t-i-1}$である。入力系列の長さが4のとき、これは入力系列に対して下の図5.8のように展開される。

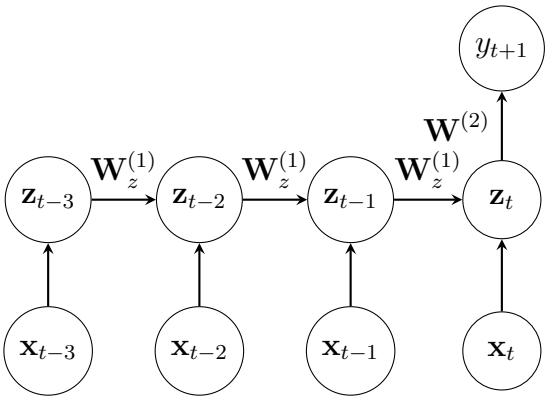

図. 5.8: 時間展開されたRNN

RNNとFFNNの違いは以下の通りである[13]。

- RNNはタイムステップを順番に一つずつ見ていく。
- RNNには、RNNが一つのタイムステップから次のタイムステップにかけて保存する状態($\mathbf{z}_{t-i}$)があり、この状態とその静的パラメータこそが、各タイムステップでの各新情報に対する応答の更新を決定する。

- RNNには、隠れ状態をはじめとする状態を、あるタイムステップから次のタイムステップに更新するパラメータ郡($\mathbf{W}_z$)がある。

一般化された自己回帰モデルとしてのRNN　伝統的な時系列モデルである自己回帰モデルは線形性や弱定常性などの強い制約を持つモデルであるが、未来のデータは過去のデータに依存するという、予測可能な時系列データの基本的な特徴をモデリングしたものである。RNNは自己回帰モデルの、非線形性をモデリングするような一般化と捉えることができる。また、入力ベクトルの長さをp、隠れユニットが一つ、活性化関数を持たない非常に単純なRNNはオーダーpの自己回帰モデルAR(p)であると示すことができる[23]。

5.2.3　長・短期記憶型について

RNNとFFNNの構造的な違いから、RNNはFFNNとは異なる方法でパラメータの更新を行う。RNNのパラメータ更新方法はBPTT（Back Propagation Through Time)と呼ばれており、これは通時的誤差逆伝播法とも知られている。BPTTは、RNNを時間展開した図5.8のようにRNNをFFNNのような構造として捉え、誤差逆伝播を適用する。しかしこの時、式5.8のようにある時点の隠れ層がその一つ前の隠れ層を使って回帰式で表されたのと同様に、ある時点での誤差e_{t-i-1}はタイムステップが一つ先の誤差e_{t-i}によって表される。
これによって入力系列が長くなればBPTTにおける勾配も発散・消失しやすく、RNNが長い系列データを扱うことを困難にしている。

この問題を解消するために、隠れ層とは別に過去の層からの出力を取り入れるセルと呼ばれるユニットc_tを追加し、これと隠れ層の前後に情報の流れを制御するための4つのゲートを追加したネットワークアーキテクチャが設計された。これは長・短期記憶型ニューラルネットワーク（LSTM）と呼ばれる。
LSTMの4つのゲートはそれぞれ入力ゲート(input gate)、出力ゲート(output gate)、忘却ゲート(forget gate)、セルゲート(cell gate)と呼ばれる。一般的なLSTMの隠れ層を構成する、2つのユニットと4つのゲートからなるLSTMブロック一つは図5.9のようになる。なお、この図でh, gはtanhを表し、σはロジスティックシグモイド関数(式5.5)を表している。

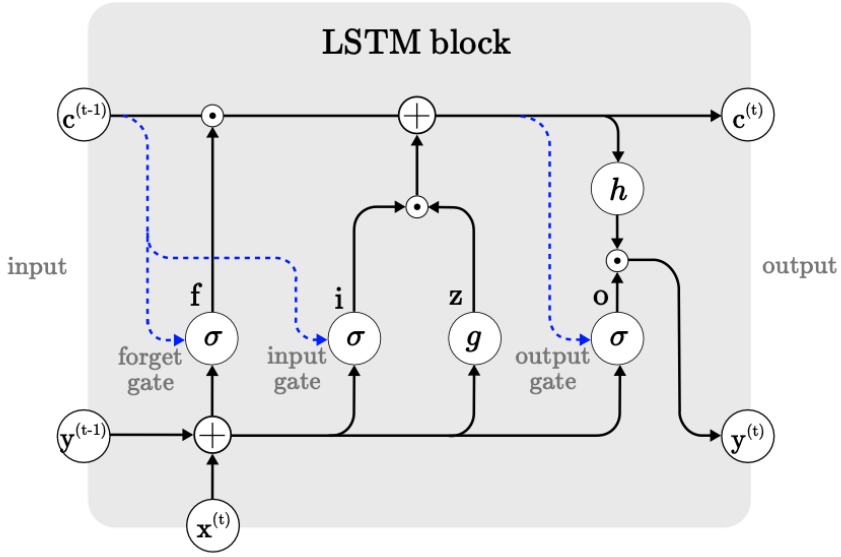

図. 5.9: LSTMブロック[24]

入力ゲート$i^{(t)}$、出力ゲート$o^{(t)}$、忘却ゲート$f^{(t)}$、セルゲート$z^{(t)}$はそれぞれ

$$i^{(t)} = \sigma(W_{ii}x^{(t)} + b_{ii} + W_{yi}y^{(t-1)} + b_{yi}) \tag{5.10}$$

$$o^{(t)} = \sigma(W_{io}x^{(t)} + b_{io} + W_{yo}y^{(t-1)} + b_{yo}) \tag{5.11}$$

$$f^{(t)} = \sigma(W_{if}x^{(t)} + b_{if} + W_{yf}y^{(t-1)} + b_{yf}) \tag{5.12}$$

$$z^{(t)} = \tanh(W_{iz}x^{(t)} + b_{iz} + W_{yz}y^{(t-1)} + b_{yz}) \tag{5.13}$$

と表され、隠れ層$y^{(t)}$、セルユニット$c^{(t)}$、はそれぞれ

$$y^{(t)} = o^{(t)} \odot \tanh(c^{(t)}) \tag{5.14}$$

$$c^{(t)} = f^{(t)} \odot c^{(t-1)} + i^{(t)} \odot z^{(t)} \tag{5.15}$$

と表される。なお、Wは重みパラメータ、bはバイアスパラメータで、$\odot$は要素ごとの積である。

モデルの特性　式5.12において$f^{(t)} = 0$とすると、LSTMのセルユニットに入る情報は毎度リセットされることになり、このときLSTMは普通のRNNとして振る舞う。また、5.11において$o^{(t)} = 0$とすると、LSTMの回帰的な構造がなくなり、FFNNとして振る舞う。したがって、通常のLSTMはRNNとFFNNの中間のような存在であると捉えることができる。

LSTMは長期記憶をセルに保管し、そのコピーを隠れ状態の更新に使うが、セルの情報もまた忘却ゲートによってリセットされ得る。このようにしてLSTMは長期的データと短期的データを、その重要度に応じて使用することができる。この特徴から、LSTMは様々な系列データを良くモデリングすることができている。

なお、RNNと同様にLSTMは同じ重み・バイアスパラメータを繰り返し使用するため、効率のいいモデルとなっている。

LSTM(と、広くニューラルネットワーク)はデータの特徴を学習することに長けている。

5.3 データへのフィッティング

使用したモデルの構造 先述の通り、現在のニューラルネットワークの設計において、精度を上げるために隠れ層を増やしてネットワークを「ディープ」にすることはスタンダードな手法であり、一般的に隠れ層を増やすことで精度が上がることがわかっている[25]。株価予測のためにLSTMを使った先行研究でも同様のアプローチをとっているもの[26]がる。そこで、本研究では隠れ層と隠れ層のベクトルの次元を変えていくことによって最適なネットワーク構造を探索した。隠れ層のベクトルの次元は4, 16, 64, 200と変えていき、隠れ層の数は1, 2, 3と変えていった。活性化関数は5.2.1のパラメータ推定法の段落で述べた考えをもとに、ロジスティックシグモイド関数を用いた。

ロジスティックシグモイド関数を用いたことによってLSTMの出力は連続となり、したがって3章の勾配ブースティング決定木におけるハイパーパラメータ探索と同様にArea Under Receiver Operator Curve(AUC)をモデル選択指標として使った。なお、モデルのスコープは50日であったことから、LSTMの入力系列の長さは50とした。

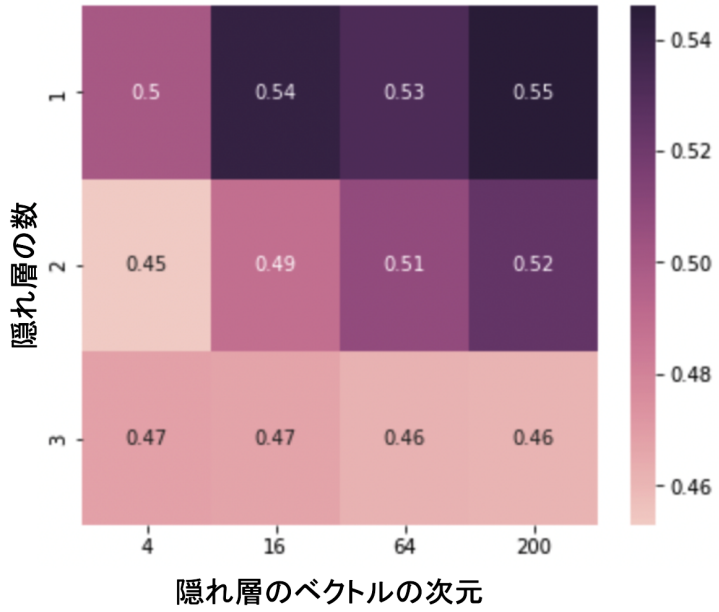

図. 5.10: LSTMの構造探索

上の図5.10は隠れ層の数を増やすことよりも、隠れ層のベクトルの次元を増やすことの方が精度が上がることを示しており、また、隠れ層の数を増やすことで精度が下がることがわかった。最も良かったモデルは隠れ層の数が1、隠れ層のベクトルの次元が200のときであった。このときのモデルの最大accuracyは0.552であった。また、選ばれたパラメータを使ってテストデータの最初の50日について予測をした結果、以下のようになった。

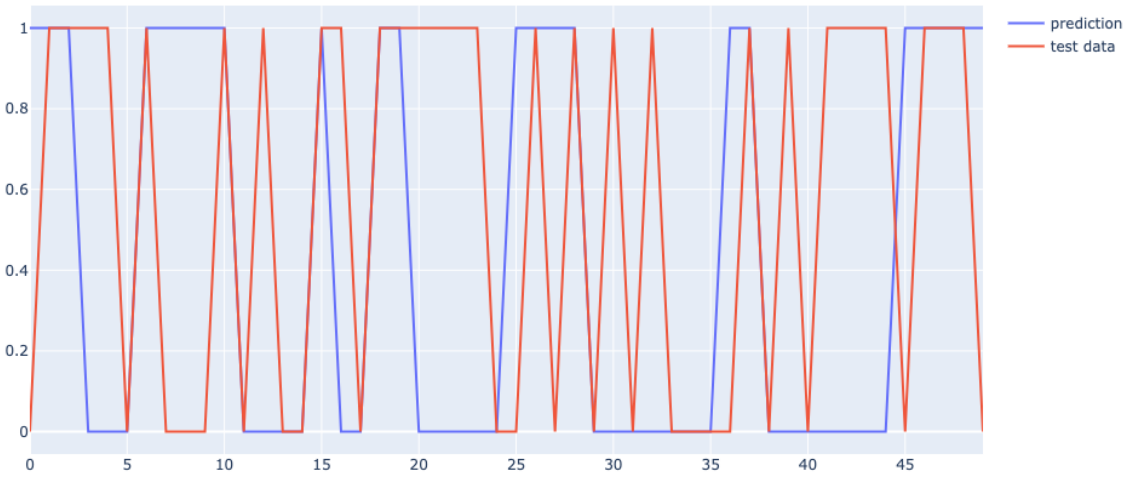

図. 5.11

6　精度の比較と考察

3章から5章のそれぞれのモデルの予測結果は以下の図6.1のようになり、ニューラルネットワークの精度が一番いいことがわかった。

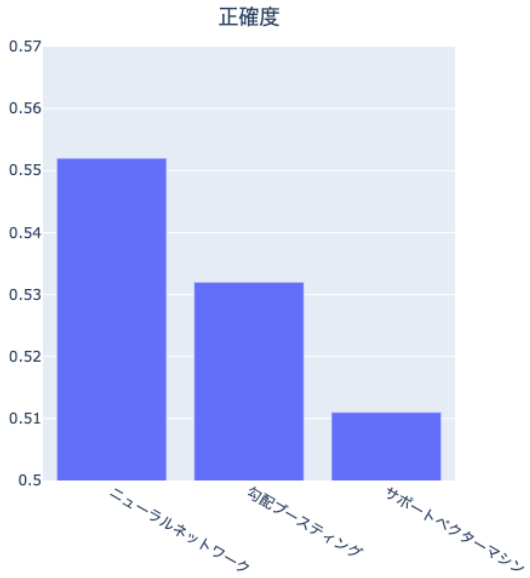

図. 6.1

また、それぞれのモデルの特徴量重要度をは以下の図6.2

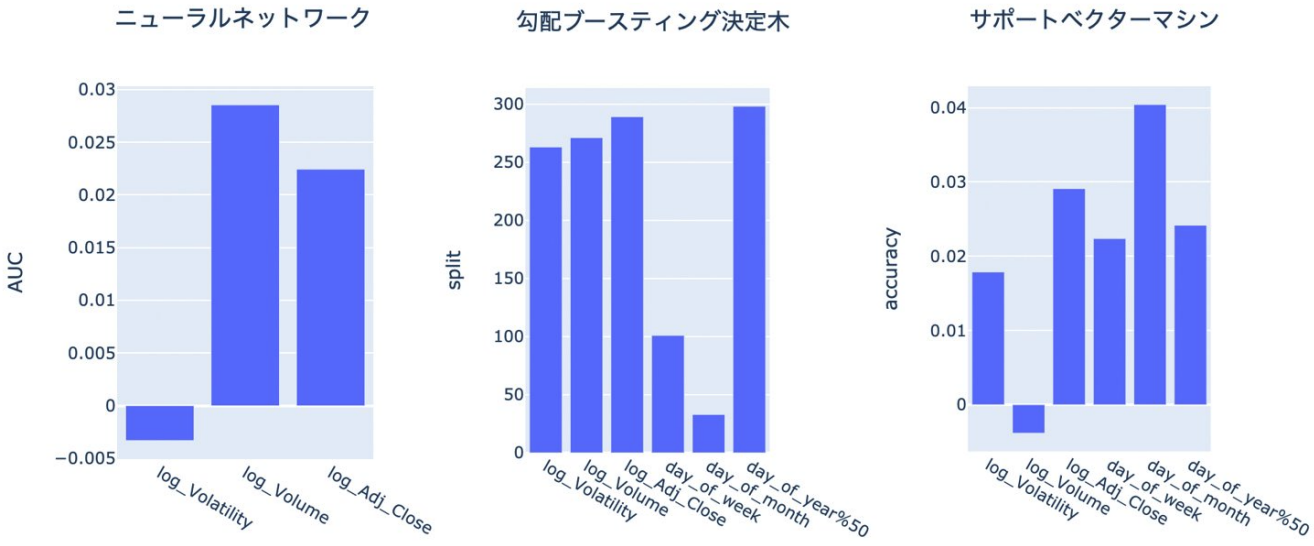

図. 6.2

のようになり、これと図6.1を比較すると、出来高の重要度は、ニューラルネットワークでは一番高く、勾配ブースティング決定木では三番目に高く、サポートベク

ターマシンでは一番低い。従って、精度の高いモデルほど、log_Volumeすなわち出来高の重要度が高いことがわかる。

株式市場では古くから「出来高は株価の燃料である」という格言がある。これは、出来高の動きの後に株価が動くということである。例えば買い手が非常に多いと出来高も大きく、このときは株の価値が上がり、値段も上がる。逆に売り手が非常に多いときにも出来高は大きくなる傾向があり、このときは株価の価値が下がり、値段も下がる。

今回高精度だったモデルは出来高の重要度が高かったため、このような株と出来高の関係性を利用して予測を上手く行うことができたと考えられる。また、サポートベクターマシンにおいては出来高の重要度が負になっており、これは出来高の重要性を全く学習できなかったことを意味している。

精度が比較的良かったニューラルネットワークと勾配ブースティング決定木を比較をすると、ニューラルネットワークは過去の収益率データよりも出来高に注目しており、特徴学習能力が活かされていることがわかる。これに対して、勾配ブースティング決定木では重要度の高い4つの特徴量の間には大きな差が見られない。勾配ブースティング決定木モデルのハイパーパラメータを見ると、モデルの複雑さを抑えるハイパーパラメータの値(lambda_l2など)が大きく、モデルが十分に特徴量の重要性の違いを学習していない可能性が考えられる。

なお、ニューラルネットワークにおいて隠れ層のベクトルの次元を増やすと精度が上がったのは、隠れ層のベクトルの次元を増やすことによってモデルの表現力が高まり、より些細な関係性も記述できるようになったからであると考えることができる。しかし、隠れ層を増やしても精度が上がらず、下がってしまう理由については明らかにできなかった。

7　まとめと展望

まとめ　本研究では勾配ブースティング決定木、サポートベクターマシン、回帰型ニューラルネットワークの一種であるLSTMを、先行研究の代表的かつ互いに性質の異なるベースモデルとして選択し、それらの精度を比較した。この結果、LSTMが一番いい精度であり、複雑な関係性を持つ株式データに対して、ニューラルネットワークの特徴である特徴学習能力の高さが発揮された。特徴量重要度とモデル精度の比較により、精度の高いモデルは株式市場で古くから知られている、出来高と株価の関係性を反映していることがわかった。また、LSTMは隠れ層を増やすことよりも隠れ層のベクトルの次元を増やした方が精度が上がり、これは近年のニューラルネットワークに関する知見に反する結果となってしまった。しかし、この機構を明らかにすることはできなかった。

展望　本研究で株価予測に対して最適なベースモデルはLSTMであることが明らかになったため、このアーキテクチャの様々な拡張を試したり、特徴量を増やしたりすることで精度をあげることができると考えられる。
アーキテクチャの拡張を使った例として、Khaledら[27]はある時点の過去のデータだけでなく、未来のデータも使用して学習するLSTM(Bidirectional-LSTM)を使用し、株価の予測精度が上がったと報告している。

また、本研究ではこれらのモデル使って実際に株価予測を行っていないため、果たして実際に利益が出るのかは明らかにできていない。したがってこれらのアルゴリズムを使って株を売買した時に実際に利益が出るのかを調べる必要がある。

本研究は短期的な取引に焦点を当てていたが、長期的な取引に焦点を当てて再度それぞれのモデルを比較するとどのような結果が得られるかを調べることも考えられる。

本研究で明らかにできなかった、LSTMの隠れ層の数と隠れ層のベクトルの次元の関係性についてであるが、今回使ったLSTMは入力系列の長さが50と短く、より長い系列長のデータを与えたり、より多くの特徴量を与えるとこの関係性がどうなるか調べる必要がある。

8　謝辞

最後に、指導教員としてご指導いただいた名古屋大学大学院情報学研究科複雑系科学専攻多自由度システム情報論講座の時田恵一郎教授、研究内容の助言をしていただき、研究の仕方について教えていただいた時田研究室の先輩方、情報学部の同級生に感謝を申し上げます。そして、学業を支えてくれた家族(スティーブン、ヒメナ、アルバ、マジパン、マプチェ、マテオ)に感謝を申し上げます。

参考文献

[1] Scott E Stickel and Robert E Verrecchia. Evidence that trading volume sustains stock price changes. *Financial Analysts Journal*, 50(6):57–67, 1994.

[2] Yaser S Abu-Mostafa and Amir F Atiya. Introduction to financial forecasting. *Applied intelligence*, 6(3):205–213, 1996.

[3] Eugene F Fama. The behavior of stock-market prices. *The journal of Business*, 38(1):34–105, 1965.

[4] Burton G Malkiel. The efficient market hypothesis and its critics. *Journal of economic perspectives*, 17(1):59–82, 2003.

[5] Isaac Kofi Nti, Adebayo Felix Adekoya, and Benjamin Asubam Weyori. A systematic review of fundamental and technical analysis of stock market predictions. *Artificial Intelligence Review*, 53(4):3007–3057, 2020.

[6] Shubharthi Dey, Yash Kumar, Snehanshu Saha, and Suryoday Basak. Forecasting to classification: Predicting the direction of stock market price using xtreme gradient boosting. *PESIT South Campus*, 2016.

[7] Rebwar M Nabi, Saeed Soran Ab M, and Habibollah Harron. A novel approach for stock price prediction using gradient boosting machine with feature engineering (gbm-wfe). *Kurdistan Journal of Applied Research*, 5(1):28–48, 2020.

[8] Yuling Lin, Haixiang Guo, and Jinglu Hu. An svm-based approach for stock market trend prediction. In *The 2013 international joint conference on neural networks (IJCNN)*, pages 1–7. IEEE, 2013.

[9] David MQ Nelson, Adriano CM Pereira, and Renato A de Oliveira. Stock market's price movement prediction with lstm neural networks. In *2017 International joint conference on neural networks (IJCNN)*, pages 1419–1426. IEEE, 2017.

[10] Khaled A Althelaya, El-Sayed M El-Alfy, and Salahadin Mohammed. Evaluation of bidirectional lstm for short-and long-term stock market prediction. In *2018 9th international conference on information and communication systems (ICICS)*, pages 151–156. IEEE, 2018.

[11] Kai Chen, Yi Zhou, and Fangyan Dai. A lstm-based method for stock returns prediction: A case study of china stock market. In *2015 IEEE international conference on big data (big data)*, pages 2823–2824. IEEE, 2015.

[12] John M Bates and Clive WJ Granger. The combination of forecasts. *Journal of the Operational Research Society*, 20(4):451–468, 1969.

[13] Aileen Nielsen. *Practical Time Series Analysis*. O'Reilly Media, 2019.

[14] Yoav Freund, Robert Schapire, and Naoki Abe. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612, 1999.

[15] G Sathyadevi. Application of cart algorithm in hepatitis disease diagnosis. In *2011 International Conference on Recent Trends in Information Technology (ICRTIT)*, pages 1283–1287. IEEE, 2011.

[16] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[17] lightgbm document. https://lightgbm.readthedocs.io/en/latest/Features.html. Accessed: 2022-1-30.

[18] optuna. https://optuna.org/. Accessed: 2022-2-1.

[19] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[20] 栗田多喜夫. サポートベクターマシン入門. 産業技術総合研究所脳神経情報研究部門. *July*, 2002.

[21] sklearn. https://scikit-learn.org/stable/. Accessed: 2022-2-1.

[22] Jeff Heaton. Ian goodfellow, yoshua bengio, and aaron courville: Deep learning, 2018.

[23] Matthew F Dixon, Igor Halperin, and Paul Bilokon. *Machine Learning in Finance*. Springer, 2020.

[24] Greg Van Houdt, Carlos Mosquera, and Gonzalo Nápoles. A review on the long short-term memory model. *Artificial Intelligence Review*, 53(8):5929–5955, 2020.

[25] 岡谷貴之. ディープラーニング. 映像情報メディア学会誌, 68(6):466–471, 2014.

[26] Slawek Smyl. A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. *International Journal of Forecasting*, 36(1):75–85, 2020.

[27] Khaled A Althelaya, El-Sayed M El-Alfy, and Salahadin Mohammed. Evaluation of bidirectional lstm for short-and long-term stock market prediction. In *2018 9th*

international conference on information and communication systems (ICICS), pages 151–156. IEEE, 2018.