人工知能による日経平均ボラティリティー・インデックスの予測

あすかアセットマネジメント株式会社
南正太郎

1　はじめに

高いリターンを追求すればリスクも高くなる。積極的に高いリターンを追求したいヘッジファンドにおいては、市場のリスク[*1]の大きさを知ることは重要な情報の一つである。なぜならば、ボラティリティーが大きい環境でこそ、マネージャの銘柄選定力やトレード技術などのスキルがアルファとして現れるかが試されるからである[*2]。市場のリスクの大きさを図る指標として、日本経済新聞社から日経平均ボラティリティー・インデックスが公表されており、これは将来の日経平均株価の変動の大きさがどのくらいなのかについて市場（投資家）がどのように考えているかを示したものである。また、大阪取引所では日経平均 VI 先物が提供されており、将来のボラティリティー水準に投資することも可能となっている。

本稿は資産運用への AI の応用[*3]として、日経平均ボラティリティー・インデックスの予測をディープラーニングと呼ばれる AI 手法の一つである LSTM-RNN を用いて行った結果をまとめたものである。

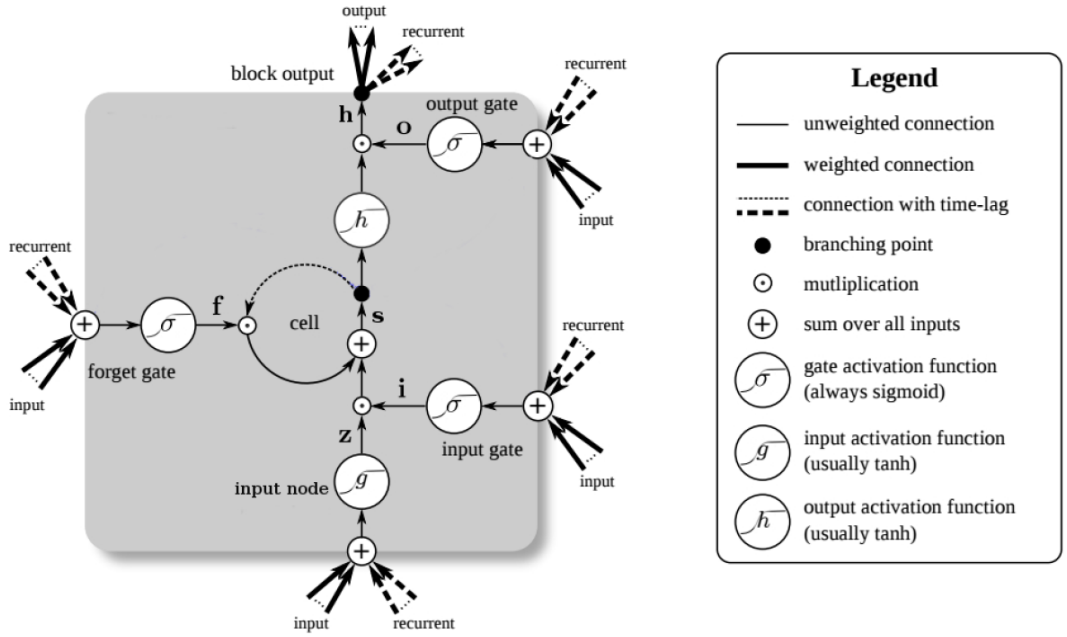
2　理論及び分析の枠組み

LSTM-RNN　人工知能分野におけるアルゴリズムの一つにニューラルネットワーク（Neural Network）がある。これは、人間の脳機能にみられる特徴（ニューロン (Neuron) と呼ばれる神経細胞間のネットワークを情報伝達する様）をモデル化したものである。ニューロンのネットワーク内では階層構造により情報が処理されており、階層を深くすることによって複雑なネットワーク構造をモデル化することができる。分類問題や回帰問題に応用できるニューラルネットワークであるが、時系列データを分析できるよう拡張した手法に再帰的ニューラルネットワーク（Recurrent Neural Network）がある。本稿で提案する LSTM-RNN（Long Short Term Memory Recurrent Neural Network) は、再帰的ニューラルネットワーク（Recurrent Neural Network) の一種であり、通常の再帰的ニューラルネットワークが持つ長期の時系列データを学習する際に発生する勾配問題を LSTM ブロックと呼ばれる複雑なニューロンを用いることで解決したものである。そのた

[*1] ここでのリスクは収益率の変動であるボラティリティーの大きさを意味する。

[*2] Juliana[1] では、代表的なロング・ショート・ファンドの平均エクスポージャと VIX Index の関係を見ており、歴史的に株式市場のボラティリティーが高い時にマネージャはネット・エクスポージャを低下させ、ボラティリティーが低下すると、ネット・エクスポージャを高めていることを示している。一般的なロング・ショート・ファンドのエクスポージャはロング・バイアスであり、株式市場のボラティリティーが高いときにネット・エクスポージャを低下させるのは、過度なリスクを取りたくないマネージャの心理が行動に表れているためとみられる。

[*3] 応用例として Minami[2] では、LSTM-RNN を用いてコーポレートアクションとプレスリリース情報を織り込んだ株価予測を行っている。



出所 Akash[4], Figure 2.4, P16

図1: LSTM ブロックの概略図

め、長期の時間依存も短期の時間依存も学習できる特性がある。

定式化　以下では、1997 年に発表された Hochreiter[3] に基づき、標準的な LSTM-RNN について定式化する。x_t を t 時点の入力値ベクトル、h_t を t 時点の出力値ベクトルとするとき、

$$
\begin{aligned}
z_t &= \tanh(W^z x_t + R^z h_{t-1} + b^z) && input \\
i_t &= \sigma(W^i x_t + R^i h_{t-1} + b^i) && inputgate \\
f_t &= \sigma(W^f x_t + R^f h_{t-1} + b^f) && forgetgate \\
o_t &= \sigma(W^o x_t + R^o h_{t-1} + b^o) && outputgate \\
s_t &= z_t \odot i_t + s_{t-1} \odot f_t && cellstate \\
h_t &= \tanh(s_t) \odot o_t && output
\end{aligned}
$$

ただし、

$$\sigma(x) = sigmoid(x) = \frac{1}{1+e^{-x}}$$

となる。$\odot$ は要素ごとの積、W^* は入力ウェイト、R^* はリカレントウェイト、b^* はバイアスウェイト、s_t は t 時点のメモリセルの状態ベクトル (cellstate) である。また、i_t は新しい情報を得るための重みベクトルである入力ゲート（inputgate）、f_t は古い情報を記憶する重みベクトルである忘却ゲート（forgetgate）、o_t は出力候補のベクトルを出す出力ゲート (outputgate) を表す。活性化関数のうち、$sigmoid(*)$ はシグモイド関数、$\tanh(*)$ は双曲線正接関数である。

予測評価方法　実運用での利用を想定すると、実績値とモデルによる予測値との間に乖離がないほうが望ましい。そこで、RMSE(Root Mean Squared Error）を使ってモデルの評価をする。$\hat{y}_t$ をモデルにより出力された予測値、y_t は実績値であるとき、

$$RMSE = \sqrt{\frac{1}{N}\sum(\hat{y}_t - y_t)^2}$$

となる。その他の評価尺度としては、MAE(Mean Absolute Error)= $\frac{1}{N}\sigma_{i=1}^{N} \parallel y_i - \hat{y}_i \parallel$ があるが、局所的な誤差の影響よりも大きな誤差に対する影響を見る方が適していると思われるため RMSE を用いている。

データについて　データは、２００１年１月から２０１８年１月までの日経平均ボラティリティー・インデックスの月末値を用いている。LSTM-RNN に学習させる前に次のデータ変換を行っている。まず、全体の９５％を訓練用データとし残りの５％をテスト用データに分けている。これは訓練用のデータを用いてモデルを作成し、モデル作成で使わなかったテスト用データにおいても予測できているのかを調べるためである。次に、現在の観測値と 1 期前の観測値との差分をとることで定常過程に変換している。これはトレンドなどの非定常な性質を持っていると望ましくない結果をもたらす恐れがあるためである。観測値からトレンドを排除した上で学習と予測を行い、出力された予測結果を元のスケールに戻すことで同等のエラースコアを求めることができる。次に、時系列データのスケールを合わせるために、-1 ～ 1 の範囲に変換している。LSTM ブロックの中で用いている活性化関数 (activation function) は、シグモイド関数もしくは双曲線正接関数であり、この活性化関数を用いて変数を使うか使わないかを判断することから、-1 ～ 1 の範囲であることが望ましいためである。スケーリングのための係数（最大値・最小値）は訓練用データから算出し、テスト用データと出力された予測値を元のスケールに戻すために用いている。

その他の条件　機械学習のフレームワークとしては、フロントエンドに Keras、バックエンドには Tensorflow を使用して実装[*4]している。確率的勾配降下法の最適化を行う上で、Adam(Adaptive Moment Estimation) により学習率を設定。また、GPU は NVIDIA GeForce GTX 1080 Ti 11GB を用いた。プログラミング言語は Python である。学習率とは、活性化関数の尤度関数を最大化するパラメータの収束しやすさを調整するパラメータである[*5]。分析に際しての設定条件として、隠れ層は 1 つの LSTM 層、Epoch 数を１５００回、ニューロン数を 4 個で計算している。精度を上げるためには隠れ層内のニューロンの数を増やすことや隠れ層の数そのものを増やすことが考えられる。隠れ層の数を増やすことで深い層の学習ができ、より複雑な分布を持つ時系列データの予測が可能となるが、過学習の問題を引き起こしたり、膨大な計算時間になる場合もある。よって、隠れ層は 1 つにしている。Epoch 数は繰り返しデータをシャッフルして学習する回数を意味している。確率的勾配降下法では最適なパラメータを探索によって探すため、学習の回数を設定する必要がある。

3　実証分析

図２は、テスト用データによる直近１２カ月の実績値とモデルによる予測値を表示したものである。比較対象として、1 期前の値を予測に用いた線形予測モデル（左図）を併せて表記している。LSTM-RNN モデルの

[*4] Brownlee[5] には keras と Tensorflow による基本的な実装の仕方が記載されている。

[*5] 実際には尤度関数の偏微分を計算する際にシグモイド関数などの活性化関数の尤度は関数が積の形となり計算が煩雑となるため、計算を簡単にするために、対数をとって符号を入れ替えることで交差エントロピー誤差関数に変換し、この関数を最小化することを行う。損失関数として呼ばれることもある。

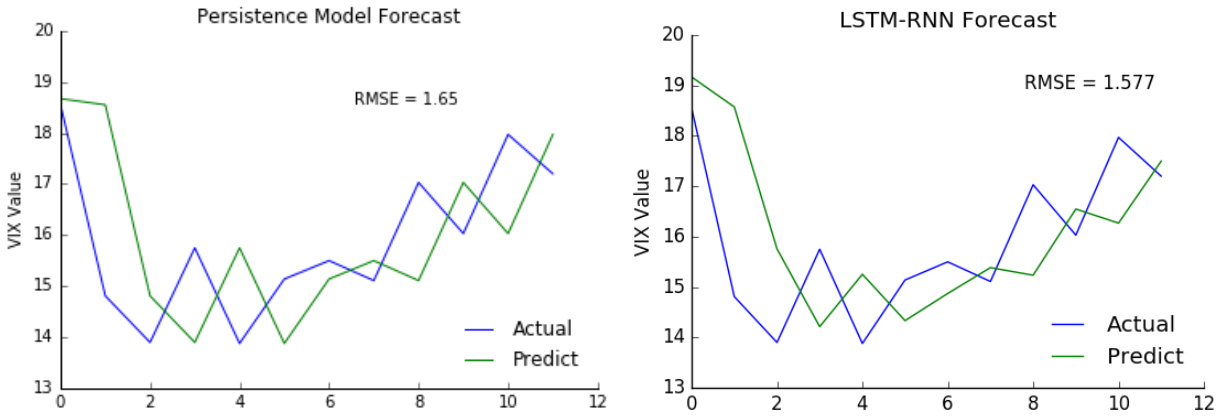


図2: 直近 12 カ月の予測と実績：線形予測モデル（左図）、LSTM-RNN モデル（右図）

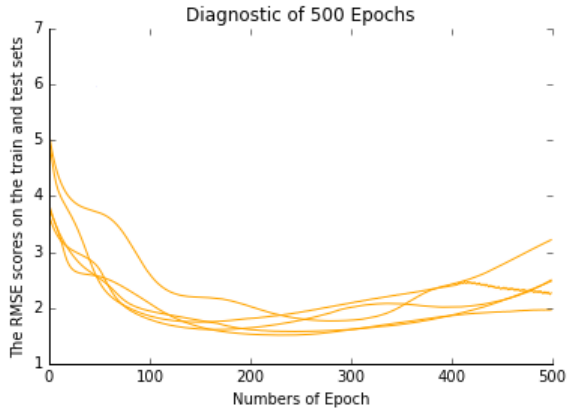


図3: Epoch 数別の RMSE

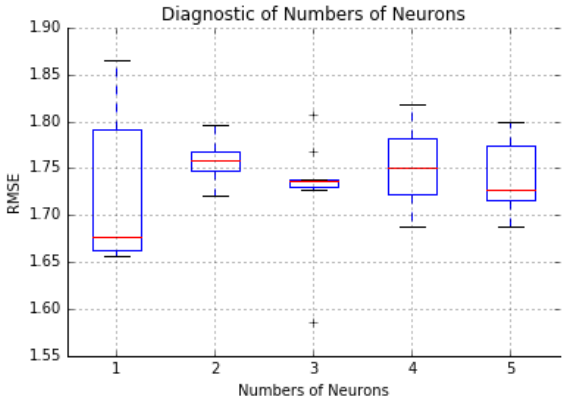


図4: ニューロン数別の RMSE：ボックスプロット

１２カ月の RMSE は 1.57 であり、線形予測モデルの 1.65 よりも予測誤差が小さいことから、LSTM-RNN モデルの方が予測ができている結果となった。時系列の振舞いを見ても、実績値との乖離は小さいように見える。ただし、注意しなければならないのは、ここでの結果は、上記で示した条件の下でたまたま良かった可能性がある。そこで、Epoch 数別の RMSE とニューロン数別の RMSE について計算を行った。図３は、縦軸は RMSE、横軸は Epoch 数である。全体の計算を５回繰り返し、各 Epoch 数時の RMSE をプロットした結果を示している。２００～３００ Epoch 数近辺まで低減したのち、それ以降は緩やかに増加している。これは、３００ Epoch 数を境に過学習が生じてしまっているためであり、効率的な学習としては２００～３００ Epoch 数で十分と言えそうである。図４は、ニューロン数別の計算結果を示したものである。効率化のため、３００ Epoch 数とし、各ニューロン数別に全体の計算を１０回繰り返したときの RMSE の結果をボックスプロットしたものである。最も RMSE が小さかったのはニューロン数が 1 の時であるが、最大値と最小値との差が最も大きいことから予測精度が高いこともあれば低いこともあるようだ。ニューロン数が 3 の時は、RMSE が安定している結果となったが、1.70 ～ 1.75 の範囲内と相対的には RMSE が低い結果となった。

最後に実運用を想定した場合の予測結果を示す。実務においては、例えば、当月末時点までのデータを用いて翌月末あるいは翌々月の予測を行うことを毎月繰り返すような使い方が想定されるだろう。図５は、線形予

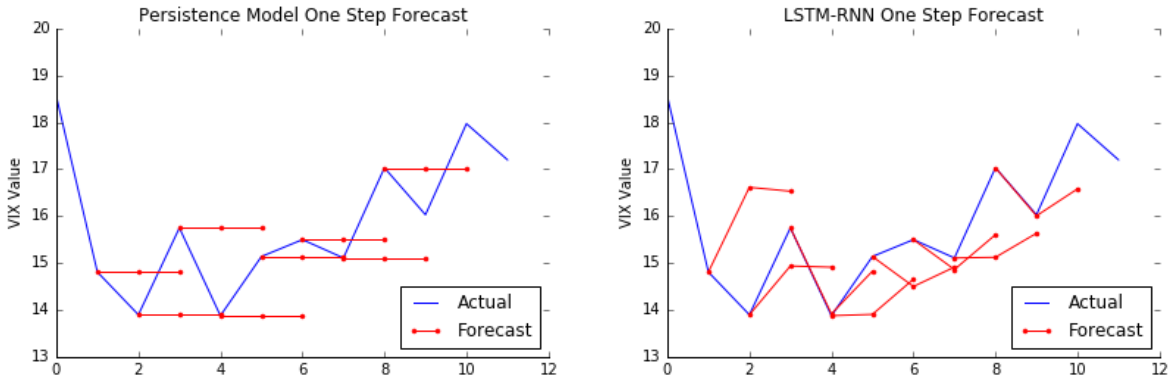


図5: 2 期先までの予測と実績：線形予測モデル（左図）、LSTM-RNN モデル（右図）

測モデルと LSTM-RNN モデルそれぞれについて、各時点ごとにその時点から 2 期先までの予測を示したもの（Forecast）である。線形予測モデルと比較すると、LSTM-RNN では予測の方向性がある様や 1 期先が上昇時に 2 期先は低下するといった予測を行っている様子が見てとれる。

4 おわりに

本稿では、ディープラーニングと呼ばれる人工知能の手法である LSTM-RNN を用いて日経平均ボラティリティー・インデックスの予測を行った。分析結果によれば線形予測モデルよりも精度が高くなる可能性は示唆されたものの、実運用に用いる上ではより精度が高く安定した結果をもたらすようなモデルのチューニングが必要であろう。Epoch 数やニューロンの数、今回は検証していないが隠れ層の数などを変えることでより精度の高い結果を得ることができるかもしれない。冒頭でも触れたように将来のボラティリティー水準について知ることはアルファにつながることから重要な情報である。予測精度が高まれば、先物取引により直接利益を獲得することもできるであろう。資産運用への AI の応用はまだ始まったばかりであり、本稿の検証を通じてより深めていきたいと思う。

参考文献

[1] Juriana Hadas, Andrea Pomoili. (2014) The Case for Long/Short Equity: Four Reasons to Include the Strategy in a Portfolio, NEUBERGER BERMAN, REPORT. https://www.nb.com/web/japan/pdf/20141106_E_The_Case_For_LS.pdf

[2] S.Minami. (2018) Predicting Equity Price with Corporate Action Events Using LSTM-RNN, Jounal of Mathematical Finance, 2018, 8, 58-63. http://file.scirp.org/pdf/JMF_2018013014134167.pdf

[3] Hochreiter, S.and Schmidhuber, J. (1997) Long Short-Term Memory. Neural computation,9,1735-1780. https://doi.org/10.1162/neco.1997.9.8.1735

[4] Akash Singh. (2017) Anomaly Detection for Temporal Data using Long Short-Term Memory(LSTM), Degree Project in Information and Communication Technology, Second Cycle, 30 Cred-

its, StockHolm, Sweden. http://www.diva-portal.org/smash/get/diva2:1149130/FULLTEXT01.pdf

[5] Brownlee, J. (2017) Long Short-Term Memory Networks with Python Develop Sequence Prediction Models with Deep Learning. Machine Learning Mastery, EBook.