F-011

電力供給量データを用いた時系列予測問題に対する各種機械学習手法及び提案手法の比較検証
Comparative verification of machine learning method to predict time series data

有村 和真† 新谷 洋人† 本木 実†
Kazuma Arimura Hirohito Shintani Minoru Motoki

1. はじめに

商品の売り上げ推移や株価、気温など時系列情報は様々な分野に存在し非常に多種多様である。その活用や予測は非常に重要度の高い技術とされており、ARIMA モデルや状態空間モデルといった様々な解析・予測手法が提案されてきた。

また、昨今様々な分野でニューラルネットワークによる目覚ましい成果が報告されている。しかし、画像処理の分野における Convolutional Neural Network（以下、CNN）の活躍や自然言語処理分野における Recurrent Neural Network[1]（以下、RNN）の活躍ほどの成果が時系列情報予測の分野ではあまり報告されていない。

本研究では予測手法としてニューラルネットワークに着目し、RNN や Quasi-Recurrent Neural Network[2]（以下、QRNN）、当研究室の提案手法である Time Shift Autoencoder（以下、TSA）やそれを多段に重ねたモデル Multiple Time Shift Autoencoder（以下、MTSA）及びそれらのアンサブルモデルを予測手法に選び、電力供給量データを用いた比較検証を行った。

2. 研究方法

2.1 予測手法

2.1.1 RNN

RNN は通常のニューラルネットワークに加え直前のデータの出力を再帰的に入力するモデルである。そのため連続的な情報を処理することに優れており、様々な手法が提案されてきた。今回はその中から最も単純なモデルである Simple Recurrent Network（以下、SRN）を用いた検証を行った。

2.1.2 QRNN

QRNN は RNN の性質上１データずつ処理しなくてはならないことに起因する並列性の悪さを克服するために CNN で擬似的に RNN を再現したモデルである。今回は RNN との性能・速度比較のために用いた。

2.1.3 TSA

本モデルは当研究室の提案手法であり次元削減の手法として用いられるニューラルネットワークの一種であるオートエンコーダーを時系列情報に応用したモデルである。入力データに対して出力データを時間軸方向にシフトするニューラルネットワークであり、その中間層は入力データ、時間軸方向にシフトされたデータの両方の特徴量を持ち合わせたオートエンコーダーになる。

†熊本高等専門学校 National Institute of Technology, Kumamoto College

2.1.4 MTSA

本モデルは前項の TSA を多段に重ね合わせたモデルである。前項に記述したように TSA の中間層は入力データと時間軸方向にシフトされたデータの特徴量を持ち合わせている。この特性を利用して中間層出力を学習するような TSA を多段に用意することにより時間軸方向にシフトされた中間層出力を得ることが出来る。この予測された中間層出力を前段にフィードバックすることにより、さらに離れた時刻のデータの予測が可能となる。

2.2 検証データ

今回これらの手法を比較検証を行うにあたり、関西電力のホームページで公開されている電力需要量データを用いた。3 分毎、2013/1/1 から 2016/11/27 までの約 4 年分の電力需要量のデータ（全 525600 点）で、そのうち前半 4 分の 4 を学習用データとして、後半 4 分の 1 を検証用データとして用いた。また、学習用データのうち 90％を train dataset、10％を validation dataset に分けて学習を行った。

3. 検証方法

3.1 検証環境

検証実験を行った環境は以下の表 1 の通りである。また、使用した言語、ライブラリのバージョンは以下の表 2 の通りである。

表１　実験環境

OS	CentOS Linux 7.5.1804
CPU	Intel Core i5-6600 CPU
GPU	GeForce GTX TITAN BLACK
言語	Python 3.5.2

表２　使用したライブラリ一覧

Tensorflow-GPU	1.4.0
Keras	2.1.0
numpy	1.13.3
pandas	0.21.0
scikit-learn	0.19.1
matplotlib	2.1.1

また、QRNN に関しては GitHub 上に公開されている Keras 向けのものを用いて実装した[3]。

3.2 検証方法

今回は検証として、1 点先のデータ予測と 20 点先のデータ予測の 2 種類を行った。また、データ全体で min-max 正規化を行った。20 点先を選んだ理由としては、電力需用量予測の分野では 1 時間先の予測が重要であり、今回の場合 1 点あたりの時刻が 3 分であるためである。

各ネットワークには連続した 64 点分の電力需要量データを入力し、入力データから任意の n 点だけ時間軸方向に進んだ 64 点を出力するように学習を行わせた。また、MTSA に関しては 1 点先を予測する TSA を 20 個用意し、1 段目の TSA では入力された電力需要量データを、2 段目以降の TSA では前段の TSA の中間層出力を学習させた。最終的な出力の算出には全 TSA を用い、前段の TSA に中間層出力の予測をフィードバックすることで任意の点数離れた出力を得ることが可能となる。

4. 検証結果

4.1 1 点先の予測

以下の表 3 に 1 点先の予測結果を示す。

表３　１点先予測結果

モデル名	平均誤差（%）	誤差最大（%）
RNN	0.799367	16.556952
QRNN	1.155152	12.415363
TSA	0.216696	4.590472
アンサンブル（TSA＋QRNN）	0.843280	4.665853

検証結果から 1 点先(3 分先)の予測では TSA が最も精度が高く予測を行えていることがわかった。また、その誤差はどれも 1%前後もしくはそれ以下であることから、これらのモデルを使って電力需要量の予測が可能であることがわかる。

4.2 20 点先の予測

以下の表 4、図 1 に 20 点先の予測結果を示す。また、表 5 に出力全体の誤差を示す。

表４　２０点先予測結果

モデル名	平均誤差（%）	最大誤差（%）
RNN	5.068405	31.80008
QRNN	3.249148	16.664414
TSA	2.959879	20.877629
MTSA	3.663426	19.432109
アンサンブル（TSA＋QRNN）	3.146441	18.582104

表５　２０点先予測結果（出力全体）

モデル名	平均誤差（%）	最大誤差（%）
RNN	3.059800	31.80008
QRNN	1.325396	16.664414
TSA	0.983792	20.877629
MTSA	1.347096	19.432109
アンサンブル（TSA＋QRNN）	0.754649	18.582104

検証結果より 20 点先（1 時間先）の予測では TSA が精度が高く予測可能であった。また、最大誤差をみると QRNN が最小となっており、TSA と QRNN のアンサンブルモデルではその両方の特性を持ち合わせた結果となった。

また、出力全体で見るとアンサンブルモデルが最良の結果となっており、TSA より 0.2%ほど平均誤差が小さい結果となった。

次にアンサンブルモデルの出力の中で、学習データに近い箇所と遠い箇所での誤差を以下の表 6 に示す。

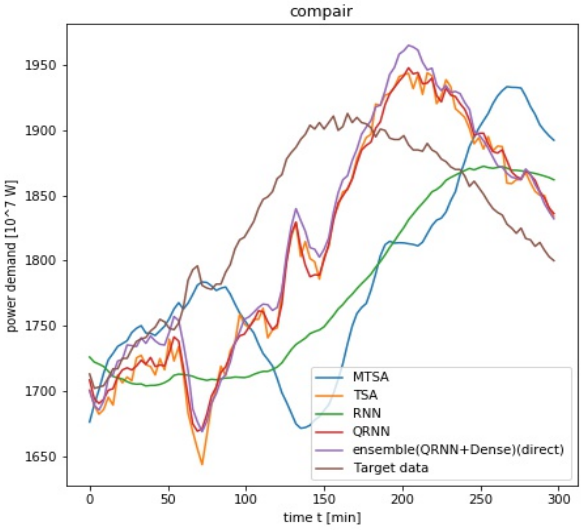

図１ 予測結果の比較

表６　検証データ内での誤差の比較

データ点	平均誤差（%）	最大誤差（%）
検証データ（全体）	0.754649	18.582104
検証データ（先頭１万点）	0.804394	17.088980
検証データ（末尾１万点）	0.785761	14.867084

この結果より、学習データと検証データの時間的な差は予測結果に影響を与えないことがわかった。

5. 結論・まとめ

時系列情報の予測手法として複数モデルを電力需要量データで検証したところ、TSA を用いた予測で優れた結果を得ることができた。また、通常の予測手法で用いられる曜日や季節、天気等のパラメータを用いず電力需用量データのみからの予測が可能であったことから、他の時系列データへの応用も可能かと思われる。

今回の結論からは提案手法である MTSA を用いて予測を行なった結果最良のモデルにはならなかったが、今回のように 20 点先を予測するように学習を行なった場合でも、中間層を予測する TSA を削減することにより 20 点先以下の任意の時刻のデータの予測が可能となるためメリットが大きい。今回は MTSA を用いて作成したネットワークの追加学習まで行えなかったため、今後検証を行う。

謝辞

本研究を行うにあたり、多くのアドバイスをいただいた九州大学　内保光太郎様に深く感謝いたします。

参考文献

[1] T, Mikolov, “Recurrent neural network based language model”, Interspeech, (2010).
[2] J. Bradbury, “Quasi-Recurrent Neural Networks”, ICLR2017, (2017).
[3] DingKe, Quasi-recurrent Neural Networks for Keras, https://github.com/DingKe/qrnn , (2018/6/28 現在)