再帰型ニューラルネットワークを用いた
人身事故による電車遅延の発生予測について

uguisuheiankyo

１．はじめに

本取り組みでは、再帰型ニューラルネットワーク(RNN)を用いて人身事故による電車遅延の発生を予測することを目的とする.

鉄道における人身事故は、電車の運営に多大な影響を与えることから、電車遅延への直接原因として明確である。また、通学・通勤で鉄道を利用する人にとっては、電車内混雑や緊急停止ボタンの押下と比較して、人身事故は大きく遅延する原因となり、遅刻に直結することから、看過できない要因のひとつとして挙げられる。

本稿では、東京圏の鉄道における人身事故の発生を予測することが可能かどうかの疑問の元、過去の人身事故の発生タイミングを学習データとして、人身事故を原因とする電車遅延の発生を予測する試みについて記載する。

２．人身事故発生予測の目標

電車の遅延原因は大きく 2 つの種類に分けられる。1 つは、車両や施設等の故障により遅延となる部内原因。もう 1 つは、急病人搬送や混雑による乗車時間超過、人身事故等により遅延となる部外原因である。国土交通省の調査[1]によれば、30 分以上の遅延となる要因の約 68%が部外原因にあり、またその内訳をみると、線路立ち入りや自殺等の人身的な事故が約 65%を占めている（図 1)。

このように、通勤や通学において遅刻の直接的な原因となり得る 30 分以上の電車遅延は、約 65%が人身事故を原因とするものであり、人身事故の発生を予測することは、早めの出発や代替路線への乗車等により、遅刻回避のための手段を取る 1 つの尺度になり得ると考える。

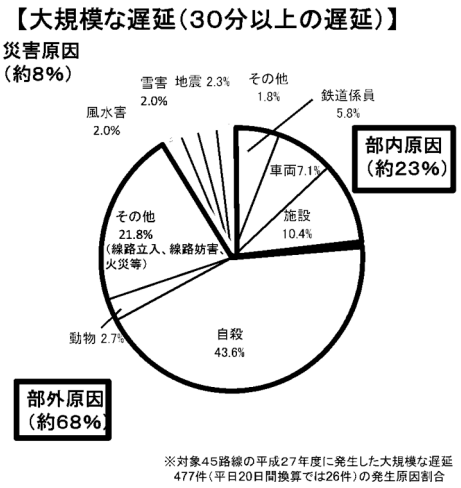

図 1. 大規模遅延の原因内訳

ここで、今回のテーマにおいて目標とする予測は、過去の人身事故発生データを元に学習したモデルを利用した「任意の路線と日付におけるその日の人身事故の発生」を予測することとする。

３．学習データの準備

学習データは、2010 年 01 月以降に発生した鉄道人身事故について、その件数を駅別、路線別、鉄道会社別にランキング形式でまとめている Web ページ[2]より、python の Selenium と PhantomJS ライブラリを利用してスクレイピングしたものを利用する(図 2)。

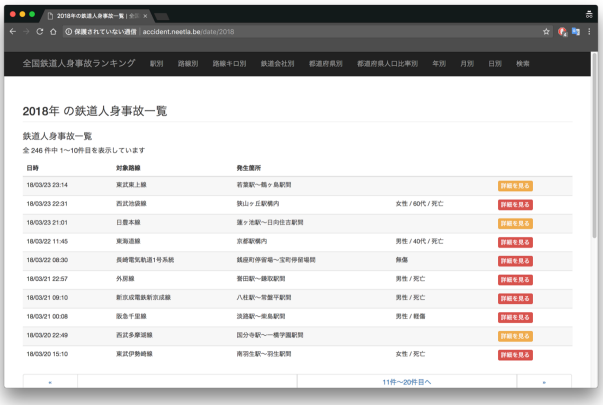

図 2. 学習データとした Web ページ

スクレイピングしたデータは、人身事故が発生した日時、対象路線、発生場所、事故内容のカラムをもつ。今回は、発生日時と対象路線に着目し、学習データを形成した。

前処理として、文字列である発生日時を数値に変換するため、Unixtime （1970 年 1 月 1 日基準の経過秒数）に変換し、さらに 2010 年 1 月 1 日基準の経過日数に修正した。図 3 は、人身事故発生数の多い東京圏 10 路線における 2010 年 1 月 1 日からの発生数を示したものである。横軸が 2010 年 1 月 1 日からの累積経過日数、縦軸が各路線の人身事故累積発生数を示す。

系列データの生成には、Lesson3 の create_dataset 関数をそのまま流用し、訓練用とテスト用の系列データを生成した。ここで、参照する過去データの範囲を指定する look_back の値は、30 日(1 ヶ月)とした。尚、学習データの内、８割を訓練用データ、残りの 2 割をテスト用データとしている。

当初は、時間や分単位の予測を試みていたものの、時間や分単位における人身事故累積発生数の変化は非常に微少なものであった。また、それに従って指定すべき look_back の値も大きくなることから必要な学習時間が膨大になったため、今回

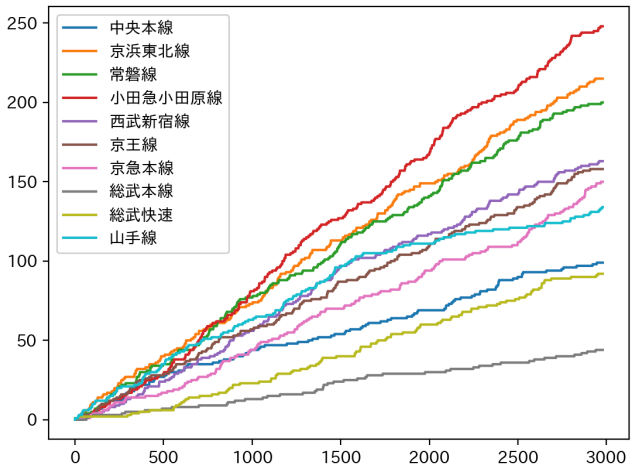

図 3. 東京圏 10 路線の人事事故発生数の推移

のテーマでは日単位データでの学習と予測とした。

4．モデルの構築と学習

ベースとして構築したモデルは、Lesson3 を流用したものであり、損失関数に平均二乗誤差（MES）、最適化アルゴリズムに Adam を利用した。学習時のエポック数は 100 、バッチサイズは 128 とした。

```
#モデル構成
adam = Adam(lr = 0.001)
model.add(SimpleRNN(units=10, input_shape=(look_back, 1)))
model.add(Dense(1))
model.compile(loss='mse', optimizer=adam)

#モデルの学習
model.fit(trainX, trainY, epochs=100, batch_size=128)
```

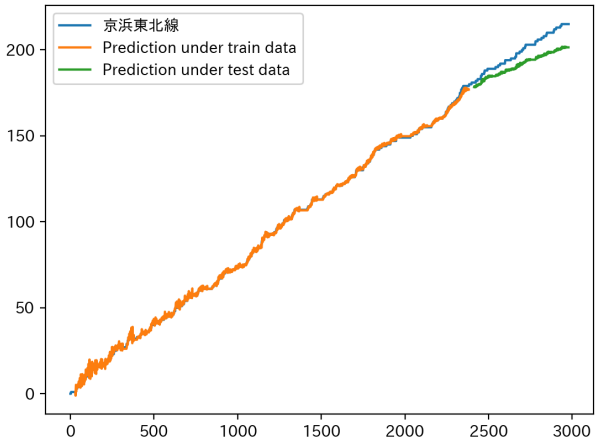

図 4. 初回実行時のモデル評価

図 4 に、対象路線として京浜東北線を選定した学習データを用いたモデルの評価を示す。Train RMSE は 1.14、Test RMSE は 8.63 となった。これをベースとして、ユニット数や学習率、エポック数の変更、LSTM や GRU といったモデルの変更を試しながら、精度向上にあたった。

5．パラメータチューニング

5.1 SimpleRNN

最初に、SimpleRNN を利用したモデルのチューニングを試

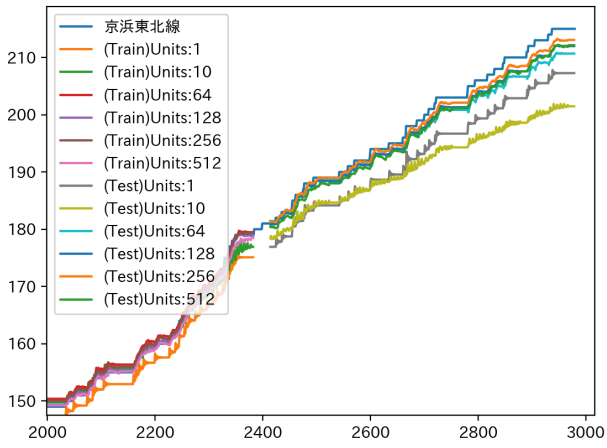

図 5. ユニット数変更時の各モデルの評価

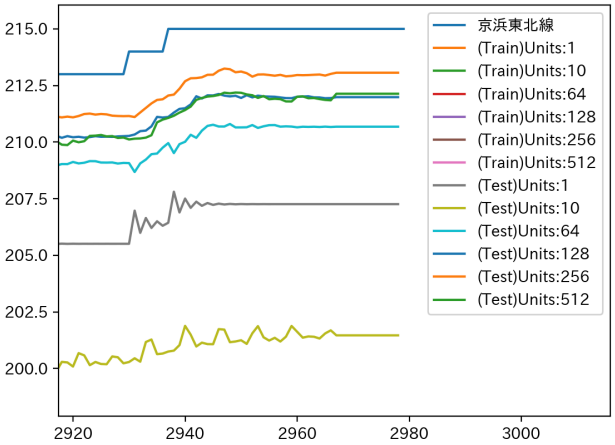

図 6. 各モデル予測データの終盤を拡大したもの

みる。対象のチューニングとするパラメータは、ユニット数（1/10/64/128）、学習率（0.001/0.005/0.01）エポック数（100/200/400）である。

まずは、今回のテーマにおいて最適なユニット数を探ることから始める。学習率を 0.001 、エポック数を 100 と固定し、ユニット数を変更した時の各モデルの予測を評価する。図 5、図 6 は各モデルの予測精度を視覚的に示したものである。予測データのプロットしたグラフの終盤を拡大した図 6 をみてみると、ユニット数が 128, 256, 512 のときに予測精度が高いことが読み取れる。また、ユニット数が 128 の時と 512 の時で ほとんど精度に差異がみられないことから、最適なユニット数は 128 もしくは 256 であるとここでは予想される。

次に、最適なエポック数について考える。ここでは、モデル学習時に算出した損失値の推移に着目し、損失関数の収束具合を調査する。model.fit 関数の validation_data には、データセット中の 2 割であるテストデータを指定している。図 7、図 8 は、ユニット数が 128, 256、エポック数が 400 とした各モデルの損失値の推移である。エポック数 0 から 150 近辺の損失値を示した図 7 に注目すると、損失値が上下に振れていることが見て取れる。エポック数 200 から 400 近辺を示した図 8 に注目すると、特にユニット数が 256 のモデルにおいては、上下の振れ幅は小さくなってきていることがわかる。図 7, 8 より、必要なエポック数は 100 以上であることが考えら

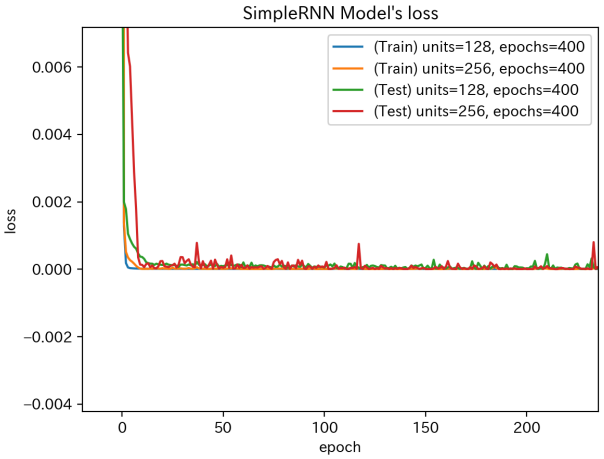

図 7. SimpleRNN における損失値の推移（1）

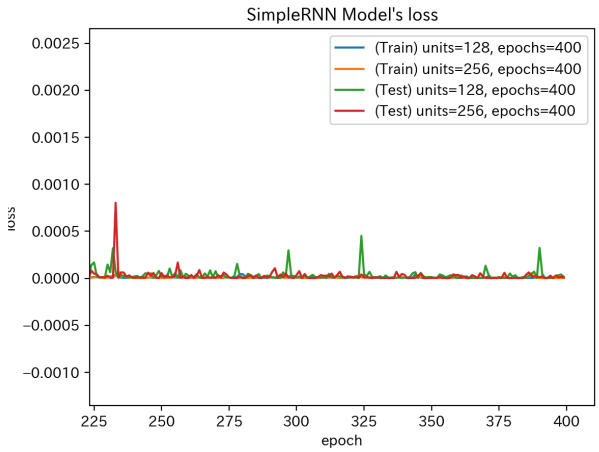

図 8. SimpleRNN における損失値の推移（2）

れる。

最後に、各パラメータ変更時のモデルにおける、平均二乗誤差の平方根（RMSE）値を表 1 に示す。黄色塗りしたカラムは、各パラメータチューニング時において精度の高かったものを示す。最適な学習率については、学習率が小さいもの方が、予測精度が高いことが表 1 より明らかである。ユニット数とエポック数については、ユニット数が 128 の場合において、エポック数は 400 と大きい方が精度が良い。ユニット数が 256 の場合においては、エポック数が 200、400 の時のテストデータからの予測精度に差異はないものの、訓練データの予測精度は、エポック数が 400 と大きい方が良かった。

以上から、SimpleRNN においては ユニット数 256、エポック数 400、学習率 0.001 が最適なパラメータと考えられる。

5.2 Long Short Term Memory(LSTM)

2 つ目の検証用モデルとして、LSTM を取り入れたモデルを考える。学習率は 0.001 と固定し、ユニット数を 128, 256、エポック数を 100, 200, 400 に変更した時の予測精度の変化を検証する。図 9 は、LSTM を用いた各パラメータ変更児における予測データをプロットしたグラフである。図 10 は、各エポック数における損失値の推移をプロットしたグラフである。図 9 を見る限りでは、ユニット数 128、エポック数 400 が最も学習元データに近い予測をしている。図 10 では、エポック数 400 までの損失値の推移を示している。ユニット数 256

モデル	エポック	学習率	ユニット数	Train RMSE	Test RMSE
SimpleRNN	100	0.001	1	1.64	6.09
			10	1.14	8.63
			64	0.92	2.54
			128	0.56	1.89
			256	0.68	1.29
			512	0.67	2.23
	100	0.001	128	0.56	1.89
		0.005		0.63	4.28
		0.01		4.41	9.37
	100	0.001	128	0.56	1.89
	200			0.53	0.93
	400			0.48	0.85
	100	0.001	256	0.68	1.29
	200			0.68	0.64
	400			0.42	0.64

表 1. SimpleRNN における RMSE

においては、エポック数 100 近辺で安定した推移を示すものの、以降、エポック数が大きくなるにつれて不安定な推移を示すようになっていた。表 2 において、LSTM の各パラメータにおける RMSE を示しているが、図 9、図 10 で表れていたように、ユニット数 128、エポック数 400 が最も良い値となっている。また、ユニット数 256 においては、エポック数が大きくなるほど、RMSE 値も大きくなっていた。

5.3 Gated Recurrent Unit(GRU)

3 つ目の検証用モデルとして、GRU を取り入れたモデルを考える。学習率は 0.001 と固定し、ユニット数を 128, 256、エポック数を 100, 200, 400 に変更した時の予測精度の変化を

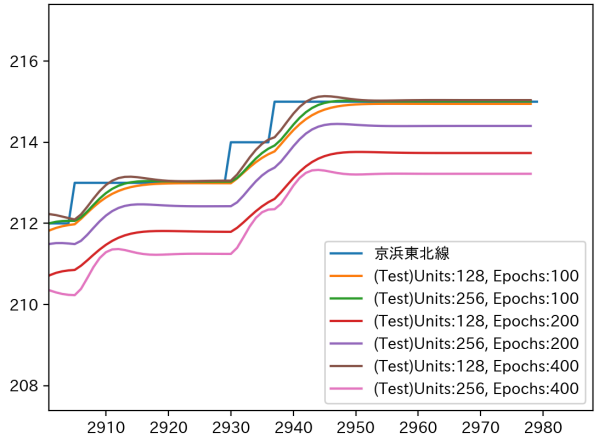

図 9. LSTM における予測データのプロット

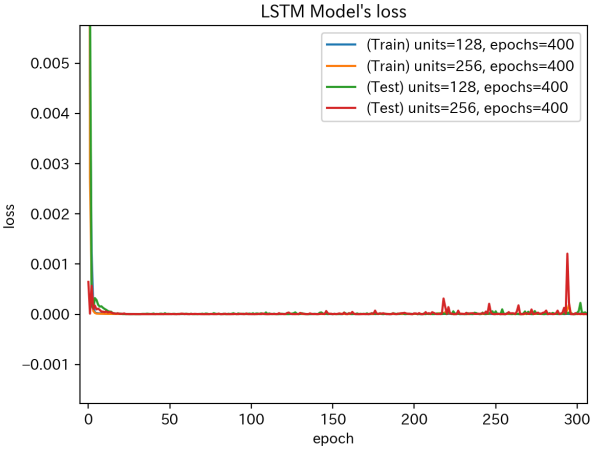

図 10. LSTM における損失値の推移

モデル	学習率	エポック	ユニット数	Train RMSE	Test RMSE
LSTM	0.001	100	128	0.52	0.45
			256	0.48	0.44
		200	128	0.63	1.24
			256	0.54	0.79
		400	128	0.44	0.39
			256	1.01	1.78

表 2. LSTM における RMSE

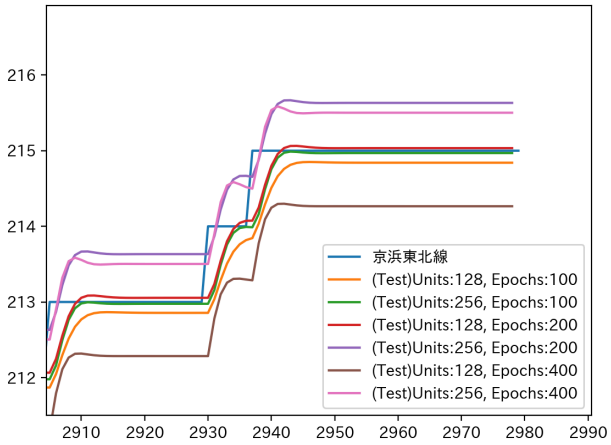

図 11. GRU における予測データのプロット

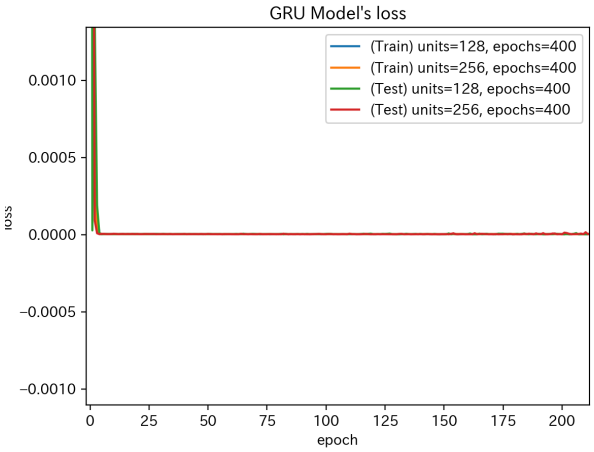

図 12. GRU における損失値の推移

検証する。図 11 は GRU における予測データをプロットしたものである。視覚的な差異はほとんど見受けられないが、ユニット数 256、エポック数 100 の場合と、ユニット数 128、エポック数 200 の場合が、元のデータセットに近い予測値を示している。図 12 は、各エポック数における損失値を示したものである。こちらも視覚的な差異はほとんど見受けられず、則ち、エポック数の大小に依らず損失値の推移は安定している。

ここで、RMSE の値に注目する。表 3 に各パラメータ変更時における RMSE の値を示す。表 3 の値を考慮してみても、GRU を用いたモデルにおいては、ユニット数とエポック数の大小に依らないと考えられる。

以上から、GRU を用いたモデルにおいては、ユニット数 128、エポック数 200 または ユニット数 256、エポック数 100 が最適なパラメータと考えられる。

６．実際の発生予測の検証

実際に人身事故が発生した日付をピックアップし、学習し

モデル	学習率	エポック	ユニット数	Train RMSE	Test RMSE
GRU	0.001	100	128	0.42	0.43
			256	0.4	0.37
		200	128	0.42	0.35
			256	0.47	0.59
		400	128	0.44	0.75
			256	0.42	0.5

表 3. GRU における RMSE

たモデルによって発生日の予測が可能かを検証する。対象とする路線は京浜東北線とし、フォーカスする発生日は 2018 年 1 月 16 日とした。ここでは、2018 年 1 月 15 日までのデータを利用し、翌日の人身事故の発生が予測できるか、則ち、累積発生数＋１を予測できるかを検証する。

まず、2018 年 1 月 16 日の人身事故発生の予測を試みた結果を図 13 に示す。2937 日目の 215 人目が、2018 年 1 月 16 日の発生を示す値であるが、実際にモデルが 215 の近い値を予測した発生日は 2942 日目であった。他の発生日に着目しても約 2 日～4 日程度のズレがみられた。このズレを考慮するならば、本モデルによって 4 日先までの予測値を算出し、累積人数に変化がみられるのであれば、翌日の人身事故発生確率は高いと判断できる、という仮説を立ててみる。2018 年 1 月 15 日時点までのデータを利用して学習させたデータを元に、16 日以降の連続した予測を試行する。Lesson3 のコードのままであると、翌日の人身事故の発生しか予測できないため、下記のように、予測したデータを次の予測値算出の入力とするようなコードに変更した。

```
# f_days:連続して予測する日数
testX = train[-look_back:]
for i in range(f_days):
    testX= np.append(testX, model.predict(
            np.array([testX[i:i+look_back]])), axis = 0)

testPredict = testX[-f_days-1:]
```

予測結果を図 14 に示す。ユニット数 256、エポック数が 200 の時、仮説の通り、2 日程度のズレが発生しつつも累積発生数は 215 に増加している。しかしながら、以降も値が単調増加しているため、本モデルが正しく予測できるとは言い切れない結果となっている。

念のため、2018 年 3 月 17 日に発生した人身事故について

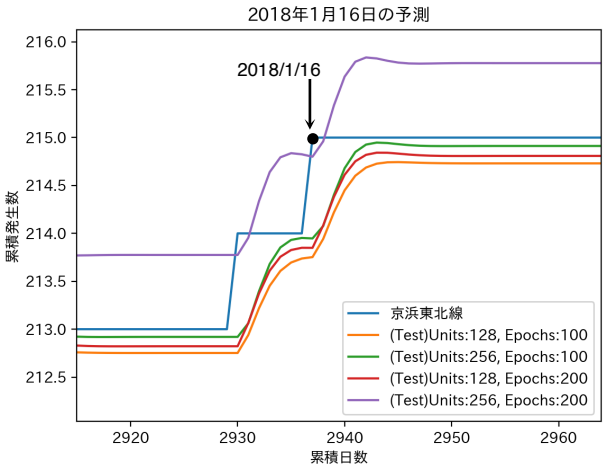

図 13. 2018/1/16 の予測

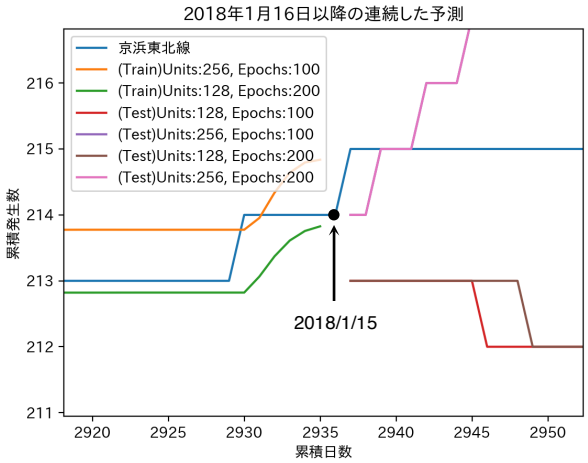

図 14. 連続した予測値の算出

も予測を試みる。図 15 に示すとおり、やはり 2018 年 3 月 17 日（2997 日目）の予測においても、累積発生数が 216 となるのに 4 日程度の遅れがみられる。次に、2018 年 3 月 16 日以降を連続して予測したものを図 16 に示す。ここでもユニット数 256、エポック数 200 の予測値に着目すると、今回は 7 日程度の遅れがあって、累積発生数は 216 を超えた。

これまでの検証より、今回生成したモデルでは、人身事故発生の予測に 2 日～1 週間程度の遅れがあることが分かった。この遅れを考慮し、ある時点から翌日の人身事故発生を予測する場合においては、連続して 1 週間程度の先読みを実施することが有効であると考える。ただし、この 1 週間程度という度合いは、本稿の中で検証した 2018 年 1 月 16 日と 2018 年 3 月 17 日に発生していた人身事故より仮定したものであり、他の発生予測においても有効であるかどうかはさらなる検証が必要である。

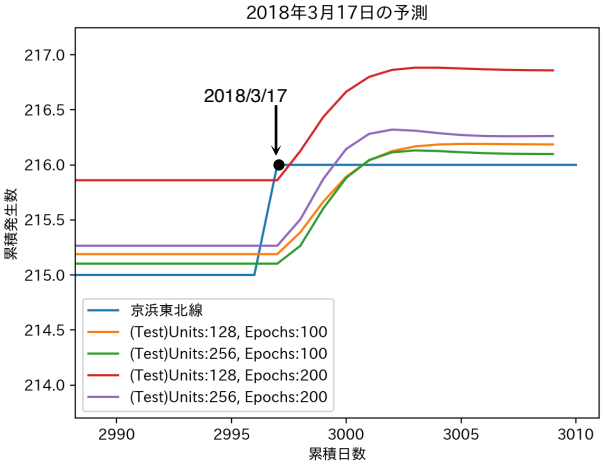

図 15. 2018/3/17 の予測

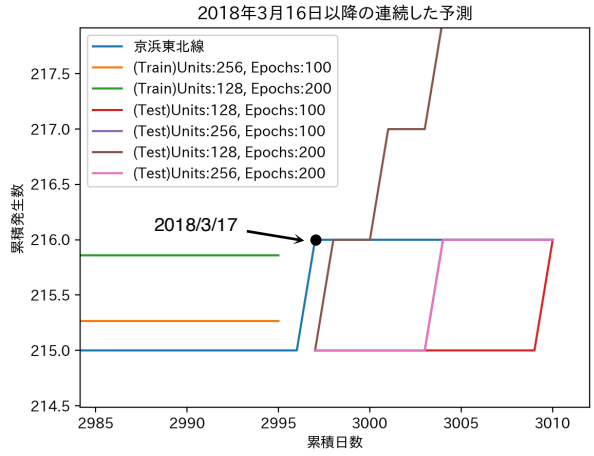

図 16. 連続した予測値の算出

発生の予測に遅れがある点については、学習データ量の不足、look_back の不足等が考えられるが、こちらもさらなる検証が必要である。

7．おわりに

本取り組みでは，再帰型ニューラルネットワーク（Simple RNN, LSTM, GRU）を用いて、人身事故による電車遅延の発生予測をおこなった。今回生成したモデルで実施した過去の人身事故発生予測では、2 日～1 週間程度の予測の誤差が見受けられた。これは、学習データ量やチューニングのまだ不十分であることが考えられる。今後は実運用に耐えられるように、モデルやパラメータのチューニング、学習データとして天気、曜日といったデータの追加、今回は実装が間に合わなかった QRNN (Quasi-RNN)といった新しいニューラルネットワークの利用によって、さらなる精度向上に努めたい。行く行くは、データ公開元の許可が得られれば、人身事故発生を予測する Twitterbot の作成も検討したい。

謝辞

本取り組みは、東京大学松尾研究室による高度な Deep Learning 技術者を育成することを目的とした、アプリケーション指向の無償オンライン教育プログラムの支援を受けたものであり、このような充実したコンテンツや検証環境等による学習の機会を頂けたこと、此処に感謝申し上げます。

文　献

[1] 遅延の「見える化」，国土交通省, http://www.mlit.go.jp/common/001215328.pdf

[2] 鉄道人身事故ランキング， http://accident.neetla.be/

[3] QUASI-RECURRENT NEURAL NETWORKS, James Bradbury, Stephen Merity, Caiming Xiong & Richard Socher, Salesforce Research, ICLR 2017