시계열 자료 예측에 대한 SimpleRNN / LSTM / GRU 성능분석

[월별 산불 발생 통계를 사용하여]

2016122015
응용통계학과
김선우

1. 개요

전통적으로 시계열 예측은 ARIMA와 같은 통계모델이 주를 이루었다. 하지만 최근에는 딥러닝을 활용하여 주가의 예측이나, 주가의 이상 변동을 포착하는 모델이 많이 알려져 있다. 시계열 자료에 적합한 Recurrent Neural Network(이하 RNN)는 실제로 다양한 시뮬레이션 시계열 데이터에서 높은 성능을 보이는 것으로 확인되었다. 그러나 사회의 실제 데이터는 큰 변동성을 갖고 있고, 패턴이 유사하다고 할 뿐 과거의 데이터가 완전히 반복되지는 않는다. 본 보고서는 RNN, 그리고 RNN의 Gradient Vanishing Problem을 해소한 LSTM과 GRU가 실제 시계열 데이터 예측에 있어서 어떠한 특성을 보이고, 어느 수준의 성능을 낼 수 있는지 확인해보도록 하겠다.

2. 데이터의 수집과 처리

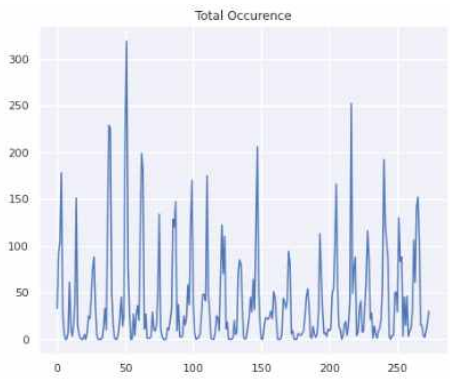
본 보고서는 통계청에서 발표한 1997~2019년의 월별 산불 발생 통계량을 시계열 자료로 사용했다. 본 데이터는 입력데이터로 활용되기 위하여 사전에 엑셀을 통해 행 정렬과 같은 간단한 처리가 선행되었다. 다음은 데이터 설명이다.

데이터

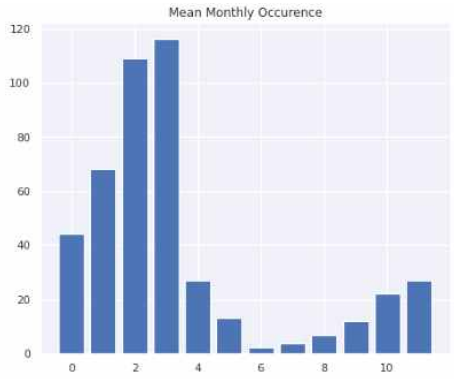
	0
0	33
1	92
2	104
3	178
4	28
...	...
269	4
270	2
271	9
272	19
273	30

274 rows × 1 columns

데이터 시각화



월별 평균 산불 발생 횟수



23개년도 12개월 자료이므로 274개의 데이터가 존재한다. 월별로 뚜렷한 차이가 존재하고, 가장 건조한 시기인 늦겨울과 봄 시기에 가장 많은 산불이 발생하는 것을 확인할 수 있다. **1997년부터 2017년까지 21개년도 250개의 데이터를 Train Data**로 사용하고, **'18 & '19년도 데이터를 Test Data**로 활용하였다. **Validation Data는 생성하지 않았다.**

Training Data는 다음과 같이 두 종류가 존재한다. 일반적인 **Raw Train Data**와, 데이터가 전체적으로 분산이 고르지 못하다고 판단하여 로그를 씌운 **Log Train Data** 두 개를 생성하였다. 두 데이터를 모두 사용하여 성능 비교를 하여, RNN이 전처리 된 데이터에 더 좋은 성능을 내는지 파악해 보았다.

3. 딥러닝 모델 RNN, LSTM, GRU

딥러닝 모델로는 일반적인 Simple RNN, LSTM, 그리고 GRU를 사용하였다. 이때 입력데이터는 다음과 같다.

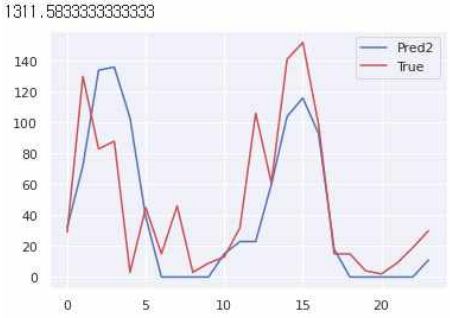
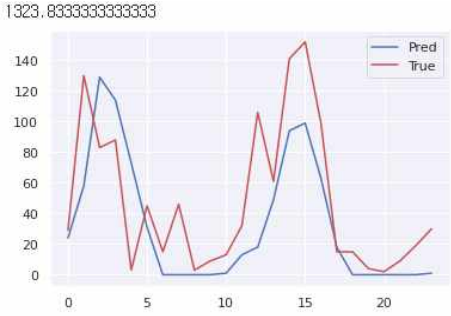
예측하고자 하는 이전 12개 시점이 input으로 들어온다. 모델은 이를 토대로 해당 시점의 데이터를 예측하고, 주어진 true 데이터와의 값 비교를 통해 가중치를 업데이트한다. Test Data에 대해서는 이전 시점에서 예측한 값을 이후 시점의 input으로 사용했다. 이로 인하여 기존 250개의 Train data는 240개로 축소된다. 이를 표로 표현하면 다음과 같다.

Training Procedure			Prediction Procedure		
n	input	label	n	input	Pred
형태	$[x_1\, x_2\, x_3\, x_4 \ldots x_{11}\, x_{12}]$의 12개	y 1개	형태	$[x_1\, x_2\, x_3\, x_4 \ldots x_{11}\, x_{12}]$의 12개	$\hat{y}$ 1개
1	1997.1, 1997.2, --- , 1997.12	1998.1	1	2017.1, 2017.2, --- , 2017.12	$\widehat{2018.1}$
2	1997.2, --- , 1997.12, 1998.1	1998.2	2	2017.2, --- , 2017.12, $\widehat{2018.1}$	$\widehat{2018.2}$
3	1997.3, --- , 1998.1, 1998.2	1998.3	3	2017.3, --- , $\widehat{2018.1}$, $\widehat{2018.2}$	$\widehat{2018.3}$
:	:	:	:	:	:
240	2016.12, --- , 2017.10, 2017.11	2017.12	24	$\widehat{2018.12}$, --- , $\widehat{2019.10}$, $\widehat{2019.11}$	$\widehat{2019.12}$

학습 수준이 크게 복잡하지 않고, 데이터 자체가 많지 않기 때문에, 층은 단일 층과 이중 층 두 개의 모델을 고안했다. Loss와 평가 기준은 모두 MSE로 설정했다. Drop Out Rate, Optimizer, Activation은 모두 동일하게 적용했는데, 저들은 가장 좋은 성능을 낸 조합이었다. 그러나 Node 개수는 조금 다르다. 먼저 single_layer_result와 double_layer_result 함수를 통해 다양한 n에 대한 개요를 잡았는데, 이는 최적화를 위함이 아니었다. 이 이유는 결론에서 밝히겠다.

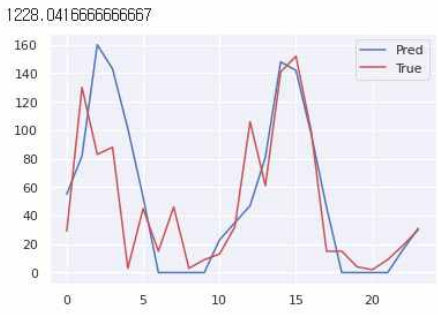
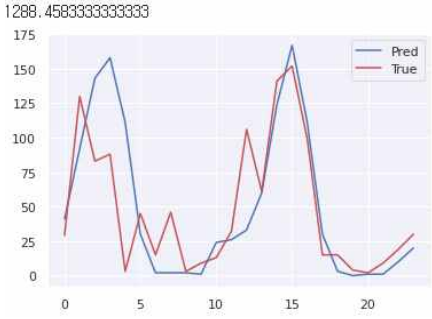
본 보고서에서 중요한 부분으로 채택한 것은 모델의 정확한 예측값보다, 모델이 데이터의 추이를 제대로 포착하였는지 여부이다. 예측값보다 추이를 우선시한 이유 역시 뒤의 결론에서 자세히 설명하겠다.

3.1) Simple RNN

	Single Layer RNN	Double Layer RNN
Parameter	Node : 50 Drop Out Rate : 0.3 Optimizer : Adam Activation : Relu	Node : 20, 60 Drop Out Rate : 0.2 Optimizer : Adam Activation : Relu
Model Summary	Layer (type) Output Shape Param # simple_rnn_9 (SimpleRNN) (None, 50) 3150 dropout_9 (Dropout) (None, 50) 0 dense_9 (Dense) (None, 1) 51 Total params: 3,201 Trainable params: 3,201 Non-trainable params: 0	Layer (type) Output Shape Param # simple_rnn_72 (SimpleRNN) (None, 1, 20) 660 dropout_72 (Dropout) (None, 1, 20) 0 simple_rnn_73 (SimpleRNN) (None, 60) 4860 dropout_73 (Dropout) (None, 60) 0 dense_43 (Dense) (None, 1) 61 Total params: 5,581 Trainable params: 5,581 Non-trainable params: 0
Training Procedure		
Prediction Result		

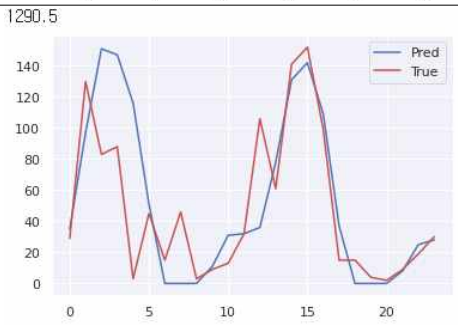
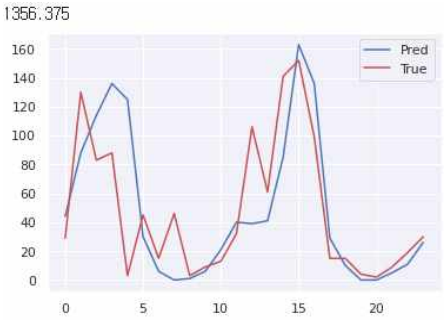
Simple RNN(이하 SRNN)의 학습 결과이다. 단층과 이층 모델 모두 늦겨울과 초봄에 많은 산불이 발생한다는 특징은 잘 학습했지만, 전체적으로 test data보다 작은 값을 예측했다. 또한, 예측값이 다양하지 않고 평이하게 완만한 곡선을 그린다는 것을 확인할 수 있다. 특이점으로는 두 개의 구조 모두 '18년도의 산불을 '19년도보다 크게 예측한 것으로, 이는 Vanishing Gradient의 현상이 일정 부분 발생한 것으로 보인다. SRNN은 '19년 11, 12월의 산불이 다시 증가한다는 일반적인 패턴에 대해서는 잘 예측하지 못한 것으로 보인다. 두 개 층 모델의 경우 node 수가 층별로 2~3배 정도 차이가 있을 때 추이를 더 잘 학습했다.

3.2) LSTM

	Single Layer LSTM	Double Layer LSTM
Parameter	Node : 45 Drop Out Rate : 0.3 Optimizer : Adam Activation : Relu	Node : 40, 40 Drop Out Rate : 0.2 Optimizer : Adam Activation : Relu
Model Summary	Model: "sequential_56" _____ Layer (type) Output Shape Param # ===== lstm_13 (LSTM) (None, 45) 10440 _____ dropout_86 (Dropout) (None, 45) 0 _____ dense_56 (Dense) (None, 1) 46 ===== Total params: 10,486 Trainable params: 10,486 Non-trainable params: 0	_____ Layer (type) Output Shape Param # ===== lstm_83 (LSTM) (None, 1, 40) 8480 _____ dropout_85 (Dropout) (None, 1, 40) 0 _____ lstm_84 (LSTM) (None, 40) 12960 _____ dropout_86 (Dropout) (None, 40) 0 _____ dense_51 (Dense) (None, 1) 41 ===== Total params: 21,481 Trainable params: 21,481 Non-trainable params: 0
Training Procedure		
Prediction Result		

전반적인 산불 발생 추세를 잘 학습한 모습이다. Training Data에 대해 학습을 하는 과정 자체는 SRNN보다 유의미한 차이를 보였다고 할 수 있다. 예측 데이터는 전반적으로 test data보다 큰 값을 예측했고, '18년도의 예측값과 '19년도의 예측값이 많은 차이를 보이지 않았다. LSTM의 장점 중 하나인 Vanishing Gradient를 완화 시키는 효과가 발생한 것으로 예상된다. SRNN과 다르게 LSTM은 '19년 11, 12월에 산불 발생이 다시 증가한다는 점을 잘 포착했다. 구체적인 node의 수는 중요하지 않지만, 이중 층의 경우에는 두 층의 node 수가 유사할 때 더 좋은 성능을 냈다.

3.3) GRU

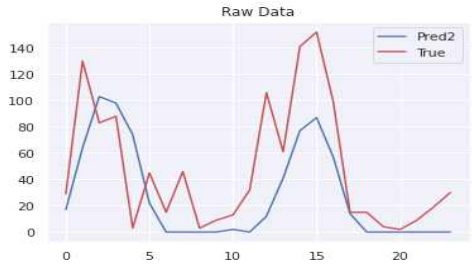
	Single Layer GRU	Double Layer GRU
Parameter	Node : 70 Drop Out Rate : 0.3 Optimizer : Adam Activation : Relu	Node : 60, 60 Drop Out Rate : 0.2 Optimizer : Adam Activation : Relu
Model Summary	Layer (type) — Output Shape — Param # gru_85 (GRU) — (None, 70) — 17430 dropout_375 (Dropout) — (None, 70) — 0 dense_338 (Dense) — (None, 1) — 71 Total params: 17,501 Trainable params: 17,501 Non-trainable params: 0	Layer (type) — Output Shape — Param # gru_100 (GRU) — (None, 1, 60) — 13140 dropout_390 (Dropout) — (None, 1, 60) — 0 gru_101 (GRU) — (None, 60) — 21780 dropout_391 (Dropout) — (None, 60) — 0 dense_346 (Dense) — (None, 1) — 61 Total params: 34,981 Trainable params: 34,981 Non-trainable params: 0
Training Procedure		
Prediction Result		

GRU는 전체적인 패턴은 LSTM과 유사한 형태를 보였다. 패턴을 잘 파악하고, '19년도 말에 산불이 다시 증가하는 추세를 잘 표현했다. 그러나 Simple RNN이나 LSTM보다 좀 더 많은 수, 그리고 좀 더 넓은 범위에서 적절한 Node의 range가 형성되었다.

4. 결론

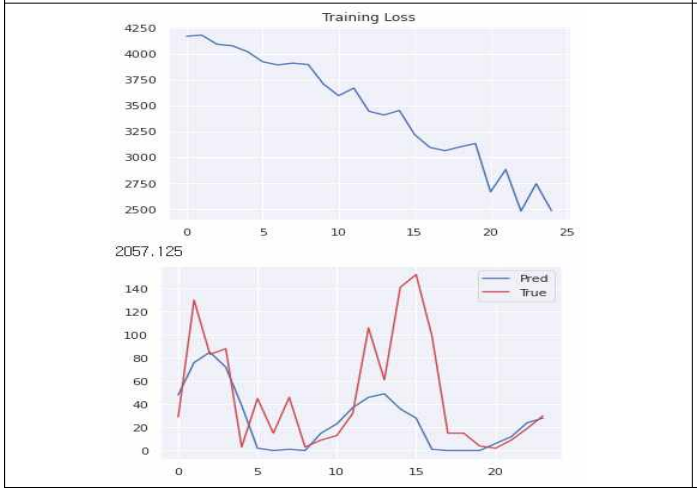
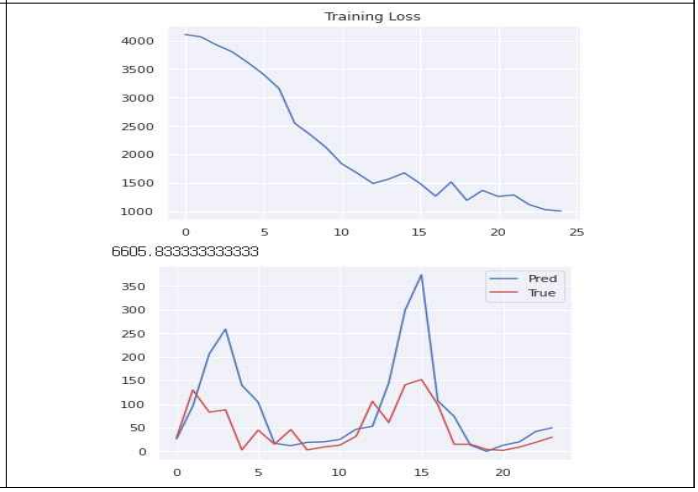
4.1) 자료의 전처리 및 검정

해당 데이터는 이분산성(Heteroskedasticity)이 명확했다. 실제로 SARIMAX에 데이터를 넣고 돌렸을 때 log를 취해주어야 제대로 된 학습을 할 수 있었다. 그러나 이 딥러닝 모델의 경우에는 이분산성과 무관하게 Raw Data를 사용해야 더 좋은 효율이 나왔다. 이 모델의 경우 수치화 데이터 전처리의 필요성이 통계모델보다는 비교적 작은 것을 확인할 수 있다. 아래 그림은 같은 parameter 조건에 대하여 일반 데이터와 로그 분산 안정화 데이터의 예측 결과 차이를 시각화한 것이다.



4.3) Node의 수와 예측성능의 관계

Node 개수에 대해서는 모든 모델에 대해서 공통적인 특성이 하나 발견되었다.
1) Node 수가 많을 경우, Training Loss는 감소하지만 과적합되어 기준보다 큰 값을 예측한다.
2) Node 수가 적을 경우, Training Loss는 조금 감소하는 과소적합으로 기준보다 작은 값을 예측한다.
그러나 이번 경우에서 최적 node를 찾는 것은 매우 어려운데, 그 이유는 다음에서 다루겠다.

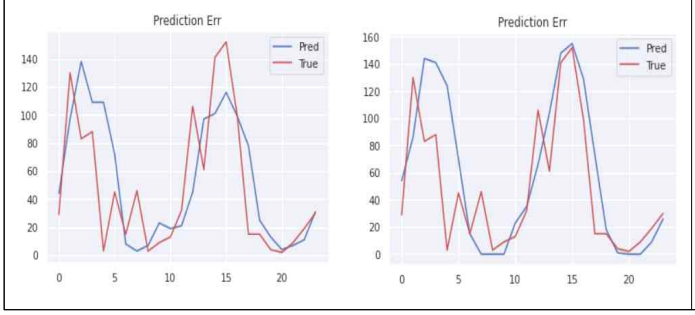
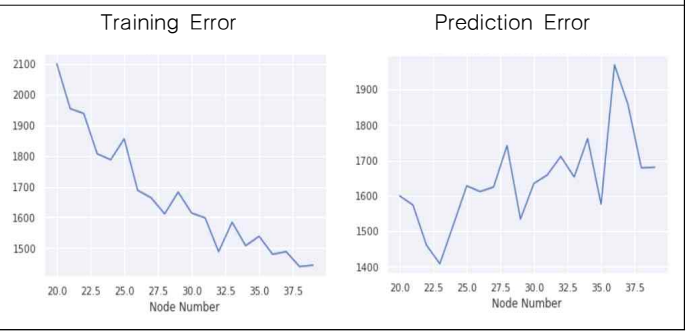
Node의 수가 적정 수보다 적음	Node의 수가 적정 수보다 많음
	

4.2) Importance of Batch Selection / Bias-Variance Tradeoff / 추세와 MSE

이 보고서에서 내가 가장 강조하고 싶은 부분이다. 하단 좌측의 plot과 같이, 같은 parameter setting임에도 불구하고, 모델은 학습과 예측성능에서 정말 큰 차이를 보인다. 이는 batch sampling의 차이에서 기인하는데, 이번 프로젝트에서는 최적의 batch selection을 발견하지 못하였다. **딥러닝 모델은 Random Sampling 되는 batch에서 최적의 성능을 내는데, 이러한 부분이 시계열 예측의 본질적 특성에 치명적으로 작용**하는 것 같다.

여기서도 Bias-Variance Trade Off와 Underfitting&Overfitting을 확인할 수 있다. 우측의 plot을 보면 node 개수가 증가할수록 학습 Error는 감소하지만, 예측에 있어서 이것이 비례하지 않는다. 그렇다고 23개의 node가 LSTM 단일 층에서 최적의 예측을 하는 것은 아니다. 보고서처럼 45개의 node가 더 적절한 Training과 Prediction의 균형을 이룬 것처럼, 최적의 예측을 하는 node - batch 짝은 따로 존재할 수 있다.

또한, 나는 구체적인 MSE 값보다 추세 학습을 더 중요시하였다. 전반적으로 과대예측의 경우가 과소예측보다 더 큰 prediction MSE를 냈는데, 좌측의 plot에서 오른쪽 경우가 추세를 더 잘 예측했지만, MSE 값은 좌측이 훨씬 낮았다. MSE와 추세의 경중에는 이견이 존재할 수 있다. 그러나 나는 최적화가 불가능하다고 판단하여 추세를 우선시하였다.

같은 Parameter Setting의 다른 결과	LSTM에 대해 node 숫자 별로 30번의 Simulation 결과
	

4.4) 뛰어난 패턴탐색

한가지 고무적인 부분은, [SRNN, LSTM, GRU] 모두 산불 발생에 대한 전반적인 추이는 잘 파악했다는 점이다. SRNN은 두 번째 년도('19)를 과소예측했지만, LSTM과 GRU는 전체적으로 마지막 산불 반등까지 추세를 완벽하게 파악했다. 이를 확장한다면, 시계열 데이터에 대해서 인간이 포착할 수 없는 미세한 패턴을 모델이 학습할 수 있다는 뜻으로 해석될 수 있다. 이것이 실제 금융시장에서 금융사기를 포착하는 AI의 핵심 기술이라고 생각된다. 추후 RNN의 기술이 더욱 발달하여, 보다 넓은 범위에서 활용되는 RNN의 모습을 기대해 본다.

Reference.

1. 데이터 관련 :

1.1) 데이터 수집
http://kosis.kr/statisticsList/statisticsListIndex.do?menuId=M_01_01&vwcd=MT_ZTITLE&parmTabId=M_01_01#SelectStatsBoxDiv
** 해당 링크에서) 범죄◦안전 → 산불통계 → 월별 발생 건수

1.2) 산불 관련 기초정보
https://terms.naver.com/entry.nhn?docId=3576505&cid=58947&categoryId=58981

2. 기초 Numpy, Pandas, Matplotlib 문법참고

2.1) Jake VanderPlas, **파이썬 데이터 사이언스 핸드북**, 김정인 옮김, 위키북스, 2017년

3. 딥러닝 모델구현

3.1) Francois Chollet, **케라스 창시자에게 배우는 딥러닝**, 박해선 옮김, 길벗, 2018년

3.2) RNN, LSTM, GRU parameter 참고 : https://keras.io/api/

3.3) 수치 데이터 예측 모델 참고 1 :
https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/

3.4) 수치 데이터 예측 모델 참고 2 :
https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/

3.5) 수치 데이터 예측 모델 참고 3 :
https://tykimos.github.io/2017/09/09/Time-series_Numerical_Input_Numerical_Prediction_Model_Recipe/

3.6) 수치 데이터 예측 모델 참고 4 :
https://towardsdatascience.com/time-series-forecasting-with-rnns-ff22683bbbb0

4. 비교를 위한 통계모델 구현

4.1) SARIMAX 코드 참고 :
https://machinelearningmastery.com/sarima-for-time-series-forecasting-in-python/

4.2) SARIMAX 코드 참고 2 :
https://towardsdatascience.com/how-to-forecast-sales-with-python-using-sarima-model-ba600992fa7d