LSTM기반의 강남역 시간별 지하철 탑승인원 예측

The Prediction of the number of passengers in Gangnam subway Station using LSTM

요 약

지하철의 포화도는 계속 증가하는 추세이며, 이용객들에게 큰 불편을 안겨주고 있다. 따라서 본 논문에서는 LSTM을 기반으로 한 강남역의 시간별 탑승 인원을 예측하는 방법을 제안한다. 제안된 모델은 강남역의 승차 인원에서 하차 인원의 수를 뺀 값인 탑승 인원을 입력으로 순환 신경망 네트워크 구조를 이용해서 실험을 하였다. 실험 결과로 매우 높은 정확도를 보였고, 이를 통해 전 역의 지하철 포화도를 계산, 이용객들에게 현재 시각 지하철의 포화도 제공, 이용자들의 분산효과, 정부 기관의 지하철 시간표 조정과 예산할당의 지표로 활용하는 등의 효과를 기대할 수 있다.

1. 서 론

출퇴근 시간대 혹은 특정 호선들의 차량은 수용률이 150% 이상을 넘어가며, 이는 시간이 지날수록 증가하는 추세이다. 이로 인해 현대인들이 지하철 이용에 불편함을 겪고 있다. 광역 버스에는 남은 좌석 수가 표시가 되지만 지하철에는 탑승 인원 표시를 지원하지 않는다. 따라서 본 논문에서는 서울특별시에서 제공받은 지하철 호선별 역별 승·하차 인원 정보 데이터[1]를 이용하여 순환 신경망(RNN, Recurrent Neural Network)기반의 강남역 탑승 인원 예측 모델을 제시한다.

2. 관련기술

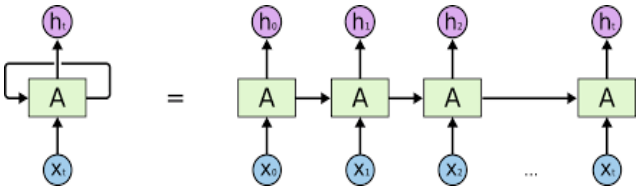

Fig. 1 RNN 모델 [3]

순환 신경망은 Fig 1과 같이 입력 데이터와 이전 데이터가 함께 다음 입력으로 들어가도록 연결되어 있다. 따라서 과거의 데이터를 이용해 다음 데이터에 영향을 주는지를 학습할 수 있고 이러한 특성으로 시계열 데이터의 형태를 지닌 데이터의 학습이 용이하다.[4]

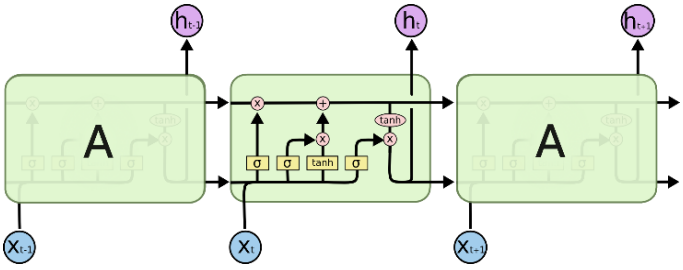

Fig. 2 LSTM구조 [5]

LSTM은 RNN을 기반으로 하는 모델로써 여러 개의 게이트(gate)가 붙어있는 셀(cell)로 이루어져 있으며 이 셀들이 하는 역할은 정보를 버리고, 저장하고, 업데이트하고, 내보내는 기능들로 이루어져 있다. 각 셀은 셀에 연결된 게이트의 값을 보고 무엇을 저장할지, 언제 정보를 내보낼지, 언제 쓰고 언제 지울지를 결정한다. LSTM은 이러한 셀 구조를 가짐으로써 RNN의 대표적인 문제인 기울기 소실 및 발산 문제와 장기 의존성 문제를 해결할 수 있다.

3. 데이터셋

서울시에서 운영하는 서울열린데이터광장[6]에서 제공하는 서울시 지하철 호선별 역별 승하차 인원 정보를 데이터를 이용하였다. 사용한 데이터의 기간은 2008년 1월 1일부터 2017년 09월 30일 까지이며, 지하철 운행 시간대별로 구간을 나누어 승차 및 하차 인원을 역별로 나누어 저장하였다. 본 연구에서 원하는 예측값은 시간대별로 탑승 인원을 구하는 것이므로, 승차 인원에서 하

차 인원을 뺀 값에서 시간대별로 나누어진 구간별 데이터를 하나의 일일 데이터로 생성하였다.

3. 실험환경 및 모델 설정

본 실험에서는 python[7]3.6과 Keras[8]를 이용한다. 데이터셋의 탑승 인원(승차 인원 - 탑승 인원)의 값을 입력으로 받는다. 탑승 인원은 출퇴근 시간대에 가장 큰 값을 가지고, 05시, 11시 이후에는 급격히 감소하는 것을 알 수 있다. 데이터의 각 구간별 격차가 매우 크기 때문에 입력의 범위의 격차가 커지므로 경사 하강(Gradient Descent)을 적용하기 까다로워지지만, 데이터를 정규화하면 쉽고 빠르게 최적화 지점을 찾을 수 있다.[9] 따라서 원본데이터를 0과 1사이의 값으로 정규화를 실시하였다. 이 중에서 가장 많은 승·하차 인원을 가진 강남역을 기준으로 실험을 진행하였다. 강남역의 데이터의 총 개수는 71,22개이며, 학습에 80%, 테스트에 20%의 데이터를 할당하였다. 원본데이터가 1일을 20개의 시간으로 나누어져 있기 때문에 얼마만큼의 과거 데이터를 사용하여 학습할 것인지를 뜻하는 lookback을 1일에 해당하는 20으로 설정하였다.

모델은 LSTM을 기반으로 학습한다. 모델의 성능 평가 척도는 평균 제곱 오차(MeanSquareError, 이하 MSE)를 사용한다. MSE는 실제 데이터값과 예측값의 차이의 제곱을 의미한다. 차이가 크면 클수록 MSE는 더욱 커지게 된다. 따라서 이 값의 차이를 최대한 줄임으로써 실제 값과 예측값의 차이를 줄이는 것이 이번 실험의 목표이다. MSE를 식으로 정리하면 다음 식 (1)과 같다.

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2 \qquad (1)$$

위 수식과 같이 입력값 y_i에서 예측값 $\tilde{y}_i$를 뺀 값을 제곱한 뒤 모두 합하여 평균을 낸다.

학습의 손실 함수는 MSE, 최적화는 rmsprop, 에폭은 10, 미니배치의 크기는 16으로 설정하였다. 모델의 마지막에는 하나의 출력값을 내보내기 위해 유닛 하나를 추가하였다 따라서 본 실험 예측 모델의 네트워크를 다음과 같이 구성한다.

3.1 네트워크 구조

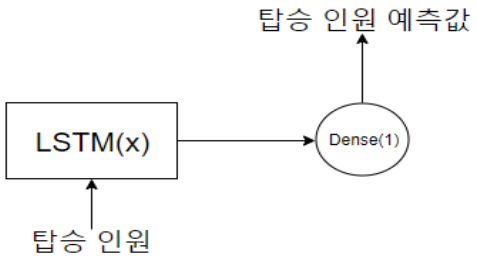

Fig. 3.1 (a)

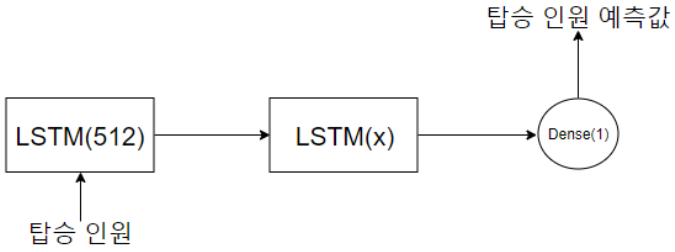

Fig. 3.2 (b)

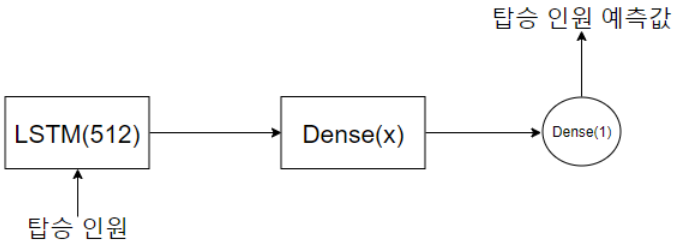

Fig. 3.3 (c)

여기서 x는 유닛의 크기를 뜻하는 변수이다.

4. 실험 및 분석

본 실험에서는 3.1의 (a), (b), (c) 모델을 이용하여 최솟값의 MSE를 얻을 수 있는 구체적인 실험 모델을 구현하였다. 각 결과를 시각화하여 네트워크를 설명하겠다.

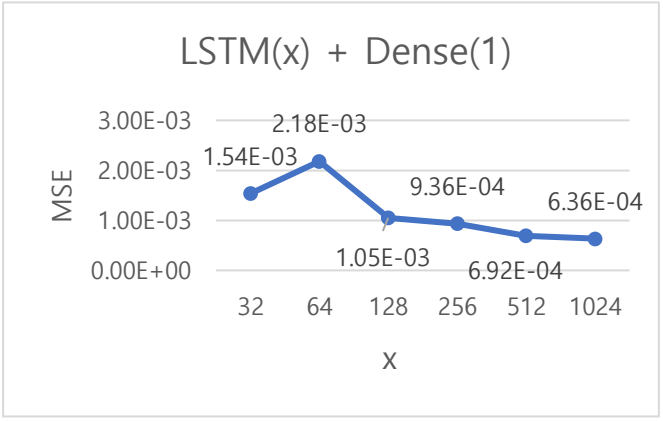

Fig. 4.1 (a) 모델의 MSE 시각화

(a)모델의 경우 단순 LSTM layer만으로도 매우 낮은 오차를 보인다. 하지만 더욱 낮은 오차를 구하기 위하여 Hidden Layer를 추가한다. 첫번째 층 LSTM 유닛의 개수를 512로 설정하는 가장 큰 이유는 1024개의 유닛과 MSE를 비교하면 차이가 적기 때문이다. 또한, 유닛을 줄이면 학습속도 개선과 과적합의 방지의 효과가 있으므로 유닛의 개수를 512로 설정한다.

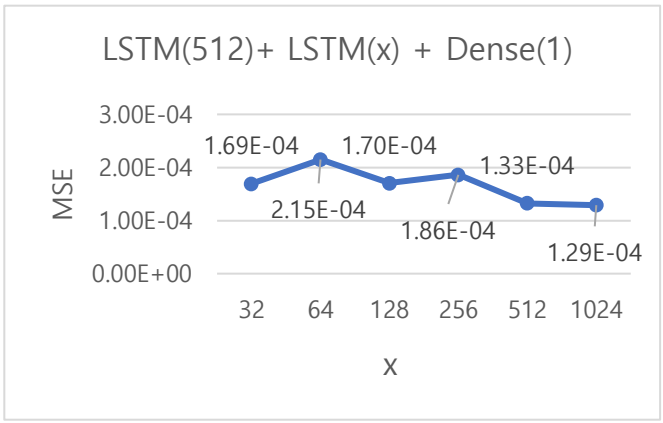

Fig. 4.2 (b) 모델의 MSE 시각화

(b) 모델의 경우 Unit의 수가 많아질수록 오차가 줄어드는 것을 알 수 있다. 하지만 (a) 모델에 비해 유닛의 개수가 많아질수록 오차가 크게 떨어지지 않는 것을 확인할 수 있다. 이는 입력 데이터의 크기에 비해 모델이 너무 복잡하여 과도한 파라미터를 가지므로 오히려 학습 능력이 줄어든다는 것을 알 수 있다. 또한, 과적합이 발생할 가능성이 크다.

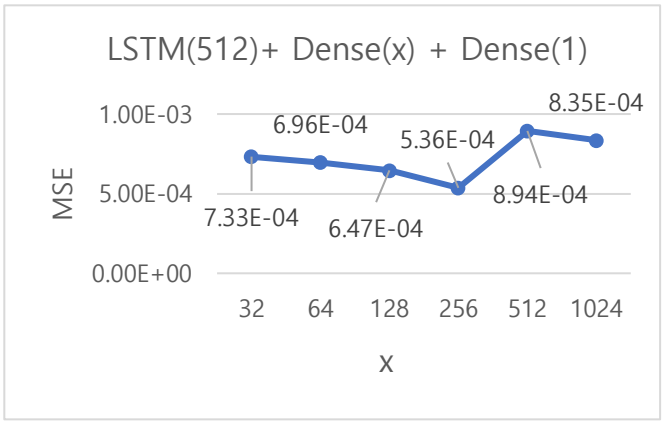

Fig. 4.3 (c) 모델의 MSE 시각화

(c) 모델이 (b) 모델보다 오차가 큰 이유는 완전 연결(Fully-Connected)이 시계열 데이터의 예측을 정확하게 하지 못한다는 것을 의미한다. 그 이유는 과거의 데이터를 고려하지 않았기 때문이다.

5. 결론 및 향후 연구

본 논문에서는 강남역의 시간별 탑승 인원에 대해 Keras의 LSTM layer와 Dense layer를 이용하여 MSE를 줄이는 방향으로 연구를 진행하였다.

이번 실험에서는 강남역 하나에 대한 탑승 인원 예측을 하였다. 이를 확장해 모든 역에 대한 탑승 인원을 예측하면 지하철 수용인원 대비 탑승 인원을 계산하여 전역의 지하철 포화도를 계산할 수 있을 것이다. 이를 통해 이용객들에게 출퇴근 시간대 특정 시점에서의 지하철 포화도를 제공할 수 있으며, 비교적 적은 포화도에서 편안한 지하철 이용을 가능하게 하는 서비스를 만들 수 있다. 많은 사람이 이러한 포화도를 보고 지하철을 이용한다면 분산효과를 기대할 수 있으며 이는 지하철 이용의 불편 감소로 이어지는 긍정적인 영향을 가져올 수 있다. 또한, 정부 기관에서는 미래의 탑승 인원을 예상하여 지하철 시간표를 조정할 수 있고, 예산할당의 지표로 삼을 수 있다.

현재 데이터셋은 시간별로 구간이 나누어진 데이터를 합쳐서 예측한 모델이지만, 학습된 시계열 모델을 선형회귀모델로 바꾸어 분 단위의 탑승 인원 예측이 가능하다. 또한, LSTM을 사용했던 모델의 개선을 통해 모델의 정확도를 더욱 높일 수 있다.

참 고 문 헌

[1] https://data.seoul.go.kr/dataList/datasetView.do?iinfl=OA-12921&srvType=F&serviceKind=1&currentPageNo=1

[2] S. Hochreiter and J. Schmidhuer, “Long Short-term Memory”, Neural Computation, Vol9, No.8, pp.1735-1780, 1997.

[3] https://towardsdatascience.com/recurrent-neural-networks-and-lstm-4b601dd822a5

[4] http://aikorea.org/blog/rnn-tutorial-1/

[5] http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[6] https://data.seoul.go.kr/

[7] https://www.python.org/

[8] https://github.com/keras-team/keras

[9] https://stackoverflow.com/questions/4674623/why-do-we-have-to-normalize-the-input-for-an-artificial-neural-network

.