



저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:



저작자표시. 귀하는 원저작자를 표시하여야 합니다.



비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.



변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 [이용허락규약\(Legal Code\)](#)을 이해하기 쉽게 요약한 것입니다.

[Disclaimer](#)

석 사 학 위 논 문

딥러닝을 이용한 무기체계
수리부속의 간헐적 수요예측

A Study on Weapon System Spare Parts
Intermittent Demand Forecasting
Using Deep Learning

고려대학교 컴퓨터정보통신대학원

소프트웨어공학 전공

오 병 훈

김 현 철 지도교수
석 사 학 위 논 문

딥러닝을 이용한 무기체계
수리부속의 간헐적 수요예측

A Study on Weapon System Spare Parts
Intermittent Demand Forecasting
Using Deep Learning

이 논문을 공학 석사학위 논문으로 제출함

2017 년 12 월 25 일

고려대학교 컴퓨터정보통신대학원
소프트웨어공학전공

오 병 훈



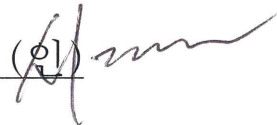
오 병 훈의 공학 석사학위 논문 심사를 완료함

2017 년 12 월

위원장 김 현 철

(인) 

위 원 서 태 원

(인) 

위 원 임 희 석

(인) 



요 약

국방 군수 분야에서는 적정한 수리부속을 확보하고 있어야 군 무기체계의 효율적인 운용이 가능하다. 연간 소모 소요를 예측하기 위한 수요예측은 재고 관리를 위해 선행되어야 할 중요한 역할을 차지하고 있다.

하지만 간헐적인 수요는 수요예측을 어렵게 만드는 주요 원인 중 하나이다. 간헐적인 수요는 수요의 발생이 드문 경우가 빈번하거나 수요 발생 간격이 불규칙한 수요를 말한다. 이러한 수요는 정규분포를 따르지 않기 때문에 기존 시계열 기법으로 예측하는데 있어 어려움이 크다. 간헐적 수요예측의 정확도를 높이기 위해 다양한 연구들이 진행되어 오고 있다.

최근 인공지능은 금융, 경제, 인문학 등 다양한 분야에서 기존에 어려웠던 문제를 딥러닝을 통해 해결하며 주목을 받고 있다. 특히 순환신경망의 일종인 LSTM의 경우 딥러닝 알고리즘 중 시계열 데이터를 예측하는데 높은 성능을 보이고 있다. 순환신경망은 데이터의 선후관계를 인지하고 학습할 수 있는 능력을 가지고 있으므로 시계열 데이터에서 강점을 갖고 있다.

본 논문에서는 간헐적 품목의 수요예측에 순환신경망을 비롯한 딥러닝 알고리즘을 적용시켜보고 성능을 검증해 보았다. 실험에 사용한 알고리즘은 MLP, RNN, LSTM, GRU이며 실험 결과 LSTM의 RMSE, MAE의 예측 오차가 다른 알고리즘들에 비해 가장 적게 나타나 성능의 우수함을 확인하였다.

주제어 : 간헐적 수요예측, 순환신경망, LSTM, RNN, 딥러닝



Abstract

In the field of defense logistics, effective operation of the military weapons system is possible only when adequate repair parts are secured. Demand forecasting to predict annual consumption plays an important role that should precede inventory management.

However, intermittent demand is one of the main reasons why demand forecasting is difficult. Intermittent demand is defined as demand that is often generated in rare instances or that has an irregular interval between demand and generation. These demand are not according to the normal distribution, so it is very difficult to predict using existing time-series techniques. Various studies have been conducted to increase the accuracy of the forecast of intermittent demand.

Recently artificial intelligence is receiving the spotlight for solving difficult problems in various fields, such as finance, economics, and humanities by deep-learning. LSTM, a type of recurrent neural network, in particular has performed well in predicting time-series data among deep-learning algorithms. The recurrent neural network has an advantage in serial-series data, as it has the ability to recognize and learn about the after-trade of data.

In this paper, we applied the deep-learning algorithm, including recurrent neural network, to forecast the demand for intermittent items and tested its performance. The algorithms used in the experiment were MLP, RNN, LSTM, and GRU, and the results showed the lowest error in LSTM's RMSE and MAE, which led to superior performance.



목 차

1. 서론	1
1.1 연구배경 및 목적	1
1.2 관련 연구	2
1.3 논문의 구성	2
2. 이론적 배경	2
2.1 다층 퍼셉트론 (Multi-Layer Perceptron, MLP)	3
2.2 순환 신경망 (Recurrent Neural Network, RNN)	6
2.3 장단기 기억 메모리 (Long-Short Term Memory, LSTM)	7
2.4 게이트 된 순환 유닛(Gated Recurrent Unit, GRU)	10
3. 연구 방법	11
3.1 실험 데이터 및 전처리	11
3.2 실험 환경	13
3.3 평가 방법	13
3.4 모델 설계	14
4. 결과	19
4.1 학습 결과	19
4.2 결과 검증 및 평가	21
5. 결론 및 향후 과제	26
참고 문헌	27



그림 목차

그림 1. 다층 퍼셉트론의 구조	3
그림 2. 활성화 함수	4
그림 3. Sigmoid 함수	5
그림 4. ReLu 함수	5
그림 5. RNN 구조	6
그림 6. RNN의 모듈 구조	7
그림 7. LSTM의 모듈 구조	8
그림 8. LSTM의 망각 게이트(Forgot gate)층	8
그림 9. LSTM의 입력 게이트(Input gate)층	9
그림 10. LSTM의 셀 상태(cell state) 값	9
그림 11. LSTM의 셀의 출력값	9
그림 12. GRU의 모듈 구조	10
그림 13. 수요발생 유형 구분	11
그림 14. 간헐적 품목 월간 수요 데이터 조회 현황	13
그림 15. 간헐적 품목 ADI, CV 데이터 조회 현황	13
그림 16. 모델 설계 순서	15
그림 17. Python Import Library	15
그림 18. 다층퍼셉트론 모델 학습 코드	16
그림 19. 심층 다층퍼셉트론 모델 학습 코드	16
그림 20. 순환신경망 모델 학습 코드	17
그림 21. 기본형 LSTM 모델 학습 코드	17
그림 22. 메모리 단위 LSTM 모델 학습 코드	18
그림 23. 상태유지 LSTM 모델 학습 코드	18
그림 24. 상태유지 스택 LSTM 모델 학습 코드	19

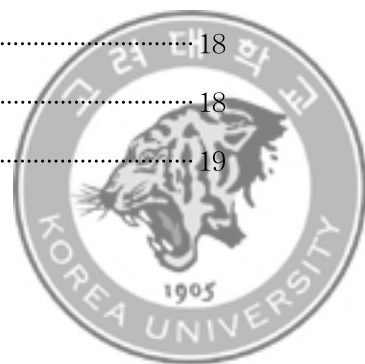


그림 25. GRU 모델 학습 코드	19
그림 26. 기본형 LSTM 모델 실행 결과 화면	20
그림 27. 알고리즘 별 학습 결과 비교	21
그림 28. 알고리즘 별 실제값과 예측값 그래프(1)	22
그림 29. 알고리즘 별 실제값과 예측값 그래프(2)	23
그림 30. 특정 간헐적 품목의 RMSE, MAE 실험 결과 그래프	24
그림 31. 전체품목 RMSE, MAE 실험 결과 그래프	25



표 목차

표 1. 실험 데이터 선별 기준	12
표 2. 실험 환경	14
표 3. 특정 간헐적 품목의 알고리즘 간 실험 결과	23
표 4. 전체 품목의 알고리즘 간 실험 결과	25



1. 서론

1.1 연구배경 및 목적

수요예측은 적정량의 재고 관리를 유지하기 위해 선행되어야 할 중요한 부분이라 할 수 있다. 특히 군수분야에서는 적정한 수리부속을 확보하는 것이 매우 중요하다. 획득된 물자로 보급, 정비 등 군수근무지원 행위가 이루어진다. 이때 물자의 과부족으로 적시에 공급이 이루어지지 않는다면 군의 전투준비태세를 떨어뜨리는 결과를 초래한다. 무기체계 수리부속의 수요예측의 정확도는 연간 예측 수요를 판단하고 예측 결과를 활용하여 적정 재고관리를 위한 중요한 근간이 된다. 정확한 예측을 할 경우 적정 재고를 유지시킬 수 있으며 효율적인 운용이 가능하다. 반면 정확도가 낮을 경우 재고 과부족 현상으로 인해 장비가동률 보장이 어렵게 되거나 과대예측으로 인해 국방 예산 손실이 발생 할 수 있다.

이는 장비가동률 보장과 및 효율적인 국방예산 운영과 연계된다는 측면에서 볼 때 중요한 부분을 차지하고 있다. 이처럼 수요예측의 정확도 향상이 적정한 재고를 유지하기 위한 토대가 된다고 할 수 있다.

하지만 간헐적인 수요는 수요예측을 어렵게 만드는 주요 원인 중 하나이다. 간헐적인 수요는 수요의 발생이 드문 경우가 빈번하고 수요 발생 간격이 불규칙한 수요를 말한다. 이러한 수요는 정규분포를 따르지 않기 때문에 기존 시계열 기법으로 예측하는데 있어 어려움이 크다.

최근 빅데이터와 GPU의 발전, 신경망 학습방법이 고도화 되면서 인공지능이 괄목할 성장을 이루고 있는 가운데 금융, 경제, 인문학 등 다양한 분야에서 기존에 어려웠던 문제를 딥러닝을 통해 해결하며 주목을 받고 있다. 딥러닝은 인공신경망을 기반으로 하는 기계학습의 한 분야로써 알고리즘 개선으로 기존에 발생했던 문제들을 극복하며 활용 가치가 높아지고 있다.

본 연구에서는 LSTM, GRU를 비롯한 RNN, MLP 딥러닝 알고리즘들을 활용하여 간헐적 품목의 수요예측을 실시하고 성능을 검증해 보았다.



1.2 관련 연구

과거 정성적인 방법을 사용하여 예측하던 시기가 있었지만 이후 이동평균법, 지수평활법과 같은 시계열 분석기법을 사용하여 예측을 시도하였다. 하지만 이는 간헐적 수요에 대한 특성을 고려하지 않은 방법으로 정확도가 낮게 나타났다.

초기에는 이동평균법, 지수평활법과 같은 시계열 분석 기법을 사용하여 분석을 하였다. 하지만 간헐적 수요는 정규분포를 따르지 않아 일반적인 시계열 방법으로는 잘 예측이 되지 않는다는 점이 있다. Croston은 지수평활법을 변형시킨 시계열 예측 방법을 제안하였다[5]. Syntetos과 Boylan은 기존 Croston 기법이 bias 문제가 발생하는 것에 대한 개선된 방법을 제시하고 향상된 정확도를 입증하였다[6]. Lev'en과 Segerstedt는 Croston 기법의 수정된 다른 개선된 대안을 제시하지만 과대예측을 할 수 있다는 문제점이 Syntetos과 Boylan에 의해 다시 제기되었다[7][8]. 그밖에 부트스트래핑 활용 및 분포를 사용하는 방법 연구가 있었으며 확률개념을 도입한 한 Croston 기법에 대한 연구도 진행되었다[10][11]. 2008년에 Gutierrez는 Neural Network를 이용하여 Lumpy 수요에 대해 기존 방법들보다 효과성이 있음을 보였다.[13] 2013년에는 Merve Şahin에 의해 Lev'en과 Segerstedt가 제안한 Croston 방법보다 MLP 방법이 간헐적 수요 패턴을 가진 데이터에서 예측오차가 더욱 작음을 입증한 사례가 있다.[16]

1.3 논문의 구성

본 연구의 구성은 다음과 같다. 1장에서는 연구배경 및 목적, 관련연구를 소개한다. 2장에서는 모델을 구축하기 위한 이론적 배경을 설명하고 3장에서는 연구 방법에 대해 설명한다. 4장에서는 실험 결과를 기법별로 비교하였으며, 5장에서는 결론 및 향후 연구에 대해 기술한다.



2. 이론적 배경

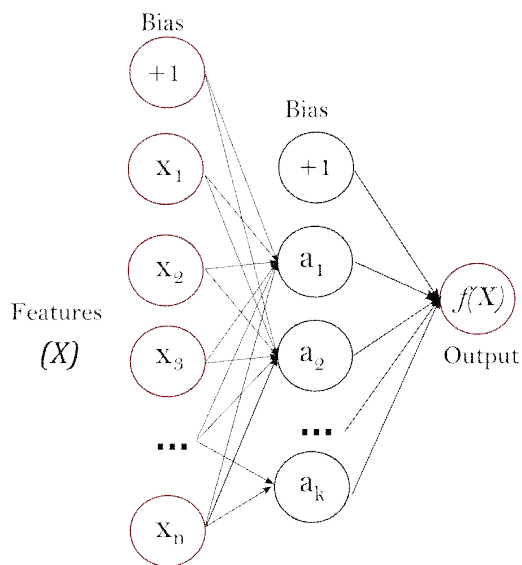
2.1 다층 퍼셉트론 (Multi-Layer Perceptron, MLP)

다층 퍼셉트론은 퍼셉트론으로 이루어진 층을 여러 개 연결한 것으로 정방향 인공신경망(Feed-Forward Deep Neural Network, FFDNN)으로 부르기도 한다.

기존에 단층퍼셉트론은 선형 분리만 가능하여 XOR에 대한 학습을 할 수 없다는 한계를 가지고 있었다. 하지만 다층 퍼셉트론은 입력층과 출력층 사이에 은닉층(Hidden Layer)을 두고 학습함으로써 선형분리가 불가능했던 데이터를 선형 분리가 가능하도록 하여 단층 퍼셉트론이 가진 한계를 극복하였다.

보통 2개 이상의 은닉층을 가지고 있을 경우 심층 신경망(Deep Neural Network)라고 하고 이러한 심층 신경망을 학습하기 위한 알고리즘들을 딥러닝(Deep Learning)이라고 한다.

다층 퍼셉트론에서 입력층에서 전달되는 값이 은닉층 노드로 전달되며 은닉층 노드에서 출력층 노드로 전달되는데 이러한 형식으로 전달되는 방식을 순전파(Feedforward)라고 한다.



<그림 2> 다층 퍼셉트론의 구조

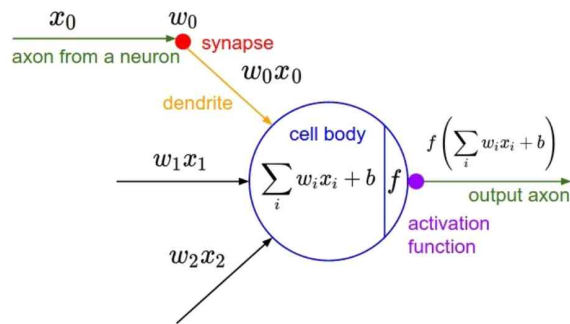


<그림 1>은 특정 요소(X)로부터 $f(x)$ 의 값이 출력되는 다층 퍼셉트론을 나타낸다. 입력층은 X_1 에서 X_n 까지며 은닉층은 A_1 에서 a_k 까지 구성되어 있다.

입력층과 은닉층에는 Bias 값을 설정하는데 보통 1로 둔다. 표현된 다층 퍼셉트론은 입력의 노드수가 n 개이며 은닉층의 노드수가 k 개, 출력층은 하나로 구성되어 있다. 이렇게 구성된 다층퍼셉트론은 $n-k-1$ 퍼셉트론으로 부른다.

다층 퍼셉트론의 동작원리는 단층 퍼셉트론과 같이 활성화함수가 내놓는 결과값과 실제값의 오차가 최소가 되도록 입력층에서 전달되는 값들에 곱해지는 가중치의 값을 결정하게 된다.

활성화 함수(activation function)는 입력신호의 총합을 그대로 사용하지 않고 출력신호로 변환하는 함수이다. 입력 신호의 총합이 활성화를 일으키는지 여부를 판단하는 역할을 한다. 입력 신호의 합이 특정값 이상일 경우 1에 가까운 숫자를 나타내도록 하는 것이 활성화 함수의 역할이다. 활성화 함수로는 비선형 함수(non-linear function)를 사용하며 어떤 활성화 함수를 선택하느냐에 따라 뉴런의 출력값이 달라질 수도 있다. <그림 2>에서 활성화함수는 Cell body로 표현된 부분이며 이를 뉴런의 형태를 수학적 모델로 표현해 놓은 것이다.



<그림 2> 활성화 함수 <출처:cs231n.github.io>

입력값의 합은 입력값에 가중치를 곱한 합이며 다음과 같이 표현된다.

$$w_0x_0 + w_1x_1 + w_2x_2$$

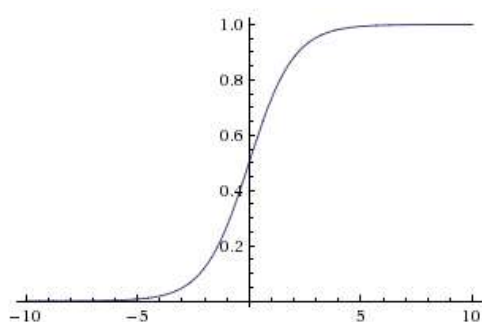


b는 임계값을 말하며 바이어스(bias)라고도 한다. 이를 입력값에 더하여 활성화 함수로부터 활성화 여부를 판단하게 된다.

$$f(\sum_i w_i x_i + b)$$

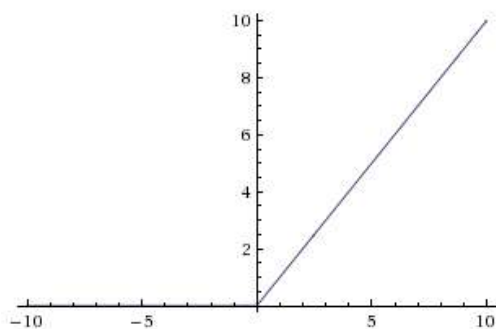
활성화 함수 종류에는 주로 sigmoid 함수와 ReLu 함수가 사용된다.

sigmoid 함수는 로지스틱 함수로도 불리며 부드러운 곡선 형태로 입력에 따라서 출력이 연속적으로 변화한다.



<그림 3> Sigmoid 함수

Relu(Rectified Linear Unit) 함수는 입력이 0을 넘게 되면 그 입력을 그대로 출력하고 0이하이면 0을 출력하는 함수이다. 다시 말해 0이하의 값을 무시해버린다. 입력이 작을 때의 출력은 0에 가깝고 입력이 클 때의 출력은 1에 가깝다.

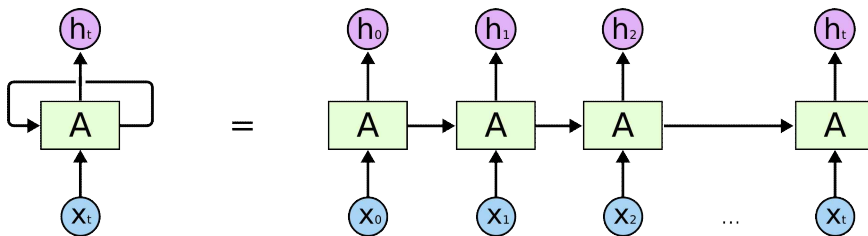


<그림 4> ReLu 함수



2.2 순환 신경망 (Recurrent Neural Network, RNN)

순환 신경망(Recurrent Neural Network, RNN)은 순차적인 데이터를 학습하는 인공신경망의 방법 중 하나로써 각 메모리 블록에는 특정 시점의 데이터 정보가 저장되고, 이를 다음 시점으로 전달하는 순환구조로 이루어져 있다. 기존의 인공신경망은 각 층이 뉴런으로 연결되어 있는 구조였다면 순환 신경망은 과거 자신의 정보의 가중치를 기억하고 이를 다음 단계 학습 시 반영하게 된다.



<그림 5> RNN 구조 <출처:colah.github.io>

인공 신경망과 다르게 순환 신경망은 은닉층의 데이터를 저장하고 있으며 경사 하강법(gradient descent)과 역전파(backpropagation)를 이용해 학습을 한다.

시퀀스 길이에 관계없이 입력과 출력을 받아들일 수 있는 구조이기 때문에 필요에 따라서 다양하고 유연하게 구조를 만들 수 있다는 점이 순환 신경망의 가장 큰 장점이라 볼 수 있다.

기존의 신경망(Neural Network)이 역전파(Backpropagation) 알고리즘을 사용했지만 순환 신경망은 시간의 흐름에 따라 BPTT(Back-Propagation Through Time)을 사용한다.

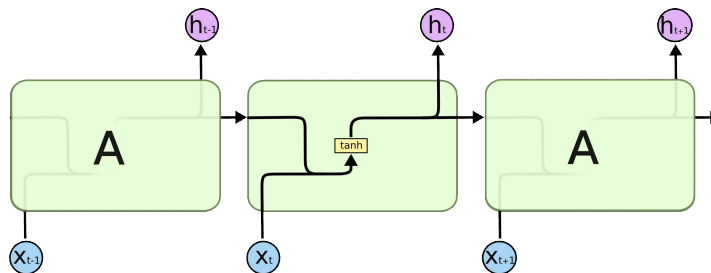
이러한 순환 신경망(RNN)의 특징을 적용시켜 음성인식, 언어모델링, 번역, 작곡 등 많은 분야에서 쓰이고 있으며 순차적인 시계열 데이터에서도 강력한 성능을 보여준다.



2.3 장단기 기억 메모리 (Long-Short Term Memory, LSTM)

LSTM은 RNN의 가지고 있는 장기 의존성 문제를 해결한 발전된 알고리즘으로 1997년에 Hochreiter 와 Schmidhuber에 의해 제안되었다. 현재 대중화되면서 다양한 문제에 적용되기 시작하였으며 많은 분야에서 사용되고 있다.

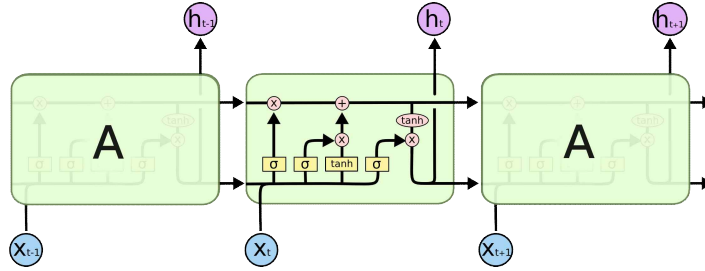
기존 순환 신경망(RNN)은 시간이 뒤로 갈수록 과거의 gradient가 점차 줄어들어 학습능력이 크게 저하되는 문제점이 발생한다. 이러한 단점으로 인해 그라디언트 소실(vanishing gradient problem)이 발생하며 이를 장기 의존성 문제라고 한다. 이를 해결하기 위해 메모리를 도입한 방법이 장단기 기억 메모리이며 LSTM으로 불린다. LSTM은 장기 의존성을 학습할 수 있는 특별한 종류의 신경망이다. LSTM은 장기 의존성 문제를 피하기 위해 설계되었고 오랫동안 기억을 보존하기 위함이 LSTM의 기본 동작이다.



<그림 6> RNN의 모듈 구조 <출처:colah.github.io>

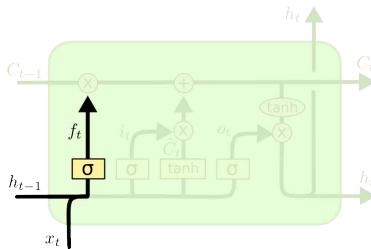
모든 일반적인 순환신경망(RNN)은 사슬형태의 반복되는 신경망 모듈로 구성되어 있으며 각 모듈에는 한 개의 tanh층을 가진 단순한 구조로 이루어져 있다.





<그림 7> LSTM의 모듈 구조 <출처:colah.github.io>

RNN과 다르게 LSTM에서 반복되는 모듈은 4개의 상호작용하는 층을 포함하고 있으며 각 모듈을 통해 지나가는 셀 상태(cell state)가 핵심적인 요소로 작용한다. LSTM은 셀 상태에 정보를 더하거나 혹은 지움으로써 각 게이트에서 정보가 선택적으로 지나가게 한다. 게이트는 sigmoid 신경망 층과 각 요소별 곱셈 연산이 이루어지며 sigmoid 층에서 각 출력값이 얼마나 많은 요소가 통과되어야 하는지를 0에서 1사이의 숫자로 출력하게 된다. LSTM에서는 세 종류의 gate를 통해 셀 상태를 제어한다.

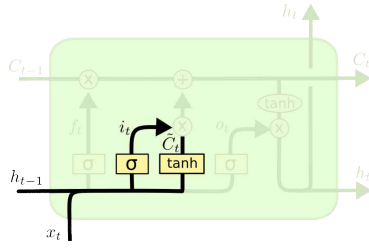


$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

<그림 8> LSTM의 망각 게이트(Forget gate)층 <출처:colah.github.io>

sigmoid층의 하나인 망각 게이트(forget gate)층에서 셀 상태에서 잊어버릴 정보를 결정하게 된다. f_t 는 과거정보를 잊기 위한 망각 게이트로써 h_{t-1} 과 x_t 를 통해 sigmoid를 취해준 값을 내보내게 된다. 출력이 0에 가까울수록 데이터를 잊어버리고 1에 가까울수록 이전 데이터를 그대로 전달하게 된다.



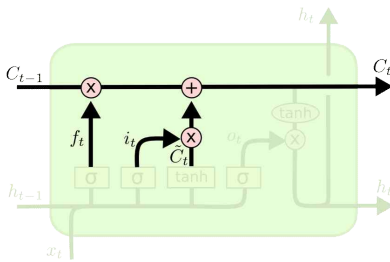


$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

<그림 9> LSTM의 입력 게이트(Input gate)층 <출처:colah.github.io>

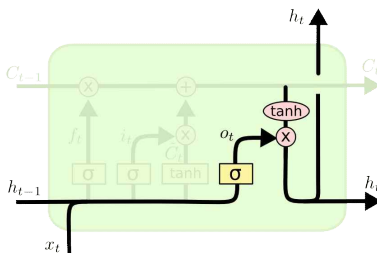
다음 단계에서는 어떠한 새로운 정보를 셀 상태에 저장할지 여부를 결정한다. 입력게이트 층 t_t 은 새로운 값을 갱신할지를 결정하고, tanh 층은 셀 상태에 더해질 수 있는 새로운 결정할 후보값 \tilde{C}_t 를 만든다.



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

<그림 10> LSTM의 셀 상태(cell state) 값 <출처:colah.github.io>

셀 상태 C_t 는 기존의 셀 상태 C_{t-1} 에 망각 게이트 f_t 의 출력값을 곱한 값과 입력값 i_t 와 이전 값의 출력값을 처리한 결과 \tilde{C}_t 를 곱한 값을 더하여 새로운 값을 만들게 된다.



$$o_t = \sigma(W_o [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh(C_t)$$

<그림 11> LSTM의 셀의 출력값 <출처:colah.github.io>



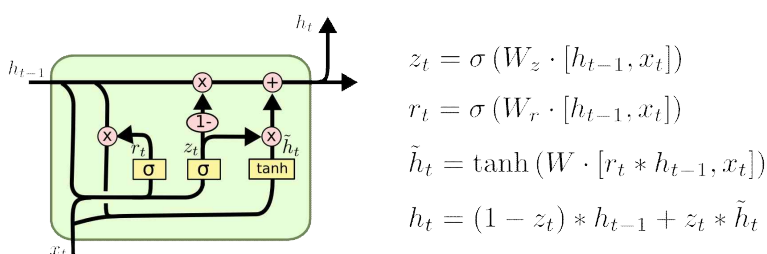
출력게이트는 무엇을 출력할지를 결정한 sigmoid 값 o_t 를 산출하고, 0에서 1사이의 값을 갖도록 한 tanh에 셀 상태 C_t 를 넣고 이 둘을 서로 곱하여 새로운 출력 값 h_t 을 생성한다.

기존 RNN에서 BPTT를 통한 연산 중 chain rule에 의한 연결의 길이가 매우 길어지게 되고 깊은 신경망에서 발생하는 vanishing gradient 문제가 있었지만, LSTM의 장기간 메모리가 필요한 문제를 해결함으로써 RNN의 단점을 극복하게 되었다.

이러한 장점으로 인해 LSTM은 시계열 데이터 처리에 있어서 ARIMA와 같은 기존 시계열 기법보다 좋은 성능을 내는 것으로 알려져 있다.

2.4 게이트 된 순환 유닛(Gated Recurrent Unit, GRU)

GRU는 LSTM의 변형 모델 중 하나로 2014년에 처음 소개되었으며 LSTM이 가진 장점을 유지하면서 계산의 복잡성을 낮춘 구조를 가지고 있다. Gradient Vanishing문제를 극복한 LSTM의 장점을 유지하면서 게이트의 일부를 생략하고 update gate(z_t)와 reset gate(r_t) 두 가지만 가지고 있다.



<그림 12> GRU의 모듈 구조 <출처:colah.github.io>

update gate(z_t)에 LSTM이 가지고 있던 forget gate와 input gate를 하나로 통합을 하고 별도의 셀 상태와 hidden state를 하나의 hidden state로 묶었다. 이와 같은 구조로 단순한 구조를 갖게 되었음에도 불구하고 LSTM에 거의 근접한 결과물을 나타내고 있어 관심을 받고 있는 알고리즘이다.



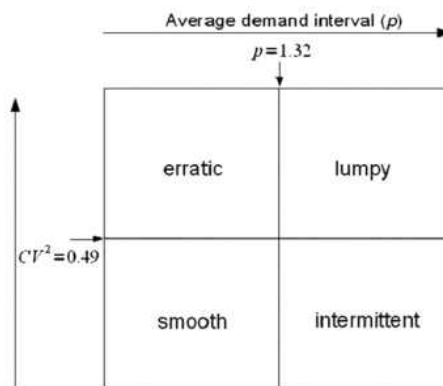
3. 연구 방법

3.1 실험 데이터 및 전처리

본 연구에서는 해군 장비정비정보체계 DELIIS¹⁾의 특정 함정 장비를 대상으로 정비실적에 근거한 소모실적 월별 수요 자료를 활용하여 분석을 실시하였다.

기간은 2010년부터 2016년까지의 7개년 간 데이터를 사용하였으며 데이터 유형은 월별 시계열 자료로써 품목별 소모 수량을 Raw Data로 활용하였다.

간헐적 품목은 수요발생유형을 구분 기준에 따라 평균수요발생구간(ADI)와 변동계수(CV)를 사용하여 식별하였다.



<그림 13> 수요발생 유형 구분

수요발생 유형을 <그림 13>과 같이 구분할 수 있다. 원만한 수요(smooth demand)는 수요발생 기간과 수량이 규칙적으로 발생하여 예측하기 쉬운 품목이며 간헐적인 수요(intermittent demand)는 수요수량의 변동이 거의 없지만 수요 간격이 크게 변화한다. 불규칙한 수요(Erratic demand)는 수요 간격은 불규칙한 편은 아니지만 수요량이 불규칙하게 발생하는 품목이며 무더기 수요(Lumpy demand)는 수요발생 간격도 크며 특정시기에 수요량이 불규칙하게 발생하는 품

1) DELIIS : 군 장비정비정보체계로써 정비 및 보급, 조달에 관련 업무를 관리하는 체계 (KIDA)



목이다.

수요발생구간(Average Demand Interval, ADI)는 수요발생 간 평균 간격을 나타내며 변동계수 (coefficient of variation, CV)는 수요의 표준편차를 평균으로 나눈 값으로 산출된다.

$$ADI = \frac{\sum_{i=1}^N t_i}{N} \quad (3.1)$$

$$CV = \frac{\sqrt{\frac{\sum_{i=1}^N (\epsilon_i - \epsilon)^2}{N}}}{\epsilon} \quad (3.2)$$

실험에 포함한 적용한 간헐적 수요 범위는 Intermittent 수요와 Lumpy 수요를 간헐적 수요로 구분 짓고 실험을 실시하였다.

분류기준에 따라 전체 품목 중 간헐적인 수요패턴을 가진 16,848품목에 대한 84개월 시계열 데이터 총 1,415,232레코드를 추출하였으며 이 중 실험 표본 300품목에 대한 84개월 시계열 데이터 25,000레코드를 선별하였다.

학습 데이터(Training Data)는 2010년 1월부터 2014년 12월까지 사용하고 테스트 데이터(Test Data)는 2014년 1월부터 2016년 12월까지의 데이터를 사용하였다.

<표 1> 실험 데이터 선별 기준

데이터 선별	기간	개월	비율
Training Data	2010.01~2014.12	60	71.4%
Test Data	2014.01~2016.12	24	28.6%



MODEL_NMIN	MONTH	QY
001247370	201412	0
001247370	201501	9
001247370	201502	0
001247370	201503	2
001247370	201504	0
001247370	201505	1
001247370	201506	0
001247370	201507	0
001247370	201508	0
001247370	201509	4
001247370	201510	1
001247370	201511	2
001247370	201512	1
001247370	201601	2
001247370	201602	0
001247370	201603	0
001247370	201604	0
001247370	201605	7
001247370	201606	3
001247370	201607	0
001247370	201608	0
001247370	201609	0
001247370	201610	4
001247370	201611	0
001247370	201612	3

<그림 14> 간헐적 품목 월간 수요 데이터 조회 현황

<그림 14>는 수요 RAW 데이터로써 2010년 1월부터 2016년 12월까지의 7개년 간 간헐적 품목의 월간 수요 데이터 조회 현황이다.

MODEL_NMIN	DEMAND_INTERVAL	DEMAND_CNT	ADI	CV
12F127334	84	57	1.4737	0.7735
12F147052	83	27	3.0741	2.062
12F140493	34	1	34	0
12F174937	83	18	4.6111	0.7428
12F175921	83	12	6.9167	0.426
12F176568	4	1	4	0
12F177943	83	5	16.6	0.1172
12F178788	14	1	14	0
12F176548	82	20	4.1	0.4969
12F178573	74	3	24.6667	0
12F178143	81	28	2.8929	4.5064
12F177555	60	1	60	0
12F173598	55	3	18.3333	0.1875
12F170945	68	5	13.6	0.4066
12F170053	34	3	11.3333	0.8148
12F173388	78	8	9.75	0.6297
12F173392	84	27	3.1111	0.4994
12F170069	78	15	5.2	0.7259
12F170354	10	1	10	0
12F170989	83	2	41.5	0.125

<그림 15> 간헐적 품목 ADI, CV 데이터 조회 현황

<그림 15>는 월간 수요 데이터로부터 ADI, CV값을 생성한 후 간헐적 품목들에 대한 조회 현황을 나타낸 것이다.



3.2 실험 환경

실험을 실시한 환경은 <표 2>와 같다.

<표 2> 실험 환경

Category	Value
CPU	intel i7-4770 3.40GHz
Memory	8.00GB
OS	Window 10 64bit
Language	Python 3.5
Library	Keras 2.0.9

3.3 평가 방법

본 연구에서는 성능 평가를 위하여 각각의 예측 오차값에 대표적으로 측정되는 RMSE(Root Mean Square Error)와 MAE(Mean Absolute Error)를 사용하였다.

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y})^2} \quad (3.3)$$

n은 예측치의 개수를 의미하며 y_i 는 실측치, \hat{y}_i 는 예측치를 의미한다.

RMSE는 평균제곱근오차로 정량적으로 예측값과 실제값의 차이의 제곱한 값을 합하여 예측치의 개수로 나눈 값에 제곱근을 취한 것이다. 제곱근을 사용함에 따라 오차의 편차가 심할수록 값이 커지게 된다.

$$MAE = \frac{1}{n} \sum_{j=1}^n |y_j - \hat{y}_j| \quad (3.4)$$

n은 예측치의 개수를 의미하며 y_j 는 실측치, \hat{y}_j 는 예측치를 의미한다.

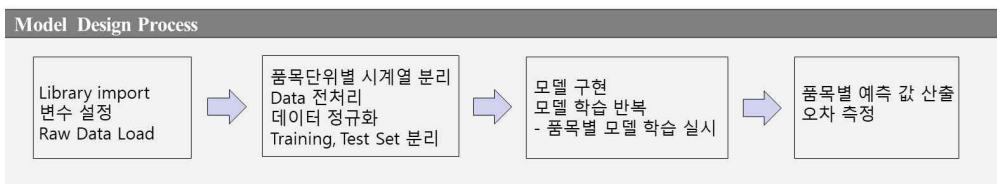
MAE는 절대평균오차로 예측치와 실측치 차이의 오차의 절대값으로 표시하고 이를 합하여 예측치의 개수만큼 나누는 방식이다.

오차가 0에 가까울수록 예측치가 실측치에 가깝다는 것을 의미한다.



3.4 모델 설계

모델의 설계 순서의 첫 번째 단계는 Library import 과정과 변수설정, 그리고 Raw data를 읽어온다. 두 번째 단계에서 품목단위별 시계열 데이터를 분리하는 작업을 통해 품목별에 대한 데이터 전처리와 정규화 과정을 거친 후 학습 데이터와 테스트 데이터로 분리하였다. 세 번째 단계에서 각 60개의 시계열 데이터에 대한 각 모델의 학습을 품목별로 실시한다. 마지막으로 각 품목에 대한 24개월에 대한 예측값을 산출한 후 오차를 측정하도록 설계하였다.



<그림 16> 모델 설계 순서

실험에 사용한 Python Library는 <그림 17>과 같다. 딥러닝 알고리즘을 위한 Keras, 데이터 처리를 위한 Pandas, 통계 처리를 위한 Library 등을 사용하였다.

```
import numpy
import matplotlib.pyplot as plt
import pandas as pd
import math
import time
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from pandas import DataFrame, Series
```

<그림 17> Python Import Library



본 논문에서는 딥러닝을 이용한 시계열 데이터 분석을 위해 MLP, RNN, LSTM, GRU 모델 대상으로 실험을 실시하였다.

다층 퍼셉트론(Multi-layer perceptron, MLP) 모델은 일반적인 다층 퍼셉트론 신경망 모델과 2개의 은닉층을 가진 심층(Deep) 다층 퍼셉트론 신경망 모델을 설계하였다.

다층 퍼셉트론의 모델 학습을 위한 설계 방법은 다음과 같다.

다층 퍼셉트론의 모델의 첫 번째 Layer는 8개의 뉴런을 가지도록 구성하였으며 활성화 함수는 오류역전파가 용이한 relu 함수를 사용하였다. 두 번째 Layer는 하나의 예측값을 나타내기 위한 1개의 뉴런을 가지며 별도의 활성화 함수는 사용하지 않았다.

```
# create and fit Multilayer Perceptron model
model = Sequential()
model.add( Dense( 8, input_dim=look_back, activation='relu' ) )
model.add( Dense( 1 ) )
model.compile( loss='mean_squared_error', optimizer='adam', metrics=["accuracy"] )
hist = model.fit( trainX, trainY, epochs=200, batch_size=1, verbose=2, validation_data=(testX, testY))
```

<그림 18> 다층퍼셉트론 모델 학습 코드

심층 다층 퍼셉트론의 모델은 총 3개의 Layer로 구성하였으며 첫 번째, 두 번째 Dense Layer는 각각 8개의 뉴런을 가진다. 활성화함수는 relu 함수를 사용하였다. 출력 Layer는 하나의 수치값을 예측하기 위해 1개의 뉴런을 가지도록 구성하였다.

```
# create and fit Deep Multilayer Perceptron model
model = Sequential()
model.add( Dense( 8, input_dim=look_back, activation='relu' ) )
model.add( Dense( 8, activation='relu' ) )
model.add( Dense( 1 ) )
model.compile( loss='mean_squared_error', optimizer='adam', metrics=["accuracy"] )
hist = model.fit( trainX, trainY, epochs=200, batch_size=1, verbose=2, validation_data=(testX, testY))
```

<그림 19> 심층 다층퍼셉트론 모델 학습 코드



순환 신경망(RNN) 모델 학습을 위한 설계 방법은 다음과 같다.

단순한 순환 신경망(RNN) 모델로 4개의 뉴런이 각각의 RNN Block을 구성하고 인수의 벡터는 한 개의 출력 값을 가지도록 설계하였다. 손실 함수로는 mean-square-error를 사용하였고 최적화 방법으로 adam을 사용하였다.

```
# create and fit the Simple RNN network
model = Sequential()
model.add(SimpleRNN(4, input_dim=1, input_length=3))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam', metrics=["accuracy"])
hist = model.fit(trainX, trainY, epochs=200, batch_size=1, verbose=2, validation_data=(testX, testY))
```

<그림 20> 순환신경망 모델 학습 코드

장단기 기억메모리(LSTM) 모델 실험은 다양한 설정 방법을 통하여 다각도 분석을 실시하였다.

기본형 장단기 기억메모리(LSTM) 모델 학습을 위한 설계 방법은 <그림 21>과 같다. 네트워크에는 1개의 입력을 가진 Layer, 4개의 메모리 블록과 뉴런이 있는 숨겨진 Layer, 단일 값을 예측하는 출력 Layer로 구성되어 있다. 활성화 함수는 기본적으로 많이 사용하는 Sigmoid를 사용하였다.

```
look_back = 1
trainX, trainY = create_dataset(train, look_back)
testX, testY = create_dataset(test, look_back)

trainX = numpy.reshape(trainX, (trainX.shape[0], 1, trainX.shape[1]))
testX = numpy.reshape(testX, (testX.shape[0], 1, testX.shape[1]))

model = Sequential()
model.add(LSTM(4, input_shape=(1, look_back)))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam', metrics=["accuracy"])
hist = model.fit(trainX, trainY, epochs=200, batch_size=1, verbose=2, validation_data=(testX, testY))
```

<그림 21> 기본형 LSTM 모델 학습 코드



메모리 단위 LSTM 예측을 실시하기 위해 최근의 데이터를 일정 기간씩 묶어서 다음 단계에 적용시키는 방법으로 시계열 데이터를 배열의 메모리로 구분한 뒤 순차적으로 각 기간의 데이터를 다음 예측할 메모리에 적용시키는 방법이다.

```
trainX, trainY = create_dataset(train, look_back)
testX, testY = create_dataset(test, look_back)

trainX = numpy.reshape(trainX, (trainX.shape[0], 1, trainX.shape[1]))
testX = numpy.reshape(testX, (testX.shape[0], 1, testX.shape[1]))

model = Sequential()
model.add(LSTM(4, input_shape=(1, look_back)))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')
hist = model.fit(trainX, trainY, epochs=200, batch_size=1, verbose=2)
```

<그림 22> 메모리 단위 LSTM 모델 학습 코드

상태유지 LSTM 모델은 현재 학습된 상태가 다음 학습 시 초기 상태로 전달되며 시퀀스 데이터의 길이가 길수록 성능을 발휘하게 된다. 긴 시퀀스 데이터를 잘라서 학습하더라도 LSTM의 내부적으로 기억을 유지해야 할 중요한 정보만을 이어갈 수 있도록 상태가 유지되게 된다. 이 모델은 LSTM 계층 내 stateful 옵션을 상태유지 시킴으로써 Keras의 LSTM 네트워크의 내부의 상태가 지워지는 것을 좀 더 세밀하게 제어할 수 있다.

```
batch_size = 1
model = Sequential()
model.add(LSTM(4, batch_input_shape=(batch_size, look_back, 1), stateful=True))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')
for i in range(200):
    model.fit(trainX, trainY, epochs=1, batch_size=batch_size, verbose=2, shuffle=False)
    model.reset_states()
```

<그림 23> 상태유지 LSTM 모델 학습 코드



상태유지 스택 LSTM 모델은 상태유지 신경망을 여러 층으로 쌓아올린 형태로써 순환 신경망 모델과 동일하나 LSTM의 장점을 이용하여 deep network 구조를 활용한 메모리 배열이다. stateful=True, return_sequence=True를 설정한 후 심층 LSTM 모델을 구성하였다.

```
batch_size = 1
model = Sequential()
model.add(LSTM(4, batch_input_shape=(batch_size, look_back, 1), stateful=True, return_sequences=True))
model.add(LSTM(4, batch_input_shape=(batch_size, look_back, 1), stateful=True))
model.add(Dense(1))

model.compile(loss='mse', optimizer='adam', metrics=['mse', 'mae'])
for i in range(200):
    model.fit(trainX, trainY, epochs=1, batch_size=batch_size, verbose=2, shuffle=False)
    model.reset_states()
```

<그림 24> 상태유지 스택 LSTM 모델 학습 코드

GRU 모델은 내부 메모리 정보를 저장하지 않고 output gate 없이 reset gate와 update gate만을 이용한다. 네트워크 구성은 1개의 입력을 가진 Layer, 4개의 메모리 블록과 뉴런이 있는 숨겨진 Layer, 단일 값을 예측하는 출력 Layer로 구성하였다.

```
model = Sequential()
model.add(GRU(4, input_shape=(look_back, 1)))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam', metrics=["accuracy"])
hist = model.fit(trainX, trainY, epochs=200, batch_size=1, verbose=2, validation_data=(testX, testY))
```

<그림 25> GRU 모델 학습 코드

각 모델 실험 시 공통적으로 적용한 사항은 학습 epoch를 200, batch size=1, input dim=1로 설정하였으며 optimizer 알고리즘은 adam을 사용하였다. 또한 학습 Loss 측정 지표로써 mean square error를 사용하였다.



4. 결과

4.1 학습 결과

각 알고리즘별 손실함수를 측정하였다. 손실함수(loss function)는 비용함수(cost function)라고도 불리며 지도학습을 진행하며 실측치에 대한 에러(error)를 계산하고 정답에 가까울수록 작은 값이 나온다. 손실함수 측정 지표로써 MSE를 사용하였다.

학습 시 epoch은 학습의 횟수를 나타내며 batch size는 1회 학습 시 데이터를 처리하는 데 사용되는 메모리의 개념이다. 손실함수의 곡선을 확인하면서 적당한 수의 epoch를 정하는 것이 필요하다. batch size가 클수록 한 번에 데이터를 처리하는 양이 많아지게 되지만 정확도가 낮아질 수 있다. 반대로 batch size가 작을수록 한 번에 처리되는 양이 최소화됨으로써 정확도가 향상되지만 학습 시간이 오래 걸리게 된다. 따라서 적절한 epoch와 batch size를 정하는 것이 중요하다고 할 수 있다. <그림 26>은 기본형 LSTM 모델 실행 결과 화면이다.

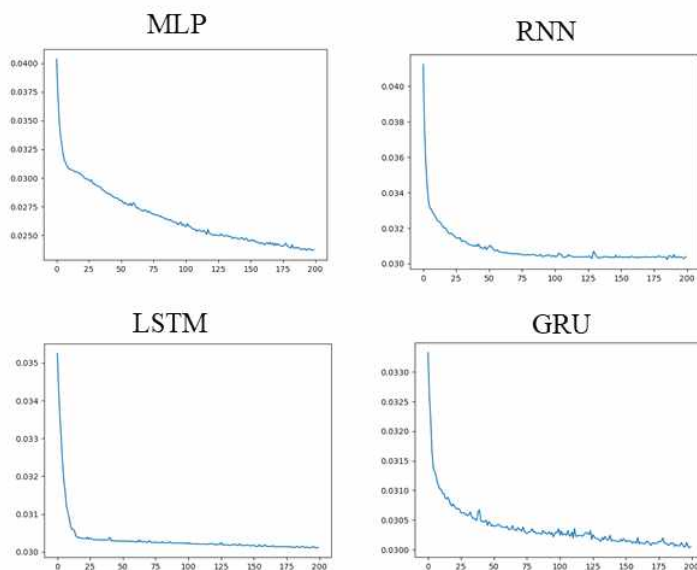
```
Epoch 194/200
- 0s - loss: 0.0301 - acc: 0.4643 - val_loss: 0.0380 - val_acc: 0.3333
Epoch 195/200
- 0s - loss: 0.0301 - acc: 0.4643 - val_loss: 0.0380 - val_acc: 0.3333
Epoch 196/200
- 0s - loss: 0.0301 - acc: 0.4643 - val_loss: 0.0381 - val_acc: 0.3333
Epoch 197/200
- 0s - loss: 0.0301 - acc: 0.4643 - val_loss: 0.0380 - val_acc: 0.3333
Epoch 198/200
- 0s - loss: 0.0301 - acc: 0.4643 - val_loss: 0.0381 - val_acc: 0.3333
Epoch 199/200
- 0s - loss: 0.0301 - acc: 0.4643 - val_loss: 0.0381 - val_acc: 0.3333
Epoch 200/200
- 0s - loss: 0.0301 - acc: 0.4643 - val_loss: 0.0380 - val_acc: 0.3333

Train Score: 16.6539 RMSE , 2.9603 MAE
Test Score: 18.7197 RMSE , 3.2415 MAE
```

<그림 26> 기본형 LSTM 모델 실행 결과 화면



<그림 27>은 각 MLP, RNN, LSTM, GRU 학습 과정의 수렴 결과이다. 알고리즘 별로 일관성 있는 비교를 위해 하나의 간헐적 품목을 선택하여 결과를 분석해 보았다.



<그림 27> 알고리즘 별 학습 결과 비교

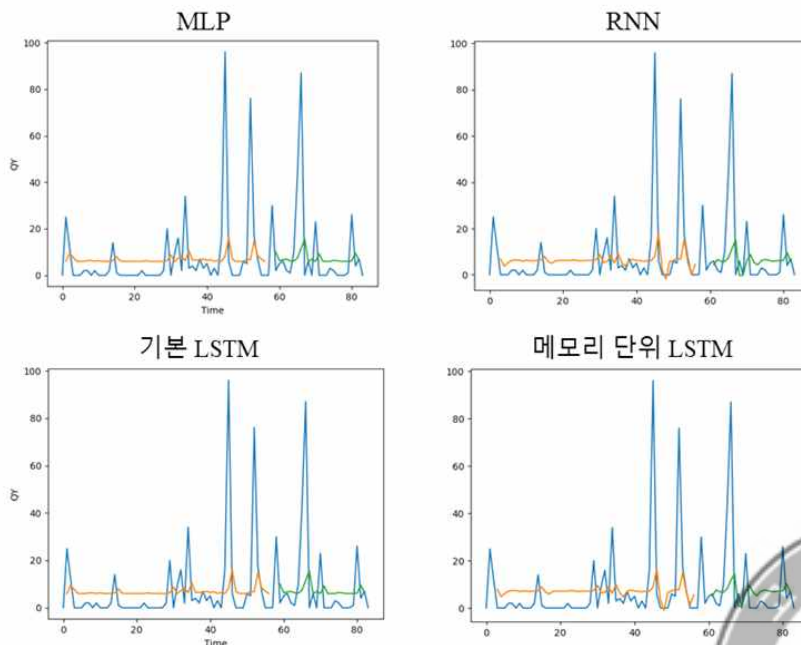
학습은 200epoch 동안 진행되었으며 LSTM이 학습 횟수에 비해 손실률이 가장 적고 빠르게 수렴해 나갔다. LSTM이 RNN, GRU 보다 학습이 근소한 차이로 빠르게 수렴되었다. 상대적으로 MLP는 다른 알고리즘에 비해 학습이 다소 늦게 수렴해가는 것을 확인 할 수 있다.



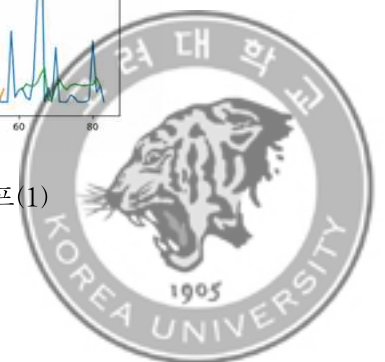
4.2 결과 검증 및 평가

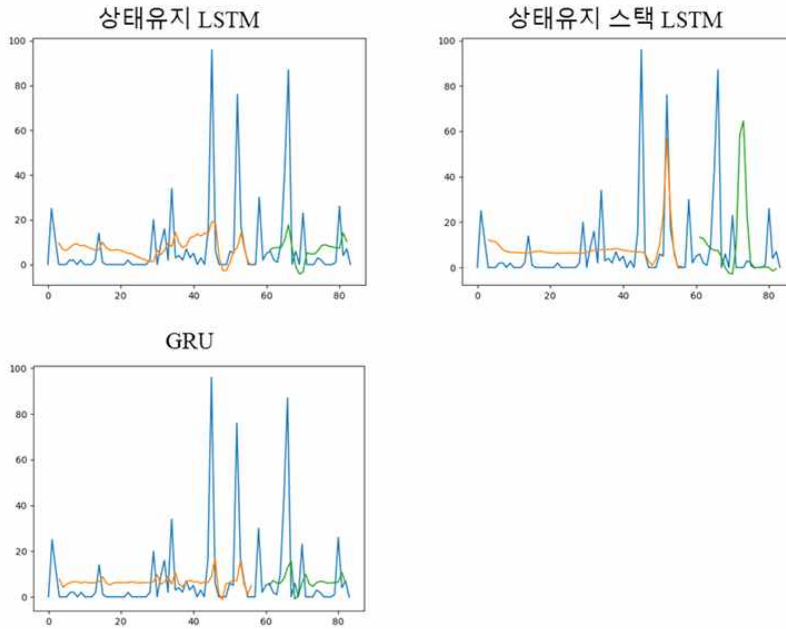
실험은 총 간헐적 품목 총 16,847품목 중 300품목을 대상으로 실험을 실시하였다. 일관성 있는 알고리즘 간 비교를 위해 품목 단위에 대한 결과 분석과 전체 품목에 대한 결과를 각각 분석하였다. 전체 품목에 대한 결과 비교는 품목들의 예측 오차들의 평균으로 분석하였다.

각 알고리즘별 학습과정을 실시 후 학습 데이터 기간 이후 테스트 데이터 시작 시점부터 예측 값을 표시하도록 하였다. 성능 평가를 위해 72.4:24.6의 비율로 학습 데이터와 테스트데이터를 구분하였고 품목별 총 84개의 시계열데이터를 활용하였다. 2010년 1월부터 2014년 12월까지의 60개 시계열 데이터를 이용하여 학습을 실시하였으며 2015년 1월부터 2016년 12월까지 24개의 시계열 예측을 실시하였다. 파란색으로 표시된 선은 전체 시계열 기간의 실제 값을 나타내며 노란색 선은 학습을 실시한 데이터이며 녹색 선이 예측 값을 나타낸다. <그림 28>은 간헐적 품목에 대한 알고리즘 간의 실제 값과 예측 값의 데이터를 그래프로 나타낸 것이다



<그림 28> 알고리즘 별 실제값과 예측값 그래프(1)



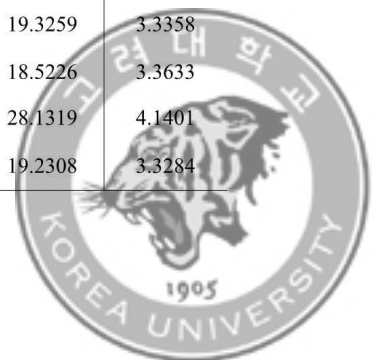


<그림 29> 알고리즘 별 실제값과 예측값 그래프(2)

MLP는 레이어가 총 3개인 다층퍼셉트론 모델로 설계하였고 LSTM은 메모리 셀 및 상태유지 세부 옵션을 조정해가면서 4가지 모델을 설정하여 실험을 실시하였다. <표 3>은 특정 간헐적 품목의 알고리즘 별 실험 결과를 나타낸 것이다.

<표 3> 특정 간헐적 품목의 알고리즘 간 실험 결과

Algorithm	Method	Estimation			
		Training RMSE	Training MAE	Test RMSE	Test MAE
MLP	Multi Layer Perceptron	14.7541	2.7675	19.7426	3.4931
RNN	Recurrent Neural Network	16.7255	2.8778	19.4585	3.3132
LSTM	기본형 LSTM	16.6538	2.9602	18.7193	3.2415
	메모리 단위 LSTM	16.6999	2.9672	19.3259	3.3358
	상태유지 LSTM	16.0317	2.9948	18.5226	3.3633
	상태유지 스택 LSTM	14.5328	2.8603	28.1319	4.1401
GRU	Gated Recurrent Unit	16.6358	2.8980	19.2308	3.3284



<그림 30>은 <표 3>에 대한 알고리즘 간 예측오차를 비교해 놓은 그래프이다. LSTM의 결과는 성능이 가장 좋은 기본형 LSTM으로 비교하였다.



<그림 30> 특정 간헐적 품목의 RMSE, MAE 실험 결과 그래프

성능 비교를 위해 알고리즘 간 테스트 데이터의 예측값 RMSE와 MAE를 비교하였다.

<표 3>에서 볼 수 있듯이 기본형 LSTM의 RMSE가 18.7193, MAE가 3.2415로 예측 오차의 값이 다른 알고리즘에 비해 가장 낮아 성능이 가장 우수한 결과도출 되었다. 간헐적인 수요에 대해 24개의 예측 기간 중 가장 낮은 수량 오차를 보이며 안정적인 수요예측을 나타냈다고 볼 수 있다.

메모리 단위의 LSTM이나 상태유지 LSTM은 RNN, GRU와 비슷한 수준의 성능을 보였으나 상태유지 스택 LSTM은 모든 알고리즘들 중에 가장 큰 예측오차를 보였다. Training RMSE가 14.5328로 다른 알고리즘에 비해 낮아 학습 데이터가 과적합 된 예측 모형이 설계되어 Test RMSE가 28.1319로 다소 높은 오차값을 나타내게 된 것으로 추정된다. <그림 30>에서 상태유지 스택 LSTM의 학습 모형 중 간헐적으로 크게 나타나는 50번째 수요에 대해 과적합되었고 그 결과 예측 수량의 오차가 큼을 알 수 있다. GRU는 기본형 LSTM에 비해 성능이 약간 못 미치지만 거의 근접한 성능을 보였고, MLP나 RNN보다 예측 오차가 적은 것으로 나타났다. MLP와 RNN 간의 비교에서 RNN의 예측 오차가 작게 나타나 성능이 우위에 있음을 알 수 있다.

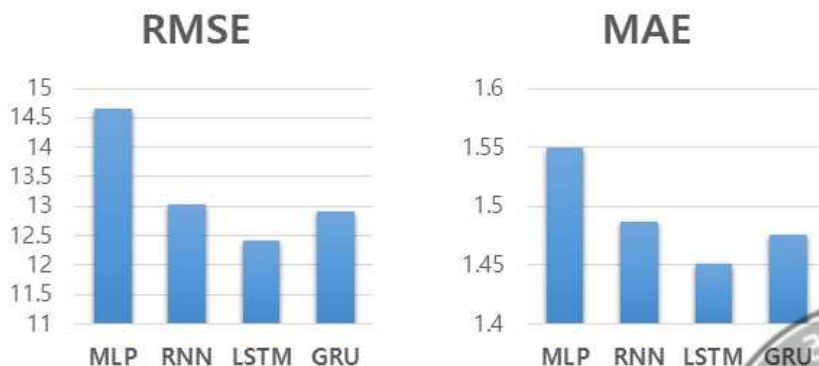


알고리즘의 공정한 평가를 위해서 실험을 실시한 300품목에 대해 RMSE, MAE의 평균값으로 비교하였다. <표 4>은 실험에 사용된 전체 품목에 대한 MLP, RNN, LSTM, GRU 간 실험 결과를 나타낸 것이다.

<표 4> 전체 품목의 알고리즘 간 실험 결과

Algorithm	Method	Estimation			
		Training RMSE	Training MAE	Test RMSE	Test MAE
MLP	<i>Multi Layer Perceptron</i>	16.4335	1.5753	14.6483	1.5490
RNN	<i>Recurrent Neural Network</i>	12.8622	1.4303	13.0347	1.4864
LSTM	<i>기본형 LSTM</i>	13.0335	1.4378	12.4288	1.4508
	<i>메모리 단위 LSTM</i>	12.5584	1.4078	14.2795	1.5089
	<i>상태유지 LSTM</i>	12.7676	1.4065	12.5133	1.4662
	<i>상태유지 스택 LSTM</i>	12.7618	1.4078	13.1161	1.5325
GRU	<i>Gated Recurrent Unit</i>	12.9233	1.4176	12.9023	1.4752

<그림 31>은 <표 4>에 대한 알고리즘 간 예측오차를 비교해 놓은 그래프이다. LSTM의 결과는 성능이 가장 좋은 기본형 LSTM으로 비교하였다.



<그림 31> 전체품목 RMSE, MAE 실험 결과 그래프



전체 품목을 대상으로 실험한 결과도 앞서 분석한 품목 단위에 대한 결과와 동일하게 나타났다. 실험 결과를 통해 기본형 LSTM의 예측 오차(RMSE: 14.4288, MAE: 1.4508)가 다른 알고리즘들 보다 낮게 나타나 성능이 가장 우수함을 확인할 수 있다. RNN이 가지고 있는 gradient 소실문제를 해결한 LSTM이 RNN보다 오차가 상대적으로 낮게 나타난 것으로 추정된다. 반면 LSTM 간 비교에서 옵션을 변화시킨 메모리 단위 LSTM이나 상태유지 LSTM은 기본형 LSTM에 비해 뛰어난 결과를 보이지 않았다. 두 번째로 우수한 성능을 보인 GRU가 기본형 LSTM에 근접한 성능을 보였다. GRU는 LSTM과 유사하지만 출력게이트를 생략한 형태의 단순한 구조로 이루어져 LSTM과 거의 동일한 성능을 보인다고 알려져 있다. 실험을 통해서도 이와 같은 사실을 확인할 수 있다. 앞선 결과를 종합해 볼 때 순환신경망 알고리즘들이 시계열 데이터와 같은 순차적 데이터를 분석함에 있어서 MLP보다 높은 성능을 보였다고 볼 수 있다. 이는 과거의 상태를 저장하는 메모리를 가지고 있는 순환신경망이 시계열 데이터 분석에 더 적합하다는 것이 실험을 통해서 나타난 것이라 볼 수 있다. 즉, 시간의 의존성을 갖고 있는 문제 해결에 더 적합하다고 판단된다. 간헐적 수요는 일반적인 패턴이 없이 극단적으로 수요가 갑자기 발생하는 경우가 일어나는 경우가 많다. 순환 신경망 알고리즘 중 LSTM은 극단적인 수요 변동에 대해서도 유연하게 대응하여 타 알고리즘들에 비해 예측 값의 오차를 가장 최소화하였다고 본다.



5. 결론 및 향후 과제

국방 분야의 특성상 불규칙적인 수요 패턴을 가진 품목이 상당부분 차지하고 있다. 따라서 원활한 무기체계 관리 및 효율적인 예산 활용을 위해 간헐적 품목의 수요예측은 중요한 부분이다.

간헐적 수요에 대한 예측방법 연구는 기존부터 다양한 방법을 통해 지속되어 왔다. 기존에는 지수평활, 가중이동평균, Croston 방법과 관련된 다양한 연구들이 제안되어 왔다. 이후 인공신경망, MLP 방법의 효과성이 제안되었고 최근에는 딥러닝 알고리즘 중 순환신경망인 RNN의 우수성이 입증되었다.

본 논문에서는 RNN의 단점을 개선한 LSTM과 GRU의 알고리즘을 통해 간헐적 수요예측에 대해 실시해보고 성능을 검증해보았다. 간헐적 수리부속의 소모자료를 식별 후 MLP, RNN, LSTM, GRU 4가지 딥러닝 알고리즘을 적용시켜 비교 분석을 실시하였다.

본 연구 결과를 통해 다른 알고리즘에 비해 LSTM 성능이 우수함을 확인할 수 있었다. 또한 순환신경망 계열의 RNN, GRU 알고리즘이 다층퍼셉트론의 MLP의 예측 오차보다 낮았다. 시계열 기법에서 좋은 성능을 보이고 있는 순환신경망 계열(RNN, LSTM, GRU)의 딥러닝 기법을 무기체계 수리부속의 간헐적 수요에 적용해 봄으로써 활용 가능성 확인할 수 있었다.

데이터 흐름에 따라 데이터의 선후 관계를 인식하는 학습이 가능한 순환신경망이 간헐적 품목의 수요예측 정확도를 높이는데 강점이 있다고 판단된다.

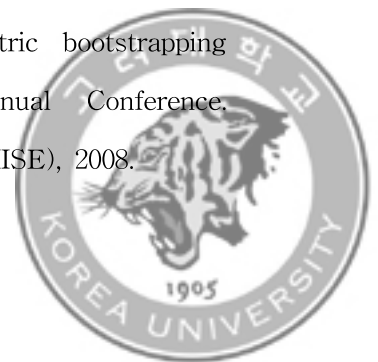
향후 운용시간 및 MTBF²⁾, TCI품목 등과 같은 수요영향 요인 변수를 반영한 지속적인 연구 및 다양한 신경망 모델에 대한 연구가 병행된다면 더욱 높은 예측력을 가진 모델을 만들 수 있을 것이라 기대한다.

2) MTBF : 고장 간 평균시간 (Mean Time Between Failure)



참고 문헌

- [1] 이혁수 외 10명, 2016, “2016 수리부속 소요산정 모형 개발 연구”, KIDA
- [2] 장기덕, 2012, “군수관리의 이론과 실제”, KIDA
- [3] Bishop, Christopher M. Neural networks for pattern recognition. Oxford university press, 1995.
- [4] Bishop, Christopher M. Pattern recognition and machine learning. springer, 2006.
- [5] Croston, J. Do. "Forecasting and stock control for intermittent demands." Operational research quarterly (1972): 289-303.
- [6] Syntetos, Aris A., and John E. Boylan. "On the bias of intermittent demand estimates." International journal of production economics 71.1 (2001): 457-466.
- [7] Levén, Erik, and Anders Segerstedt. "Inventory control with a modified Croston procedure and Erlang distribution." International journal of production economics 90.3 (2004): 361-367.
- [8] Syntetos, Aris A., and John E. Boylan. "The accuracy of intermittent demand estimates." International Journal of forecasting 21.2 (2005): 303-314.
- [9] Syntetos, Aris A., and John E. Boylan. "On the stock control performance of intermittent demand estimators." International Journal of Production Economics 103.1 (2006): 36-47.
- [10] Willemain, Thomas R., Charles N. Smart, and Henry F. Schwarz. "A new approach to forecasting intermittent demand for service parts inventories." International Journal of forecasting 20.3 (2004): 375-387.
- [11] Varghese, Vijith, and Manuel Rossetti. "A parametric bootstrapping approach to forecast intermittent demand." IIE Annual Conference. Proceedings. Institute of Industrial and Systems Engineers (IIE), 2008.



- [12] Teunter, Ruud H., and Laura Duncan. "Forecasting intermittent demand: a comparative study." *Journal of the Operational Research Society* 60.3 (2009): 321-329.
- [13] Gutierrez, Rafael S., Adriano O. Solis, and Somnath Mukhopadhyay. "Lumpy demand forecasting using neural networks." *International Journal of Production Economics* 111.2 (2008): 409-420.
- [14] Amin-Naseri, M. R., and B. Rostami Tabar. "Neural network approach to lumpy demand forecasting for spare parts in process industries." *Computer and Communication Engineering*, 2008. ICCCE 2008. International Conference on. IEEE, 2008.
- [15] Kocer, Umay Uzunoglu. "Forecasting intermittent demand by Markov chain model." *International Journal of Innovative Computing, Information and Control* 9.8 (2013): 3307-3318.
- [16] Sahin, Merve, Recep Kizilaslan, and Ömer F. Demirel. "Forecasting Aviation Spare Parts Demand Using Croston Based Methods and Artificial Neural Networks." *Journal of Economic and Social Research* 15.2 (2013): 1.
- [17] Kourentzes, Nikolaos. "Intermittent demand forecasts with neural networks." *International Journal of Production Economics* 143.1 (2013): 198-206.
- [18] Hua, Zhongsheng, and Bin Zhang. "A hybrid support vector machines and logistic regression approach for forecasting intermittent demand of spare parts." *Applied Mathematics and Computation* 181.2 (2006): 1035-1048.
- [19] Syntetos, Aris A., John E. Boylan, and J. D. Croston. "On the categorization of demand patterns." *Journal of the Operational Research Society* 56.5 (2005): 495-503.
- [20] Willemain, Thomas R., et al. "Forecasting intermittent demand in manufacturing: a comparative evaluation of Croston's method." *International*



journal of forecasting 10.4 (1994): 529–538.

[21] Hyndman, Rob J. "Another look at forecast-accuracy metrics for intermittent demand." *Foresight: The International Journal of Applied Forecasting* 4.4 (2006): 43–46.

[22] Syntetos, Aris A., and John E. Boylan. "On the stock control performance of intermittent demand estimators." *International Journal of Production Economics* 103.1 (2006): 36–47.

[23] Johnston, F. R., and John E. Boylan. "Forecasting for items with intermittent demand." *Journal of the operational research society* 47.1 (1996): 113–121.

[24] Boylan, John E., Aris A. Syntetos, and G. C. Karakostas. "Classification for forecasting and stock control: a case study." *Journal of the operational research society* 59.4 (2008): 473–481.

[25] Froung, Nguyen Khoa Viet, et al. "Intermittent Demand forecasting by using Neural Network with simulated data." *Proceedings of the 2011 International Engineering and Operations Management Kuala Lumpur, Malasia* (2011): 723–728.

[26] Callegaro, Andrea. "Forecasting methods for spare parts demand." (2010).

[27] Lolli, F., et al. "Single-hidden layer neural networks for forecasting intermittent demand." *International Journal of Production Economics* 183 (2017): 116–128.

[28] Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." *Neural computation* 9.8 (1997): 1735–1780.

[29] Chung, Junyoung, et al. "Empirical evaluation of gated recurrent neural networks on sequence modeling." *arXiv preprint arXiv:1412.3555* (2014).

[30] Werbos, Paul J. "Backpropagation through time: what it does and how to do it." *Proceedings of the IEEE* 78.10 (1990): 1550–1560.



- [31] Mikolov, Tomas, et al. "Recurrent neural network based language model." Interspeech. Vol. 2. 2010.
- [32] Narendra, Kumpati S., and Kannan Parthasarathy. "Identification and control of dynamical systems using neural networks." IEEE Transactions on neural networks 1.1 (1990): 4-27.
- [33] Sutskever, Ilya. "Training recurrent neural networks." University of Toronto, Toronto, Ont., Canada (2013).
- [34] Rumelhart, David E., James L. McClelland, and PDP Research Group. Parallel distributed processing. Vol. 1. Cambridge, MA, USA:: MIT press, 1987.
- [35] Multi-layer Perceptron,
http://scikit-learn.org/stable/modules/neural_networks_supervised.html
- [36] activation function, sigmoid function, Relu function,
<http://cs231n.github.io/neural-networks-1>
- [37] LSTM, <http://colah.github.io/posts/2015-08-Understanding-LSTMs>

