심층학습을 이용한 공정 모델링

요약

기계학습 중 하나인 심층학습을 이용하여 공정을 모델링하는 기법을 소개하고자 한다. 심층학습은 지도학습으로 회귀작업이 가능하고 종단간 학습으로 사람의 개입을 더 배제하는 방법이다. 또한, 응답 곡선 데이터로부터 시간지연이 있는 1차나 2차 모델로 근사하는 방법을 이용했다. 훈련데이터는 지침에 따라 MATLAB을 이용하여 얻었고 학습은 Python 라이브러리들을 이용한 알고리즘을 통해 하였다. 이를 통해 정확도 99% 이상의 회귀작업을 할 수 있었으며 실제 액위 제어 장치에 적용해 실제 물리적 시스템에서도 성능을 유지할 수 있는지 알아보았다. (~~~ 액위 실험 내용 추가)

1. 서론

공정이 효율을 올리거나 비용을 낮추기 위해 더 복잡해지면서, 발전된(advanced) 방법을 사용하고자 하는 동기가 있으며 지속해서 새로운 방법들이 발달하고 있다. 이러한 방법 중에는 다양한 종류의 모델 기반 설계 방법들도 있다. 이런 모델 기반 방법들은 이미 산업적으로 수용되고 있고, PID 제어기를 설계하는 데 사용되며 feedforward 제어기와 발전된(advanced) 제어계 설계에도 이용된다[1]. 실제로, 대표적인 모델 기반 제어인 feedforward 제어는 1960년대까지는 제어 산업에 많이 이용되지 않았지만, 그 이후부터 산업적 제어에 광범위하게 적용되고 있다[2]. 또 다른 예로서는, 내부모델 제어가 있다. 이는 새로운 제어개념이며 엄밀한 수학적 기초와 정확한 공정모델을 바탕으로 하는 것으로, 대략 1980년대 이후부터 집중적인 연구과제가 되어왔다[3]. 이러한 모델 기반 기법들은 모델이 더 정확할수록 더 좋은 성능을 보여주는 것이 일반적이다. 예를 들어, 시간지연 보상 방법의 하나인 smith predictor는 시간지연이나 공정이득이 실제 값의 30% 안에 있지 않으면 시간지연 보상을 사용하지 않은 PID보다 좋지 않은 성능을 보여준다[4]. 따라서 공정의 모델을 더 정확하게 찾는 기법들을 개발하고자 하는 지속적인 노력이 필요하다. 여기서는 기계학습의 한 분야인 심층학습을 이용하여 공정을 모델링 하는 기법을 소개한다.

화학 공정 제어에서 사용되는 모델링 방법은 두 가지가 있다. 첫 번째는, 화학적, 물리적 법칙에 기반을 둔 이론적인 모델로 나타내는 방법이다. 하지만 엄격한 이론적 모델을 만들어내는 것은 모델이 많은 수의 공정변수와 알려지지 않은 변수((예) 화학적, 물리적 물성)를 포함한 많은 수의 방정식이 필요하다면 복잡한 공정에 대해 실용적이지 않을 수 있다. 다른 방법은 실험적 데이터로부터 직접 실험적(empirical) 모델을 만드는 것이다. 이러한 과정을 공정 혹은 계의 인식(process or system identification)이라고 한다. 이 방법은 다수의 변수와 방정식을 포함한 복잡한 모델에 대해 맞지 않을 수 있다. 하지만 일반적으로, 실험적인 동적 모델은 이론적 모델보다 간단하고 실시간으로 계산될 수 있다는 장점이 있다. 즉, 모델의 해답에 필요한 계산 시간이 실제 공정 반응 시간보다 훨씬 짧다[5]. 그래서 실험적 데이터로부터 직접 실험적(empirical) 모델을 만드는 방법을 사용할 것이다.

실험적 데이터로부터 직접 모델을 만드는 방법에는 선형회귀, 비선형회귀와 계단 시험을 이용하여 일차나 이차 모델로 근사하는 방법 등이 있다. 대부분 화학 공정은 시간지연이 존재하는 1차나 2차 공정 모델로 나타낼 수 있고 실제로 많은 복잡한 화학 공정들은 시간지연이 존재하는 1차 공정 모델의 형태로 아주 효과적으로 나타난다[6]. 그래서 여기서는 계단 시험을 이용하여 일차나 이차 모델로 근사하는 방법을 이용하고자 한다. 대표적인 방법으로는 응답곡선법과 smith’s method가 있다. 두 방법은 공정을 각각 시간지연이 있는 1차와 2차 공정으로 나타내는 그래프 기법이다[7].

한편, 기계학습 시스템 중 지도학습은 타깃 수치를 예측하는 회귀(regression)작업이 가능하다. 지도학습은 알고리즘에 주입하는 훈련데이터에 레이블이라는 원하는 답을 함께 주입하는데 (그래서 지도학습이라 한다)[8] 기계학습 시스템이 레이블을 포함한 많은 양의 훈련데이터를 학습하면 그다음에 새로운 데이터를 줬을 때 그 데이터의 레이블이 무엇인지 예측할 수 있다는 것이다. 심층학습은 대표적인 지도학습 기법이고 이 심층학습을 이용한 회귀작업을 통해 시상수와 시간지연 같은 공정 변수를 예측하여 모델링 하고자 한다.

서포트 벡터 머신(SVM) k-최근접 이웃(KNN) 같은 다른 지도학습 알고리즘도 회귀 작업이 가능하지만, 심층학습을 사용하는 이유는 심층학습이 종단간 기계학습이라는 점이다. 종단간 기계학습에서는 데이터(입력)에서 목표한 결과(출력)를 사람의 개입 없이 얻을 수 있다. 다른 기계학습 방법들은 데이터에서 특징을 추출하고 그 특징의 패턴을 기계학습 기술로 학습한다. 이때, 사용하는 특징은 여전히 사람이 설계하는 것이다. 즉, 사람이 적절한 특징을 생각해내야 한다. 하지만 심층학습은 데이터를 있는 그대로 학습한다. 즉, 심층학습은 데이터에 포함된 중요한 특징까지도 기계가 스스로 학습한다. 즉, 심층학습은 다른 기계학습에서 사용하던 방법보다 사람의 개입을 더욱 배제할 수 있게 해주는 중요한 특성이 있다[9]. 사람의 개입을 더 배제할수록 완전한 자동화에 더 가까워질 것이다.

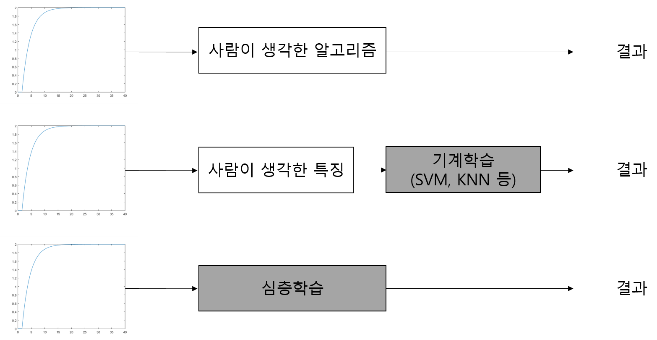


그림1-1. 종단간 학습과 다른 방법과 비교: 가장 위쪽 방법은 규칙을 ‘사람’이 만든다. 가운데 방법은 ‘사람’이 적절한 특징을 생각해야 한다. 반면, 심층학습은 주어진 데이터 그대로를 입력 데이터로 활용하여 학습한다[10].

추가로 고려해야 할 사항이 있다. 시간 지연이 있는 1차 모델과 감쇄비가 1 이상인 시간지연이 있는 2차 모델은 형태가 비슷하여 잘 구분되지 않는다. 그렇다면, 계단 시험을 하여 얻은 그래프가 S자 곡선 모양 혹은 언덕 모양과 같은 형태를 나타냈을 때, 시간 지연이 있는 1차 모델과 감쇄비가 1 이상인 시간지연이 있는 2차 모델 둘 중에 어느 모델로도 근사할 수 있을 것이다. 여기서는 이 경우 시간 지연이 있는 1차 모델로 근사하기로 한다. 실제, overshoot이 없는 경우에는 시간지연이 있는 1차 모델로 나타내도 큰 무리는 없다[11]. 이렇게 되면 overshoot 없다면 1차로 근사하고 overshoot이 있으면 2차로 근사하면 된다. 정리하면 다음과 같다.

* **공정의 계단 응답에 대한 가정**

**공정의 계단 응답이 overshoot이 없는 동적인 양상(즉, overdamped 형태)을 나타낸다면 시간지연이 있는 1차 공정으로 근사를 한다. 공정의 계단 응답이 overshoot이 존재하는 동적인 양상(즉, underdamped 형태)을 나타낸다면 시간지연이 있는 2차 공정으로 근사를 한다. 이때 단순화하기 위해 역응답은 고려하지 않는다(즉, 분자에 양수인 pole이 존재하지 않는 경우만 고려한다.).**

즉, 어떤 공정이 S자 곡선 형태의 계단응답을 보여주면 그 공정을 시간지연이 있는 1차 공정으로 생각하고 진동형태의 계단응답을 보여주면 그 공정을 시간지연이 있는 2차 공정으로 생각한다.

정리하자면, 계단 시험 데이터와 심층학습을 이용하여 공정을 시간지연이 있는 1차나 2차 공정으로 근사하는 방법을 소개하고자 한다. 시간지연이 있는 1차나 2차 공정은 여러 형태가 있지만, 여기에선 가장 간단한 형태인 와 형태로 근사한다. 즉, 심층학습을 이용한 회귀작업을 통해 공정변수인 시상수(), 시간지연(), 감쇄비()를 구하는 것이 목표이다. (공정이득(은 계단시험을 하면 바로 알 수 있다) 우선 MATLAB으로부터 시간지연이 있는 1차 공정에 대한 데이터와 시간지연이 있는 2차 공정에 대한 데이터를 대량으로 얻어 심층학습을 학습시킨다. 여기서 심층학습을 학습시킨다는 것은 훈련데이터로부터 손실 함수의 결괏값을 가장 작게 만드는 가중치 매개변수의 최적화된 값을 자동으로 획득하는 것을 뜻한다[12]. 그런 다음 학습으로 얻어진 가중치 매개변수를 다시 불러와서 공정 변수를 예측하는 데 사용할 것이다. 다음은 실험 순서이다.

* **실험순서**

1. **MATLAB을 이용하여 데이터 얻기**
2. **심층학습에 맞게 데이터 처리**
3. **심층학습을 통한 학습**
4. **추가적인 성능 향상**
5. **실제 액위계 모델링과 제어**

(3) 과정에서 대부분의 심층 학습은 큰 어려움 없이 이루어졌다. 하지만 2차 공정의 시간지연 변수를 학습하는 데 어려움이 있었으며 이는 추가적인 특성을 추가함으로써 해결할 수 있었다. (4)에서는 Scikit-learn 라이브러리의 메서드인 RandomizedSearchCV를 이용한 교차시험을 통해 최적의 하이퍼파라미터를 찾았다. (5) 과정에서는 (~~~ 액위 실험 내용 추가)

이 실험에 대한 전체적인 개요를 그림1-2를 통해 설명하였다.

1. 실험

2-1. 훈련 데이터 준비

심층학습을 이용하기 위해서 훈련데이터를 잘 준비하는 것은 매우 중요하다. 이를 위해 Hands-On Machine Learning with Scikit-Learn＆ Tensorflow라는 책의 지침을 이용했으며 그 내용은 다음과 같다[13].

* **학습 데이터 준비 지침**

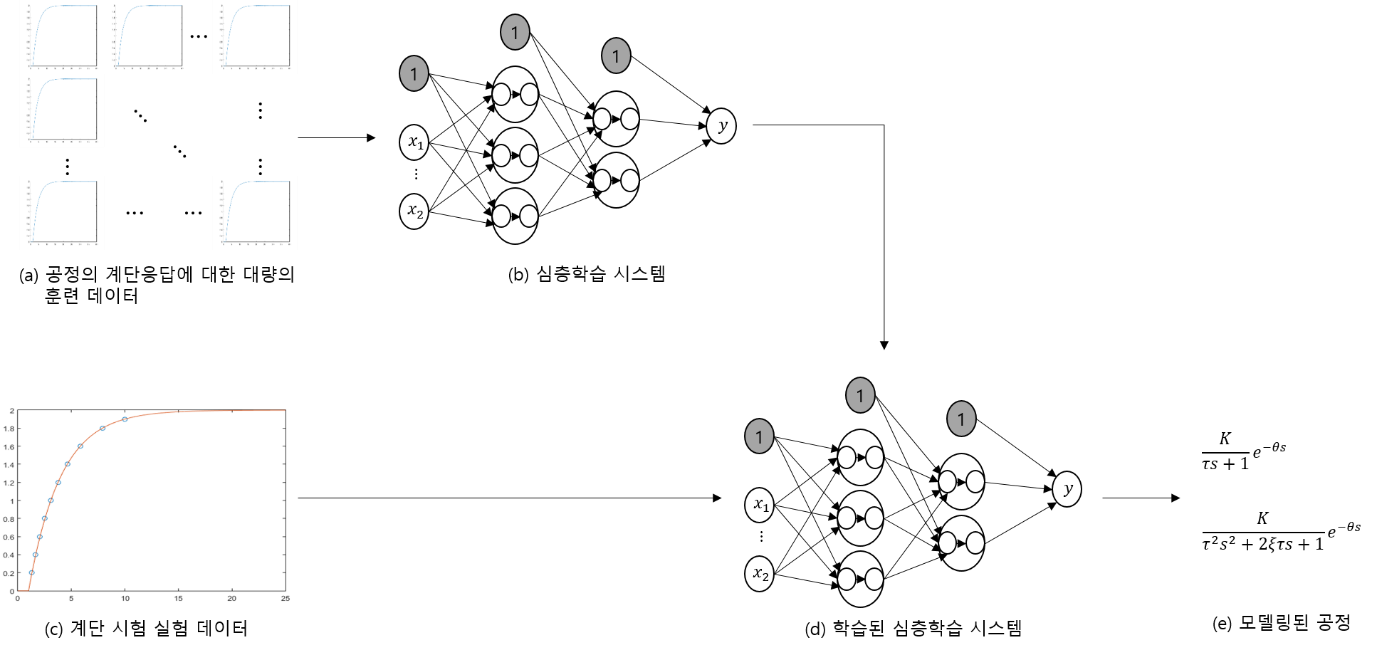


그림1-2. 실험에 대한 개요, 먼저 MATLAB에서 얻은 데이터(a)를 심층학습 알고리즘에 주입하여 학습한다(b). 이 학습된 심층학습 시스템(d)에 실제 공정의 계단 시험에서 얻은 실험 데이터(c)를 주입하면 그 공정의 공정변수가 무엇인지 알 수 있다(e)

1. **양이 충분해야 한다.**
2. **대표성이 있어야 한다.**
3. **에러, 이상치, 측정 장치의 낮은 성능으로 인한 잡음 같은 것이 없어야 한다.**
4. **관련 없는 특성이 없어야 한다.**
5. **훈련 데이터 과적합을 고려해야 한다.**

다시 언급하면, 시간지연이 있는 1차나 2차 공정에 대한 데이터는 MATLAB을 이용하여 얻었으며 MATLAB 내장 함수와 script를 이용하여 최대한 자동화했다.

MATLAB을 통해 얻은 모든 계단 시험 데이터를 신경망에 주입하여 학습할 수는 없으므로 지침 (2)에 따라 대표성이 있는 데이터를 선정해야 했다. 여기서 선택된 데이터 중 하나를 속성이라 하고((예) RiseTime) 이 속성과 값이 합쳐진 것을 특성이라 한다. 이 선정된 데이터만 심층학습 시스템에 주입했다. 먼저, 시간 지연이 있는 1차 공정에서는 공정출력이 정상상태 값의 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%, 95%에 도달한 시간으로 특성을 정했다. 시간 지연이 있는 2차 공정에서는 RiseTime, SettlingTime, SettlingMin, SettlingMax, Overshoot, PeakTime 이라는 값으로 특성을 정했다. 이 값들은 MATLAB에서 stepinfo라는 내장함수를 이용하여 구할 수 있고 MATLAB 공식 문서에서 찾을 수 있는 정의는 다음과 같다[14]

* **시간지연이 있는 2차 공정 입력 특성의 MATLAB 공식문서에서 정의**
* **RiseTime: 정상상태 응답의 10%에서 90%까지 상승하는 응답에 걸리는 시간**
* **SettlingingTime: 정상상태 응답과 응답의 차이인 오차가 정상상태 값의 2%보다 작아지는 데 걸리는 시간**
* **SettlingMax: 일단 응답이 상승한 이후 출력의 최댓값**
* **SettlingMin: 일단 응답이 상승한 이후**

**출력의 최솟값**

* **Overshoot: 최종값에 상대적으로 더 넘어간 퍼센트 값**
* **PeakTime: 출력 고점의 절댓값**

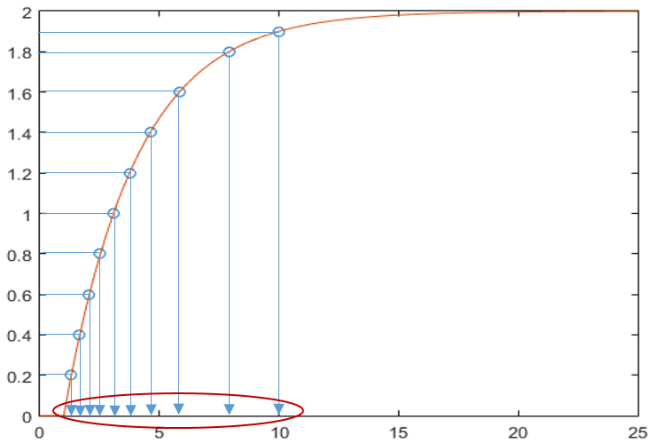


그림2-1. 시간지연이 있는 1차 공정의 학습 알고리즘에 주입되는 입력 특성: 공정출력이 정상상태 값의 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%, 95%에 도달한 시간

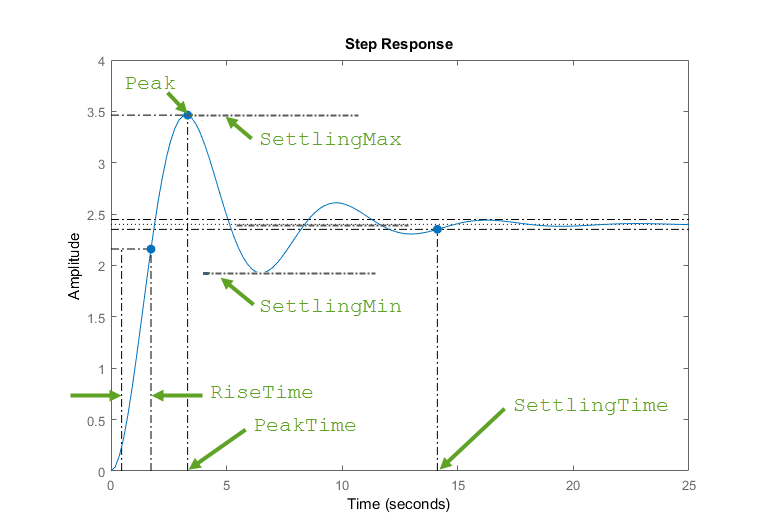


그림2-2. 시간지연이 있는 2차 공정의 학습 알고리즘에 주입되는 입력 특성[15]: MATLAB의 stepinfo 내장함수에 의해 얻어지는 6개의 값(RiseTime, SettlingingTime, SettlingMin, SettlingMax, Overshoot, PeakTime)

추가로, 공정의 이득은 계단 시험에 의해서 바로 얻을 수 있고 공정의 중요한 정보이므로 시간지연이 있는 1차 공정과 2차 공정 모두 위에서 언급한 특성에 공정이득을 추가했다. 다음은 위에 언급된 지침을 고려하며 데이터를 심층학습에 맞게 가공하는 과정이다.

* **데이터 처리**
* **시간지연이 있는 2차 공정의 공정변수 중 하나가 매우 작으면 stepinfo 함수에 의해 계산된 값들이 매우 크거나 작아질 수가 있다. 이런 값들은 학습할 때 학습이 잘 이루어지지 않게 할 수 있고 사실상 데이터로써 큰 의미가 없다. 그래서 이를 방지하기 위해 공정 변수에 아주 작은 값을 발생시키지 않게 하한과 상한을 설정한다. 공정이득은 0.01~30 사이, 시상수는 0.2~50 사이, 시간지연은 0.01~20 사이, 감쇄비는 0.02~0.95 사이의 수로 하였다. (지침(3))**

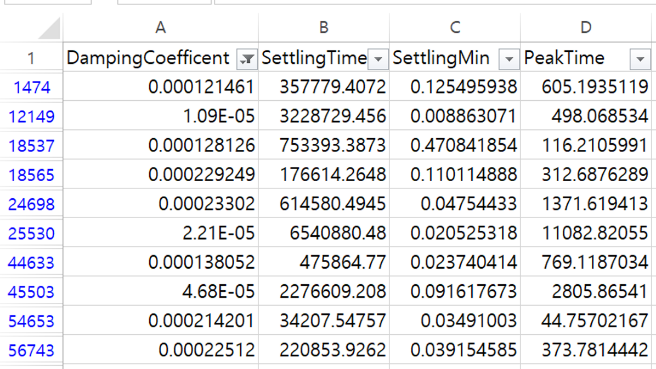


그림2-3-(1). 감쇄비가 가장 작은 10개(공정변수의 크기에 제한이 없을 때): SettlingTime과 SettlingMin, PeakTime의 값이 지나치게 크거나 작다.

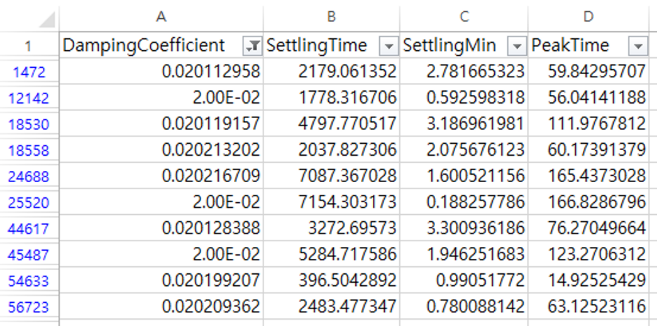


그림2-3-(2). 감쇄비가 가장 작은 10개(공정변수의 크기에 제한이 있을 때): SettlingTime과 SettlingMin, PeakTime의 값이 지나치게 크거나 작은 정도가 감소한다.

* **시간지연이 있는 2차 공정의 훈련데이터를 얻기 위해 사용했던 stepinfo 함수는 위에 서술한 값 이외에도 UnderShoot과 Peak라는 값을 계산하지만 지금 경우에는 UnderShoot은 모두 0으로 계산되고(역응답이 없음) Peak는 SettlingMax와 같다. 그래서 훈련데이터에 포함하지 않았다. 또한, 시간지연이 있는 2차 공정에서 감쇄비가 1에 가까운 경우에 Overshoot이 0인 경우도 있다. 이 경우도 서론에서의 가정에 따라 훈련 데이터에 포함하지 않는다. (지침(4))**
* **심층학습 시스템을 학습시킬 때 과적합되는 경우가 발생할 수 있는데 과적합 여부와 정도를 알기 위해 훈련데이터에서 20%를 따로 떼어내서 테스트 데이터를 준비해야 한다. 이 과정은 scikit-learn 라이브러리에 의해 구현했다. (지침(5))**

추가로, 심층학습에 주입하는 데이터는 스케일링할 필요가 있다. 입력 특성들의 크기가 모두 다르면 이는 학습을 매우 어렵게 하는 요소가 된다. 그래서 스케일링을 이용해서 입력 특성들의 스케일을 비슷하게 맞춰준다[16]. 이 또한 scikit-learn 라이브러리에 의해 구현할 수 있다.

정리하면, 공정이득은 0.01~30 사이, 시상수는 0.2~50 사이, 시간지연은 0.01~20 사이, 감쇄비는 0.02~0.95 사이의 수로 하고 MATLAB을 이용하여 계단응답 데이터를 얻어 입력 특성으로 사용했다. 시간 지연이 있는 1차 공정에서는 공정출력이 정상상태 값의 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%, 95%에 도달한 시간이고 시간 지연이 있는 2차 공정에서는 stepinfo라는 내장함수를 이용하여 구할 수 있는 6개의 값이다. 여기에 두 경우 모두 공정 이득을 특성으로 추가했다. 훈련 데이터와 함께 심층학습 알고리즘에 주입되는 레이블은 예측하고자 하는 타깃 값에 따라 시상수, 시간지연, 감쇄비이다. 그리고 학습의 성능을 향상하기 위해 데이터에 몇 가지 조작들을 했고 그중 일부는 scikit-learn 라이브러리에 의해 구현했다.

2-2. 심층학습 시스템 학습

실제로 학습을 할 때는 모든 훈련 데이터를 이용하지는 않으며 훈련 데이터 중 무작위로 일부만 골라 학습한다. 이 일부를 미니배치라 하고 이러한 학습을 미니배치학습이라 한다[17]. 이 실험에서도 학습할 때 미니배치학습을 이용했다. 훈련 데이터로부터 얻은 미니배치로 학습하고 또 새로운 미니배치를 무작위로 얻어 다시 학습하는 작업을 반복하여 학습했다. 그리고 이 반복작업에 의해 훈련 데이터를 모두 소진했을 때의 횟수를 1 에포크라 한다. 앞서 언급했듯이 학습이란 훈련데이터로부터 손실 함수의 결괏값을 가장 작게 만드는 가중치 매개변수의 최적화된 값을 자동으로 획득하는 것이고 미니배치 학습은 반복작업마다 손실함수 결괏값을 줄이면서 가중치 매개변수를 최적화된 값으로 갱신한다. 따라서 심층학습 시스템이 제대로 학습되고 있는지 판단하는 지표를 손실함수로 하였다[18]. 에포크가 커지면서 손실함수의 결괏값이 가장 작은 값으로 점점 수렴해 간다면 학습이 잘 되고 있다고 판단할 수 있을 것이다. 회귀문제의 전형적인 손실함수는 평균 제곱근 오차(RMSE)이다[19]. 평균 제곱근 오차의 전역 최솟값은 예측값과 실제값이 같을 경우의 0이지만 실제 0으로 수렴하기는 굉장히 어려우며 최대한 0으로 가깝게 수렴할 때 좋은 학습이라 할 수 있다. 추가로, 정확도도 다음과 같이 정의하여 또 하나의 부수적인 지표로 삼았다.

* **정확도 = (1-(abs(예측값-실제값)/실제값)**

먼저, 은닉층의 수=2, 에포크=20, 은닉층의 뉴런 수=100, momentum=0.9, 학습률=0.0001, 배치 크기=50, 옵티마이저: 네스테로프 가속경사, 활성화 함수: relu로 하여 학습이 원활하게 이루어지는지 에포크마다 RMSE와 정확도를 출력하면서 판단하였다.

추가로, 심층학습을 구현하면서 Python 라이브러리 패키지인 아나콘다에서 Pandas, numpy, scikit-learn의 라이브러리를 이용했으며 개발환경으로 Jupyter Notebook을 이용했다. 대부분의 인공신경망 메서드는 tensorflow 라이브러리에서 이용했고 알고리즘은 Hands-On Machine Learning with Scikit-Learn ＆ Tensorflow의 알고리즘을 참고했다.

* **시간지연이 있는 1차 공정**
* **시상수를 학습하기 위한 학습**

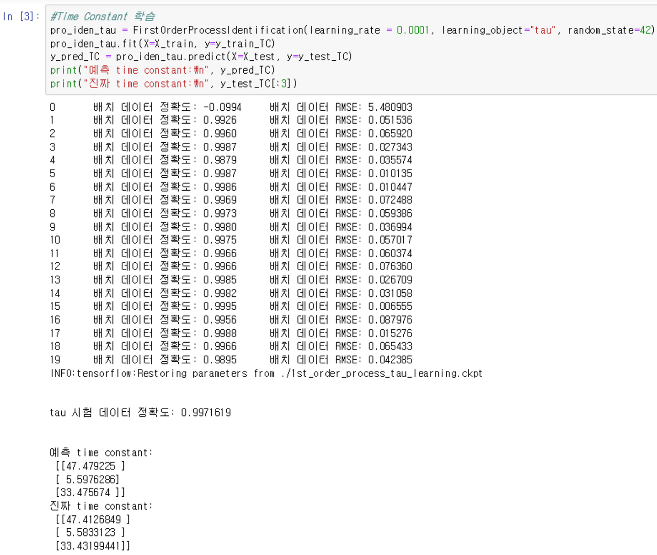


그림2-4-(1). 시간지연이 있는 1차 공정의 시상수를 학습하는 심층학습 시스템의 에포크에 따른 정확도와 RMSE

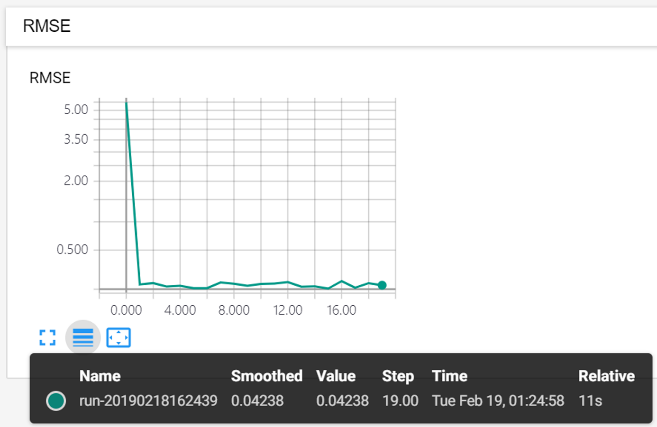


그림2-4-(2). 시간지연이 있는 1차 공정의 시상수를 학습하는 심층학습 시스템의 에포크에 따른 RMSE의 그래프

그림2-4-(1)를 보면 무난하게 학습이 이루어지는 것을 알 수 있다. 매우 높은 정확도를 얻을 수 있고 손실함수 값도 충분히 0에 가까워진다. 또한, 시험 데이터의 정확도가 훈련 데이터의 정확도와 큰 차이를 보이지 않는 것에서 과적합이 나타나지 않는다는 것을 알 수 있다. 그림2-4-(1)의 하단은 학습데이터로 학습된 심층학습 시스템에 시험 데이터를 주입한 결과이다. 매우 근사한 값으로 예측했다는 것을 알 수 있다.

* **시간지연을 학습하기 위한 학습**

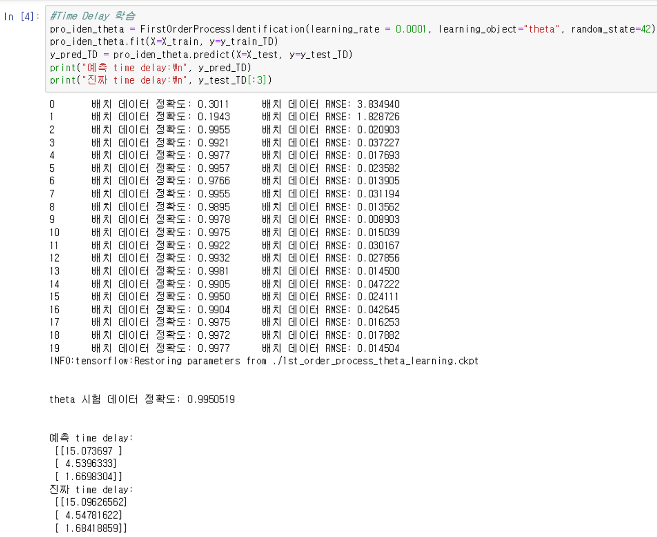


그림2-5-(1). 시간지연이 있는 1차 공정의 시간지연을 학습하는 심층학습 시스템의 에포크에 따른 정확도와 RMSE

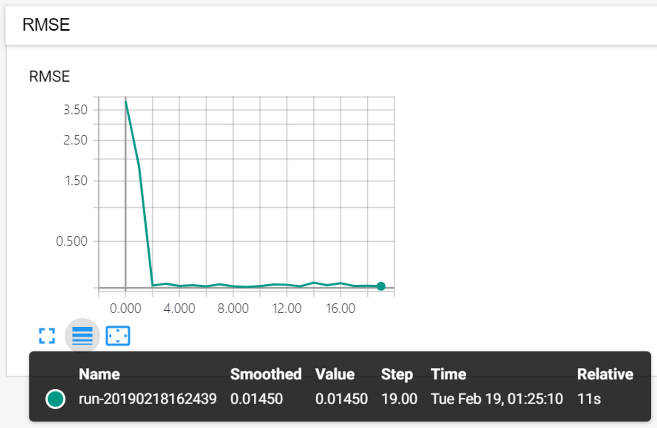


그림2-5-(2). 시간지연이 있는 1차 공정의 시간지연을 학습하는 심층학습 시스템의 에포크에 RMSE의 그래프

이번 과정 역시 무난하게 학습이 이루어지는 것을 알 수 있다. 매우 높은 정확도를 얻을 수 있고 손실함수 값도 적당히 작아진다. 또한, 과적합도 없다.

* **시간지연이 있는 2차 공정**
* **시상수를 학습하기 위한 학습**

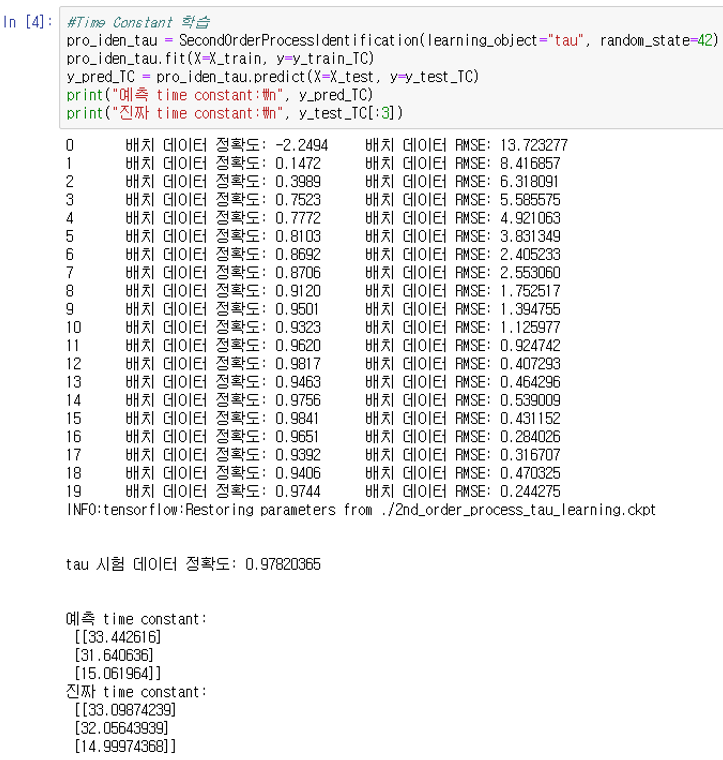
****

그림2-6-(1). 시간이 있는 2차 공정의 시상수를 학습하는 심층학습 시스템의 에포크에 따른 정확도와 RMSE

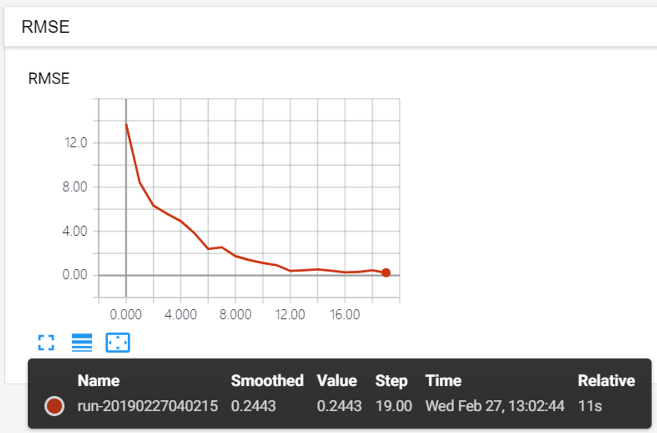


그림2-6-(2). 시간이 있는 2차 공정의 시상수를 학습하는 심층학습 시스템의 에포크에 따른 RMSE의 그래프

이번 과정 역시 무난하게 학습이 이루어지는 것을 알 수 있다. 매우 높은 정확도를 얻을 수 있고 손실함수 값도 적당히 작아진다. 또한, 과적합도 없다.

* 감쇄비를 학습하기 위한 학습

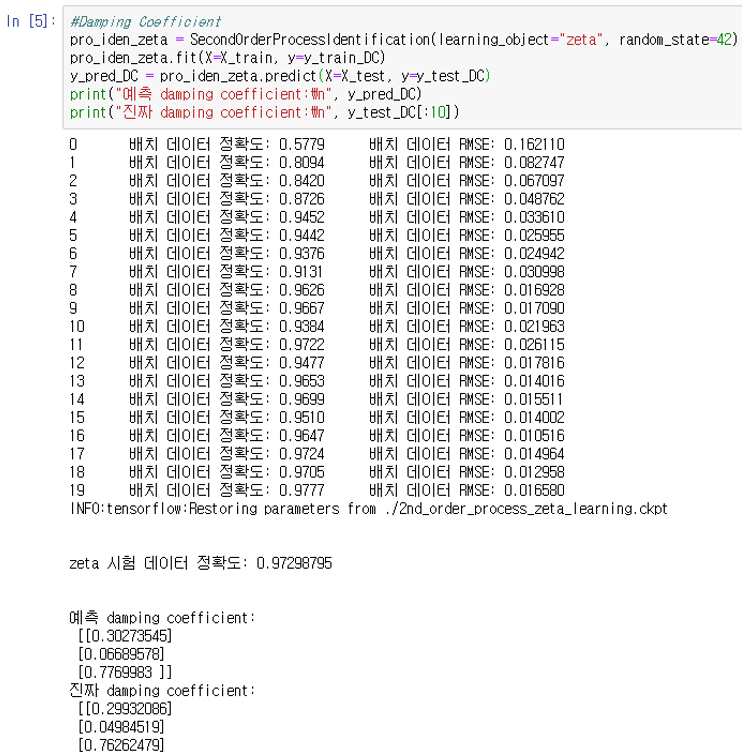


그림2-7-(1). 시간이 있는 2차 공정의 감쇄비를 학습하는 심층학습 시스템의 에포크에 따른 정확도와 RMSE



그림2-7-(2). 시간이 있는 2차 공정의 감쇄비를 학습하는 심층학습 시스템의 에포크에 따른 RMSE의 그래프

이번 과정 역시 무난하게 학습이 이루어지는 것을 알 수 있다. 매우 높은 정확도를 얻을 수 있고 손실함수 값도 적당히 작아진다. 또한, 과적합도 없다.

* **시간지연을 학습하기 위한 학습**

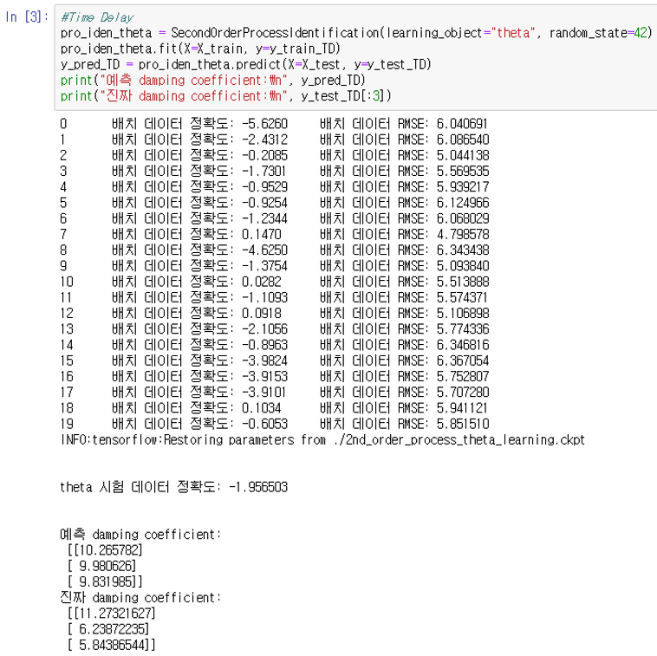
****

그림2-8-(1). 시간이 있는 2차 공정의 시간지연을 학습하는 심층학습 시스템의 에포크에 따른 정확도와 RMSE

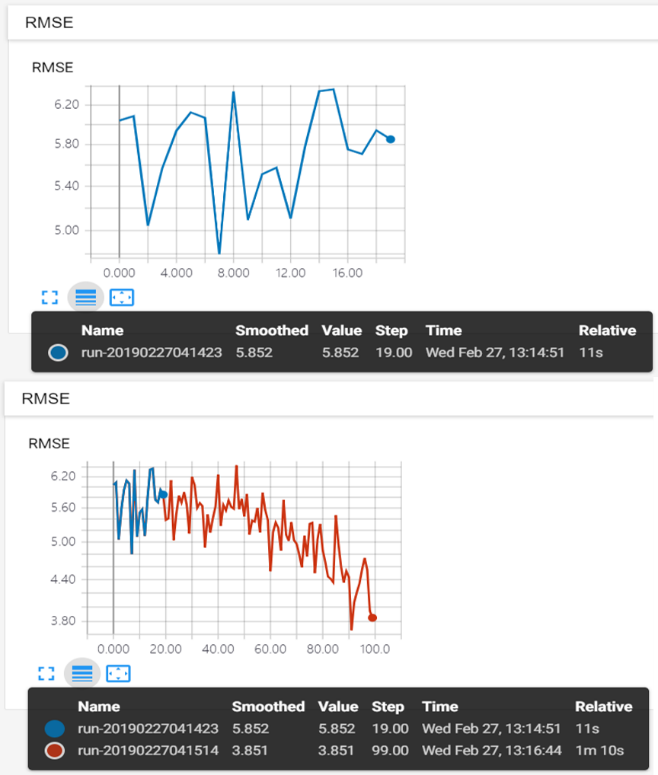


그림2-8-(2). 시간이 있는 2차 공정의 시간지연을 학습하는 심층학습 시스템의 에포크에 따른 반복 횟수에 따른 RMSE의 그래프: 반복 횟수를 20으로 한 경우(위), 반복 횟수를 100으로 한 경우(아래)

그림2-8-(2)을 보면 손실함수가 최적값에 수렴하지 못하며 학습이 이루어지지 않는 것을 볼 수 있다. 위의 그래프는 에포크 수가 20인 학습의 손실함수 그래프이고 아래의 그래프는 에포크를 80번 더 추가(빨간색 부분)하여 100번을 진행한 그래프이다. 아래 검은 창의 손실함수의 값(value)을 봐도 마지막 손실함수의 값이 0으로 수렴하지 못하는 것을 알 수 있다.

하지만 수렴하지 못하는 이유가 적절하지 않은 하이퍼파라미터 때문일 수 있거나 에포크 수가 적었을 수 있으므로 scikit-learn의 RandomizedSearchCV를 사용하여 학습률이나 momentum 같은 하이퍼파라미터를 바꾸어보는 교차시험을 해서 여러 경우의 손실함수를 구해보았다. 또한, 이 알고리즘은 최적의 하이퍼파라미터를 찾지 못한다면 1000번까지 학습을 반복하므로 학습 횟수도 충분하다.

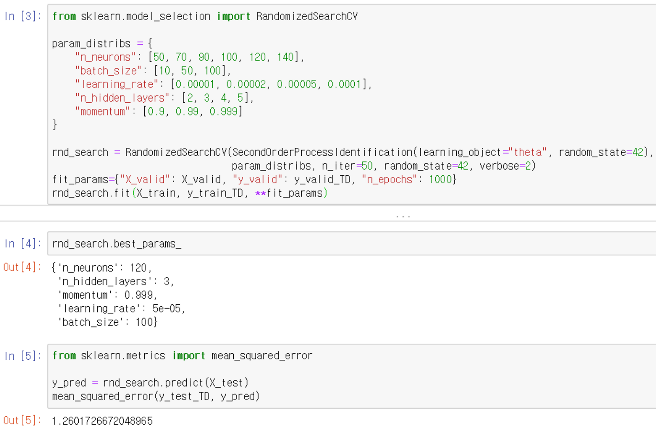


그림2-8-(3) 여러 교차 시험에도 수렴하지 않는 손실함수: RandomizedSearchCV를 이용하면 “뉴런 개수”: [50, 70, 90, 100, 120, 140], "미니배치 크기": [10, 50, 100], "학습률": [0.00001, 0.00002, 0.00005, 0.0001], "은닉층 수": [2, 3, 4, 5], "momentum": [0.9, 0.99, 0.999] 중 하나씩 무작위로 선택해 100번의 교차 시험을 통해 최고의 성능을 보이는 하이퍼파라미터 조합이 무엇인지 알 수 있다.

RandomizedSearchCV 통해 구해진 최적의 하이퍼파라미터는 뉴런 개수가 120, 미니배치 크기가 100, 학습률이 0.00005, 은닉층 수가 3, momentum이 0.999이다. 하지만 이 최적의 하이퍼파라미터로 학습한 심층학습 시스템에 시험 데이터를 주입하여 얻은 RMSE값도 1.26 정도로 0으로 수렴하지 않는 것을 알 수 있다. (그림 2-8-(3))

이때 시도해 볼 수 있는 것은 입력 특성을 추가하는 것이다. 당연히 훈련 데이터와 관련 없는 특성이 적고 관련 있는 특성이 충분해야 시스템이 학습할 수 있다[20]. 지금 2차 공정에서 훈련 데이터로 이용하는 공정이득, RiseTime, SettlingingTime, SettlingMin, SettlingMax, Overshoot, PeakTime이 시간지연의 학습에 어느 정도 영향을 미치는지 알아보기 위해 특성을 하나씩 제외하면서 손실함수의 변화를 알아보았다.

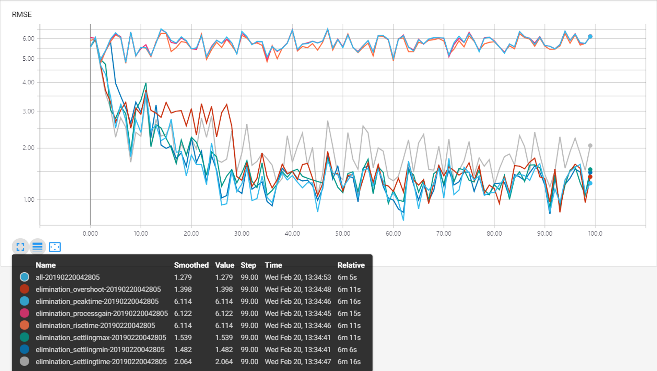


그림2-9. 특성을 제외한 후 반복 횟수에 따른 손실함수의 변화

그림 2-9를 보면 특성에서 제외했을 때 성능의 큰 변화가 있는 것은 공정이득, RiseTime, PeakTime이고 SettlingTime, SettlingMin, SettlingMax, Overshoot을 제외하는 것은 성능의 큰 변화는 없다. 성능에 도움되는 특성을 좀 더 추가할 필요가 있어 보였다.

시간지연과 관련 있는 특성은 무엇인가에 대한 아이디어는 시간지연이 있는 1차 공정에 대한 심층학습에서 얻을 수 있었다. 이 심층학습에서 공정출력이 정상상태 값의 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%, 95%에 도달한 시간의 값을 이용하여 시간지연을 좋은 성능으로 예측하였다. 이와 같게 2차 공정의 훈련데이터에 공정출력이 정상상태 값의 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%, 95%에 도달한 시간을 추가한다. 다음은 학습데이터를 추가한 훈련 데이터를 이용하여 시간지연이 있는 2차 공정의 시간지연을 학습한 과정이다.

* **훈련 데이터에 새로운 특성을 추가한 후 시간지연을 학습하기 위한 학습(은닉층의 수=2, 에포크=20, 은닉층의 뉴런 개수=100, momentum=0.9, 학습률=0.0001, 배치 크기=50, 옵티마이저: 네스테로프 가속경사, 활성화 함수: relu**

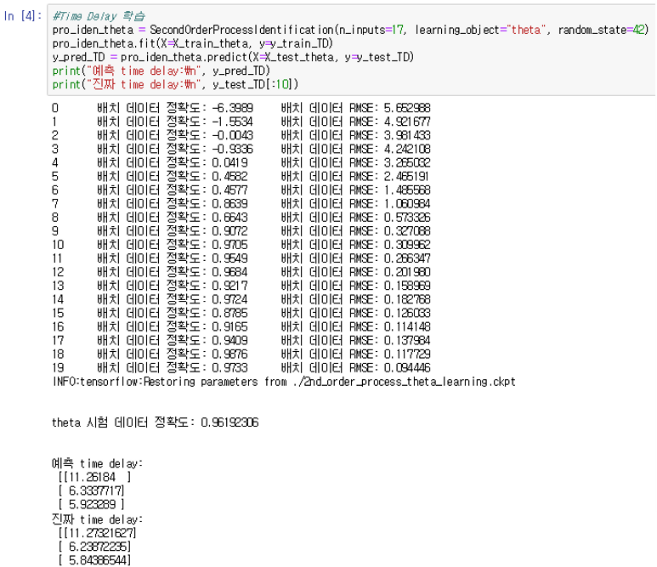


그림2-10-(1). 훈련 데이터에 새로운 특성을 추가한 후 시간이 있는 2차 공정의 시간지연을 학습하는 심층학습 시스템의 에포크에 따른 정확도와 RMSE

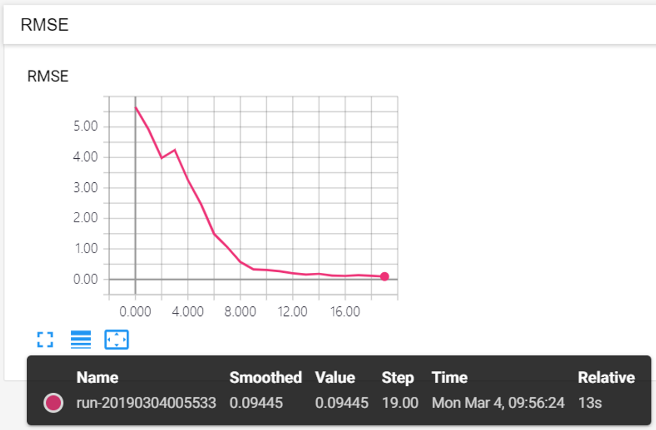


그림2-10-(2). 훈련 데이터에 새로운 특성을 추가한 후 시간이 있는 2차 공정의 시간지연을 학습하는 심층학습 시스템의 에포크에 따른 RMSE의 값

새로운 특성을 추가한 후 학습은 그림2-10-(1)를 보면 무난하게 학습이 이루어지는 것을 알 수 있다. 매우 높은 정확도를 얻을 수 있고 손실함수 값도 충분히 0에 가까워졌다. 하지만 추가된 이 데이터에는 약간의 문제가 있다. 만약 MATLAB에서 공정출력이 정상상태 값의 10%에 도달한 시간을 구하기 위해 stepinfo의 옵션을 (‘RiseTimeThreshold’를 [0, 0.1])로 설정하고 계산한다고 하면 제대로 된 값을 계산하지 못한다. 그래서 시간지연이 없는 상태에서 (‘RiseTimeThreshold’를 [0, 0.1])로 설정하고 계산한 값에 시간지연을 더하여 계산하였다. 다른 퍼센티지의 값도 이런 방식으로 구했는데 이 값은 실제의 값과 약간의 차이가 있다. 그러나 이 차이가 데이터 준비를 위한 지침의 지침(3)처럼 에러, 이상치, 측정 장치의 낮은 성능으로 인한 잡음 같은 것이 있어 공정변수를 예측하는 성능을 낮추는 정도는 아니라고 판단했다.

* **RamdomizedSearchCV를 이용한 하이퍼파라미터 조정**

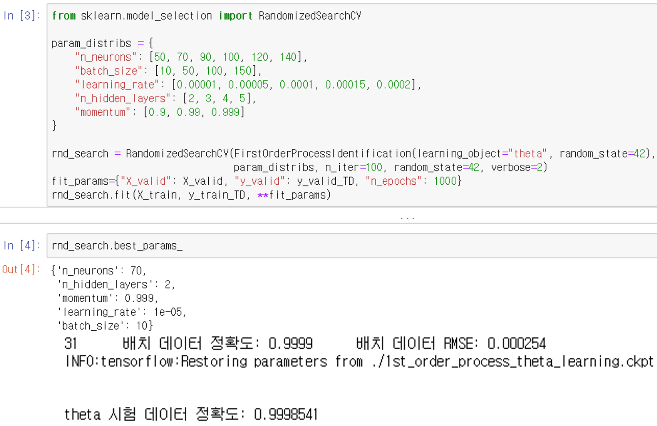
마지막으로, RandomizedSearchCV를 이용하여 최적의 하이퍼파라미터를 찾아봤다. 앞에서 봤듯이 RandomizedSearchCV는 교차시험을 통해 최적의 하이퍼파라미터를 찾는 과정을 자동화해준다. “뉴런 개수”: [50, 70, 90, 100, 120, 140], "미니배치 크기": [10, 50, 100], "학습률": [0.00001, 0.00002, 0.00005, 0.0001], "은닉층 수": [2, 3, 4, 5], "momentum": [0.9, 0.99, 0.999] 중 하나씩 무작위로 선택해 100번의 교차 시험을 하도록 했으며 이에 따른 최적의 하이퍼파라미터는 다음과 같다.

* **시간지연이 있는 1차 공정 시상수 학습**

****

그림2-11-(1). 교차시험을 통해 얻은 최적의 하이퍼파라미터와 마지막 에포크에서의 정확도, RMSE (뉴런 개수: 100, 은닉층 수: 2, momentum:0.999, 학습률: 0.00001, 배치 크기: 50)

* **시간지연이 있는 1차 공정 시간지연 학습**

****

**그림2-11-(2). 교차시험을 통해 얻은 최적의 하이퍼파라미터와 마지막 에포크에서의 정확도, RMSE** **(뉴런 개수: 70, 은닉층 수: 2, momentum:0.999, 학습률: 0.00001, 배치 크기: 10)**

* **시간지연이 있는 2차 공정 시상수 학습**

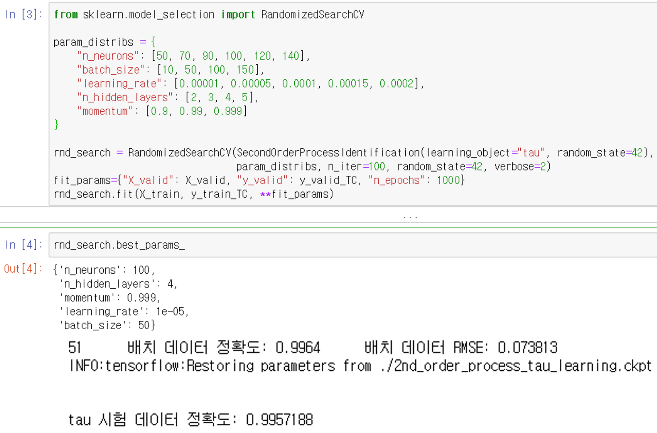
****

그림2-11-(3). 교차시험을 통해 얻은 최적의 하이퍼파라미터와 마지막 에포크에서의 정확도, RMSE (뉴런 개수: 100, 은닉층 수: 4, momentum:0.999, 학습률: 0.00001, 배치 크기: 50)

* **시간지연이 있는 2차 공정 감쇄비 학습**

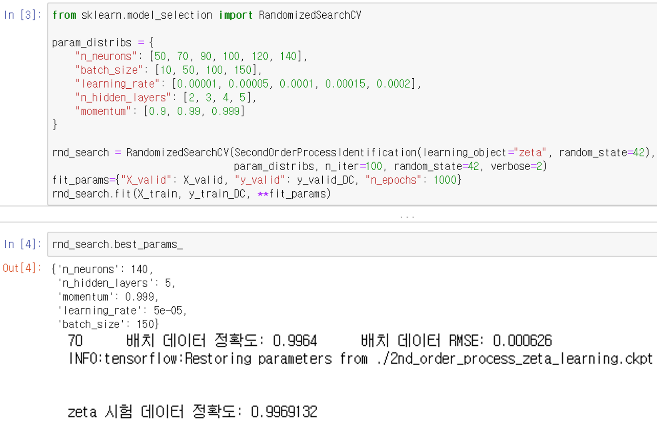
****

그림2-11-(4). 교차시험을 통해 얻은 최적의 하이퍼파라미터와 마지막 에포크에서의 정확도, RMSE (뉴런 개수: 140, 은닉층 수: 5, momentum:0.999, 학습률: 0.00005, 배치 크기: 150)

* **시간지연이 있는 2차 공정 시간지연 학습**

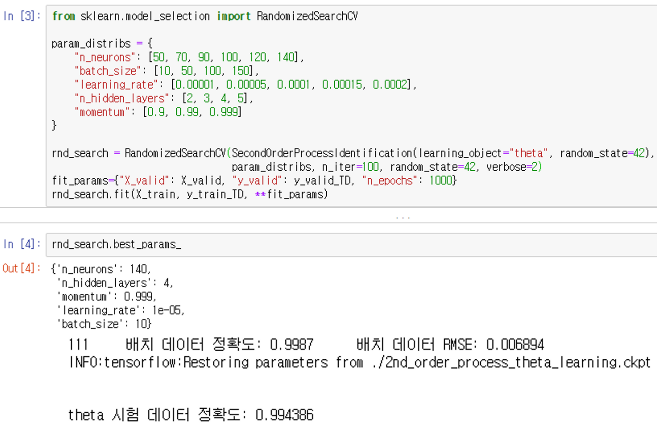
****

그림2-11-(5). 교차시험을 통해 얻은 최적의 하이퍼파라미터와 마지막 에포크에서의 정확도, RMSE (뉴런 개수: 140, 은닉층 수: 4, momentum:0.999, 학습률: 0.00001, 배치 크기: 10)

다음의 그림 2.12는 심층학습에 이용된 심층 학습 시스템의 계산 그래프이다.

최종으로, 교차시험을 통해 찾은 최적의 하이퍼파라미터를 이용하여 실제 액위계의 모델링을 해서 이 방법이 실제 물리적 장치에서 효과가 있는지에 대해 알아보았다.

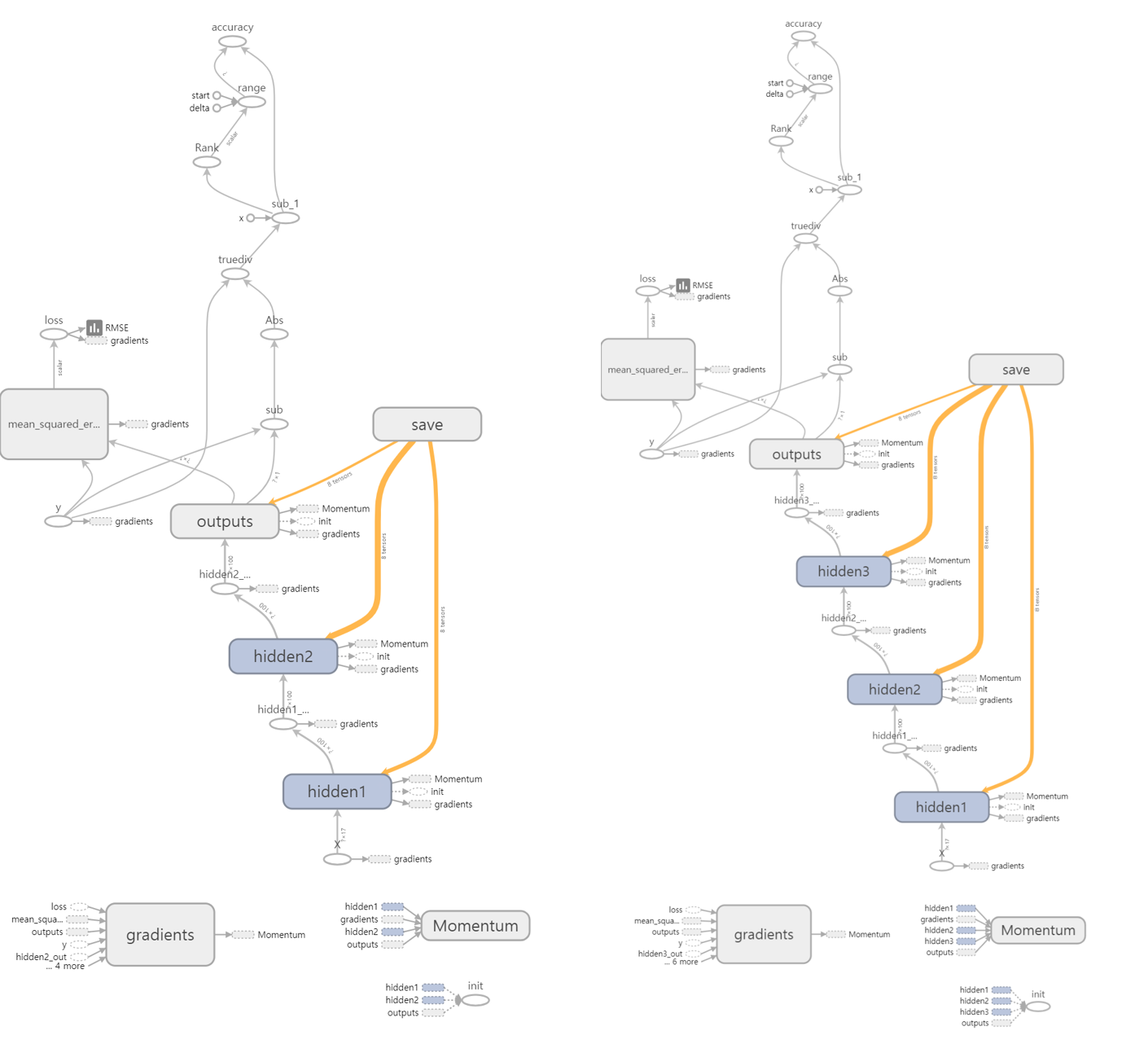


그림2- 12. 실험에서 사용된 심층학습 시스템의 계산그래프: 은닉층이 2개(왼쪽)인 것과 3개(오른쪽)인 것이 사용됐다. 아래쪽에서 데이터가 주입되면 은닉층(hidden)을 따라 outputs(출력)에 도달하고 손실함수인 mean\_squared\_error\_root에서 손실함수의 결괏값이 계산된다. Save에서 가중치가 저장되는 것도 다시 불리는 것도 확인할 수 있다

* **액위계의 공정 모델링**

(~~~ 액위 실험 내용 추가)

1. 결과 및 고찰

(~~~ 액위 실험 내용 추가)

reference

1. seborg, mellichamp, doyle, edgar, process dynmics and control techniques 4th ed. , John Wiley ＆ Sons.(2017)

2. seborg, mellichamp, doyle, edgar, process dynmics and control techniques 4th ed. , John Wiley ＆ Sons.(2017)

3. coughanowr, LeBlanc, 최신공정제어, 3rd Ed., McGraw-Hill.(2010)

4. seborg, mellichamp, doyle, edgar, process dynmics and control techniques 4th ed. , John Wiley ＆ Sons.(2017)

5. seborg, mellichamp, doyle, edgar, process dynmics and control techniques 4th ed. , John Wiley ＆ Sons.(2017)

6. 어영구, 최신공정제어공학-MATLAB의 활용과 실습 2nd ed., McGraw-Hill.(2014)

7. seborg, mellichamp, doyle, edgar, process dynmics and control techniques 4th ed. , John Wiley ＆ Sons.(2017)

8. Aurélien Géron, Hands-On Machine Learning with Scikit-Learn ＆ Tensorflow, O’REILLY.(2018)

9. 사이토 고키, 밑바닥부터 시작하는 딥러닝, O’REILLY(2017)

10. 사이토 고키, 밑바닥부터 시작하는 딥러닝, O’REILLY(2017)

11. 어영구, 최신공정제어공학-MATLAB의 활용과 실습 2nd ed., McGraw-Hill.(2014)

12. 사이토 고키, 밑바닥부터 시작하는 딥러닝, O’REILLY(2017)

13. seborg, mellichamp, doyle, edgar, process dynmics and control techniques 4th ed. , John Wiley ＆ Sons.(2017)

14. MATLAB 공식문서

15. MATLAB 공식문서

16. seborg, mellichamp, doyle, edgar, process dynmics and control techniques 4th ed. , John Wiley ＆ Sons.(2017)

17. 사이토 고키, 밑바닥부터 시작하는 딥러닝, O’REILLY(2017)

18. 사이토 고키, 밑바닥부터 시작하는 딥러닝, O’REILLY(2017)

19. Aurélien Géron, Hands-On Machine Learning with Scikit-Learn ＆ Tensorflow, O’REILLY.(2018)

20. Aurélien Géron, Hands-On Machine Learning with Scikit-Learn ＆ Tensorflow, O’REILLY.(2018)