* 1. **친환경적 폭발성 폐기물 처리 공정 운전 최적화**

|  |  |
| --- | --- |
| **학습 내용** | |
| [문제] | 폭발성 폐기물 처리 공정의 이해 |
| [방법] | 폭발성 폐기물 처리 공정의 대리 모델(Surrogate model) 구축 |
| [응용] | 대리 모델(Surrogate model)의 최적 운전 조건 도출 |
| [요약] | * 폭발성 폐기물 공정에 대한 이해 * 대상 시스템의 입력과 응답의 관계를 인공신경망으로 모사 * 기울기 정보를 사용하지 않는 최적화를 수행하여 최적 조건 도출 |

### **[공정 설명] 폭발성 폐기물 처리 공정의 이해**

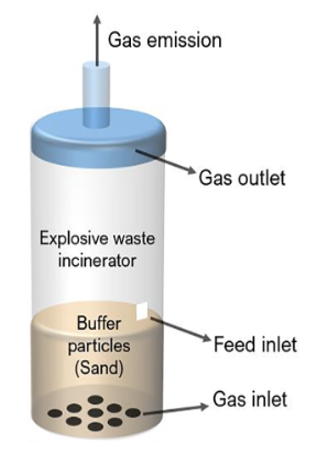


그림 1. 폭발성 폐기물 처리를 위한 유동층 반응기 구조

화학 반응을 이용한 군용 무기 제작 시 2-methyl-1,3,5-trinitrobenzene(TNT)와 같은 폭발성 물질들이 다량 사용되며 그 결과 많은 양의 폭발성 폐기물이 배출된다. 이러한 폐기물은 영구적으로 보관될 수 없으며, 안전을 고려하여 조속히 폐기되어야 한다. 일반적으로, 폭발성 폐기물 처리를 위해 open detonation 방식과 open burning 방식을 사용하지만 이들은 많은 양의 폐기물을 처리할 수 없을 뿐만 아니라 다량의 질소산화물(Nitrogen oxides; NOx) 발생을 동반하여 환경적으로도 한계를 지닌다.

이러한 한계를 극복하고자 고온가스를 이용한 로터리 킬른(Rotary kiln) 반응기 기반의 소각 공정에 대한 논의가 이루어져 왔다. 소각 공정은 많은 양의 폐기물을 연속적으로 처리할 수 있고 기존 방법에 비하여 NOx 배출량을 90% 이상 줄일 수 있다는 장점이 있다. 하지만, 로터리 킬른 반응기는 반응기 내부 온도 조절이 불가능하며, 이로 인해 반응기 내부에서 온도가 급격히 증가하는 핫스팟(hot spot)과 그로 인한 NOx 배출량 증가 시 대처가 어렵다는 단점을 지닌다. 유동층 반응기(Fluidized bed reactor)는 로터리 킬른 반응기에 비하여 반응기 내부 온도 조절이 용이하기 때문에 핫스팟 발생을 억제할 수 있고, 결과적으로 NOx 배출량도 대폭 감소시킬 수 있다.

폭발성 폐기물을 다루는 공정의 특성상 실제 실험 전에 반응기 내부 유동의 정확한 모사가 필수적이다. Computational fluid dynamics(CFD)는 식 (1), (2)의 Navier-Stokes 방정식 등을 기반으로 유체의 복잡한 전달 현상(transport phenomena)을 정확하게 모사할 수 있는 전산모사 방법이다.

|  |  |
| --- | --- |
|  | (1) |
|  | (2) |

CFD는 높은 정확도를 보이지만 많은 계산을 필요로 한다. CFD를 이용하여 최적화를 수행하는 경우, 계산량 감축을 위해 일반적으로 각 변수를 특정 범위 내에서 임의의 간격으로 분할하여 부분집합을 생성함으로써 수행한다. 하지만 이러한 접근법은 국부해(local optimum)로 수렴되어 전역해(global optimum)를 구하지 못할 가능성이 높다. 계산량 감축과 전역해 도달을 위한 한 가지 대안으로, 실제 함수(underlying function)를 모사하는 대리 모델(surrogate model)을 생성하고, 이를 통해 최적화를 수행하는 방법이 있다. 대리 모델은 많은 반복 계산을 필요로 하는 최적화 과정에서, 계산량을 대폭 감축시킴으로써 빠른 연산을 가능케 한다.

최적화는 크게 기울기 기반 방법(derivative-based optimization)과 기울기를 사용하지 않는 방법(derivative-free optimization)이 있다. 기울기 기반 방법은 수렴 속도가 빠른 반면에 국부해로 수렴될 가능성이 높으며, 이로 인해 많은 경우 유전 알고리즘(genetic algorithm) 등의 기울기를 사용하지 않는 방법이 선호된다.

본 챕터에서는 CFD 시뮬레이션 데이터를 통해 인공신경망 대리 모델을 구축하고 기울기를 사용하지 않는 최적화 방식을 사용하여 효율적인 최적화를 수행하는 것을 목적으로 한다.

### **[문제]**

**온도, 압력, 가스 유입 속도 등 반응기 운전 조건과 유입되는 입자의 조성, 입자의 크기 등 원료 조건이 질소산화물 배출에 미치는 영향을 나타내는 인공신경망 모델을 구축하고 메타휴리스틱(Metaheuristics) 최적화 방법을 적용하여 질소산화물 배출량을 최소로 하는 운전조건을 규명하자.**

* **MATLAB의 Statistic and Machine Learning Toolbox를 이용하자**
* **표준화된 데이터 ‘RawX.xlsx’과 ‘RawY.xlsx’를 이용하자**

**\* 데이터는 보안 문제로 인하여 표준화된 형태로 제공됨**

### **[방법] 폭발성 폐기물 처리 공정의 인공신경망 모델 구축**

#### 관련 라이브러리 설치 및 데이터 불러오기.

1. MATLAB을 사용하는 경우에 Statistics and Machine Learning Toolbox를 설치해야 한다. 해당 Toolbox를 사용하여 간편하게 인공신경망을 구축하고 최적화를 수행할 수 있다.

xlsread 를 통해 해당 경로에 있는 엑셀 파일을 불러올 수 있다.

|  |
| --- |
| RawX = xlsread('RawX.xlsx');  Response = xlsread('RawY.xlsx'); |

RawX는 xlsread 를 통해 해당 경로에 있는 엑셀 파일을 불러올 수 있으며, 아래 그림과 같이 Workspace에 두 변수 double 형태로 만들어진다.



#### 데이터 표준화

1. 질소산화물 배출에 영향을 미치는 다섯 파라미터를 표준화하여 모델의 정확도를 향상시킬 수 있다. RawX(:, 1)은 배열 RawX을 인덱싱하는 것이다. x1에 RawX 중 1번째 열(column)에 있는 모든 값을 할당한다.

모든 파라미터를 0부터 1 사이의 값으로 변환시키기 위하여 각 파라미터의 최댓값과 최솟값을 정의하면 Std\_x1과 같이 표준화된 파라미터가 생성된다.

\*데이터 보안 상 실제 수치는 다를 수 있다.

표준화를 마친 다섯 변수 Std\_x1, Std\_x2, … , Std\_x5를 학습에 사용하기 위하여 다시 결합하여 StdX 라는 배열을 생성한다.

|  |
| --- |
| x1 = RawX(:, 1); %배열 인덱싱  x2 = RawX(:, 2);  x3 = RawX(:, 3);  x4 = RawX(:, 4);  x5 = RawX(:, 5);  min\_x1 = k1; % 각 파라미터 최댓값 및 최솟값 설정  max\_x2 = k2;  Std\_x1 = (x1-min\_x1)/(max\_x1-min\_x1); %표준화  Std\_x2 = (x2-min\_x2)/(max\_x2-min\_x2);  Std\_x3 = (x3-min\_x3)/(max\_x3-min\_x3);  Std\_x4 = (x4-min\_x4)/(max\_x4-min\_x4);  Std\_x5 = (x5-min\_x5)/(max\_x5-min\_x5);  StdX = [Std\_x1, Std\_x2, Std\_x3, Std\_x4, Std\_x5]; %데이터 결합 |

테이블이(가) 표시된 사진

자동 생성된 설명

#### 인공신경망 구조 설정.

1. 인공신경망에 사용하는 데이터는 열(column)을 기준으로 한다. 따라서 표준화된 데이터를 전치하여 데이터 형태를 맞춘다. Training function은 경사하강법 등 여러 가지를 사용할 수 있지만 일반적으로 다른 방법에 비하여 비교적 빠르게 수렴하는 Levenberg-Marquardt 방법을 사용한다. 은닉층의 노드(node) 개수는 hiddenLayerSize로 정의하는데 은닉층(hidden layer)이 한 층인 경우는 아래와 같이 선언하며 여러 층인 경우는 [10, 5, 4]와 같이 행벡터(Row vector)로 표현한다. net = fitnet(hiddenLayerSize,trainFcn)은 은닉노드의 크기가 hiddenLayerSize이고 trainFcn을 training function으로 하는 신경망 구조를 생성한다. 전체 데이터 중 학습데이터를 70%, 검증데이터를 15%, 평가데이터를 15%로 설정하였다. 학습을 진행할 최대 epochs를 1500으로 설정하였으며 학습은 최대 100초 동안 진행한다.

\*net에 대한 자세한 설명은 MATLAB Command window에 net을 입력하면 볼 수 있다.

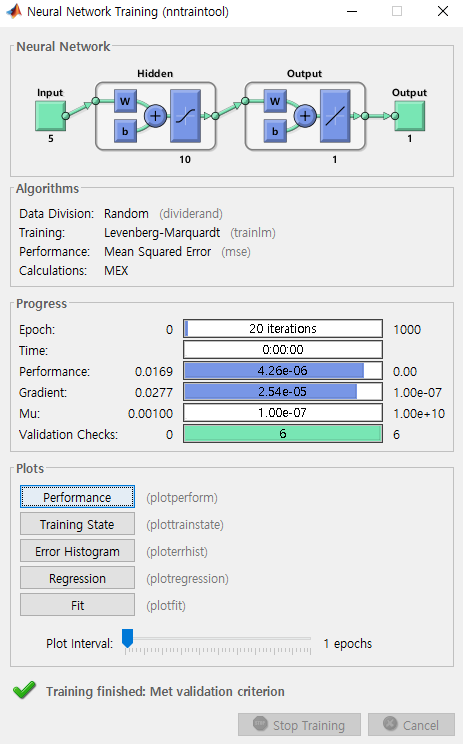
|  |
| --- |
| x = StdX' ; %데이터 구조 변환  t = Response' ; %데이터 구조 변환  trainFcn = 'trainlm'; %Training function 설정  hiddenLayerSize = 10; %은닉층 노드(hidden node) 설정  net = fitnet(hiddenLayerSize,trainFcn); %신경망 생성  net.divideParam.trainRatio = 70/100; %학습데이터 비율 설정  net.divideParam.valRatio = 15/100; %검증데이터 비율 설정  net.divideParam.testRatio = 15/100; %평가데이터 비율 설정  net.trainParam.epochs = 1500; %학습할 최대 Epoch 횟수 설정 (default : 1000)  net.trainParam.time = 100; %학습할 최대 시간 설정(단위 : 초) (default : inf) |

위 단계를 수행하면 workspace에 class가 network인 net이 생성된다.

#### 인공신경망 학습.

1. Train 커맨드를 사용해서 신경망을 훈련시키며 train(net, x, t)와 같이 사전에 선언한 신경망 구조와 입력과 응답을 입력한다. gsubtract 커맨드를 통해 실제 응답(t)과 신경망 응답(y)의 차이를 구하고, perform 커맨드를 통해 신경망 성능을 정량화한다. net과 훈련 기록을 반환하며, workspace에 class가 struct인 훈련 기록 tr이 저장된다(그림 2). view 커맨드는 직접 구성한 신경망 구조를 확인할 수 있다(그림 3). Parity plot을 통해서 인공신경망의 성능과 오버피팅(overfitting) 여부를 확인한다(그림 4).

|  |
| --- |
| [net,tr] = train(net,x,t); %신경망 훈련 및 net, tr(훈련 기록) 반환  y = net(x);  e = gsubtract(t,y); %두 배열 또는 셀의 성분의 차를 구함  performance = perform(net,t,y); %신경망 성능을 계산함  view(net) %신경망 구조 반환 |

테이블이(가) 표시된 사진

자동 생성된 설명

그림 2. 훈련 결과(좌)와 훈련 기록(우)

텍스트, 시계이(가) 표시된 사진

자동 생성된 설명

그림 3. 신경망 구조

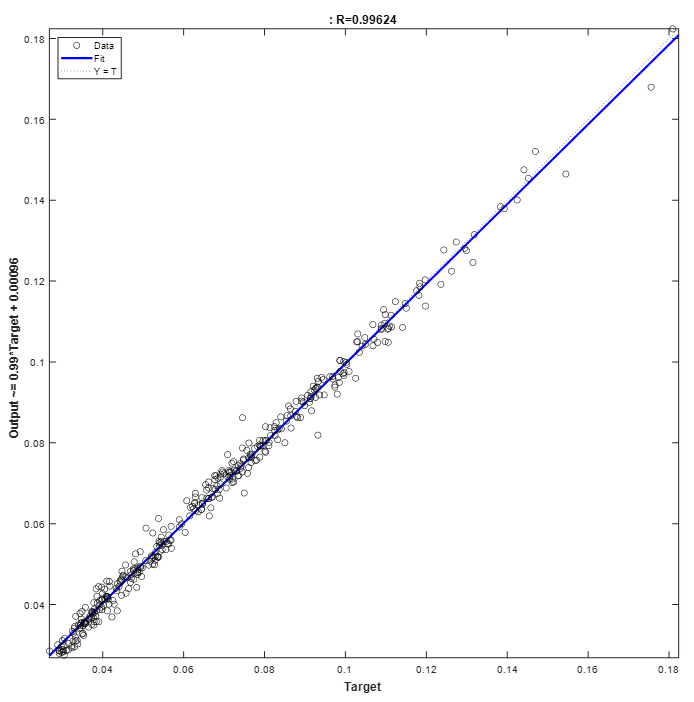


그림 4. Parity plot을 통한 모델 결과 확인

#### 인공신경망을 함수로 불러오기.

1. genFunction을 통해서 학습한 신경망인 net을 함수 형태로 저장할 수 있다. NNfunction.m 이라는 파일이 MATALB 경로에 생성된다. MATLAB은 이처럼 m파일(‘-.m’형태의 파일)을 함수처럼 사용할 수 있는 기능을 제공한다. fun = @(x) NNfunction(x)는 NNfunction이라는 함수를 x에 대한 함수 fun으로 선언하는 것을 의미한다. 이때, @(x) 는 NNfunction이 x, y, z에 대한 함수인 경우에도 x만을 변수로 한다. Function handle 성능 확인을 위해 x의 첫 번째 열을 인덱싱한 후 fun의 응답을 확인하고 이를 y와 비교한다.

인공신경망 함수를 최적화의 목적 함수로 사용하기 위하여 형태를 변형시켜야 한다. 먼저, genFunction은 함수를 셀(cell)형으로 반환하기 때문에 계산된 값이 스칼라가 아닌 셀형태로 반환된다. 스칼라로 목적 함수를 반환받기 위하여 ‘MatrixOnly’를 ‘Yes’로 선언한다. 다음으로, 데이터 입력 형태에 관한 변환이다. 현재 열을 기준으로 학습되어 있는 인공신경망을 최적화에 사용하기 위하여 행으로 변환해야한다. 따라서 NNfunction의 인수를 전치하여 입력한다.

|  |
| --- |
| genFunction(net,'NNfunction', ‘MatrixOnly’, ‘yes’); %학습된 신경망 코드 생성  fun = @(x) NNfunction(x’); %신경망 구조를 갖는 함수 생성(function handle)  [Command Window]  >> y1 = fun(x(:, 1)) %function handle 결과 확인  ans = 0.0495 |

### **[응용] 메타휴리스틱 방법을 통한 최적 조건 규명**

대상 함수의 기울기 정보를 사용하는 최적화는 기울기 정보가 정확하고 얻기 쉬운 경우에는 최적해를 빠르게 얻을 수 있다. 하지만, 실제로 기울기에 대한 정보를 얻을 수 있는 경우는 제한적이며 이에 대한 신뢰도를 보장할 수 없다. 이러한 한계는 기울기 정보를 사용하지 않는 derivative-free optimization(DFO)를 통해 극복할 수 있다. 유전알고리즘(Genetic algorithm)은 메타휴리스틱 최적화 방법 중 가장 널리 사용되는 방법으로 변이(mutation), 교배(crossover) 등을 기반으로 최적화를 진행한다.

|  |
| --- |
| Numvar = 5; %입력 파라미터 개수  lowerbound = [0 0 0 0 0]; %각 변수의 최솟값  Upperbound = [1 1 1 1 1]; %각 변수의 최댓값  opts = optimoptions(@ga, 'PlotFcn',{@gaplotbestf}); %최적화 옵션  [x,fval,exitFlag,Output] = ga(fun,Numvar,[],[],[],[],lowerbound,Upperbound,[], opts); |

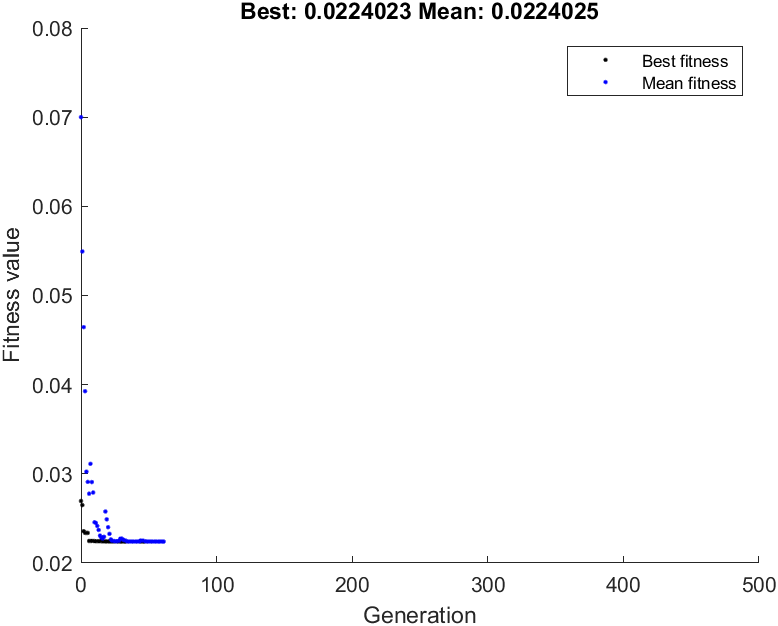


그림 5. 유전알고리즘 수행 과정

최적화 옵션을 통해 다양한 파라미터를 설정할 수 있으며 그림 5와 같이 최적화가 수행되는 과정을 도식화 할 수 있다**.** 최적화 결과 질소산화물의 최솟값은 0.0224로 구해지며 그때의 운전조건은 다음과 같다.

하지만, 이처럼 유전 알고리즘과 같은 DFO 알고리즘을 통해 얻은 해를 전역 최적해(global optimal)이라고 보장할 수 없으며 다양한 알고리즘을 적용하여 최대 혹은 최솟값을 도출하는 알고리즘의 해를 선택한다.

### **[결론]**

공학 분야에서 대상으로 하는 대부분의 공정은 입∙출력 변수들 간의 관계가 매우 복잡하여, 시스템의 응답을 정확하게 예측하고 최적화하는데 어려움이 있다. 본 챕터에서는 폭발성 폐기물을 처리하는 공정에서 발생하는 질소산화물 저감을 목적으로, 인공신경망 기반의 대리 모델을 구축하고, 기울기 정보를 사용하지 않는 최적화기법을 통해 최적화함으로써, 최적의 운전 조건 및 원료 조건을 도출하였다. 모든 변수에 대하여 최적 운전조건이 결정론적으로 도출되었으며 해당 운전조건에서 0.0224 만큼의 질소산화물이 배출되는 것을 확인하였다.

### **[학습 결과]**

* 학습 내용

공정 운전 조건 최적화를 위한 신경망 대리 모델 구축 및 최적화 적용 방법 익히기

* 학습 결과 확인하기

신경망을 통한 시스템 모사 방법 및 파라미터 변경 방법, 최적화 진행 과정 분석하기

* 학습 결과 응용하기

수학적으로 표현하기 어려운 시스템을 대리 모델 구축과 기울기를 사용하지 않는 최적화를 이용하여 효율적으로 최적화하기

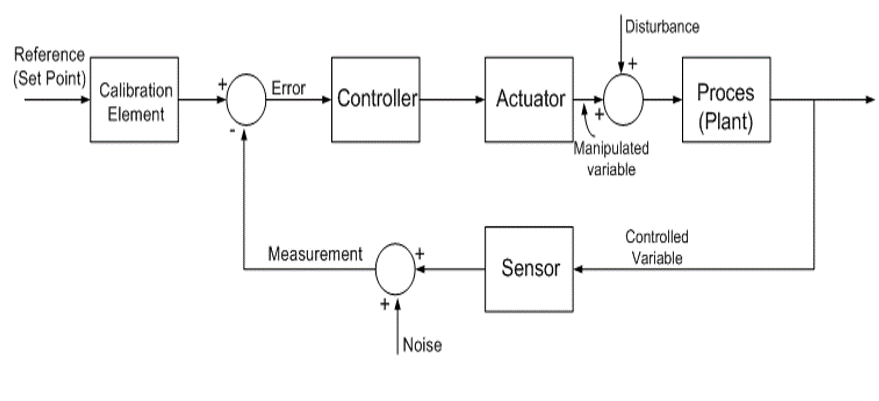
* 1. **PID 제어 시스템을 이용한 공정제어**

|  |  |
| --- | --- |
| **학습 내용** | |
| [문제] | PID 제어 시스템의 이해 |
| [방법] | PID 제어 시스템의 조율 |
| [응용] | 딥러닝을 적용한 PID 제어 시스템의 조율 |
| [요약] | * PID 제어 시스템의 이해 * 기존 PID 제어 시스템의 조율 방법 * 딥러닝을 이용한 PID 제어 시스템의 조율 |

### **[이론] PID 제어 시스템**

### **공정제어의 개요**

**공정제어**(process control)란 온도, 압력, 유량 등 공정의 상태를 측정하여, 해당 상태가 우리가 원하는 조건으로 도달될 수 있도록 공정의 변수를 조작하는 행위를 일컫는다. 실제 공정의 운전에서는 외란으로 인해 공정을 원하는 운전조건으로 일정하게 유지하는 것이 어려우며, 제어가 제대로 이루어지지 않을 경우, 제품의 품질 불량, 수율 감소, 안전 사고 등의 원인이 될 수 있다. 한 예로, 냉각수의 유량을 조절하여 발열 반응이 진행되는 반응 내부 온도를 제어하는 시스템에서, 외부 온도, 냉각수 온도, 냉각수 압력 등 외부적인 요인의 변화, 즉 외란의 변화에 의해 시간의 흐름에 따라 반응기 내부 온도가 계속해서 변화하며, 원하는 반응 전환율을 도달하기 위해 또한 반응을 안전한 온도 범위에서 유지시키기 위해 정밀한 제어가 필요하다.



**그림 1. 피드백 제어 시스템의 블록 흐름도 예시**

제어시스템은 제어기를 중심으로, 제어기가 조절하는 조작 변수(manipulated variable; MV), 제어기가 제어하고자 대상이 되는 제어 변수(controlled variable; CV), 도달하고자 하는 설정 변수(목표값, set variable; SV), 그리고 제어 변수에 영향을 미치나 제어기가 조작할 수 없는 외란 변수(disturbance variable; DV)로 구성된다. 공정의 관점에서, 조작 변수는 공정에 입력되는 공정 입력(process input)이며, 제어 변수는 공정 출력(process output)이다. 그림 1에 보인 바와 같이, 제어기는 센서를 이용하여 공정의 상태인 제어 변수 값을 측정한 후, 이를 설정 변수와 비교하여, 그 차이가 최소화되도록 조작 변수의 값을 결정한다.

### **PID 제어의 이해**

PID 제어는 산업계에서 가장 널리 이용되는 제어기법으로, 계산량이 작고 그 적용이 쉽다는 장점이 있다. PID 제어는 P 제어(proportional control; 비례 제어), I 제어(integral control; 적분 제어), 그리고 D 제어(derivative control; 미분 제어)로 구성된다. P 제어와 I 제어는 단독으로도 사용되며, 경우에 따라서는 PD 제어, PI 제어 등의 형태로 변형되어 사용된다.

텍스트, 전자기기이(가) 표시된 사진

자동 생성된 설명

**그림 2. PID 제어 시스템의 블록 흐름도**

P 제어는 현 시점의 제어오차에 비례하는 제어동작을 취한다. P 제어는 계산이 간단하여 응답이 빠르다는 장점이 있으나 잔류오차(offset)가 발생한다는 단점이 있다. 여기에서, 잔류 오차란 시간의 흐름에 따라 설정 변수와 제어 변수의 차이가 감소하지 않고 유치된 채 지속되는 오차를 가리킨다. P 제어는 식 (1)과 같이 나타낼 수 있다.

식 (1)

여기서 ubias는 제어오차가 0일때 조작 변수가 머무르는 특정 값(편차)를 나타낸 것이다. KC는 비례 이득(proportional gain) 혹은 제어 이득(control gain)으로 P 제어의 성능을 결정하는 조율 인자(tuning factor)이다. KC가 클수록 P 제어기는 같은 크기의 제어오차에 대해 더욱 민감하게 반응한다. 한편, PB(proportional band)는 100/KC로 정의된다.

I 제어는 시간의 흐름에 따라 계속해서 제어오차를 누적하고, 누적된 제어오차에 비례하여 제어동작을 산출한다. 이러한 제어오차의 누적과 이를 통한 제어동작은, 제어오차가 0이 될 때까지 지속되므로, P 제어의 단점인 잔류 오차 문제를 해결할 수 있다. 다만, I 제어의 경우, 충분한 제어오차가 누적되어야 큰 제어동작을 산출하므로, 제어 응답이 느리다는 단점을 지닌다. I 제어의 식은 다음과 같다.

식 (2)

KI는 적분 시간(integral time) 또는 리셋 시간(reset time)으로, 제어오차가 일정하게 지속되는 경우 적분 동작은 매 KI마다 비례제어동작에 해당하는 제어동작을 누증시키는 것을 의미한다. 즉, KI가 작을수록 I 제어는 더 자주 제어오차를 누증시키므로, 더 큰 제어동작을 취하게 된다. KI의 역수를 리셋 속도 (reset rate; Ri)라 하며, 1/시간의 단위를 갖는다.

D 제어는 제어오차의 기울기에 비례하는 제어동작을 산출한다. D 제어는 P 제어와 마찬가지로 잔류오차를 발생시키며, 미분(즉, 제어오차의 기울기)에 기반하여 제어 동작을 산출하므로, 높은 주파수를 갖는 노이즈에 민감한 단점을 지닌다.

식 (3)

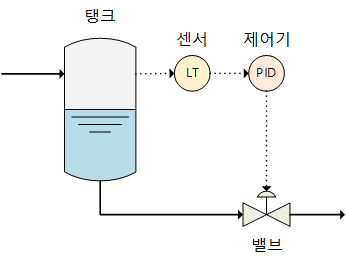
일반적으로, D 제어는 그 특성상, P 제어나 I 제어와 달리 단독으로 사용하지 않는다. 식 (3)에 PD 제어를 나타내었다. 이때, KD는 미분 시간(derivative time) 또는 선행 시간(preact time)으로, 제어오차가 일정한 속도로 증가하는 경우, 미분동작은 초기에 비례동작보다 KD 앞선 제어동작을 출력함을 의미한다.

### **인공지능 기반의 PID 제어**

PID 제어는 비례 이득, 적분 시간, 미분 시간으로 구성된 조율 변수에 따라 그 성능이 결정된다. 오랜 기간 동안 이러한 PID 제어의 최적 조율 변수 값을 결정하기 위한 많은 방법들이 제안되었으나, 그 적용 과정이 복잡하고 많은 시행착오 과정을 동반하여 널리 적용되지 않고 있다. 본 챕터에서는 인공지능 학습을 통한 PID 제어의 조율 방법을 소개한다.

### **[문제]**

**수조에 찬 물의 수위를 PID 제어 시스템을 통해 제어하고자 한다. 관련 데이터는 ‘matplotlib.pyplot’를 불러와 이용할 때, PID 제어 시스템에 딥러닝을 적용해 PID 제어 시스템의 KP, KI, KD를 조율하라. 반복 계산 횟수는 100,000번으로 하고, 절댓값 오차를 사용한다. 초기의 조건은 KP=1.0, KI=0.5, KD=0.5이고, 오차축적 횟수는 100, 학습률(learning rate)은 0.001, 초기 오차는 10이다.**



### **[방법] PID 제어 시스템의 조율**

#### PID 제어 시스템의 조율 방법 중 가장 대표적인 지글러-니콜스 조율(Ziegler-Nichols tuning) 방법에 대해 설명하여라.

1. 지글러-니콜스 조율은 1942년 Ziegler와 Nichols에 의해 제안된 방법 중 하나로, 경계안정상태의 특성에 근거한 제어기 조율 방법 중 하나이다. 지글러-니콜스 조율은 먼저 I 제어와 D 제어를 끄고 P 제어기만 켠 상태 또는 KI에 무한에 가까운 수를 넣고, KD에는 0을 입력한 상태에서 진행한다. 이 상태에서 그림 3과 같이 공정 출력에 지속적인 진동 현상이 나타날 때까지 KC를 작은 값에서부터 서서히 증가시킨다.

텍스트, 하늘, 다른이(가) 표시된 사진

자동 생성된 설명

**그림 3. 공정 출력의 지속적인 진동 현상**

KC가 작을 때에는 목표값에 수렴하다가 KC가 점점 커지면서 오차에 민감해지기 때문에 그림 3과 같은 규칙적인 진동이 관측되기 시작한다. 만약 이보다 KC가 더 커지면 공정 출력은 발산한다. 규칙적인 진동의 주기를 Pu(ultimate period)라고 하며, 이때의 KC는 KCu(ultimate controller gain)이라고 한다. 마지막으로, KCu을 표 1에 제시된 표에 대입하여 제어 시스템을 조율한다.

테이블이(가) 표시된 사진

자동 생성된 설명

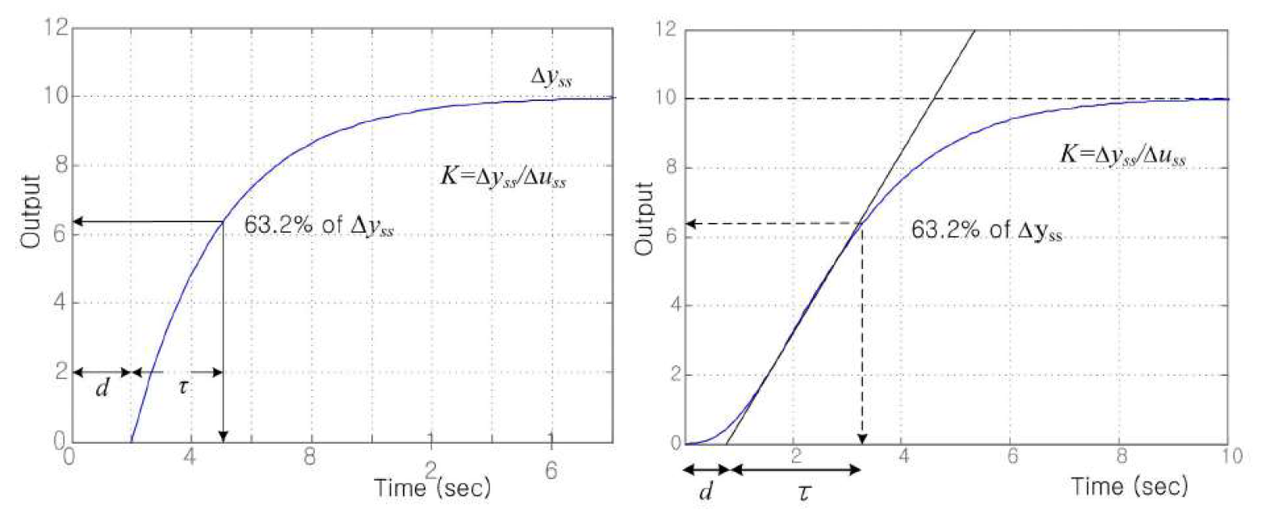
**표 1. 제어 시스템의 지글러-니콜스 조율 값**

#### 또 다른 PID 제어 시스템 조율 방법인 1차 시간지연(first-order plus dead time) 모델의 전제 조건은 무엇이며, 조율하는 방법을 설명하여라.

1. 1차 시간지연 모델은 식 (4)와 같이 나타낼 수 있다. 식 (4)는 시간 t를 라플라스 변환하여 나타낸 식이다.

식 (4)

식 (4)에서 d와 τ는 그림 4를 통해 알 수 있다. d는 지연 시간(dead time)으로, 제어기를 통해 제어출력이 나올 때까지의 시간을 의미한다. 실제 공정에서는 대부분 2차 함수와 같은 모습이 나타나는데, 이 경우 변곡점에서 접선을 그린 후 0 초부터x축과 만나는 점까지의 시간을 지연 시간으로 정의한다. τ는 시정수를 의미하며 제어 출력이 정상 상태의 제어 출력의 63.2%인 지점의 시간을 말한다.

**그림 4. (a) 1차 함수와 (b) 2차 함수의 반응응답곡선**

1차 시간지연모델의 전제 조건은 ‘0.1<d/τ<1.0’이다. d/τ<0.1일 때에는 지연 시간이 무시되는 1차 공정에 근접한다. 1차 공정의 경우 KC를 크게 할수록 제어가 잘된다. 반면, d/τ>1.0에서는 지연시간이 매우 커 일반적인 PID 제어 시스템으로는 제어성능에 한계가 생기므로 dead time compensator(ex. Smith predictor)가 필요하다.

|  |  |  |  |
| --- | --- | --- | --- |
| 제어기 | KC | KI | KD |
| P | (τ/KPd) | - | - |
| PI | 0.9(τ/KPd) | 3.33d | - |
| PID | 1.2(τ/KPd) | 2.0d | 0.5d |

**표 2. 제어 시스템의 1차 시간지연 모델 조율 값**

마지막으로, 1차 시간지연 모델로부터 지글러-니콜스 조율에 필요한 Kcu와 Pu를 구하여 지글러-니콜스 조율을 적용한다. Kcu와 Pu를 쉽게 계산하기 위해 표2와 같이 KP, d, τ값으로도 나타낼 수 있다.

### **[응용] 딥러닝을 통한 이산화탄소 배출량 예측**

#### ‘matplotlib.pyplot’을 불러오고, 탱크의 수위를 PID제어기를 정의하여라.

1. 먼저, 그림 5와 같이 윈도우 명령 프롬프트에 pip을 활용해 matplotlib을 컴퓨터에 설치한다. 그리고 아래의 절차에 따라 탱크의 수위와 PID 제어 시스템을를 정의한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

|  |
| --- |
| import matplotlib.pyplot as plt   #데이터베이스 불러오기 |

**그림 5.** matplotlib.pyplot **설치하기**

명령 프롬프트에 ‘python -m pip install -U matplotlib’를 입력하여 matplotlib.pyplot를 설치한다.

먼저, 파이썬에 설치한 matplotlib.pyplot를 불러온다.

|  |
| --- |
| class Liquid:      def \_\_init\_\_(self, error):          self.current\_error = error          self.last\_action = 0      def take\_action(self, action):          self.current\_error += 0.1 \* action          self.last\_action = action  class PID:      err\_sum = 0      old\_err = 0      def pid(self, current, goal, kp, ki, kd):          err = goal - current          self.err\_sum += err          delta\_err = err - self.old\_err          self.old\_err = err          return kp\*err + ki\*self.err\_sum + kd\*delta\_err |

위의 코드와 같이 탱크의 수위는 목표값과 실제 탱크의 수위만큼의 차인 error를 가져와 제어동작을 취함으로써 수위를 조절한다. 이때, ‘current’는 현재 탱크의 수위, ‘goal’은 제어기의 목표값을 나타내며, kp, ki, kd는 각 제어기의 조율 인자(KP, KI, KD)를 의미한다.

#### PID 제어 시스템을 조율하기 위한 함수를 정의해라

1. PID 제어 시스템 조율에 필요한 함수는 아래의 절차대로 정의한다.

|  |
| --- |
| class Derivative:      def \_\_init\_\_(self):          self.last\_x = 0          self.last\_y = 0      def get\_gradient(self, x, y):          d = (y - self.last\_y) / (x - self.last\_x)          self.last\_x = x          self.last\_y = y          return d |

위의 코드와 같이 미분 클래스를 정의한다. 미분 클래스는 기울기를 통해 조율 인자를 최적화할 때 사용된다.

|  |
| --- |
| class Train:      kp = 1.0; ki = 0.5; kd = 0.5      goal = 0      episode\_length = 100      learning\_rate = 0.001      def \_\_init\_\_(self):          self.dp = Derivative()          self.di = Derivative()          self.dd = Derivative()          self.step = 0          self.last\_loss = 0 |

위의 코드는 제어 시스템의 조율을 위해 해당 모델을 훈련시키기 위해 훈련 클래스를 정의하는 코드이다. 가장 먼저 문제에 주어진 초기 조건들을 입력하고 변수들을 정의한다. 목표값과 실제 탱크의 수위의 차이는 0이 되는 것이 이상적이므로 goal은 0으로 설정하고, 오차축적 횟수(episode\_length)는 100, 학습률은 0.001로 설정한다.

|  |
| --- |
| def abs\_mean(self, list):          sum = 0          for i in list:              sum += abs(i)          return sum / len(list)      def loss(self):          liquid = Liquid(10)          pid = PID()          error = []          for i in range(self.episode\_length):              error.append(liquid.current\_error)              liquid.take\_action(pid.pid(liquid.current\_error, self.goal, self.kp, self.ki, self.kd))          return self.abs\_mean(error)      def optimize(self):          self.kp = self.kp - self.learning\_rate \* self.dp.get\_gradient(self.kp, self.loss())          self.ki = self.ki - self.learning\_rate \* self.di.get\_gradient(self.ki, self.loss())          self.kd = self.kd - self.learning\_rate \* self.dd.get\_gradient(self.kd, self.loss())          self.last\_loss = self.dd.last\_y          print("step={}, kp={}, ki={}, kd={}, loss={}".format(self.step, self.kp, self.ki, self.kd, self.last\_loss))          self.step += 1 |

위의 코드는 훈련 클래스에서 정의된 함수들이다. ‘abs\_mean’은 문제에서 정의된 것처럼 오차를 구할 때 사용할 절댓값 평균 함수를 정의한다. ‘loss’는 for문을 통한 반복 계산을 통해 탱크의 수위를 보정하며 실제 탱크의 수위와 목표값의 차이(error)를 축적하는 함수이다. ‘loss’에서 구해진 error는 abs\_mean 함수를 통해 절댓값 평균으로 출력된다. 마지막으로, ‘optimize’는 앞서 정의한 함수들을 통해 최종적으로 조율 인자들을 최적화함으로써 PID 제어 시스템을 조율한다. 또한 이때, 뒤쪽의 그림 6과 같이 반복 계산을 하는 각 단계의 값들을 볼 수 있도록 설정했다.

#### 앞서 정의한 함수들을 통해 KP, KI, KD 값을 최적화하고 이를 도식화하라.

1. 아래와 같은 절차를 통해 조율 인자들을 최적화할 수 있다. 또한 그 결과는 그림 6과 7을 통해 각각 확인할 수 있다.

|  |
| --- |
| if \_\_name\_\_ == '\_\_main\_\_':      loss = []      kp = []      ki = []      kd = []      train = Train()      for j in range(100000):          train.optimize()          loss.append(train.last\_loss)          kp.append(train.kp)          ki.append(train.ki)          kd.append(train.kd) |

먼저, 위의 코드와 같이 계산된 데이터를 담을 배열을 생성하고, Train의 인스턴스를 저장할 변수(train)을 생성한다. 또한 앞서 정의한 ‘optimize’함수를 사용해 조율 인자들을 최적화한다. 이때 반복 횟수는 문제에서 언급한 것처럼 100000회로 설정한다.

텍스트이(가) 표시된 사진

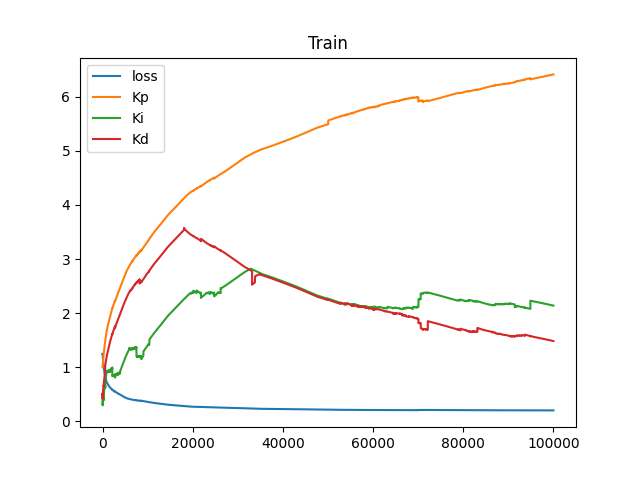
자동 생성된 설명

**그림 6. 조율 인자 최적화 결과 예시**

그림 6은 각 반복 횟수에 따른 최적화 결과를 나타낸다.

|  |
| --- |
| plt.plot(loss, label="loss")      plt.plot(kp, label="Kp")      plt.plot(ki, label="Ki")      plt.plot(kd, label="Kd")      plt.legend()      plt.title("Train")      plt.savefig("train.png") |

위의 코드는 그림 6에서 나타낸 최적화 과정을 도식화하는 코드이다. 위의 코드를 실행하면 그림 7과 같은 결과를 얻을 수 있다.



**그림 7. 조율 인자 도식화 코드 및 결과**

### **[결론]**

PID는 가장 기본적인 제어 시스템 중 하나로, 공정의 목표 값과 실제 데이터 간의 차이가 0이 되도록 조정하며 제어한다. 해당 예제에서는 딥러닝 기법을 적용하여 PID 제어 시스템을 튜닝했다. 그 결과, 반복 계산을 100,000회 반복하자 오차(loss)가 0으로 거의 수렴한 것을 확인할 수 있다. 더 많은 횟수를 반복한다면 더 정확하게 조율된 PID 제어 시스템을 얻을 수 있을 것이다.

### **[학습 결과]**

* 학습 내용

기본적인 PID 제어 시스템의 이해와 인공지능과의 접목이 필요한 이유 이해.

* 학습 결과 확인하기

대표적인 PID 제어 시스템 조율 방법 익히기.

* 학습 결과 응용하기

본 장의 학습내용에 기반해 실제 공정데이터를 이용해 딥러닝을 적용하여 PID 제어기를 조율하는데 응용.

* 1. **인공신경망 모델 기반 예측 제어를 이용한 공정제어**

|  |  |
| --- | --- |
| **학습 내용** | |
| [문제] | 모델 예측 제어(MPC)의 이해 |
| [방법] | 인공신경망 모델 기반 예측 제어 |
| [응용] | 인공신경망 모델 기반 예측 제어를 이용한 CSTR 제어 |
| [요약] | * 모델 기반 예측 제어의 이해 * Simulink를 이용한 인공신경망 모델 기반 예측 제어 구축 * 인공신경망 모델 기반 예측 제어를 이용한 CSTR 제어 |

### **[이론] 모델 예측 제어를 이용한 다변수 공정의 제어**

다변수 공정이란 입출력 변수가 두 개 이상인 공정을 의미하며, MIMO (multi-input multi-output) 공정이라 부른다. 이와 대비되는 개념으로, 단변수 공정 (single-input single-output; SISO) 공정이라 부른다. 증류탑을 비롯한 화학공학에서 우리가 다루는 대부분의 공정들은 다변수 공정에 해당하며, 변수들 간에 상호간섭이 존재한다. 이러한 상호간섭은 단변수 제어인 PID 제어의 성능을 저하시키는 요인으로 작용한다. 한 예로, 그림 1에서, PID1에 의해 공정입력 u1이 변화하는 경우, G11을 통해 y1이 변화할 뿐만 아니라, G12를 통해 y2가 변화하게 된다. 이러한 y2의 변화는 PID2에 제어오차로 작용하여, u2의 변화를 초래하며, 이는 다시 G21를 통한 y1의 변화를 초래한다. 이처럼, 다변수 공정에서는 상호간섭(G12 및 G21)이 존재하여, 하나의 공정 입력이 여러 개의 공정 출력(y1 및 y2)에 영향을 미치게 되므로, 상호간섭이 심각할 경우 단변수 제어로는 좋은 제어성능을 기대하기 어렵다. 실제 산업에 적용되는 실용형 단변수 제어기법에서는 이러한 상호간섭에 의한 영향을 최소화하고 제어성능을 개선하기 위해, 연계 제거 (decoupling) 또는 상대 이득 기반의 입력-출력 짝짓기 기법 (input-output pairing) 등이 이용된다.

한편, 실제 공정에서는 공정 입력 및 공정 출력 변수들에 제약조건이 존재한다. 예를 들어, 공정 입력 중 하나인 제어밸브의 경우, 0~100% 사이로만 운전 가능하며, 제어 대상인 반응기의 경우, 재질/두께에 따라 견딜 수 있는 최대 압력/온도가 존재한다. PID제어의 경우, 알고리즘 특성 상 이러한 제약조건을 고려할 수 없는 어려움으로 인해, low selector 또는 high selector의 도입을 통한 override 제어루프의 구성 또는 clamping 등의 방법을 통해 제어시스템을 보완하나, 그 설계가 복잡하고 성능에 한계를 지닌다.

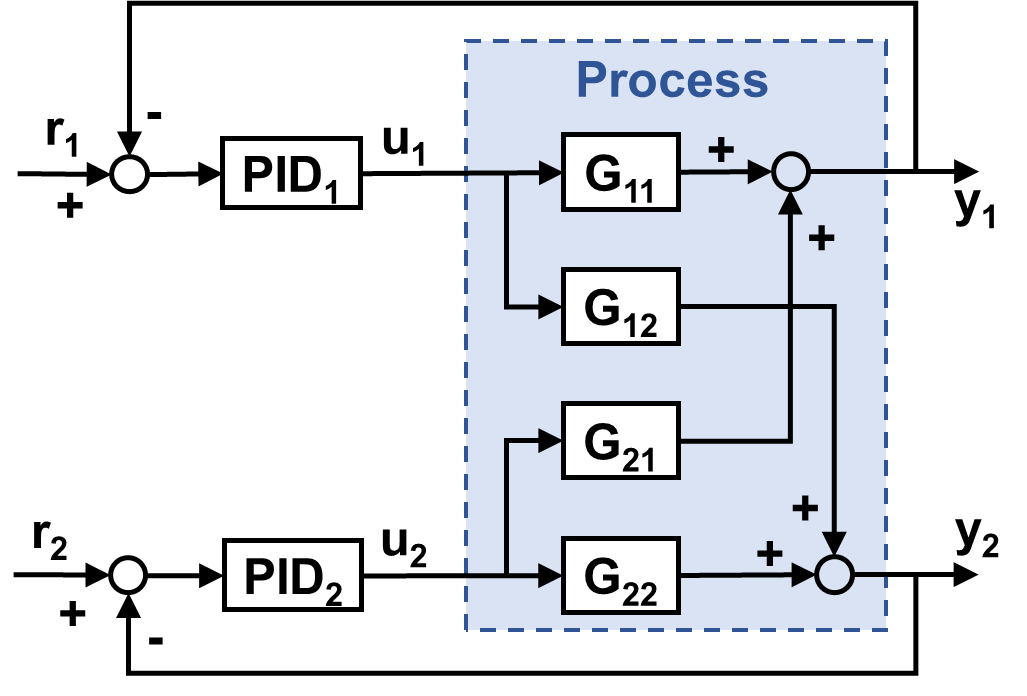


그림1. PID 제어를 이용한 다변수 공정의 제어

다변수 공정을 단변수 제어로 운전하는 것에 따른 어려움을 극복하기 위한 하나의 대안은 다변수 제어를 이용하는 것이다. 연속공정에 대한 대표적인 다변수 제어로서, 그림 2에 보인 바와 같이 모델 예측 제어(model predictive control)가 있다. 모델 예측 제어의 경우, 다변수 공정에 존재하는 변수들 간의 상호간섭을 고려하여 최적화 알고리즘을 통해 공정 입력 값을 계산하므로, 상호간섭이 심각한 다변수 공정에 대해, 보다 개선된 성능을 기대할 수 있다.

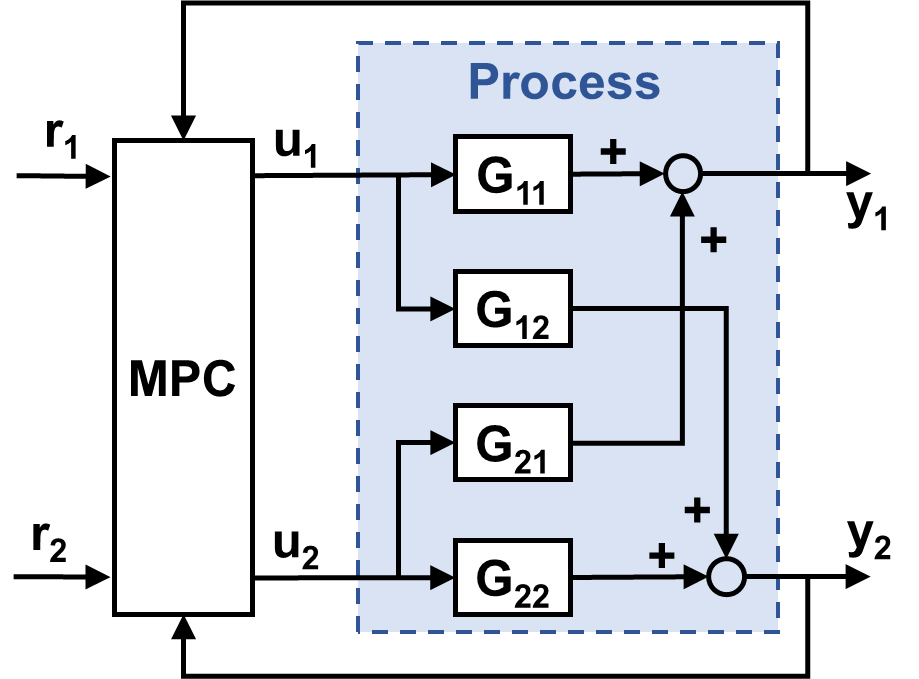


그림 2. 모델 예측 제어를 이용한 다변수 공정의 제어

그림 3은 모델 예측 제어의 동작 원리를 나타낸다. 모델 예측 제어는 실제 공정의 거동을 잘 모사할 수 있는 공정 모델(가상 공정)을 통해, 공정 입력의 변화에 따른 공정 출력을 예측하고, 이를 통해 최적의 공정 입력 시나리오(즉, 단시간 내에 제어오차를 최소화할 수 있는 공정 입력의 변화)를 도출함으로써, 공정을 제어한다. 제어기가 결정하는 공정 입력 값이, 모델을 통해 예측된 공정 출력에 의해 좌우되므로, 제어기의 성능은 공정 모델의 정확성이 크게 의존한다. 전통적으로, 이러한 공정 모델은 부공간 모델인식(subspace model identification) 기법 등을 이용하여, 공정 입력과 공정 출력의 상호관계를 잘 표현할 수 있는 식을 제작하였으나, 최근에는 인공지능 기법을 통해, 모델의 정확도, 즉, 모델 예측 제어의 성능을 개선하는 기법들이 제안되고 있다.

텍스트, 영수증이(가) 표시된 사진

자동 생성된 설명

그림 3. 모델 예측 제어기의 동작 원리

### **[문제]**

**매트랩을 이용해 인공신경망 모델 기반 예측 제어를 구축하고 이를 통해 흐름 w1(t)를 조정하여 제품농도를 유지하도록CSTR을 제어하라.**

**- 이때 입력농도는 Cb1=24.9, Cb2=0.1, CSTR 시스템의 동적 모델의 상수 k1과 k2는 각각 1이다.**

**- 또한, 계산의 복잡함을 피하기 위해 w2(t)는 0.1로 가정하고 탱크의 수위 h(t)는 제어되지 않는다고 한다.**

### **[방법] 인공신경망 모델 기반 예측 제어를 이용한 CSTR 제어**

#### 인공신경망 모델 기반 예측 제어의 첫 번째 단계는 ‘시스템 식별(system identification)’이다. 이 단계에 대해 설명하라.

1. 시스템 식별은 그림 4과 같이 플랜트의 순방향 동적 요소를 표현하도록 신경망을 훈련시키는 것을 말한다. 플랜트 출력값과 신경망 출력값 사이의 예측 오차는 신경망 훈련 신호로 사용된다.

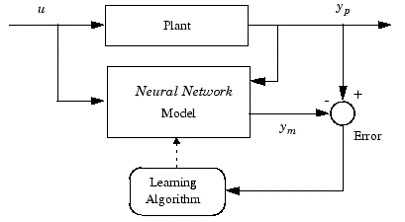


그림4. 인공신경망 훈련 개요

인공신경망 모델은 이전의 직전 입력값과 직전 플랜트 출력값을 사용하여 플랜트 출력값의 미래의 값을 예측한다. 인공신경망 모델의 구조는 그림5와 같이 표현된다.

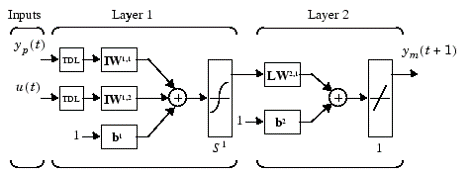


그림5. 인공신경망 모델 개요

#### 인공신경망 모델 기반 예측 제어의 두 번째 단계는 ‘예측 제어(predictive control)’이다. 이 단계에 대해 설명하라.

1. 모델 예측 제어 방법은 이동구간(receding horizon) 기법(SoHa96)을 기반으로 한다. 신경망 모델은 지정된 시간 지평에 대해 플랜트 응답을 예측한다. 예측값은 수치 최적화 프로그램이 지정된 지평에서 다음과 같은 성능 조건을 최소화하는 제어 신호를 파악할 때 사용된다.

텍스트이(가) 표시된 사진

자동 생성된 설명 식 (1)

수식 1. 수치 최적화 수식

여기서 N1, N2, Nu는 추적 오차와 제어 증분이 계산되는 지평을 정의한다. u’ 변수는 잠정적 제어 신호이고 yr은 원하는 응답, ym은 신경망 모델 응답이다. ρ는 제어 증분의 제곱합이 성능 지수에서 차지하는 비중을 결정한다.

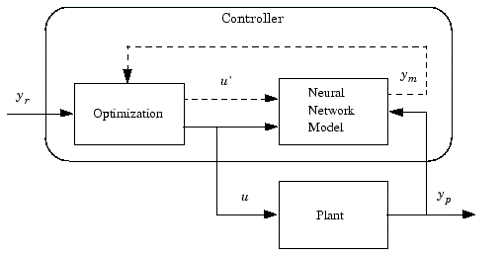


그림6. 모델 예측 제어 과정의 블록 흐름도

그림 6는 모델 예측 제어 과정을 나타낸 것이다. 제어기는 신경망 플랜트 모델과 최적화 블록으로 구성된다. 최적화 블록은 J를 최소화하는 u’의 값을 결정하며, 그런 다음 최적의 u가 플랜트에 입력된다.

#### CSTR(연속 교반 탱크 반응기; continuous stirred-tank reactor)를 도식화하고, 동적 모델을 나타내어라.

1. CSTR을 도식화하면 그림 7와 같이 나타낼 수 있다.

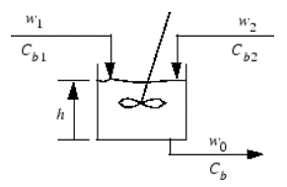


그림 7. 도식화한 CSTR

그림 7를 동적 모델로 나타내면 식 (2)과 식 (3)와 같이 나타낼 수 있다.

식 (2)

식 (3)

여기서 h(t)는 액체 수위이고, Cb(t)는 이 과정의 출력에서의 제품 농도이고, w1(t)는 농축된 공급 용액 Cb1의 유량이고, w2(t)는 희석된 공급 용액 Cb2의 유량이다.

### **[응용] 인공신경망 모델 기반 예측 제어를 이용한 CSTR 제어**

#### 매트립에서 ‘Simulink’ 기능을 활용해 공정 식별(plant identification)을 진행하라.

1. 공정 식별은 그림 8~10와 같은 절차를 통해 할 수 있다.

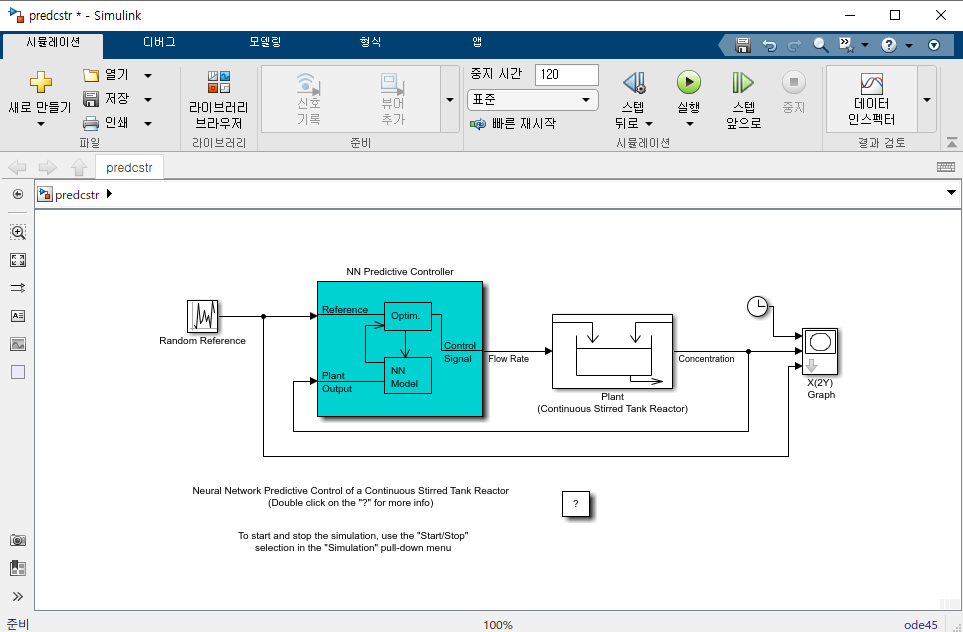


그림8. 실행된 Simulink 모델

매트랩을 실행하여 명령창에 ‘predcstr’을 입력한다. 입력한 명령을 실행하면 그림8과 같이 Simulink 편집기가 실행된다. Plant 블록에는 Simulink CSTR 플랜트 모델이 포함되어 있다. 파랗게 표시된 NN Predictive Controller 블록신호에서 ‘Control Signal은 Plant 모델의 입력에 연결되고, ‘Plant Output’은 Plant 블록 출력에 연결된다. 그리고 ‘Reference’는 Random Reference 신호에 연결된다.

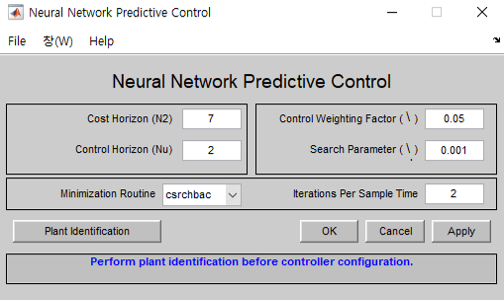


그림9. 모델 예측 제어기 설정창

NN Predictive Controller 블록을 더블 클릭하면 그림 9과 같이 모델 예측 제어기 설정창이 열린다. 이 창에서는 N2와 Nu를 변경할 수 있고, ρ도 정의할 수 있다. 이때, N1은 1로 고정되어 있다. α는 최적화를 위해 필요한 성능 감소분을 결정한다. 또한, 최적화 알고리즘이 사용할 선형 최소화 루틴(Minimization Routine)과 각 샘플 시간에 최적화 알고리즘을 몇 회 반복하여 수행할지 횟수(Iterations Per Sample Time)을 설정할 수 있다.

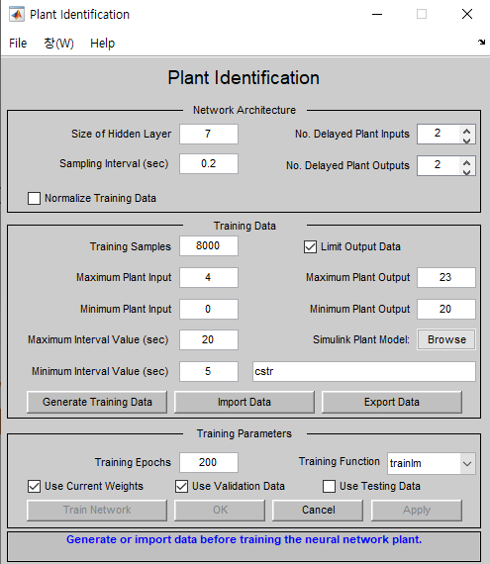


그림10. 인공신경망 플랜트 모델 설정창

플랜트 모델은 미래의 플랜트 출력값을 예측한다. 최적화 알고리즘은 이러한 예측값을 사용하여 미래의 성능을 최적화하는 제어 입력값을 결정한다. 플랜트 모델 신경망에는 앞에서 본 것처럼 하나의 은닉 계측이 있다. 이 창에서 은닉 계층의 크기, 지연 입력값과 지연 출력값의 개수, 훈련 함수를 선택한다.

#### 공정 식별 과정을 통해 생성한 인공 신경망 플랜트 모델을 훈련한 후 시뮬레이션을 실행하여 플랜트 출력값과 기준 신호를 비교하여라.

1. 인공 신경망 플랜트 모델의 훈련은 아래의 그림 11~13를 통해 이루어진다.

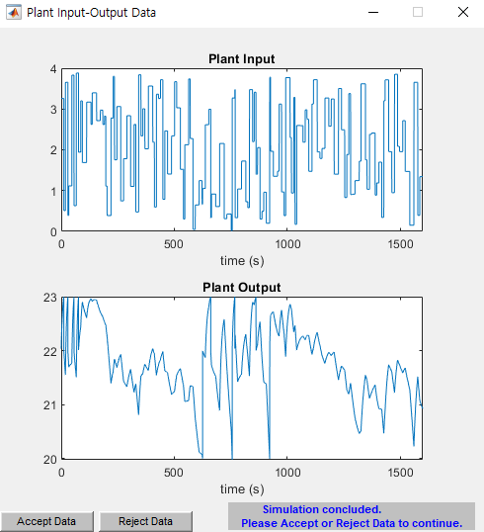


그림11. 인공신경망 플랜트 모델 훈련 결과창

그림10에서 ‘훈련데이터 생성(Generate Training Data)’을 클릭한다. 그림 11와 같이 Simulink 플랜트 모델에 일련의 무작위 스텝 입력값을 적용하여 훈련 데이터를 생성한다. 또한 이를 바탕으로 그림 11의 아래와 같이 잠정적 훈련 데이터가 표시된다. 이 데이터를 통해 미래의 공정 거동을 잘 예측한다고 판단되면 ‘Accept data’를 선택하고, 잘 예측하지 못한다고 판단되면 ‘Refuse Data’를 선택한 후 공정의 거동을 잘 예측할 수 있도록 조건을 변경한다.

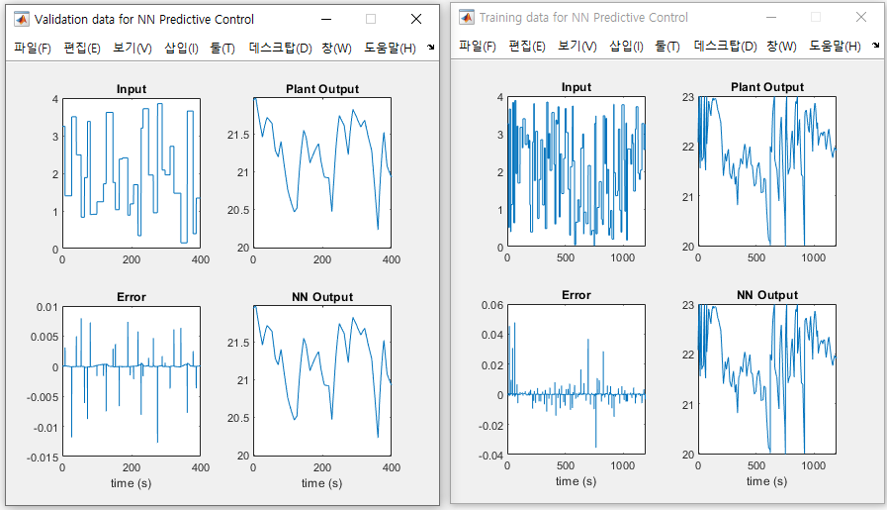


그림12. 플랜트 모델의 응답

그림 11에서 ‘Accept Data’를 클릭하고, 인공신경망 플랜트 모델 설정창(그림 10)에서 ‘Train Network’를 클릭한다. 클릭 시 플랜트 모델 훈련이 시작되고 훈련이 완료되면 그림 12과 같이 플랜트 모델의 응답이 표시된다. 이때, 검증 데이터와 테스트 데이터가 존재하는 경우 각각에 대한 별도의 플롯도 표시된다.

훈련이 완료된 후, 플랜트 모델 설정창(그림 10)에서 ‘Train Network’를 다시 선택하여 동일한 데이터 세트로 훈련을 계속할 수도 있고, ‘Erase Generated Data’를 선택하여 새로운 데이터 세트를 생성할 수도 있고, 현재 플랜트 모델을 수락하고 폐루프 시스템의 시뮬레이션을 시작할 수도 있다.

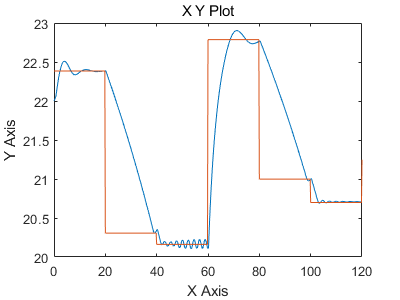


그림13. 플랜트 출력값과 기준 신호의 비교

플랜트 모델 설정창(그림 10)에서 ‘OK’를 선택하면 NN Predictive Controller 블록으로 훈련된 신경망 플랜트 모델을 불러오게 된다. 또한, 모델 예측 제어기 설정창(그림 9)에서 ‘OK’를 선택하면 NN Predictive Controller 블록으로 제어기 파라미터를 불러오게 된다. 이후 Simulink 편집기로 돌아가서 시뮬레이션을 실행하면 그림 13과 같이 플랜트 출력값과 기준 신호가 표시된다.

### **[결론]**

모델 기반 예측 제어는 고급 제어 기법 중 하나로, 공정의 거동을 잘 모사하는 모델을 생성해 이를 제어하는 방법을 통해 제어한다. 본 챕터에서는 인공신경망을 접목해 인공신경망 모델 기반 예측 제어를 통해 CSTR을 제어하였다. 같은 방법을 통해 보다 복잡한 공정으로 적용분야를 확대하여 공정 제어를 수행할 수 있다.

### **[학습 결과]**

* 학습 내용

모델 기반 예측 제어의 이해 및 인공신경망을 접목한 인공신경망 모델 기반 예측 제어 방법 익히기

* 학습 결과 확인하기

Simulink를 이용한 인공신경망 모델 기반 예측 제어 공정 모델 생성 및 설정하기

* 학습 결과 응용하기

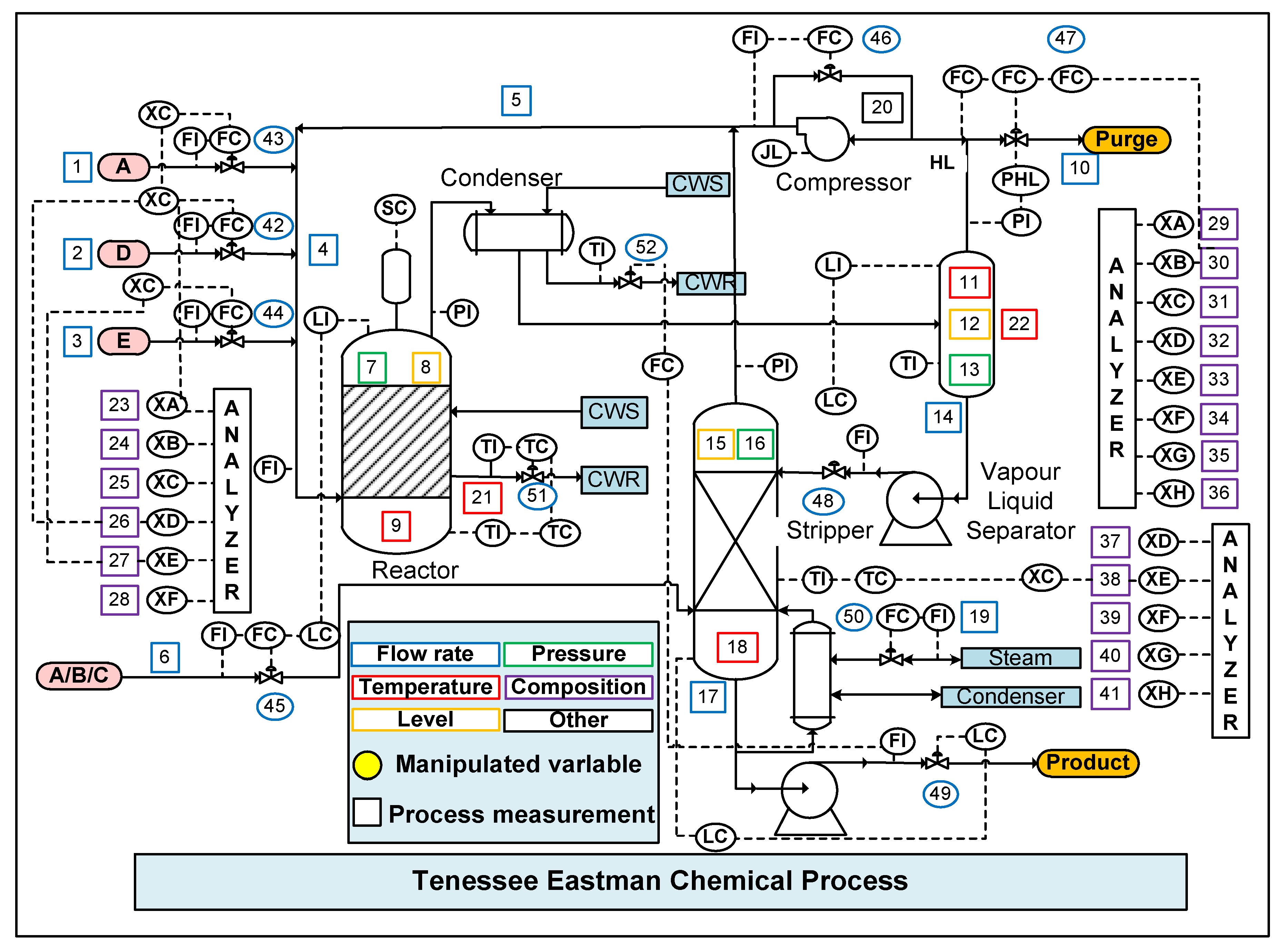
인공신경망 모델 기반 예측 제어를 통해 CSTR 제어 모델 만들기

* 1. **인공지능 기반 예지보전 및 안전**

### **화학 공정 이상 감지 및 진단**

|  |  |
| --- | --- |
| **학습 내용** | |
| [문제] | 화학 공정의 운전 상태(정상/이상) 이해 |
| [방법] | 오토인코더 모델 구축 |
| [응용] | 이상 감지 모니터링 및 기여도 분석 |
| [요약] |  |

### **[공정설명] 테네시 이스트만 공정**



테네시 이스트만 공정(Tennessee Eastman Process, TEP)은 공정 제어 및 모니터링 방법을 평가하기 위하여 Eastman Chemical Company가 제공한 가상 플랫폼으로써, 현실적인 공정을 잘 모사하는 이른바 디지털 트윈 (Digital twin)의 대표적인 예이다. 시험 공정은 성분, 반응 속도, 운전 조건 등이 수정된 실제 화학 공정의 시뮬레이션을 기반으로 한다. 본 공정은 반응기, 응축기, 압축기, 분리기, 스트리퍼 등 5가지 주요 장치로 구성되어 있고, A, B, C, D, E, F, G, H의 8가지 성분이 포함되어 있다.

기체 반응물인 A, C, D, E와 불활성 기체 B가 반응기에 투입되어 다음과 같은 반응을 통해 액체 생성물인 G와 H가 생성된다.

텍스트이(가) 표시된 사진

자동 생성된 설명

반응기 생성물은 응축기를 통해 냉각된 다음, 증기-액체 분리기로 투입된다. 분리기에서 나온 증기는 압축기를 통해 반응기 피드로 재순환된다. 분리기의 응축된 성분은 남은 반응물을 제거하기 위해 스트리퍼로 가고 스트리퍼 하단에서는 최종 생산물인 G와 H가 분리되어 나온다.

본 공정에는 41개의 측정 변수와 12개의 조작 변수가 포함되어 있다. 측정 변수는 3분마다 샘플링되고 XMEAS1부터 XMEAS22까지는 운전 변수이고 XMEAS23부터 XMEAS41까지는 성분 변수이다. 모든 측정 변수에는 가우시안 노이즈가 포함되어 있다.

TEP 시뮬레이션에는 21개의 사전 프로그래밍된 Fault가 있다. Fault 1-7은 공정 변수의 단계적 변화와 관련이 있고, Fault 8-12는 일부 공정 변수의 변동성 증가와 관련이 있다.

테이블이(가) 표시된 사진

자동 생성된 설명

### **[문제]**

**TEP 공정의 이상을 감지하고 이상 발생시 원인을 진단하여라.**

**- “./코드 및 데이터/8-1 TE process”의 데이터를 활용하여라.**

**- “data0.csv”을 학습 데이터로 “data0\_test”을 평가 데이터로 사용하여라.**

### **[방법] TEP 공정의 오토인코더 모델 구축**

#### 오토인코더를 학습시키기 위한 정상 운전 데이터를 작업 환경으로 입력하시오.

|  |
| --- |
| import pandas as pd  data\_normal = pd.read\_csv(“./8-1 TE process/data0.csv”, index\_col=0)  data\_test = pd.read\_csv(“./8-1 TE process/data0\_test.csv”, index\_col=0) |

pd.read\_csv 함수는 해당경로의 데이터를 불러와 데이터프레임 형태로 저장한다. 여기서 “index\_col=0”은 첫 열을 인덱스로 지정한다는 의미이다. data\_normal은 훈련 데이터이고 data\_test은 평가 데이터이다.

#### 입력 데이터의 구조와 정보를 분석하시오.

|  |
| --- |
| data\_normal.info() |

테이블이(가) 표시된 사진

자동 생성된 설명

info()는 데이터에 대한 간단한 설명을 보여준다. XMEAS1부터 XMEAS41까지 41개의 변수에 대해 960개의 데이터 샘플이 존재하고 결측치는 존재하지 않음을 확인할 수 있다.

|  |
| --- |
| data\_normal.describe() |



describe()는 각 변수의 샘플 개수, 평균값, 표준편차, 최솟값, 최댓값 등을 보여준다. 공정이 정상 상태일 때 각 공정 변수의 운전 조건을 알 수 있다.

#### 오토인코더 모델의 학습과 검증을 위해 데이터를 분리하시오.

|  |
| --- |
| from sklearn.model\_selection import train\_test\_split  test\_ratio = 0.2  random\_state = 42  data\_train, data\_val = train\_test\_split(data\_normal, test\_size = test\_ratio,  random\_state=random\_state)  print(data\_train.shape, data\_val.shape, data\_test.shape) |



모델의 훈련과 검증을 위해 train\_test\_split 함수를 이용하여 훈련 데이터를 분리한다. 훈련 데이터의 20%를 검증 데이터로 사용하고 평가 데이터는 이전에 불러온 data\_test를 사용한다. shape 속성을 이용하여 각 데이터셋의 크기를 출력해보았다.

#### 입력 데이터의 값을 표준화하시오.

|  |
| --- |
| from sklearn.preprocessing import StandardScaler  scaler = StandardScaler()  X\_train\_scaled = scaler.fit\_transform(data\_train)  X\_val\_scaled = scaler.transform(data\_val)  X\_test\_scaled = scaler.transform(data\_test) |

데이터 스케일링을 통해 각 공정 변수 사이의 범위를 일정한 수준으로 맞춰주어 모델이 편향성을 갖는 것을 방지한다. 여기서는 표준화를 위한 StandardScaler가 사용되었고, 모델 훈련 시 평가 및 검증 데이터의 정보가 반영되지 않도록 훈련 데이터에 맞추어 스케일링을 진행한다.

#### 공정 변수 차원 축소를 위한 오토인코더 알고리즘을 구축하시오.

|  |
| --- |
| from tensorflow import keras  from tensorflow.keras.layers import \*  from tensorflow.kears.optimizers import Adam  def Autoencoder(input\_dim, num\_nodes, num\_latents, learning\_rate):  model = keras.Sequential()  model.add(Dense(num\_nodes, input\_shape=(input\_dim, )))  model.add(BatchNormalization())  model.add(Activation(‘relu’))  model.add(Dense(num\_latents))  model.add(BatchNormalization())  model.add(Activation(‘relu’))  model.add(Dense(num\_nodes))  model.add(BatchNormalization())  model.add(Activation(‘relu’))  model.add(Dense(input\_dim, activation=’linear’))  optimizer = Adam(learning\_rate=learning\_rate, decay=0.001)  model.compile(loss='mse', metrics=’mae’, optimizer = optimizer)  return model |

오토인코더 함수는 입력 차원, 은닉뉴런 개수, 잠재공간 차원, 학습률을 하이퍼파라미터로 갖는다. 하이퍼파라미터는 문제에 따라 적절한 개수를 설정해주어야 한다. 이후에는 compile메소드를 이용하여 모델의 학습 환경을 설정한다. 최적화 알고리즘(optimizer)은 adam을, 손실 함수(loss)는 mean squared error를, 평가지표(metrics)로는 mean absolute error을 선택하였다.

다음으로 Autoencoder 함수를 이용하여 모델을 만든다.

|  |
| --- |
| input\_dim = X\_train\_scaled.shape[1]  num\_nodes = 64  num\_latents = 28  learning\_rate = 0.001  ae\_model = Autoencoder(input\_dim, num\_nodes, num\_latents, learning\_rate)  ae\_model.summary() |

테이블이(가) 표시된 사진

자동 생성된 설명

Autoencoder 함수에 입력할 인자로 입력 차원은 학습 데이터의 크기로부터 얻고, 은닉 뉴런과 잠재 공간 크기는 각각 64개, 28개로 설정하였다. 본 문제에서는 오토인코더를 차원 축소 용도로 사용하기 때문에 잠재 공간 크기는 입력 차원 수보다 작아야 한다. summary는 설정된 모델의 구조를 요약하여 출력해준다.

#### 오토인코더 모델을 정상 데이터를 이용하여 학습시켜라.

|  |
| --- |
| from kears.callbacks import EarlyStopping, ModelCheckpoint  training\_epoch = 1000  patience = 20  earlystopping = EarlyStopping(patience=patience, monitor='val\_loss')  checkpointer = ModelCheckpoint(filepath=’best model.h5’, save\_best\_only=True)  history = model.fit(X\_train\_scaled, X\_train\_scaled,  epochs=training\_epoch,  batch\_size=64,  callbacks=[earlystopping, checkpointer],  validation\_data=(X\_val\_scaled, X\_val\_scaled) |

테이블이(가) 표시된 사진

자동 생성된 설명

모델 훈련 시에는 fit 함수를 이용하고, 훈련 반복 횟수(epochs), 훈련 배치 크기(batch\_size), 검증 데이터(validation\_data)를 설정하였다.

또한, callback 함수를 통해 다양한 훈련 조건을 설정할 수 있다. 조기 종료(EarlyStopping)을 통해 정해진 횟수(patience)동안 검증 데이터에 대한 손실 값이 줄어들지 않으면 훈련을 종료한다. 학습된 모델을 저장하기 위해서 ModelCheckpoint을 사용하고 filepath로 모델을 저장할 파일 이름을 입력한다. save\_best\_only 옵션을 사용하면 검증 데이터 손실이 최고 성능 모델보다 낮아질 때만 저장한다.

#### 학습된 모델을 불러오고 평가 데이터에 대한 모델의 정확도를 평가하여라. 각 변수에 대한 parity plot을 이용하여 분석하시오.

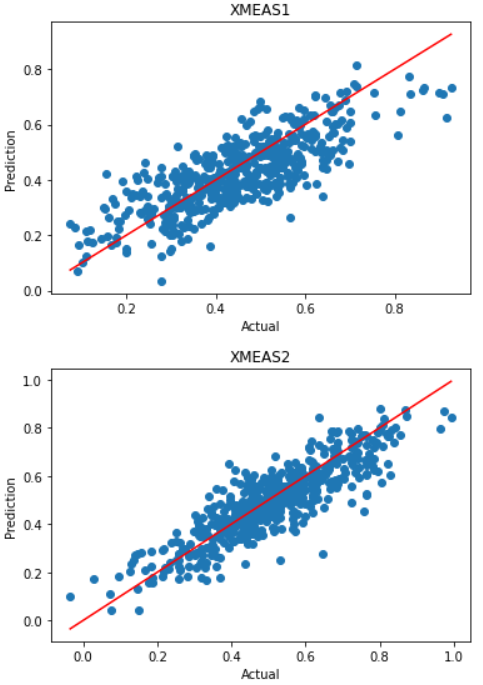
|  |
| --- |
| from keras.models import load\_model  from sklearn.metrics import mean\_squred\_error, mean\_absolute\_error  model = load\_model(‘best model.h5’)  X\_pred = model.predict(X\_test\_scaled)  mse = mean\_squared\_error(X\_test\_scaled, X\_pred)  mae = mean\_absolute\_error(X\_test\_scaled, X\_pred)  print(‘Mean squared error:’, mse)  print(‘Mean absolute error:’, mae) |

텍스트이(가) 표시된 사진

자동 생성된 설명

훈련이 끝나고 저장된 최고 성능 모델을 불러온다. 평가 데이터를 모델에 입력하여 예측값을 얻는다. 손실 값과 평가지표로 사용되었던 mean squared error와 mean absolute error를 계산하여 모델의 정확도를 평가할 수 있다. 0에 가까울수록 모델이 입력값을 잘 재구성(reconstruction)하고 있음을 뜻한다.

|  |
| --- |
| import matplotlib.pyplot as plt  for i in range(input\_dim):  plt.scatter(X\_test\_scaled[:,i], X\_pred[:,i])  plt.plot([min(X\_test\_scaled[:,i], max(X\_test\_scaled[:,i])], [min(X\_test\_scaled[:,i], max(X\_test\_scaled[:,i])], color=’r’)  plt.xlabel(‘Actual’)  plt.ylabel(‘Prediction’)  plt.title(data\_normal.columns[i])  plt.show() |



parity plot은 실제 데이터와 예측 데이터를 비교하는 산점도이고, 실제 값과 예측 값이 같으면 점이 y=x 위에 놓이게 된다. 입력 변수 41개에 대한 parity plot을 그려보면 모델의 변수 별 재구성 성능을 확인할 수 있다.

### **[응용] 오토인코더 기반의 공정 이상 감지 및 진단**

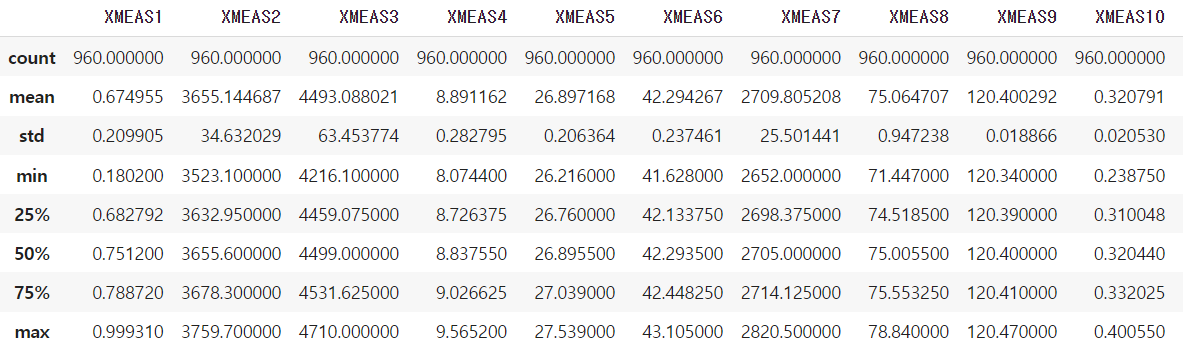
#### 공정의 상태를 판별하는 지표인 Squared prediction error(SPE)와 SPE의 control limit을 계산하는 함수를 생성하시오.

|  |
| --- |
| import numpy as np  from statsmodels.nonparametric.kde import KDEUnivariate  def SPE(reconstruction\_error):  spe = np.sum(reconstruction\_error\*\*2, axis=1)  return spe  def SPE\_CL(reconstruction\_error, alpha=0.95):  spe = np.sum(reconstruction\_error\*\*2, axis=1)  kde\_spe = KDEUnivariate(spe)  kde\_spe.fit()  spe\_cl = kde\_spe .icdf[int(alpha\*len(kde\_spe.icdf))]  return spe\_cl |

SPE는 입력 데이터에 대한 모델의 재구성 오차(reconstruction error)의 제곱의 합으로 계산된다. 통계적 한계선(control limit)은 커널 밀도 추정(kernel density estimation)을 통해 얻는다. 정상 데이터의 SPE 값에 대해 추정된 확률밀도함수의 역누적분포함수(inverse cumulative distribution function)를 구하고, 신뢰도 alpha값의 확률에 해당하는 SPE값을 계산하면 그 값이 한계선이다.

#### Q8에서 정의한 함수를 이용하여 Fault 1 데이터에 대한 SPE와 control limit을 계산하여라. 계산된 SPE와 control limit을 이용하여 모니터링 차트를 그려보아라.

|  |
| --- |
| data\_fault1 = pd.read\_csv(‘./Data/data1.csv’, index\_col=0)  data\_fault1.describe() |

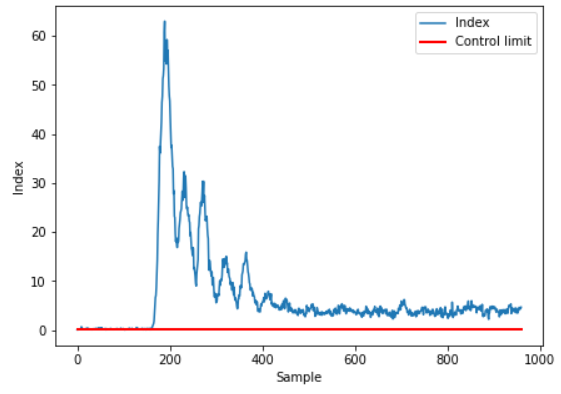


공정 모니터링 성능을 확인하기 위해 Fault 1 데이터를 불러온다. Fault 1 데이터는 960개의 샘플로 이루어져 있고 160번째 샘플부터 A/C feed ratio의 step change가 발생한다.

|  |
| --- |
| X\_fault1\_scaled = scaler.transform(data\_fault1)  X\_fault1\_pred = model.predict(X\_fault1\_scaled)  re\_fault1 = X\_fault1\_pred – X\_fault1\_scaled  spe = SPE(re\_fault1)  X\_scaled = scaler.transform(data\_normal)  X\_pred = model.predict(X\_scaled)  re\_normal = X\_pred – X\_scaled  spe\_cl = SPE\_CL(re\_normal, alpha=0.95) |

이전에 정상 데이터를 표준화했던 scaler에 적용하여 표준화를 진행한 후 훈련된 모델에 입력하였다. 모델의 예측 값과 입력 값 사이의 재구성 오차를 계산하고 이전에 정의한 SPE 함수에 입력하여 SPE 값을 계산한다. 같은 과정을 정상 데이터에 대해 반복하여 SPE\_CL 함수에 입력하면 한계선을 구할 수 있다. 신뢰도 alpha는 기본 값인 95%로 설정하였다.

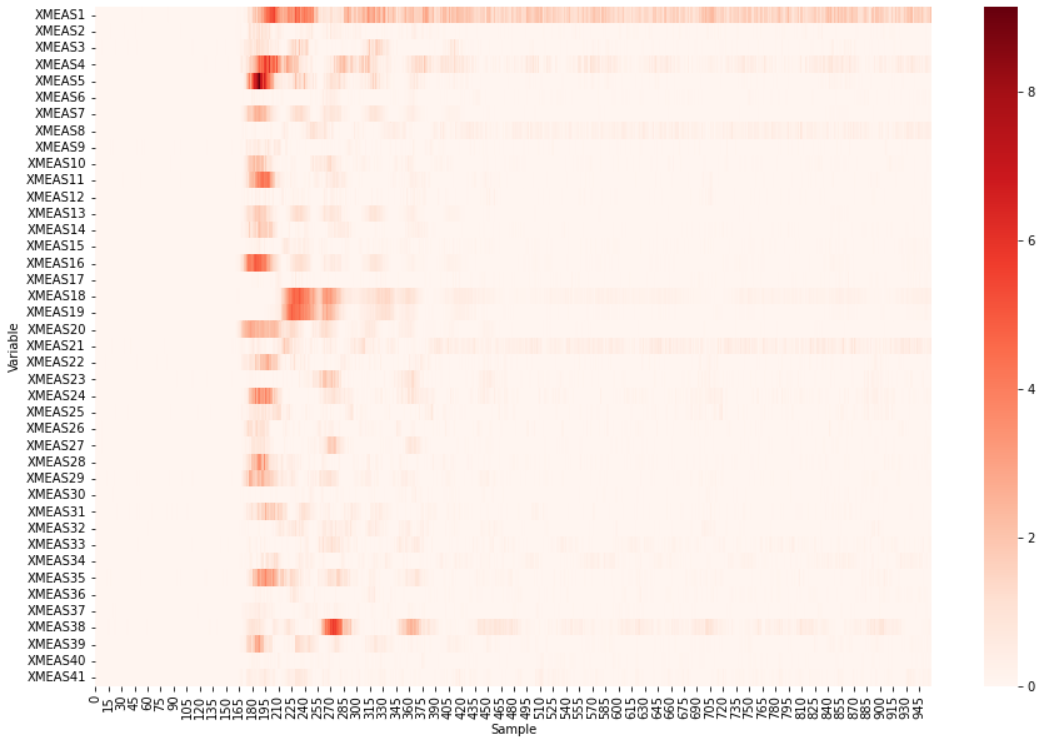
|  |
| --- |
| plt.figure(figsize=(7,5))  plt.plot(spe, label=’Index’)  control\_limit = np.full((len(spe), 1), spe\_cl)  plt.plot(np.arange(len(spe)), control\_limit, color=’r’, linewidth=2, label=’Control limit’)  plt.xlabel(‘Sample)  plt.ylabel(‘Index’)  plt.title(‘Fault1’)  plt.show() |



Sample별로 SPE 값과 한계선을 플롯해보면 SPE 값이 약 160번째 샘플 이후로 급격히 상승하여 한계선을 넘어서서 이상이 감지되는 것을 확인할 수 있다.

#### Fault 1 데이터에 대한 시간에 따른 기여도 변화를 나타내는 Contribution map을 이용하여 분석하시오.

|  |
| --- |
| import seaborn as sns  fig, ax = plt.subplots(figsize=(15,10))  sns.heatmap((re\_fault1\*\*2).T, cmap=’Reds’)  ax.set\_yticklabels(data\_fault1.columns, rotation=0)  plt.show() |



기여도는 재구성 오차의 제곱으로 계산된다. re\_fault1을 제곱한 행렬을 전치하여 행을 변수, 열을 샘플로 하는 히트맵(heatmap)을 생성하였다. 마찬가지로 160번째 샘플부터 진한 빨간색이 나타나기 시작하고 제일 윗줄인 XMEAS1(A feed rate) 변수가 전반적으로 큰 기여도를 가지고 있음을 확인할 수 있다.

### **[결론]**

본 장에서는 오토인코더 모델을 이용하여 화학 공정의 이상 상태를 감지하고 기여도를 분석하였다. 비지도 학습을 이용함으로써 정상 운전 데이터만을 사용하여 모델을 훈련시켰고, 평가 데이터로 모델을 검증하여 신뢰성을 확보하였다. 모델의 재구성 오차를 활용하여 모니터링 지표와 신뢰 구간을 정의하고 이상 데이터에 대한 모니터링 차트를 그려보았다. 또한 이상 발생 시 변수 별 기여도를 시간에 따라 나타내어 어떤 변수들에 이상이 생겼는지 확인할 수 있다.

### **[학습 결과]**

* 학습 내용

화학 공정의 운전 상태 분석 방법론 익히기

* 학습 결과 확인하기

오토인코더 모델 훈련 방법 및 공정 상태 모니터링 방식 익히기

* 학습 결과 응용하기

본 장의 학습 내용에 기반해 실시간 모니터링 모델 만들기