1. 화공산업에서의 인공지능

# **인공지능 기반 공정 운전 및 최적화**

**3.1. 인공지능 기반 공정 운전 및 최적화**

### **수증기 개질 수소 생산 공정 운전조건 최적화**

|  |  |
| --- | --- |
| **학습 내용** | |
| [문제] | 수소 생산을 위한 수증기 개질 공정 이해 |
| [방법] | 인공신경망 모델 구축 및 최적화 |
| [응용] | 인공신경망 기반 수증기 개질 공정 최적 변수 탐색 |
| [요약] |  |

### **[공정설명] 수증기 개질 공정의 이해**

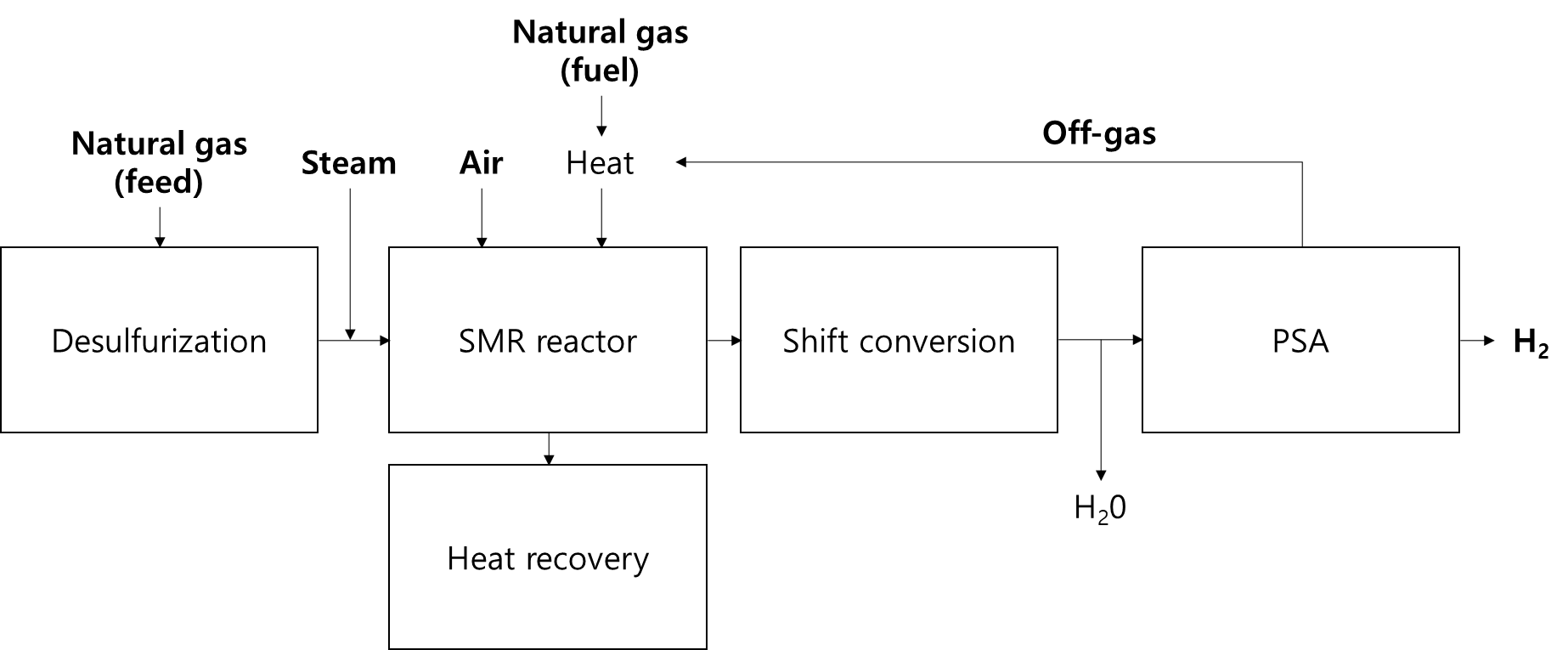


그림 1. 수증기 개질 공정의 흐름도

수증기 개질(steam methane reforming, SMR) 공정은 크게 주요 반응이 이루어지는 SMR 반응기, 황성분을 제거하는 탈황장치(hydrodesulfurization, HDS), 잔존 CO를 H2로 전환하는 전환반응기(shift reactor), 생성된 합성가스로부터 고순도의 수소를 분리하는 압력변동흡착(pressure swing adsorption, PSA) 장치, 열회수를 위한 열교환기(heat exchanger)로 구성되어 있다 (그림 1).

원료인 천연가스는 한국 전역에 설치되어 있는 천연가스 배관망을 통하여 공급된다. 천연가스는 C1~C5의 혼합물로 구성되어 있으며, 주성분은 CH4로 약 93%를 차지한다. 천연가스는 부취제로서 일부 황을 포함하고 있는데, 황 성분은 개질 반응기의 촉매를 피독하므로, 원료 투입 전에 제거되어야 한다. 원료인 천연가스는 약 300℃로 예열되어 CoMo와 ZnO 촉매로 충전된 탈황장치로 투입되고 황 성분이 제거된다. 또한, 각종 불순물이 제거된 물은 열교환기를 통하여 180℃ 이상으로 승온되어 과열 증기상태로 공급된다. 결과적으로 황 성분이 제거된 천연가스와 과열 스팀이 섞인 후 약 580℃의 조건으로 승온되어 SMR 반응기로 투입된다. SMR 반응기에서 아래의 세가지 주요 반응이 진행되며 수소가 생성된다. 전체적인 SMR 반응은 강한 흡열반응이며 약 700~800℃, 그리고 8 barg 조건에서 반응이 일어난다.

|  |  |
| --- | --- |
|  | (1) |
|  | (2) |
|  | (3) |

반응 종료된 합성가스는 CO, CO2, CH4, H2, H2O로 이루어지며 약 750℃ 조건으로 SMR 반응기를 빠져나와 후단 공정에 열을 전달한다. 이 합성가스는 180–250℃ 조건으로 저온수성가스화(low temperature shift, LTS) 반응기로 투입된다. LTS 반응기에서는 합성가스에 잔존하고 있는 CO를 제거하는 반응이 일어난다. CO는 수소 연료전지에 악영향을 미치기 때문에 최대한 제거하는 것이 좋다. 또한 LTS 반응기에서 CO가 제거되며 추가적인 H2를 생성하는 장점이 있다. LTS 반응을 거치고 나온 흐름은 40℃까지 온도가 낮춰지고, 물이 제거된 합성가스를 순도 99.999%, CO농도 0.2ppm 이하의 스펙을 갖는 수소로 분리하기 위해서 압력변동흡착 장치에 투입된다.

### **[문제] 투입되는 천연가스, 스팀, 공기의 유량과 압력변동흡착 장치의 회수율에 따라서 최종적으로 생성되는 수소의 유량을 계산하고 SMR 공정의 최적운전조건을 구하라.**

- “2. 코드 및 데이터/7-1. SMR process.csv” 데이터를 활용하여라.

- 학습 및 평가 데이터는 7:3 비율로 분리하여라.

- 민감도 분석은 평균값의 5% 범위 내에서 진행하여라.

### **[방법] SMR 공정의 인공신경망 모델 구축**

#### 데이터를 작업 환경으로 불러오자. 데이터의 불러오기와 저장하기 등 데이터 작업에 유용한 파이썬 라이브러리는 무엇인가?

1. 일반적으로 데이터는 Excel과 같은 관계형 데이터베이스에 많이 들어있다. 데이터는 pandas 라이브러리를 이용하여 다음 명령어를 통해 읽어 들인다.

|  |
| --- |
| import pandas as pd  data\_smr = pd.read\_csv(“./smr\_process.csv”, header=0) |

pd.read\_csv 함수는 해당경로의 데이터를 불러와 데이터프레임 형태로 저장한다. 여기서 “header=0”은 첫 행을 열 이름으로 지정한다는 의미이다.

#### 데이터의 기본 정보를 파악하여라. 데이터 샘플 개수, 특성 개수, 각 특성의 타입, 최솟값, 최댓값, 평균값은 어떻게 되는가?

1. head() 사용하여 불러온 데이터의 정보를 확인해보자.

|  |
| --- |
| data\_smr.head() |

테이블이(가) 표시된 사진

자동 생성된 설명

각 행은 하나의 데이터 샘플을 나타낸다. head() 괄호 안에 숫자를 입력하면 처음부터 그 숫자만큼의 행을 보여주고, 기본 값은 5개로 설정 되어있다.

info()는 데이터에 대한 간단한 설명을 나타낸다. 데이터 샘플의 수, 각 특성의 데이터 타입, 결측치 개수 등의 정보를 파악할 수 있다.

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

텍스트, 영수증이(가) 표시된 사진

자동 생성된 설명

describe()는 숫자형 데이터의 통계 정보를 보여준다.

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

테이블이(가) 표시된 사진

자동 생성된 설명

count는 결측치를 제외한 데이터 샘플의 개수, mean, min, max는 각 데이터 특성의 평균값, 최솟값, 최댓값을 나타낸다. std 행은 값이 퍼져 있는 정도를 측정하는 표준편차이다. 25%, 50%, 75% 행은 백분위수를 나타낸다.

#### 신경망 모델의 학습과 평가를 위해 데이터를 분리하여라. 학습 데이터와 평가 데이터의 차이점은 무엇인가?

1. 모델 훈련과 평가를 진행하기 위해서 사이킷런의 ‘train\_test\_split’ 함수를 이용하여 훈련 데이터와 평가 데이터를 분리한다. 학습 데이터는 모델 학습을 할 때 사용하는 데이터이며, 평가 데이터는 학습된 모델의 성능을 평가하는 데이터이다. 평가 데이터는 모델의 일반화 오류 정도를 평가하기 위해 마지막에 한번만 사용되며, 이를 통해 모델의 과적합을 방지할 수 있다.

|  |
| --- |
| from sklearn.model\_selection import train\_test\_split  test\_ratio = 0.3  random\_state = 42  data\_train, data\_test = train\_test\_split(data\_modeling, test\_size = test\_ratio,  shuffle=True, random\_state=random\_state) |

test\_ratio 값은 데이터 샘플들로부터 테스트 데이터셋을 얼마만큼 분리할 지를 결정한다. 여기서는 전체의 30% 만큼 테스트 데이터셋으로 분리하였다. random\_state는 랜덤으로 분리되는 데이터를 다음 시행에서도 동일하게 분리하기 위해(동일한 실험 결과를 얻기 위해서) 고정하는 값으로, 42로 설정되었다.

#### 모델의 입력변수와 출력변수를 정의하고, 스케일러를 이용하여 데이터의 값을 조정하여라. 훈련데이터와 평가데이터의 스케일링 시 주의사항은 무엇인가?

1. 모델이 데이터의 정보를 효율적으로 파악하고 학습할 수 있도록 모든 공정 변수의 값을 표준화하는 데이터 스케일링을 진행한다. 데이터 스케일링을 진행하기에 앞서, 스케일러 용이성을 위하여 모델의 입력변수와 출력변수를 구분하여 스케일링을 진행한다.

|  |
| --- |
| from sklearn.preprocessing import StandardScaler  var\_x = ["NG feed”, "NG fuel”, "Water”, "Air”, "PSA recovery”]  var\_y = ["H2”]  scaler\_x = StandardScaler()  scaler\_y = StandardScaler()  train\_x = scaler\_x.fit\_transform(data\_train[var\_x])  train\_y = scaler\_y.fit\_transform(data\_train[var\_y])  test\_x = scaler\_x.transform(data\_test[var\_x])  test\_y = scaler\_y.transform(data\_test[var\_y]) |

모델의 입력변수는 시스템에 주입되는 천연가스의 원료와 연료, 공기와 물의 유량, 압력변동흡착 장치에서의 회수율로 설정되었고, 출력변수는 시스템에서 빠져나가는 합성가스의 유량과 그 조성, 그리고 각 장치의 온도로 설정되었다. 스케일러는 표준화를 위한 StandardScaler가 사용되었으며, 모델 훈련 시 테스트 데이터셋의 정보가 반영되지 않도록 훈련 데이터셋에 맞추어 스케일링을 진행한다.

#### 공정 모델링을 위한 인공신경망을 구축하여라. 인공신경망 구성 시 필요한 요소들은 무엇인가?

1. 정제된 데이터를 이용하여 SMR 공정을 모델링하기 위해서는 데이터 학습을 위한 인공신경망 구축이 필요하다. 인공신경망 함수는 다음과 같이 정의된다.

|  |
| --- |
| from tensorflow import keras  from tensorflow.keras.layers import \*  def NeuralNet(num\_x, num\_y, num\_layers, num\_neurons, learning\_rate):  model = keras.Sequential()  model.add(Dense(num\_neurons,  activation="relu",  input\_shape=[num\_x]))  for n in range(num\_layers-1):  model.add(Dense(num\_neurons,  activation="relu"))  model.add(Dense(num\_y))  optimizer = keras.optimizers.Adam(learning\_rate = learning\_rate,  beta\_1=0.9, beta\_2=0.999)  model.compile(loss='mse', optimizer = optimizer)  return model |

인공신경망 함수는 입력변수 개수, 출력변수 개수, 은닉층 개수, 은닉뉴런 개수, 학습률을 인자로 갖는다. 은닉층과 은닉뉴런의 개수는 증가할수록 모델 파라미터의 수가 증가하여 더 섬세한 모델링이 가능하지만 오버피팅(overfitting) 문제가 발생할 수 있다. 따라서 문제에 따라 적절한 개수로 설정해주어야 하며, 일반적으로 하이퍼파라미터 최적화 과정을 통해 결정된다.

다음으로 인공신경망 함수를 이용하여 모델을 만든다.

|  |
| --- |
| num\_x = len(var\_x)  num\_y = len(var\_y)  num\_layers = 3  num\_neurons = 20  learning\_rate = 0.001  nn\_model = NeuralNet(num\_x, num\_y, num\_layers, num\_neurons, learning\_rate) |

신경망의 구조는 3개의 은닉층, 각 층마다 20개의 뉴런, 0.001의 학습률로 설정하였다. 그리고 학습 종료조건으로 조기종료(early stopping) 기법을 사용하였다. 이제 모델을 훈련해보자. 모델 훈련 시에는 다양한 훈련조건이 설정된다.

|  |
| --- |
| training\_epoch = 10000  patience = 30  early\_stopping\_cb = keras.callbacks.EarlyStopping(patience=30,  restore\_best\_weights= True,  monitor='val\_loss')  # 모델 저장 경로  import os  def CreateFolder(directory):  try:  if not os.path.exists(directory):  os.makedirs(directory)  except OSError:  print ('Error: Creating directory. ' + directory)  path\_save = f'{Path\_project}/Model'  model\_name = 'nn\_model'  CreateFolder(path\_save)  # 모델 학습  if not os.path.exists(f"{path\_save}/{model\_name}"):  history = nn\_model.fit(train\_x, train\_y,  epochs = training\_epoch, callbacks=[early\_stopping\_cb],  verbose = 2,  validation\_data=(test\_x, test\_y))  # 모델 저장  nn\_model.save(f'{path\_save}/{model\_name}')  # 모델 불러오기  else:  nn\_model = keras.models.load\_model(f"{path\_save}/{model\_name}") |

텍스트이(가) 표시된 사진

자동 생성된 설명

조기종료는 정해진 횟수(tolerance)동안 검증 데이터에 대한 손실 함수값이 더 이상 줄어들지 않으면 학습을 종료한다. 따라서 정해진 훈련 에포크(training epoch)만큼의 반복이 끝나기 전에 훈련이 조기 종료될 수 있으며, 이 조건은 모델이 훈련데이터에 과적합(overfitting)되는 문제를 방지할 수 있다.

#### 학습된 모델의 정확도를 평가하여라. 평가지표 R2와 RMSE의 차이점에 대해서 설명하여라.

1. 훈련이 끝난 모델은 R2와 RMSE 등의 평가지표를 통해 성능을 평가한다. R2과 RMSE 함수는 다음과 같이 설정된다.

|  |
| --- |
| import numpy as np  def R\_Squared(prediction, actual\_value):  actual\_mean = np.mean(actual\_value, axis=0)  SSR = np.sum( (prediction - actual\_value)\*\*2 , axis=0)  RSS = np.sum( (prediction - actual\_mean )\*\*2, axis=0 )  TSS = np.sum( (actual\_value - actual\_mean )\*\*2, axis=0 )  r2 = 1 - SSR/TSS  return r2  def RMSE(prediction, actual\_value):  rmse = np.sqrt(np.mean((prediction - actual\_value)\*\*2, axis=0))  return rmse |

예측(prediction)은 훈련된 모델에 입력 변수를 입력하였을 때 출력되는 예측값을 나타내고, 실제값(actual value)은 입력 변수 값에 해당하는 실제 출력 데이터값을 나타낸다. R2 지표는 0과 1사이의 값을 가지며, 1에 가까울수록 예측과 실제가 유사하다는 것을 의미한다. 반면에 RMSE 지표는 예측과 실제의 오차를 나타내는 값으로, 0에 가까울수록 두 값이 일치한다는 것을 의미한다.

|  |
| --- |
| prediction = pd.DataFrame(scaler\_y.inverse\_transform(nn\_model.predict(test\_x)),  index=data\_test.index,  columns=var\_y)  actual\_value = data\_test[var\_y]  r2 = R\_Squared(prediction, actual\_value)  rmse = RMSE(prediction, actual\_value)  display(pd.DataFrame([r2, rmse], index=["R2", "RMSE"])) |

텍스트이(가) 표시된 사진

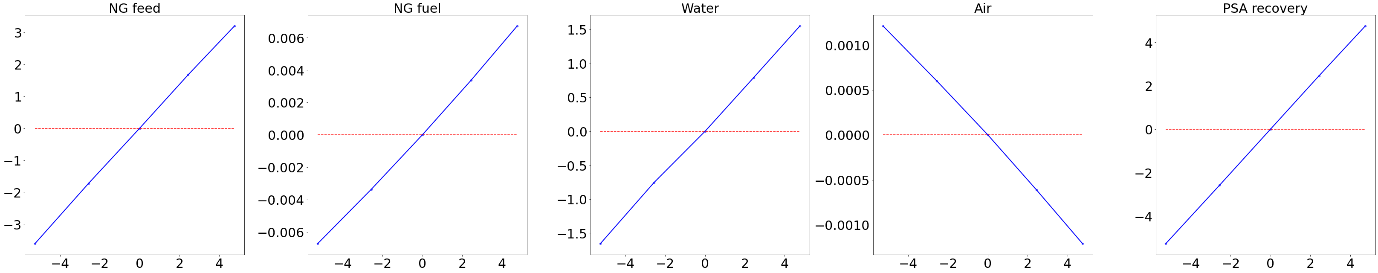
자동 생성된 설명

이 결과는 SMR 공정의 수소생산량에 대한 모델의 예측 성능을 나타낸다.

#### 민감도 분석을 이용하여 공정 모델에 대한 각 입력변수의 영향도를 분석하여라.

1. 민감도 분석은 각 입력변수를 평균값의 -5%~+5% 범위에서 변화시키면서 공정의 반응을 관찰한다. 이 분석을 통해 각 변수들의 영향도를 알 수 있다.

|  |
| --- |
| x\_base = data\_smr[var\_x].mean()  sensitivity\_points = 5  sensitivity\_result = SensitivityAnalysis(x\_base, sensitivity\_points,  nn\_model, var\_x, var\_y, scaler\_x, scaler\_y) |

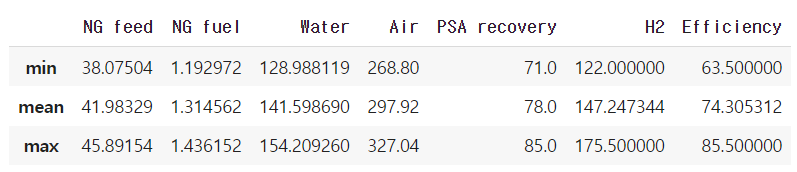
수소생산량에 대해 3 개의 변수(천연가스 원료, 물, 압력변동흡착 장치의 회수율)가 유의미한 영향도를 나타내고, 2 개의 변수(천연가스 연료, 공기)가 미미한 영향도를 나타냈다. 또한, 압력변동흡착 장치의 회수율, 천연가스 원료, 물, 천연가스 연료, 공기의 순서대로 수소 생산량에 큰 영향력을 갖는다.

### **[응용] 인공신경망 기반 최적 공정변수 탐색**

#### 학습된 모델을 기반으로 격자탐색법을 이용하여 SMR 공정의 운전 조건을 최적화하여라. 최적화하고자 하는 목적함수에 따라서 최적 운전 조건이 어떻게 달라지는가?

1. 앞에서 구축된 인공신경망 모델의 빠른 계산속도를 활용하여 최적의 운전조건을 탐색한다. 운전조건은 수집된 데이터에서 각 변수의 최솟값과 최댓값 사이에서 탐색하며, 이 범위를 ‘탐색 공간’이라고 정의한다.

|  |
| --- |
| data = data\_smr  data.describe().loc[['min', 'mean', 'max']] |



최적 조건을 찾기 위한 알고리즘으로 격자탐색법(grid search)을 사용하였다. 격자탐색법은 탐색공간 내에서 격자점을 생성하여 모든 격자점에서의 목적함수 값을 확인하며 최적 해를 탐색하는 방법으로, 간단하고 빠르지만 탐색공간이 커지거나 결정 변수의 개수가 많아질수록 계산비용이 크게 증가하는 특징이 있다.

|  |
| --- |
| bins = 11  gridsearch\_result = GridSearch(data, bins, nn\_model, var\_x, var\_y, scaler\_x, scaler\_y) |



5개 입력변수마다 11개의 격자점을 조합하여 총 161,051개의 운전조건을 생성하여 최적의 조건을 탐색하였다. 생성된 운전조건들에 대해서 수소 생산량을 계산하고, 내림차순으로 정렬함으로써 수소 생산량이 최대가 되는 운전조건을 찾을 수 있다.

|  |
| --- |
| objective\_y = "H2"  Optimization(gridsearch\_result, var\_x, objective\_y) |

테이블이(가) 표시된 사진

자동 생성된 설명

수소 생산량이 최대가 되는 최적 운전조건을 탐색한 결과, 모든 입력변수의 값이 최대일 때 수소가 가장 많이 생산되었다. 한편, 수소 생산량을 바탕으로 SMR 공정의 열효율을 계산할 수 있다. 공정의 열효율은 다음과 같이 정의된다.

|  |  |
| --- | --- |
|  | (4) |

투입되는 원료 및 연료의 양을 고려했을 때, 수소를 효율적으로 생산하기 위해서는 공정 열효율을 높이는 것이 중요하다.

|  |
| --- |
| objective\_y = "Efficiency"  Optimization(gridsearch\_result, var\_x, objective\_y) |

테이블이(가) 표시된 사진

자동 생성된 설명

공정 열효율이 최대가 되는 운전조건을 탐색한 결과, 천연가스의 원료 및 연료, 공기의 유량은 줄고 물과 압력변동흡착 장치의 회수율은 증가하였다. 이와 같이 격자탐색법을 이용하여 격자점을 한번 생성하고 나면, 원하는 목적함수에 맞는 최적의 운전조건을 빠르게 탐색할 수 있다.

### **[결론]**

본 장에서는 수소 생산을 위한 수증기 개질 공정을 인공신경망 기법을 이용하여 모델링하고, 최적의 운전조건을 탐색하였다. 문제 해결 과정에서 개발된 모델의 신뢰성 및 타당성을 검증하고, 민감도 분석을 통해 예측 변수에 가장 영향력 있는 입력 변수를 분석하였다. 그 다음, 인공신경망을 기반으로 격자탐색법을 이용하여 탐색공간 내에서 SMR 공정의 운전조건을 최적화하였다. 수소 생산량을 최대화하는 경우와 공정 열효율을 최대화하는 경우의 최적 운전조건이 다르게 나타났으며, 목적에 따라 공정의 운전조건을 적절하게 바꿔주어야 한다는 것을 이해하였다. 이와 같이, 인공신경망 모델과 격자탐색법을 공정 최적화에 적용하면 다양한 목적함수에 대해 신속하고 정확하게 최적의 해답을 얻을 수 있다.

### **[학습 결과]**

* 학습 내용

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

* 학습 결과 확인하기

민감도 분석을 통한 주요 변수 확인하기 및 목적함수에 따른 최적화 결과 분석하기

* 학습 결과 응용하기

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

**3.2. 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**

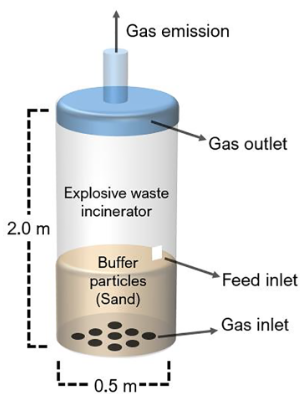


그림 1. 폭발성 폐기물 처리를 위한 유동층 반응기 구조

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 최적 조건을 규명하자.**

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

#### 관련 라이브러리를 설치하고 데이터를 작업 환경으로 입력하시오.

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.



#### 제공된 데이터를 표준화하시오.

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 구조를 설정하시오.

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.

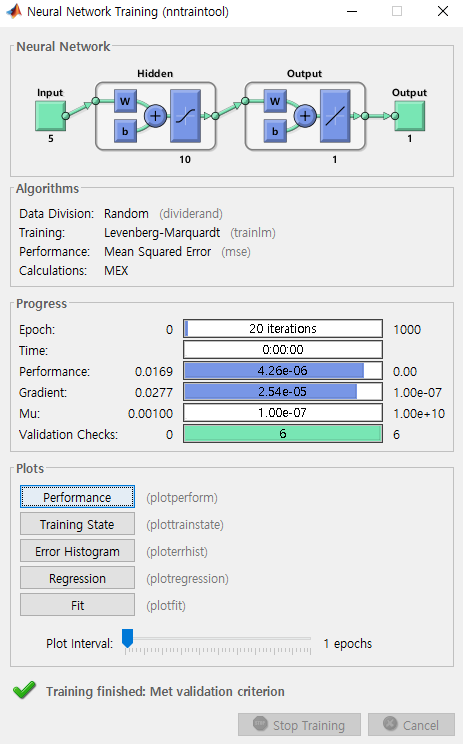
|  |
| --- |
| x = StdX' ; %데이터 구조 변환  t = Response' ; %데이터 구조 변환  trainFcn = 'trainlm'; %Training function 설정  hiddenLayerSize = 10; %은닉층 노드(hidden node) 설정  net = fitnet(hiddenLayerSize,trainFcn); %신경망 생성  net.divideParam.trainRatio = 70/100; %학습데이터 비율 설정  net.divideParam.valRatio = 15/100; %검증데이터 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.

#### 인공신경망을 학습해보자.

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

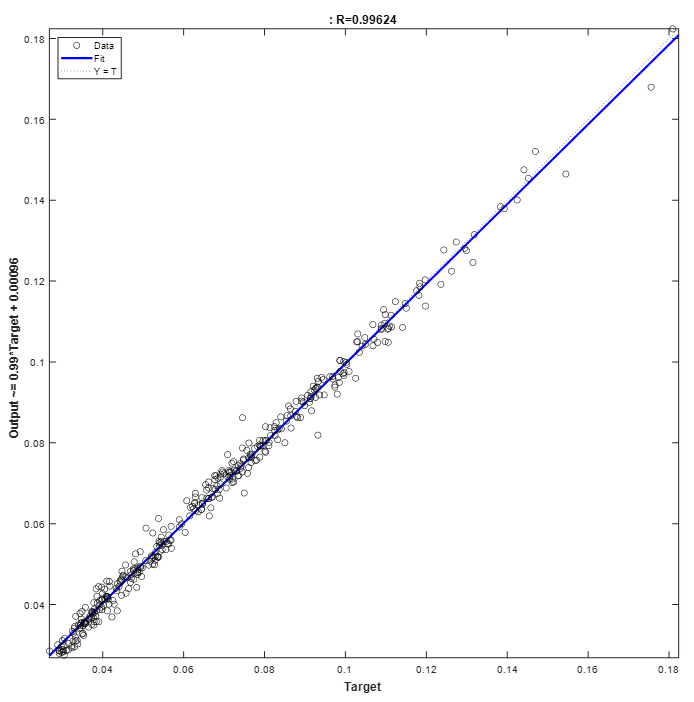


그림 4. Parity plot을 통한 모델 결과 확인

#### 인공신경망을 최적화에 사용하기 위하여 함수로 불러오자.

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); |

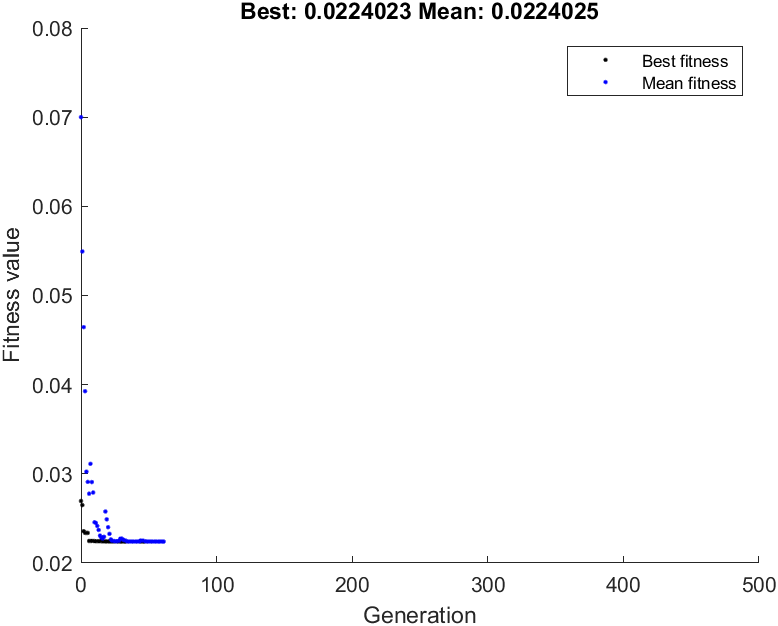


그림 5. 유전알고리즘 수행 과정

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.

### **[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