4장 인공지능 기반 물질 개발 및 거동 분석

1. 이온성 액체의 무한 희석 활성도 계수 추정

이온성 액체 (Ionic liquids, ILs)는 구조를 사용목적에 따라 선택적으로 합성 및 사용할 수 있는 designer material로, 흔히 탄화수소 추출 공정에 이용한다. 이온성 액체는 독특한 물리/화학적 특성 (전기화학적 안정성, 높은 이온전도성 등)을 갖아, 기존의 유기 용매 기반의 추출 공정을 대체할 친환경적인 대안으로 각광을 받고있다.

이온성 액체의 거동 원리는 보통 묽은 용액에서의 활동도 계수 (Infinite diluent activity coefficient, IDAC)에 기반해 이해됐다. 이를 계산하기 위한 방법으로 UNIFAC, COSMO-RS, 혹은 Abraham과 같은 mechanistic model들이 사용됐다. 하지만, 활동도 계수 계산을 위한 mechanistic model들은 낮은 정확도와 한정된 데이터를 기반으로 작동한다는 단점이 명확하다. 또한, 수많은 용매와 용질 사이의 활동도 계수 실험은 상당한 시간과 예산을 필요로 하는 작업으로, 현재의 이온성 액체 개발과 적용은 비효율적인 상태에 머물러있다.

인공신경망 (Artificial neural networks)은 데이터 기반 학습을 통한 복잡계 현상의 예측 모델을 생성하는 인공지능 알고리즘이다. 인공신경망은 고차원의 데이터 학습에 높은 성능을 보여주고 있고, 오늘날 다양한 물질 개발 및 공정 개선 연구에 활용되고 있다.

본 실습에서는 이온성 액체의 물리/화학적 특성 데이터 베이스를 활용하여, 인공신경망을 통한 활동도 계수 예측모델을 개발한다.

**[문제]**

**이온성 액체의 물리/화학적 특성 (밀도, 임계 온도 등)을 이용하여 이온성 액체의 열역학성 특성 예측 모델을 개발하고 모델의 정확도를 평가하라.**

**- “코드 및 데이터/4-2. IL\_data.csv” 데이터를 활용하여라.**

**-학습 및 평가 데이터는 7:3 비율로 분리하여라.**

**- 95% 이상의 정확도를 갖는 모델을 개발하라.**

**5.2 [방법] 인공신경망 예제**

**Q1.** 인공신경망 하이퍼파라미터 최적화를 위한 방법론을 조사하라.

**A1.** 하이퍼파라미터 최적화에는 전통적이게 manual search, grid search, random search가 있고, 통계적인 혹은 기계학습 방법으론 genetic algorithm, Bayesian optimization이 있다.

**Q2.** 인공신경망 학습을 위해 데이터의 정규화/표준화가 이루어지는 이유를 설명하고 일반적인 정규화/표준화 이외의 데이터 전처리 방법을 조사하라.

**A2.** 인공신경망은 데이터의 변위 및 크기를 인식하는 과정이 가중치를 설정에 영향을 준다. 따라서 데이터의 단위를 무시한 상대적 영향력을 파악을 위해 데이터의 정규화/표준화가 필요가 있다.

일반적인 정규화/표준화 이외의 데이터 전처리 방법은 아래와 같은 예시가 있다.

* L1 (Manhattan distance) normalization
* L2 (Euclidean distance) normalization
* L infinity normalization

**5.3 [응용] 인공 신경망 기반 이온성 액체의 열역학적 특성 예측**

예제는 Python 3.5 프로그래밍 언어를 기준으로 Jupyter 개발환경에서 작성되었다. 예제 실습을 위해 Anaconda 프로그램을 아래 URL로부터 다운로드할 수 있다.

https://www.anaconda.com/

**Q1**. 데이터를 Jupyter환경으로부터 불러오고 데이터를 확인하라.

**A1.** 다음과 같은 code를 사용하여 불러올 수 있다.

|  |
| --- |
| import numpy as np  import pandas as pd  import matplotlib.pyplot as plt |

‘pandas’ package는 데이터 편집을 위한 도구로, python 언어에서 매우 유용하게 사용된다. OSN 데이터 불러오기 및 데이터 편집 (컬럼 및 이름 설정)을 위해 본 실습에서 사용한다.

‘numpy’ package는 수학적 기능이 탑재된 도구로, 난수 발생과 대수적 계산을 위해 사용된다. 본 실습에서는 데이터의 형태 변환 및 벡터 계산에 사용한다.

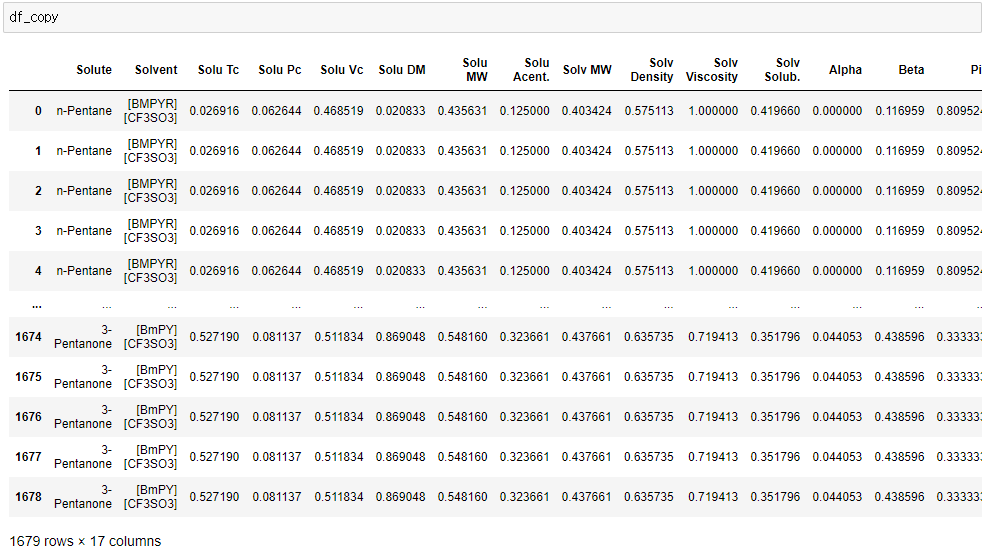
‘matplotlib.pyplot’ package는 시각화를 위한 그래프 도구로, 다양한 그래프를 그리기 위해 사용된다.

‘pandas’ package를 import하고 ‘pd’로 축약해 사용한다.

‘numpy’ package를 import하고 ‘np’로 축약해 사용한다.

‘matplotlib.pyplot’ package를 import하고 ‘plt’로 축약해 사용한다.

|  |
| --- |
| IDAC = pd.read\_csv('IL\_data.csv')  df = pd.DataFrame(data = IDAC)  df\_copy = df.copy()  df\_copy |



‘pd’를 사용하여 Jupyter 환경에 있는 ‘IL\_data.csv’ 데이터를 Jupyter script 환경에 불러온다..

‘IL\_data.csv’는 ‘pd’를 활용, 데이터 프레임 형태인 df로 정의한다.

‘.copy’ 함수를 사용하여 ‘df’ 데이터의 복사본인 ‘df\_copy’를 만든다. 이 과정은 원본 데이터의 회손을 피하기위해 실행된다..

**Q2.** 재현성있는 모델 구축을 위해 계산의 무작위성을 고정하고 데이터를 정규화하라.

**A2.** 다음과 같은 절차를 통해 무작위성 고정과 데이터 정규화가 가능하다.

|  |
| --- |
| import random as rn |

‘random’ module은 무작위 숫자 생성 등 무작위 관련 함수를 제공한다. 본 실습에서 훈련 및 검증 데이터셋 생성을 위해 사용한다.

‘random’ module을 import하고 ‘rn’으로 축약해 사용한다.

|  |
| --- |
| import sklearn  from sklearn import preprocessing  from sklearn.preprocessing import MinMaxScaler |

‘sklearn’ package는 계산을 위한 유용한 함수를 다수 내장하고 있다. ‘sklearn.preprocessing’의 ‘MinMaxScaler’ 함수를 import한다.

|  |
| --- |
| from sklearn.model\_selection import train\_test\_split  from sklearn.metrics import mean\_squared\_error, r2\_score |

‘sklearn.model\_selection’의 ‘train\_test\_split’ 함수를 import하도록 한다. ‘train\_test\_split’ 함수는 모델 학습 및 검증을 위한 모델 학습 데이터와 검증 데이터를 분할하기 위해 사용한다.

‘sklearn.metrics’의 ‘mean\_square\_error’ 함수와 ‘r2\_score’ 함수를 import하도록 한다. ‘mean\_square\_error’와 ‘r2\_score’는 모델 검증을 위한 정확도 지표를 내장하고 있는 함수이다.

|  |
| --- |
| from keras import Sequential, optimizers, metrics  from tensorflow.keras import backend as K  from keras.layers import Dense, BatchNormalization, Dropout, Activation  from keras.optimizers import Adam, SGD  from keras.initializers import lecun\_normal |

‘keras’는 ‘TensorFlow’와 같은 머신러닝 백앤드 엔진 플랫폼을 지원하는 라이브러리 이자 API (Application programming interface)이다.

‘keras’로부터 ‘Sequential’, ‘optimizer’, ‘metrics’ 함수들을 import한다.

‘tensorflow.keras’ 함수로부터 ‘backend’ import하고 ‘K’로 축약한다.

‘keras.layers’로부터 ‘Dense’, ‘BatchNormalization’, ‘Dropout’, ‘Activation’ 함수들을 import한다.

‘keras.optimizers’로부터 ‘Adam’ 함수를 import한다.

‘keras.initializers’로부터 ‘lecun\_normal’ 함수를 import한다.

|  |
| --- |
| sc = MinMaxScaler() |

‘MinMaxScaler()’ 함수를 ‘sc’로 명명하여 사용.

|  |
| --- |
| standardized\_copy = df\_copy  standardized\_copy.iloc[:,2:] = sc.fit\_transform(df\_copy.iloc[:,2:])  standardized\_copy = pd.DataFrame(standardized\_copy,  columns = df\_copy.columns)  df\_copy = standardized\_copy |

‘sc.fit\_transform’ 함수를 사용하여 ‘df\_copy’ 데이터의 numerical value만 (3열부터 17열)을 정규화한다.

정규화한 데이터는 ‘standardized\_copy’로 명명하고 데이터 프레임 형태로 변환한다.

‘standardized\_copy’ 데이터를 ‘df\_copy’ 로 명명한다.

|  |
| --- |
| X = df\_copy.iloc[:,2:-1]  y = df\_copy[['ln IDAC']] |

‘df\_copy’ 데이터의 독립변수 부분인 2열부터 16열까지를 추출하여 ‘X’로 명명한다.

‘df\_copy’ 데이터의 종속변수 부분인 ‘in IDAC’을 추출하여 ‘Y’로 명명한다.

**Q3.** 인공신경망 기반 예측 모델의 정확도를 검증하기 위해 데이터를 모델 학습 데이터와 모델 검증 데이터로 분할하라.

**A3.** 다음과 같은 절차를 통해 데이터의 분할이 가능하다.

|  |
| --- |
| tf.random.set\_seed(34)  randomstate=8 |

‘tf’의 ‘.random.set\_seed’ 함수를 사용하여 데이터 추출의 무작위성을 고정시킨다.

마찬가지로 인공신경망 학습의 무작위성을 고정시키기위해 ‘randomstate’를 생성하고 차후 학습에 사용한다.

|  |
| --- |
| X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y, test\_size = .3,  random\_state = randomstate) |

‘train\_test\_split’ 함수를 사용하여 학습데이터와 검증데이터를 생성한다.

‘X\_train’과 ‘Y\_train’은 각각 학습데이터의 독립변수와 종속변수 부분이고 마찬가지로 ‘X\_test’와 ‘Y\_test’는 각각 검증데이터의 독립변수와 종속변수 부분이다.

본 실습에서는 학습에 전체 데이터의 70%를 할당할 것이므로, ‘test\_size’ arg를 ‘.3’으로 설정한다. 모델 학습의 재현성을 위해 데이터 추출의 무작위성을 고정한다 (‘random\_state’ arg).

**Q4.** 인공신경망 기반 예측 모델을 구축하라. 모델 학습을위해 활성화함수로 ‘swish’, optimizer로 ‘adam’, 그리고 손실함수는 ‘mean\_square\_error’를 이용하라.

**A4.** 다음과 같은 절차를 통해 인공신경망 학습이 가능하다.

|  |
| --- |
| model = Sequential()  model.add(Dense(units=50, kernel\_initializer=lecun\_normal(seed=**None**),  activation='swish', input\_dim=X.shape[1]))  model.add(Dense(units=100, kernel\_initializer=lecun\_normal(seed=**None**),  activation = 'swish'))  model.add(Dense(units=150, kernel\_initializer=lecun\_normal(seed=**None**),  activation = 'swish'))  model.add(Dense(units=200, kernel\_initializer=lecun\_normal(seed=**None**),  activation = 'swish'))  model.add(Dense(units=250, kernel\_initializer='normal', activation = 'swish'))  model.add(Dense(units=150, kernel\_initializer='normal', activation = 'swish'))  model.add(Dense(units=100, kernel\_initializer='normal', activation = 'swish'))  model.add(Dense(units=50, kernel\_initializer='normal', activation = 'swish'))  model.add(Dense(units=1, kernel\_initializer='normal'))  model.compile(loss='mean\_squared\_error',  optimizer=Adam(lr=0.001, beta\_1=0.9, beta\_2=0.999,  epsilon=1e-8), metrics=[metrics.mse]) |

‘sequential()’ 함수를 사용하여 인공신경망 모델을 생성한다. 모델은 ‘model’로 명명하고 인공신경망 층은 ‘.add’를 통해 추가할 수 있다.

일반적인 신경망 모델을 사용하기 위해 ‘Dense’ 기능을 넣고, 각 노드를 지정한다.

Activation argument는 인공신경망의 활성화 함수에 해당하며, Input shape argument는 데이터의 column 차원이다.

본 실습에선 ‘swish’를 활성화 함수로 사용한다.

‘.compile’ 함수를 통해 학습의 optimizer와 loss function을 설정할 수 있다.

본 실습에서 사용된 Optimizer는 adam optimizer이고 loss function으로 MSE (Mean square error)를 사용한다.

|  |
| --- |
| model.fit(X\_train, y\_train, batch\_size=10, epochs=1000, verbose=0) |

‘.fit’ 함수를 사용하여 모델 학습을 진행한다.

모델 학습에 ‘X\_train’와 ‘y\_train’ 데이터를 사용한다. ‘batch\_size’ arg를 통해 한번에 10개 데이터셋을 고려하도록 한다.

또한, ‘epochs’ arg를 통해 반복 학습의 주기를 1000으로 지정한다.

‘verbose’ arg는 모델 학습의 진행상황을 표현하는 것으로, 0으로 설정하여 진행상황을 표시하지 않는 것으로 지정한다.

**Q5.** 인공신경망 기반 예측 모델의 정확도를 파악하라.

**A5.** 다음과 같은 절차를 통해 정확도 검증이 가능하다.

|  |
| --- |
| y\_train\_pred = model.predict(X\_train)  train\_mse = mean\_squared\_error(y\_train, y\_train\_pred)  train\_r2 = r2\_score(y\_train, y\_train\_pred) |

학습이 끝나면, ‘.predict’ 함수를 사용해서 ‘X\_train’ 데이터를 예측한다. 예측된 결과값은 ‘y\_train\_pred’로 명명한다.

‘mean\_squared\_error’와 ‘r2\_score’함수를 사용해서 ‘y\_train’과 ‘y\_train\_pred’의 정확도를 확인한다.

|  |
| --- |
| train\_mse |



|  |
| --- |
| train\_r2 |



학습의 정확도를 확인한다.

|  |
| --- |
| y\_pred = model.predict(X\_test)  test\_mse = mean\_squared\_error(y\_test, y\_pred)  test\_r2 = r2\_score(y\_test, y\_pred) |

마찬가지로 ‘mean\_squared\_error’와 ‘r2\_score’함수를 사용해서 ‘y\_test’과 ‘y\_test\_pred’의 정확도를 확인한다.

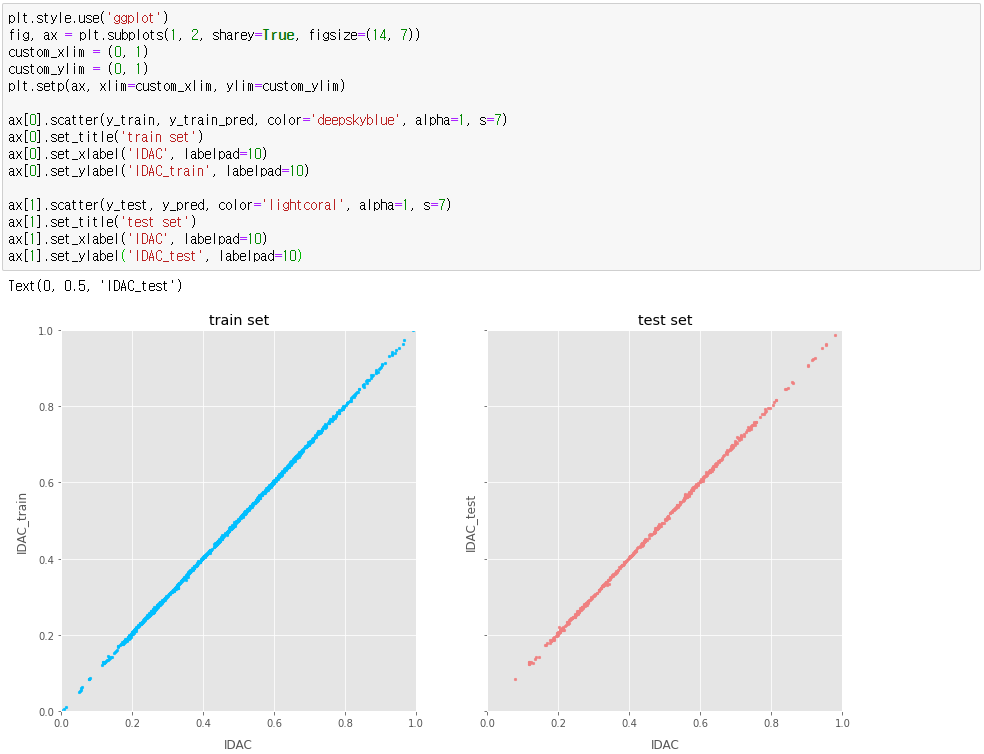
|  |
| --- |
| test\_mse |



|  |
| --- |
| test\_r2 |



|  |
| --- |
| plt.style.use('ggplot')  fig, ax = plt.subplots(1, 2, sharey=**True**, figsize=(14, 7))  custom\_xlim = (0, 1)  custom\_ylim = (0, 1)  plt.setp(ax, xlim=custom\_xlim, ylim=custom\_ylim)  ax[0].scatter(y\_train, y\_train\_pred, color='deepskyblue', alpha=1, s=7)  ax[0].set\_title('train set')  ax[0].set\_xlabel('IDAC', labelpad=10)  ax[0].set\_ylabel('IDAC\_train', labelpad=10)  ax[1].scatter(y\_test, y\_pred, color='lightcoral', alpha=1, s=7)  ax[1].set\_title('test set')  ax[1].set\_xlabel('IDAC', labelpad=10)  ax[1].set\_ylabel('IDAC\_test', labelpad=10) |



‘plt’를 사용하여 정확도를 시각화한다.

**[결론]**

본 장에서는 이온성 액체의 열역학적 특성 (활동 계수)을 예측하기위해 인공신경망을 이용했다. 데이터 정규화를 통해 데이터 전처리를 진행했다. 또한, 모델 개발을 위해 인공신경망의 하이퍼파라미터를 조정하여 모델 정확도 개선 방법을 학습했다. 결과로, 열역학적 특성 예측을 위한 모델 개발 방법을 학습했다.

**학습 결과**

* 학습 내용

추출 공정의 열역학 특성 분석을 위한 인공 신경망 기반 예측 방법론 익히기.

* 학습 결과 확인하기

인공 신경망 알고리즘의 활용 방법 및 예측 모델 학습을 위한 데이터 구조화 익히기.

* 학습 결과 응용하기

본 장의 학습내용에 기반해 추출 공정 설계를 위한 인공 신경망 적용 방법 조사하기.