


## 2차년도 주요 결과물

(과제명) 대규모 분산 에너지 저장장치 인프라의 안전한 자율운영  
및 성능 평가를 위한 지능형 SW 프레임워크 개발  
(과제번호) 2021-0-00077

- 결과물명 : 실험 데이터로 학습한 성능 진단 알고리즘 (SW)  
(BMS 실운용 데이터에 대한 OCV 라벨링 코드(SW))
- 작성일자 : 2022년 12월 07일

과학기술정보통신부 SW컴퓨팅산업원천기술개발사업  
“2차년도 주요 결과물” 로 제출합니다.

수행기관	성명/직위	확인
서울대학교	강명주/연구책임자	

정보통신기획평가원장 귀하

## <목차>

1. 개요 .....	1
가. 목적 .....	1
나. 범위 .....	1
2. 기법 .....	2
3. 소프트웨어 .....	2
가. 주요 기능 .....	2
나. 사용 환경 .....	4
다. 사용 방법 .....	4
라. 코드 .....	5

## 1. 개요

### 가. 목적

- 본 문서는 “대규모 분산 에너지 저장장치 인프라의 안전한 자율운영 및 성능 평가를 위한 지능형 SW 프레임워크 개발” 사업의 주요 결과물에 대한 보고서이다.
- 안전 AI 분석 엔진 “실험 데이터로 학습한 성능 진단 알고리즘 (SW)” 중 “BMS 실운영 데이터에 대한 OCV 라벨링 코드” 소프트웨어에 대한 설명을 한다.

[ 안전 AI 분석 엔진 SW ]

구분	1차년도	2차년도	3차년도	4차년도	5차년도
안전 AI 분석 엔진	물성 진단 지표 후보(SW)	실험 데이터로 학습한 이상 탐지 알고리즘(SW)	전이학습을 완료한 이상 탐지 알고리즘(SW)	최적화된 이상 탐지, 성능 진단 알고리즘(SW)	이상 탐지 알고리즘 및 사고 대응 알고리즘(SW)
	데이터 전처리 알고리즘(SW)	실험 데이터로 학습한 성능 진단 알고리즘(SW)	전이 학습을 완료한 성능 진단 알고리즘(SW)	개선된 이상 탐지, 성능 진단 알고리즘(SW)	상태 진단 알고리즘 및 효율적 운영 분석 결과(SW)

### 나. 범위

- 배터리의 물성 중 배터리의 현재 상태를 나타내는 지표 중 하나인 OCV 사용
- 실제 태양광 BMS 운용 데이터 대상
- 방전 구간 구분 및 이에 대한 OCV 라벨링

## 2. 기법

- 배터리의 물성 지표 중 현재 상태를 나타내는 지표 중 하나인 OCV를 계산하고, 이를 하나의 물성 지표로 활용하거나, 모델 학습 혹은 데이터 분석에 활용하도록 라벨링한다.
- 본 프로그램은 실제 태양광 BMS 운용 데이터를 대상으로 하여, BMS 내 OCV 계측을 위한 장치가 없이도 사용하는 배터리 셀 정보 및 배터리 시스템 구성 정보를 활용해 OCV를 라벨링한다.
- 본 프로그램은 BMS 사이트 중 데이터가 제공되는 시온유/판리 사이트의 데이터를 분석하며, 배터리 셀 정보를 활용할 수 있는 방전 구간에서 OCV를 계산한다.

## 3. 소프트웨어

### 가. 주요 기능

- 데이터 입출력 (메인 함수)

```
def OCV_label(data_path, site, voltage_threshold=None, save_data_path=None, save_fig_dir=None, monitor=False):  
    """  
    data_path : data path for OCV labeling  
    site : sionyu or panli  
    voltage_thres : voltage threshold for sunny days (=fully-charged day)  
    save_data_path : path for ocv labeled data (parquet or csv)  
    save_fig_dir : save figure 1)clear days 2)daily OCV labeled  
        if None, do not save figure  
    monitor : print monitoring conditions - discharge voltage drops  
    """  
    if '.csv' == data_path[-4:]:  
        file_name = data_path[:-4]  
    elif '.parquet' == data_path[-8:]:  
        file_name = data_path[:-8]  
  
    data = pd.read_parquet(data_path)  
    data = data.sort_values('TIMESTAMP')  
    data = data.reset_index(drop=True)  
  
    print(f'{file_name} OCV labeling starts')  
  
    if site == 'sionyu':  
        voltage_thres = 808 if voltage_threshold == None else voltage_threshold  
        bank_capacity, b2c_current, b2c_voltage, ocv2soc, soc2ocv = sionyu_spec()  
        data_ocv_labeled = ocv_labeling(data, site, voltage_thres, bank_capacity, b2c_current, b2c_voltage,  
                                         ocv2soc, soc2ocv, save_fig_dir, file_name, monitor=monitor)  
  
    if site == 'panli':  
        voltage_thres = 970 if voltage_threshold == None else voltage_threshold
```

```

data_ocv = []
for bank_id_panli in [1, 2]:
    print(f"BANK ID : {bank_id_panli}")
    data_i = data[data['BANK_ID']== bank_id_panli]
    bank_capacity, b2c_current, b2c_voltage, ocv2soc, soc2ocv = panli_spec(bank_id=bank_id_panli)
    ocv_labeled = ocv_labeling(data_i, site, voltage_thres, bank_capacity, b2c_current, b2c_voltage,
                              ocv2soc, soc2ocv, save_fig_dir, file_name, monitor=monitor)
    data_ocv.append(ocv_labeled)
data_ocv_labeled = pd.concat([data_ocv[0], data_ocv[1]])

if save_data_path == None:
    save_data_path = file_name + '_OCVlabeled.parquet'

if '.csv' == save_data_path[-4:]:
    data_ocv_labeled.to_csv(save_data_path)
elif '.parquet' == save_data_path[-8:]:
    data_ocv_labeled.to_parquet(save_data_path)

```

## - 배터리 셀 스펙 활용 (시온유 예시)

```

def sionyu_spec(inter_k='cubic'):
    """
    battery spec : 33J
    1 bank = 8 rack (parallel), 1 rack = 17 module (serial), 1 module = 12 core (serial), 1 core = 60 cell (parallel)

    inter_k : interpolatge method
    """
    ess_comp = {'bank': [8, 'parallel'], 'rack': [17, 'serial'], 'module': [12, 'serial'], 'core': [60, 'parallel']}

    b2c_current = reduce(lambda x, y : x*y, dict(filter(lambda v : v[1]!='parallel', ess_comp.values())).keys()) # 60*8
    b2c_voltage = reduce(lambda x, y : x*y, dict(filter(lambda v : v[1]!='serial', ess_comp.values())).keys()) # 17*12

    bank_capacity = 2.962 * 60 * 8 # 33J spec * b2c_current

    # 33J spec
    soc_table = np.arange(1., 0, -0.05)
    ocv_table = np.array([4.036, 3.985, 3.941, 3.899, 3.842,
                          3.809, 3.778, 3.731, 3.693, 3.672,
                          3.656, 3.637, 3.626, 3.615, 3.599,
                          3.577, 3.548, 3.504, 3.463, 3.427])
    ocv2soc_main = interpolate.interpld(ocv_table, soc_table, kind=inter_k)
    ocv2soc_end = interpolate.interpld(np.array([3.427, 3.213]), np.array([0.05, 0]), kind='linear')
    soc2ocv_main = interpolate.interpld(soc_table, ocv_table, kind=inter_k)
    soc2ocv_end = interpolate.interpld(np.array([0.05, 0]), np.array([3.427, 3.213]), kind='linear')

    ocv2soc = lambda x : ocv2soc_main(x) if (x > 3.427) else ocv2soc_end(x)
    soc2ocv = lambda x : np.concatenate((soc2ocv_main(x[x > 0.05]), soc2ocv_end(x[x <= 0.05])))

    return bank_capacity, b2c_current, b2c_voltage, ocv2soc, soc2ocv

```

## - OCV 라벨링

```
def ocv_labeling(data, site, voltage_thres, bank_capacity, b2c_current, b2c_voltage, ocv2soc, soc2ocv, save_fig_dir, file_name, monitor,
scaling=True):
    # Prepare data. fully-charged day
    thres_above = data[['TIMESTAMP', 'BANK_DC_VOLT']][data['BANK_DC_VOLT'] >= voltage_thres]
    clear_date = pd.to_datetime(thres_above['TIMESTAMP'], utc=True).dt.tz_convert('Asia/Seoul').dt.date.unique()

    clear_data = []
    for date in clear_date:
        clear_data.append(select_oneday_data(data, date=date))

    # plot clear_data
    if save_fig_dir != None:
        os.makedirs(save_fig_dir+f'/{file_name}', exist_ok=True)
        plt.figure(figsize=(16,12), facecolor='white')
        for oneday in clear_data:
            plt.plot(oneday['BANK_DC_VOLT'].to_numpy(), alpha=0.2, label=oneday['TIMESTAMP'].iloc[0])
        plt.legend()
        plt.title(file_name)
        plt.savefig(save_fig_dir + f'/{file_name}/{file_name}_{len(clear_date)}cleardays')

    # extract discharge data
    bank_labeled_clear = pd.DataFrame({})
    for oneday in clear_data:
        oneday_date = str(oneday['TIMESTAMP'].iloc[0])[:10]

        discharge_data = extract_discharge(oneday, site)
        discharge_data['AVERAGE_CELL_VOLT'] = discharge_data['BANK_DC_VOLT'] / b2c_voltage

        if monitor:
            print(discharge_data['AVERAGE_CELL_VOLT'].iloc[0], 'to', discharge_data['AVERAGE_CELL_VOLT'].iloc[-1]) ## monitor

        capacity = calculated_capacity(discharge_data)

        soc_init = ocv2soc(discharge_data['AVERAGE_CELL_VOLT'].iloc[0])

        # Scale capacity.
        if scaling:
            soc_end = ocv2soc(oneday['BANK_DC_VOLT'].iloc[-1] / b2c_voltage)
            scale_factor = (soc_end - soc_init) / (capacity[-1] / bank_capacity)
        else:
            scale_factor = 1

        soc_calculated = 100 * (soc_init + capacity / bank_capacity * scale_factor)

        ocv_estimated = soc2ocv(soc_calculated / 100) * b2c_voltage
        discharge_data['OCV_est'] = ocv_estimated

    # oneday = oneday[['TIMESTAMP', 'OCV_est']]
    oneday_labeled = pd.merge(oneday, discharge_data, how='inner', on='TIMESTAMP')
    oneday_labeled = oneday_labeled[['TIMESTAMP', 'OCV_est']]

    bank_labeled_clear = pd.concat([bank_labeled_clear, oneday_labeled], axis=0)

    bank_labeled = pd.merge(data, bank_labeled_clear, how='inner', on='TIMESTAMP')

    return bank_labeled
```

## 나. 사용 환경

- 본 프로그램은 python 코드로 구현되었음
- Python이 설치된 어떠한 OS 환경에서도 사용이 가능함
- 코드의 구동에는 python의 OS, numpy, pandas, scipy, matplotlib 라이브러리가 필요함

## 다. 사용 방법

- 입력과 출력 데이터는 parquet 또는 csv 형식을 사용함
- 실행 창에 `python ocv_labeling_for_discharge.py` 명령어를 입력 후 옵션값으로 `--data_path`에 BMS 수집 데이터의 파일명을 입력하고 `--site`에 BMS 사이트 명 (sionyu/panli)을 입력
- 필요 시 완전 충전 완료를 전제하는 전압값을 `--voltage_threshold`에 입력할 수 있음

## 라. 코드

OpenESS/SNU\_AI/data\_preprocess/ess\_preprocess/ocv\_labeling\_for\_discharge.py