


1단계(3차년도) 기술문서

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

- 결과물명 : 태양광 BMS의 안전도 계산을 위한 전처리 기법
및 머신러닝 기반 분석 기법 기술 보고서
- 작성일자 : 2023년 11월 20일

과학기술정보통신부 SW컴퓨팅산업원천기술개발사업
“1단계(3차년도) 기술문서” 로 제출합니다.

수행기관	성명/직위	확인
한국전자기술연구원	최효섭/책임연구원	

정보통신기획평가원장 귀하

정보통신기획평가원장 귀하

사 용 권 한

본 문서에 대한 서명은 한국전자기술연구원 내부에서 본 문서에 대하여
수행 및 유지관리의 책임이 있음을 인정하는 것임.

본 문서는 작성, 검토, 승인하여 승인된 원본을 보관한다.

작성자 :	박진원	일자 :	2022. 10. 24
-------	-----	------	--------------

검토자 :	김창우	일자 :	2022. 10. 25
-------	-----	------	--------------

승인자 :	최효섭	일자 :	2022. 10. 27
-------	-----	------	--------------

제 · 개정 이력

버전	변경일자	제·개정 내용	작성자
1.0	2023-10-24	최초 등록	박진원

목 차

1. 개요	1
2. SoS(State of Safety) 알고리즘	2
3. SoS 계산 및 머신러닝을 위한 데이터 전처리	4
4. 머신러닝 모델 학습 방법과 예측 결과	7

1. 개요

□ 목적

- 본 기술문서는 태양광 배터리 관리 시스템(BMS)의 안전성을 높이기 위한 방법론을 제시하는 것입니다. 현 시점에서 태양광 BMS는 다양한 센서를 활용하여 배터리 상태를 모니터링하고 있지만, 그 안전성은 아직 완전히 보장되지 않는다. 이를 해결하기 위해, 본 연구에서는 SoS(State of Safety) 측정 지표를 도입하여 배터리의 안전 상태를 지속적으로 모니터링한다. 또한, 머신러닝 알고리즘 Linear regression, LightGBM, XGBoost, CatBoost를 활용하여 배터리의 안전도를 사전에 예측한다. 이 연구는 태양광 BMS의 안전성 향상에 기여할 것으로 기대된다.

□ 기술 개요

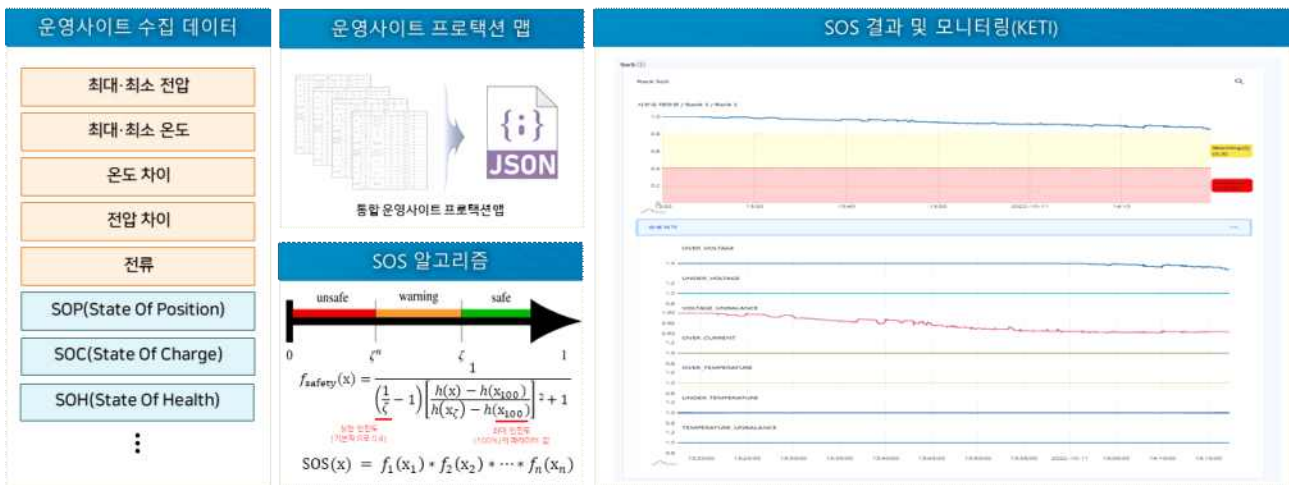


그림 2 SoS(State of Safety) 알고리즘

- 본 문서에서는 운영사이트에서 수집되는 데이터 중 Cell 데이터에 해당하는 데이터를 사용하여 SoS를 계산한다. SoS에는 배터리의 상태를 판단하는데 주요한 요소들을 넣을 수 있다. 본 연구에서는 최대·최소 전압, 최대·최소 온도, 온도차이, 전압차이, 전류를 사용하였다.
- 본 기술은 배터리의 상태를 지속적으로 모니터링 하기 위함이다. 따라서, 각 사이트마다 웹페이지 서비스를 통해 모니터링을 제공하고 있다. (<https://ess.keti-dp.re.kr/>)

2. SoS(State of Safety) 알고리즘

□ SoS 알고리즘

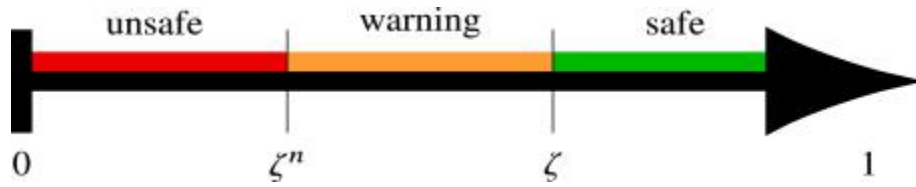


그림 3 SoS 함수에 대한 운용 영역

- 본 문서에서 사용하는 알고리즘은 논문 “Calculation of the state of safety (SOS) for lithium ion batteries” 을 참고하였다.
- 그림 2에서 ζ 는 상한안전도를 나타내며, n 은 하위함수의 개수를 나타낸다. 본연구에서는 상한안전도는 0.8으로 설정하였다.

항목	Level	설정값	보안
과충전	경고	Max Cell Volt	
	보호	Max Cell Volt	
	해제	Max Cell Volt	
과방전	경고	Cell	
	보호	Cell	
	해제	Cell	
전압 불평형	경고	[Max Cell Volt] - [Min Cell volt]	
	보호		
	해제		
충전 과전류	보호	Charging Current	
방전 과전류	보호	Discharging Current	
고온	경고	Max Temp	
	보호		
	해제		
저온	경고	Min Temp	
	보호		
	해제		
온도 불평형	경고	[Max Temp] - [Min Temp]	
	보호		
	해제		

그림 4 운영사이트 프로텍션맵

- 그림 2에서 위험과 안전의 경계의 기준이 되는 ζ^n 에서 0.4096으로 설정하였다. n 은 하위 함수의 값을 나타내고 ζ 는 0.8로 설정하였기 때문에 본연구에서 $\zeta^5 = 0.3276$ 이지만, 전압 값과 온도 값이 종속적이라 판단하였기 때문에 $n=4$ 로 설정하고 진행하였다.

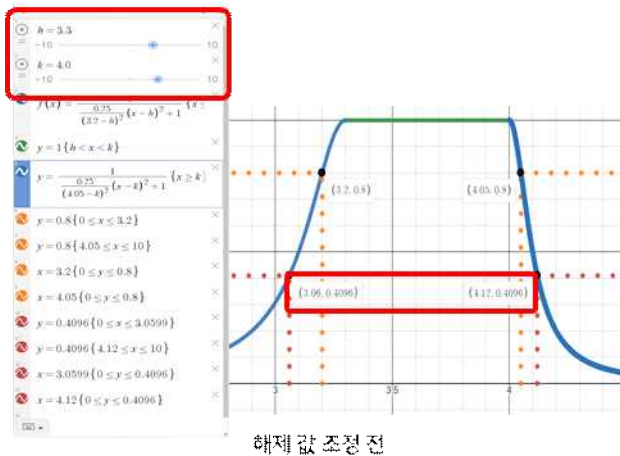
$$f_{safety} = \frac{1}{\left(\frac{1}{\zeta} - 1\right) \left[\frac{h(x) - h(x_{100})}{h(x_{\zeta}) - h(x_{100})} \right]^2 + 1} \quad (1)$$

- o 여기서, $h(x)$ 는 입력 파라미터의 현재 값, $h(x_\xi)$ 는 입력 파라미터에 해당되는 프로텍션 맵의 경고 값, $h(x_{100})$ 은 입력 파라미터에 해당되는 프로텍션 맵의 해제 값이다.
- o 각 입력 파라미터들에 대해서 f_{safety} 를 구하게 되면, 전체 시스템의 SoS 값을 식 (2)와 같은 방법으로 구할 수 있다.

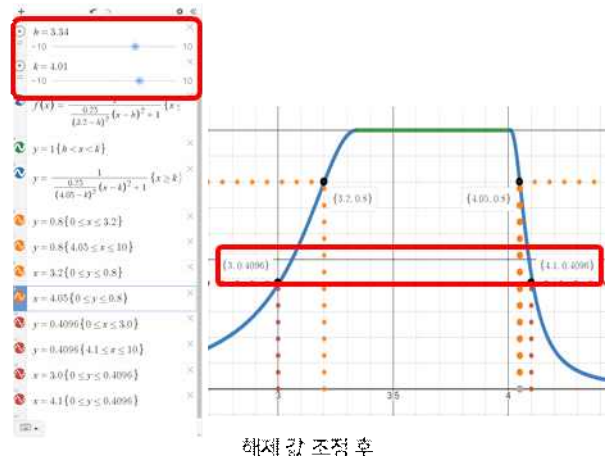
$$SoS(x) = f_1(x_1) * f_2(x_2) * \dots * f_n(x_n) \quad (2)$$

□ 해제값 조정

- o SoS의 알고리즘 적용 이전에 해제 값을 수정해야 한다. “해제 값 조정 전”에는 그림 x와 같이 Unsafe의 기준이 $\xi^4 = 0.4096$ 과 그에 해당하는 프로텍션 맵의 보호 값이 다르다. 따라서, 주요한 상황인 Unsafe의 기준을 맞추기 위해 $h(x_{100})$ 을 수정한다. 해제 값을 조정하기 위해서 Desmos 툴을 사용하였다.



해제 값 조정 전



해제 값 조정 후

그림 5 해제 값 조정

3. SoS 계산 및 머신러닝을 위한 데이터 전처리

□ 알고리즘 계산을 위한 데이터 전처리

- 운영사이트의 데이터는 매초 데이터가 들어오기 때문에 실시간으로 적용하여 값을 모니터링할 수 있다. 하지만, 운영사이트의 데이터의 획득시 카프카를 사용하기 때문에 배치 단위로 데이터가 DB에 쌓이게 된다. 따라서, 실시간으로 계산시 데이터가 없는 경우가 발생하게 됨으로써 10초의 interval 값을 주어 데이터를 쿼리한다.

데이터 전처리
<pre>try: #TimescaleDB에서 데이터 획득 bank1_query = f"""select * from rack where "TIMESTAMP" between now() - interval '10' second and now() and "BANK_ID" = 1 order by "TIMESTAMP" desc""" bank2_query = f"""select * from rack where "TIMESTAMP" between now() - interval '10' second and now() and "BANK_ID" = 2 order by "TIMESTAMP" desc""" with conn.cursor() as cur: cur.execute(bank1_query) bank1_query = dictfetchall(cur) cur.execute(bank2_query) bank2_query = dictfetchall(cur) # 데이터 수집 과정에서의 문제로 인한 2번의 쿼리 df_bank1 = pd.DataFrame(bank1_query, columns=config.col_sel) df_bank2 = pd.DataFrame(bank2_query, columns=config.col_sel) # 중복 데이터 제거 (10초 이내의 데이터중 가장 처음 데이터 획득) df_bank1 = df_bank1.dropna().drop_duplicates(["RACK_ID"], keep = 'last') df_bank2 = df_bank2.dropna().drop_duplicates(["RACK_ID"], keep = 'last') df = pd.concat([df_bank1, df_bank2], axis=0).reset_index(drop=True) # 시간이 다를수 있으므로통일을 위한 작업 df["TIMESTAMP"] = query_time</pre>

- 위 과정을 통해 얻게 되는 데이터를 SoS 계산 알고리즘에 넣게 되면, SoS 값을 구할 수 있게 된다.

SoS 알고리즘 구현
<pre>def sos_function(safety_inf, inf, value): exp1 = 0.25 / math.pow(safety_inf[inf]["upper_safety"] - safety_inf[inf]["maximum_safety"], 2) exp2 = math.pow(value - safety_inf[inf]["maximum_safety"], 2) exp3 = 1/(exp1*exp2+1) return exp3</pre>

```

def calc_sos_safety(data, safety_inf):

    f_safety = {"OVER_VOLTAGE":{},
                "UNDER_VOLTAGE":{},
                "VOLTAGE_UNBALANCE":{},
                "OVER_CURRENT":{},
                "OVER_TEMPERATURE":{},
                "UNDER_TEMPERATURE":{},
                "TEMPERATURE_UNBALANCE":{}
                }

    for inf in safety_inf:
        if inf in config.condi_1:
            for k, v in df_dict[inf].items():
                if df_dict[inf][k] < safety_inf[inf]["maximum_safety"]:
                    f_safety[inf][k] = 1
                else:
                    sos_val = sos_function(safety_inf, inf, v)
                    f_safety[inf][k] = sos_val

            elif inf in config.condi_2:
                for k, v in df_dict[inf].items():
                    if df_dict[inf][k] > safety_inf[inf]["maximum_safety"]:
                        f_safety[inf][k] = 1
                    else:
                        sos_val = sos_function(safety_inf, inf, v)
                        f_safety[inf][k] = sos_val

    sos_dict = {"SOS_SCORE":{}}
    for k in f_safety["OVER_VOLTAGE"].keys():
        sos_score = 1
        for k2 in f_safety.keys():
            f = f_safety[k2][k]
            sos_score = sos_score * f
        sos_dict["SOS_SCORE"][k] = sos_score

    f_safety.update(sos_dict)
    df3 = pd.DataFrame(f_safety)
    df3 = df3.reset_index().rename(columns={"level_0":"BANK_ID", "level_1":"RACK_ID"})
    df3 = pd.concat([df3, df["TIMESTAMP"]], axis=1)

    # OPERATING_SITE = 1 -> 시운유, 2 -> 관리
    df3["OPERATING_SITE"] = 2

    return df3

```

□ 머신러닝 학습 데이터 셋을 위한 데이터 전처리

- o SoS 값을 예측하기 위해서는 label이 필수적이다. 따라서, 위과정을 통해 SoS값을 구하여, 데이터 셋이 필요하다. 아래 그림5는 TimescaleDB에 실제 저장된 머신러닝을 위한 데이터 셋이다.

o 머신러닝 학습시 시간에 대한 부분을 넣기 위해서는 datetime 형식의 TIMESTAMP
를 분리해야한다. 또한, 좋은 학습을 위해서는 해당 날짜의 날씨를 기록해주는 것
도 좋은 방법이나 본 연구에서는 제외하고 진행하였다.

[illegible]

그림 6 머신러닝 학습을 위한 SoS 데이터셋

o 위 그림에서 데이터틀 보게되면 데이터간의 스케일이 맞지 않기 때문에 스케일을 맞춰줘야 한다. 스케일을 맞추는 방법은 다양한 방법이 있지만 사이킷런의 StandardScaler를 사용하였다.

4. 머신러닝 모델 학습 방법과 예측 결과

□ 머신러닝 모델 학습 방법

- 본 연구에서 사용하는 알고리즘은 라이브러리를 쉽게 설치한 뒤, 간편하게 사용할 수 있다. 머신러닝을 사용한 SoS 예측은 10분 뒤의 값을 예측할 것이므로 아래와 같이 코드를 구성하였으며, 그림6과 같이 trainX를 나누었으며, trainY는 해당 row에서 10분 뒤의 SoS 값이다.

학습을 위한 데이터셋

```
def get_dataset(site_num: int, rack_id: int, start: str, end: str, label: str="SOS_SCORE"):
    conn=psycopg2.connect(host="34.64.61.225",
                           dbname="ess_stats",
                           user="postgres",
                           password="keti1234!",
                           port="5432")

    query = f"""SELECT * FROM ess_sos WHERE "OPERATING_SITE"={site_num} AND "BANK_ID"=1 AND
    "RACK_ID"={rack_id} AND "TIMESTAMP" BETWEEN '{start}' AND '{end}' order by "TIMESTAMP", "RACK_ID" """

    with conn.cursor() as cur:
        cur.execute(query)
        query_dict = dictfetchall(cur)
    df = pd.DataFrame(query_dict, columns=config_columns)

    #비어있는 시간 데이터 채우기
    df.set_index("TIMESTAMP", drop=True, inplace=True)
    idx = pd.date_range(start=df.index.min(), end=df.index.max(), freq='s')
    df = df.reindex(idx, fill_value=np.nan).ffill()

    #시간 데이터 분리작업
    df = df.reset_index().rename(columns={"index": "TIMESTAMP"})
    df = df.reset_index(drop=True)
    df = time_separation(df)

    #데이터셋 만들기
    features = list(df.columns)
    features.remove(label)
    #dataX = df.loc[:, features]
    #dataY = df.loc[:, label]
    dataX = df.loc[:len(df) - 600 -1, features]
    dataY = df.loc[600:, label]
    dataX, dataY = dataX.apply(pd.to_numeric), dataY.apply(pd.to_numeric)

    return dataX, dataY

# 트레이닝 부분
print("LinearRegression learning in progress...\n")
linear_reg = LinearRegression().fit(trainX, trainY)

print("CATBoost learning in progress...\n")
cat_model = CatBoostRegressor(task_type="GPU", devices='0:1').fit(trainX, trainY)

print("LightGBM learning in progress...\n")
lgb_model = LGBMRegressor()
lgb_model.fit(trainX, trainY)
```

```
print("XGBoost learning in progress...\n")
xgb_model = XGBRegressor()
xgb_model.fit(trainX, trainY)

# 예측시 넘어가는 부위를 제거 하기 위한 부분
linear_pred = np.clip(linear_reg.predict(testX), a_min=0, a_max=1)
cat_pred = np.clip(cat_model.predict(testX), a_min=0, a_max=1)
lgb_pred = np.clip(lgb_model.predict(testX), a_min=0, a_max=1)
xgb_pred = np.clip(xgb_model.predict(testX), a_min=0, a_max=1)
```

	RACK MAX CELL VOLTAGE	RACK MIN CELL VOLTAGE	RACK CELL VOLTAGE GAP	RACK CURRENT	RACK MAX CELL TEMPERATURE	RACK MIN CELL TEMPERATURE	RACK CELL TEMPERATURE GAP	year	month	day	hour	minute	second
0	3.367	3.348	0.019	0.1	31.1	27.3	3.8	2022	6	1	0	0	0
1	3.367	3.348	0.019	0.1	31.1	27.3	3.8	2022	6	1	0	0	1
2	3.367	3.348	0.019	0.1	31.1	27.3	3.8	2022	6	1	0	0	2
3	3.367	3.348	0.019	0.1	31.1	27.3	3.8	2022	6	1	0	0	3
4	3.367	3.348	0.019	0.1	31.1	27.3	3.8	2022	6	1	0	0	4
..
15810595	3.471	3.459	0.011	0.1	19.1	16.3	2.7	2022	11	30	23	49	55
15810596	3.471	3.459	0.011	0.1	19.1	16.3	2.8	2022	11	30	23	49	56
15810597	3.471	3.459	0.011	0.1	19.1	16.3	2.8	2022	11	30	23	49	57
15810598	3.471	3.459	0.011	0.1	19.1	16.3	2.8	2022	11	30	23	49	58
15810599	3.471	3.459	0.011	0.1	19.1	16.3	2.8	2022	11	30	23	49	59

15810600 rows x 13 columns

그림 7 학습을 위한 데이터셋

□ 머신러닝 예측 결과

- o 그림7, 8은 XGBoost, LightGBM, CatBoost, Linear regression을 사용하여 결과를 나타내었다. 아래 결과에서 볼수 있듯이 별다른 하이퍼파라미터를 조정하지 않았지만 XGBoost, LightGBM, CatBoost에서 좋은 결과를 얻을 수 있다.

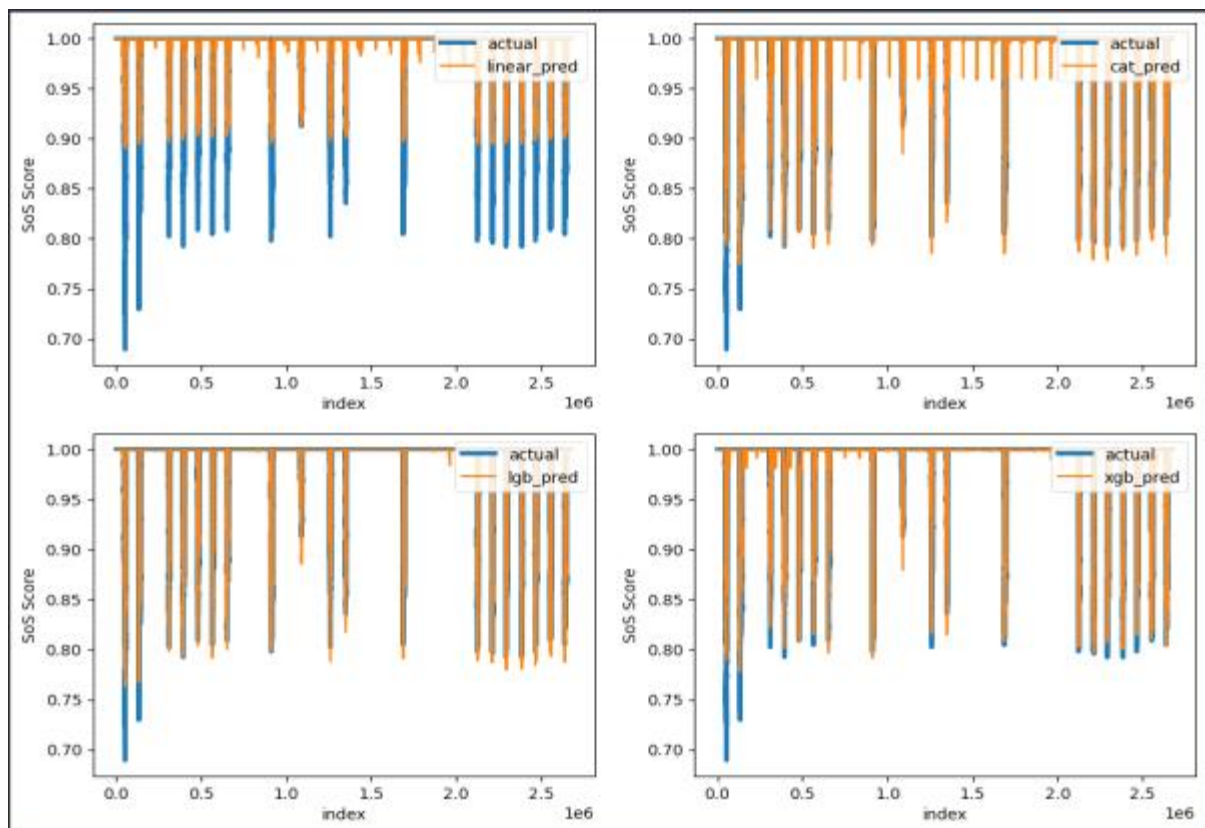


그림 8 머신러닝을 사용한 예측 실험 결과

```

=====
Linear loss(MSE) : 0.0002983013687108769
Linear loss(RMSE) : 0.01727140320619251
Linear loss(MAE) : 0.0059013722094673786
=====
CatBoost loss(MSE) : 5.8568530576894464e-05
CatBoost loss(RMSE) : 0.007653007943083194
CatBoost loss(MAE) : 0.0021987448218811424
=====
LightGBM loss(MSE) : 3.0544358522810294e-05
LightGBM loss(RMSE) : 0.005526695081403559
LightGBM loss(MAE) : 0.0014152166282314805
=====
XGBoost loss(MSE) : 6.187884009027659e-05
XGBoost loss(RMSE) : 0.007866310449650242
XGBoost loss(MAE) : 0.0023470900618534386
=====

```

그림 9 Loss 결과

- o 추후, 날씨에 대한 정보와 각각 알고리즘의 하이퍼파라미터 튜닝을 조정하면 더 좋은 결과를 낼 수 있을 것이다.