Python Week

LSTM Forecasting

한끼원

주제 선정 이유



- Deep Learning Study: RNN을 Pytorch로 구현
- Kaggle Study: ML Pipeline 구성
- Lab Intern: LSTM Paper Review

→ LSTM을 Pytorch로 구현하고, ML Pipeline 구성해보자

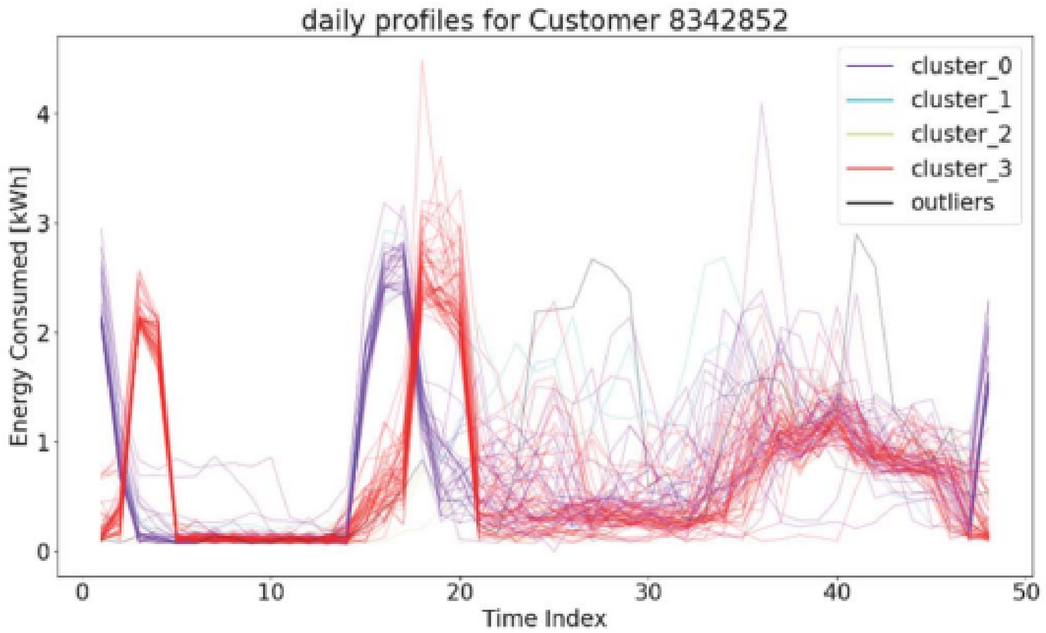
IEEE TRANSACTIONS ON SMART GRID, VOL. 10, NO. 1, JANUARY 2019

Short-Term Residential Load Forecasting Based on LSTM Recurrent Neural Network

Weicong Kong, *Student Member, IEEE*, Zhao Yang Dong, *Fellow, IEEE*, Youwei Jia, *Member, IEEE*, David J. Hill, *Life Fellow, IEEE*, Yan Xu, *Member, IEEE*, and Yuan Zhang, *Student Member, IEEE*

- 문제: 개별 전력 수요는 예측 난이도가 높음
- 제안: LSTM 기반 단기 예측 프레임워크
- 결과: 생활 패턴이 일정하지 않은 가구에서 기존 방법론보다 우세

가구별 전력 소비 패턴 클러스터링



- 한 고객 의 데이터를 30분 단위, 총 48차원 벡터로 표현
- 빨간색과 보라색 두 가지 주요 생활 루틴을 가짐

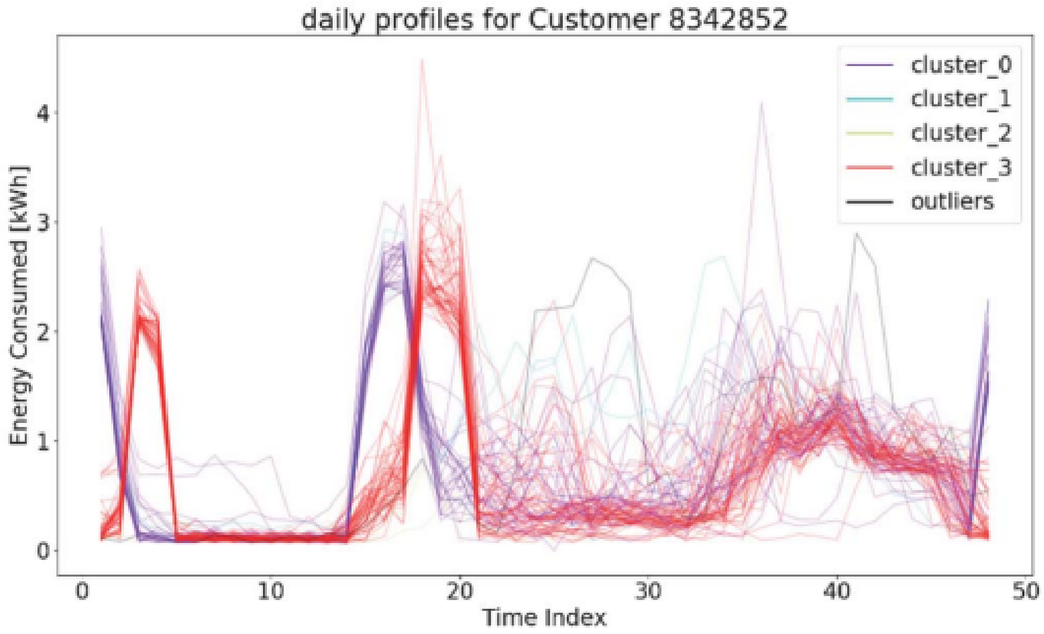
AMPds 데이터의 일일 전력 소비 패턴 클러스터링

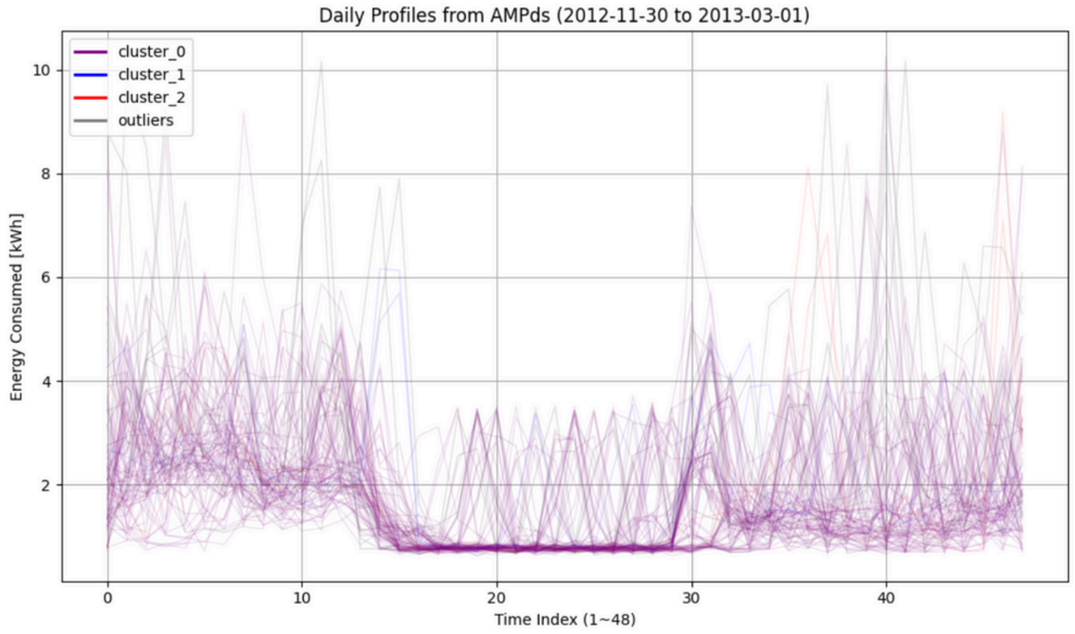
```python
# 하루 단위 reshape → 48차원 벡터 생성
daily_profiles = elec_df_30min_EDA2['Total_Power'].groupby(elec_df_30min_EDA2.index.date)\
    .apply(lambda x: x.values[:48] if len(x) >= 48 else None).dropna()
X = np.stack(daily_profiles.values)  # shape: (N_days, 48)

# DBSCAN 클러스터링
daily_total_consumption = X.sum(axis=1)
eps_val = daily_total_consumption.mean() * 0.1
db = DBSCAN(eps=eps_val, min_samples=2, metric='euclidean')
labels = db.fit_predict(X)
```

- 날짜별로 그룹화하여 전력 소비량을 하루 단위 48차원 벡터로 변환
- 평균 사용량의 10%를 기준으로 DBSCAN 클러스터링을 수행

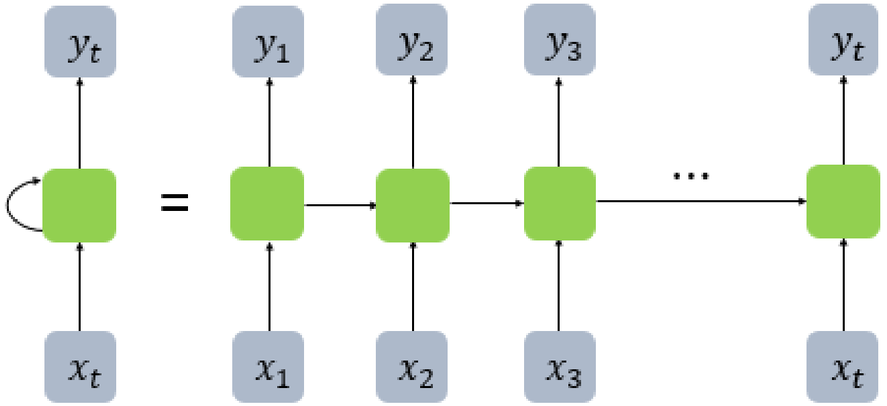
AMPds 데이터의 일일 전력 소비 패턴 클러스터링





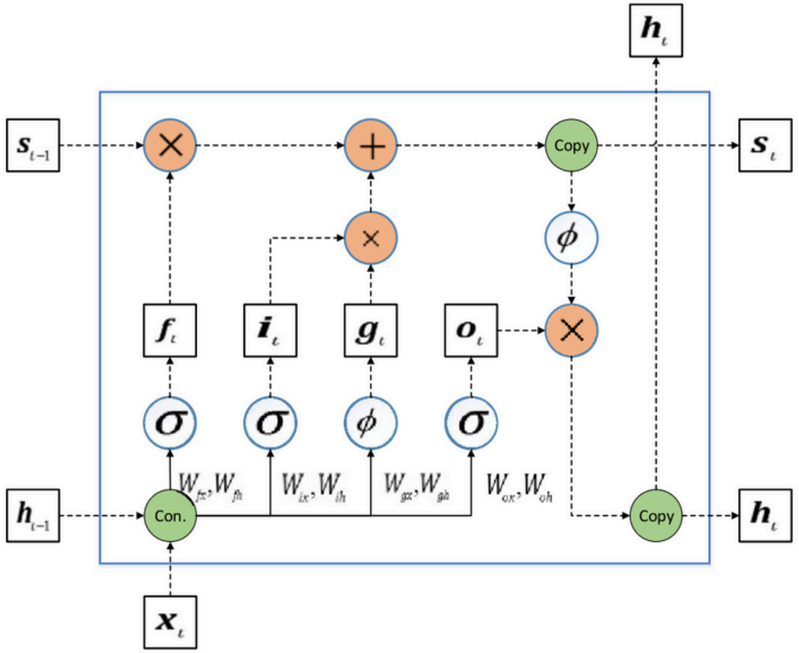
- 논문과 다른 데이터셋에서 EDA를 진행, 클러스터링 가능
- 보라색의 주요 생활 루틴을 가짐

RNN



- 입력 x_t와 이전 시점의 hidden state를 받아 현재 시점의 hidden state와 출력 y_t 계산
- BPTT → Vanishing Gradient Problem

LSTM



- LSTM은 RNN의 장기 기억 문제를 해결하기 위해 도입
- 게이트와 셀 상태를 도입한 구조로, 긴 시퀀스를 안정적으로 학습할 수 있음

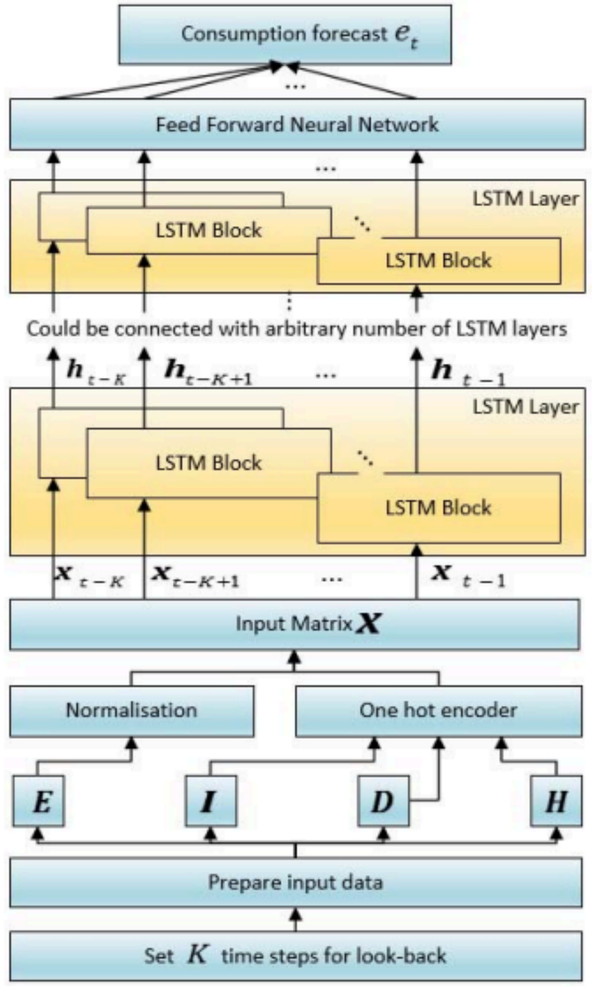
LSTM

```python
class LSTMForecast(nn.Module):
    def __init__(self, input_size=58, hidden_size=20, num_layers=2):
        super(LSTMForecast, self).__init__()

        self.lstm = nn.LSTM(
            input_size=input_size,    # 각 타임스텝당 feature 수 (예: 58차원 벡터)
            hidden_size=hidden_size,  # LSTM의 은닉 상태 차원 (출력 feature 수)
            num_layers=num_layers,    # LSTM 층의 개수 (stacked LSTM)
            batch_first=True          # 입력 텐서 shape을 (batch, seq_len, input_size)로 사용
        )

        self.fc = nn.Linear(hidden_size, 1) # LSTM 출력 → Fully Connected Layer

    def forward(self, x):
        out, _ = self.lstm(x)        # LSTM 출력: (batch_size, seq_len, hidden_size)
        out = out[:, -1, :]          # 마지막 시점 출력만 추출: (batch_size, hidden_size)
        out = self.fc(out)           # 최종 출력: (batch_size, 1)
        return out
```



- **Input: Elec. Index. Day. Holiday**
- **Model: LSTM → LSTM → FC → Output**

학습

- **Train Loss Function: MSE**
- **Test Loss Function: MAPE**

5회 반복 실험

	Mean (%)	Std (%)
Keras	43.95	2.48
Pytorch	43.31	2.18

Method/Scenario	*Avg. MAPE individual forecasts*
LSTM/2 time steps	**44.39 %**
LSTM/6 time steps	**44.31%**
LSTM/12 time steps	**44.06%**
Empirical mean	136.46%
MAPE minimisation	**46.00%**
BPNN-D/1 day	80.02%
BPNN-D/2 days	75.28%
BPNN-D/3 days	74.10%
BPNN-T/2 time steps	49.62%
BPNN-T/6 time steps	49.04%
BPNN-T/12 time steps	49.49%
KNN/2 time steps	74.83%
KNN/6 time steps	71.19%
KNN/12 time steps	81.13%
ELM/2 time steps	122.90%
ELM/6 time steps	136.49%
ELM/12 time steps	123.45%
IS-HF	96.76%

Kaggle Link

https://www.kaggle.com/code/hansupport/electricity-prediction-with-lstm-model?scriptVersionId=254440169

감사합니다