Report

Project 2: Smart Energy with Optimization Algorithms

Author: Rohit Rawat

112963417

Task 1: Solve the offline optimization problem, e.g., using tools CVX in Matlab or Python.

- Used CVXOPT library of python to calculate the offline optimization problem.
- True values are used to calculate the cost function and then took minimum cost function out of it using CVXOPT library.
- This is going to be the **datum** on which other algorithms are going to be judged.

Part1: Static solution

Fixed provision value i.e. x was used.

Part2: Dynamic solution

x varies as x (1), x (2), …, x (672).

```
For home B:
Offline Optimization Problem solution for static is:  1018.4013103622867
Offline Optimization Problem solution for dynamic is:  836.0104660550957
For home C:
Offline Optimization Problem solution for static is:  1365.6704999799842
Offline Optimization Problem solution for dynamic is:  1137.486313296603
For home F:
Offline Optimization Problem solution for static is:  6286.343226100506
Offline Optimization Problem solution for dynamic is:  5232.1411726006945
```

Task 2(i): Online gradient descent (with different step size)

- Using different kind of step sizes and by calculating partial derivative of cost function w.r.t x (t) to optimize the solution, we have calculated an optimized solution.
  Type of Step Sizes:

  step_size_type = 'A': 1/sqrt(t)

```
[53] print("For Home B,")
     OGD_results_B = OGD_online_op(homeB, p_t = 0.40, a =  4, b= 4,step_size_type = 'A')
     print("For Home C,")
     OGD_results_C = OGD_online_op(homeC, p_t = 0.40, a =  4, b= 4,step_size_type = 'A')
     print("For Home F,")
     OGD_results_F = OGD_online_op(homeF, p_t = 0.40, a =  4, b= 4,step_size_type = 'A')
```

```
For Home B,
OGD Optimization Problem solution is:  1892.720581127459
For Home C,
OGD Optimization Problem solution is:  2143.127367233312
For Home F,
OGD Optimization Problem solution is:  7521.536038487382
```

-

step_size_type = 'B': 1/(t)

```
print("For Home B,")
OGD_results_B = OGD_online_op(homeB, p_t = 0.40, a =  4, b= 4,step_size_type = 'B')
print("For Home C,")
OGD_results_C = OGD_online_op(homeC, p_t = 0.40, a =  4, b= 4,step_size_type = 'B')
print("For Home F,")
OGD_results_F = OGD_online_op(homeF, p_t = 0.40, a =  4, b= 4,step_size_type = 'B')
```

```
For Home B,
OGD Optimization Problem solution is:  1583.378957866783
For Home C,
OGD Optimization Problem solution is:  1614.6201205975656
For Home F,
OGD Optimization Problem solution is:  10516.319202234305
```

-

step_size_type = 'C': 0.01

```
[165] print("For Home B,")
OGD_results_B = OGD_online_op(homeB, p_t = 0.40, a =  4, b= 4,step_size_type = 'C')
print("For Home C,")
OGD_results_C = OGD_online_op(homeC, p_t = 0.40, a =  4, b= 4,step_size_type = 'C')
print("For Home F,")
OGD_results_F = OGD_online_op(homeF, p_t = 0.40, a =  4, b= 4,step_size_type = 'C')
```

```
For Home B,
OGD Optimization Problem solution is:  1244.8327133920006
For Home C,
OGD Optimization Problem solution is:  1771.6468845240013
For Home F,
OGD Optimization Problem solution is:  12635.242333375993
```

-

step_size_type = 'D': 1/672

```
print("For Home B,")
OGD_results_B = OGD_online_op(homeB, p_t = 0.40, a =  4, b= 4,step_size_type = 'D')
print("For Home C,")
OGD_results_C = OGD_online_op(homeC, p_t = 0.40, a =  4, b= 4,step_size_type = 'D')
print("For Home F,")
OGD_results_F = OGD_online_op(homeF, p_t = 0.40, a =  4, b= 4,step_size_type = 'D')
```

```
For Home B,
OGD Optimization Problem solution is:  1744.7828514939033
For Home C,
OGD Optimization Problem solution is:  2804.8918305921875
For Home F,
OGD Optimization Problem solution is:  17120.758057329414
```

-

- OGD is not a good optimizer use for time series data as this data is more dynamic like stock market data. OGD is better for static objective functions.

Comment about different Step size of OGD for different home:

For Home B, best step size is **step_size_type = 'C': 0.01** with **minimum cost = 1244.8327133920006**

For Home C, best step size is **step_size_type = 'B': 1/(t)** with **minimum cost = 1614.6201205975656**

For Home F, best step size is **step_size_type = 'A': 1/sqrt(t)** with **minimum cost = 7521.536038487382**

<u>Task 2(ii): Receding horizon control (with different prediction window size):</u>

- With **v = 1 and predictHorizon = window_size, fixed a = 4 and b = 4**, we have developed RHC function to optimize the cost function.
- Unlike OGD, this uses prediction values for computing. And it handles dynamic data better than OGD.
- **Prediction Algorithm 1: ARIMA**

For ARIMA Model Predictions:

```
window_size = [3,5,10]
for window in window_size:
  print("For window_size: ", window)
  [optimal_values_RHC_B, result_RHC_B] = control_algo_online_op(homeB, p_t = 0.40, a = 4, b = 4, predictionHorizon = window)
  print("For Home B, RHC Optimization Problem solution is: ", result_RHC_B)
  [optimal_values_RHC_C, result_RHC_C] = control_algo_online_op(homeC, p_t = 0.40, a = 4, b = 4, predictionHorizon = window)
  print("For Home C, RHC Optimization Problem solution is: ", result_RHC_C)
  [optimal_values_RHC_F, result_RHC_F] = control_algo_online_op(homeF, p_t = 0.40, a = 4, b = 4, predictionHorizon = window)
  print("For Home F, RHC Optimization Problem solution is: ", result_RHC_F)
```

```
For window_size:  3
For Home B, RHC Optimization Problem solution is:  750.3907458132975
For Home C, RHC Optimization Problem solution is:  989.8302788430145
For Home F, RHC Optimization Problem solution is:  4551.13984721234
For window_size:  5
For Home B, RHC Optimization Problem solution is:  750.3907458132975
For Home C, RHC Optimization Problem solution is:  989.8302788430145
For Home F, RHC Optimization Problem solution is:  4551.13984721234
For window_size:  10
For Home B, RHC Optimization Problem solution is:  750.3907458132975
For Home C, RHC Optimization Problem solution is:  989.8302788430145
For Home F, RHC Optimization Problem solution is:  4551.13984721234
```

- **Prediction Algorithm 2: Linear Regression**

For Linear Regression Model Predictions:

```
[172] window_size = [3,5,10]
      for window in window_size:
        print("For window_size: ", window)
        [optimal_values_RHC_B_LR, result_RHC_B_LR] = control_algo_online_op(homeB_LR, p_t = 0.40, a = 4, b = 4, predictionHorizon = window)
        print("For Home B, RHC Optimization Problem solution is: ", result_RHC_B_LR)
        [optimal_values_RHC_C_LR, result_RHC_C_LR] = control_algo_online_op(homeC_LR, p_t = 0.40, a = 4, b = 4, predictionHorizon = window)
        print("For Home C, RHC Optimization Problem solution is: ", result_RHC_C_LR)
        [optimal_values_RHC_F_LR, result_RHC_F_LR] = control_algo_online_op(homeF_LR, p_t = 0.40, a = 4, b = 4, predictionHorizon = window)
        print("For Home F, RHC Optimization Problem solution is: ", result_RHC_F_LR)
```

```
For window_size:  3
For Home B, RHC Optimization Problem solution is:  721.894394948552
For Home C, RHC Optimization Problem solution is:  1005.4272601940075
For Home F, RHC Optimization Problem solution is:  4414.920796928689
For window_size:  5
For Home B, RHC Optimization Problem solution is:  721.894394948552
For Home C, RHC Optimization Problem solution is:  1005.4272601940075
For Home F, RHC Optimization Problem solution is:  4414.920796928689
For window_size:  10
For Home B, RHC Optimization Problem solution is:  721.894394948552
For Home C, RHC Optimization Problem solution is:  1005.4272601940075
For Home F, RHC Optimization Problem solution is:  4414.920796928689
```

- Best window size with different prediction algorithms:

Comment about Best window_size of RHC:

For ARIMA Predictions:

For window_size: 5

For Home B, RHC Optimization Problem solution is: 750.3907458132975

For Home C, RHC Optimization Problem solution is: 989.8302788430145

For Home F, RHC Optimization Problem solution is: 4551.13984721234

For Linear Regression Predictions:

For window_size: 5

For Home B, RHC Optimization Problem solution is: 721.894394948552

For Home C, RHC Optimization Problem solution is: 1005.4272601940075

For Home F, RHC Optimization Problem solution is: 4414.920796928689

Task 2(iii): Commitment Horizon control (with different commitment levels):

- Additional to RHC parameters, CHC has another parameter which we handle with i.e **commitment level, v.** We have tried different commitment levels for all the home.
- **Prediction Algorithm 1: ARIMA**

For ARIMA Model Predictions:

```
commitmentHorizon = [3,4,5]
for commitment in commitmentHorizon:
  print("For v: ", commitment)
  [optimal_values_CHC_B, result_CHC_B] = control_algo_online_op(homeB, p_t = 0.40, a = 4, b = 4, predictionHorizon = 5, commitmentHorizon= commitment)
  print("For Home B, CHC Optimization Problem solution is: ", result_CHC_B)
  [optimal_values_CHC_C, result_CHC_C] = control_algo_online_op(homeC, p_t = 0.40, a = 4, b = 4, predictionHorizon = 5, commitmentHorizon= commitment)
  print("For Home C, CHC Optimization Problem solution is: ", result_CHC_C)
  [optimal_values_CHC_F, result_CHC_F] = control_algo_online_op(homeF, p_t = 0.40, a = 4, b = 4, predictionHorizon = 5, commitmentHorizon= commitment)
  print("For Home F, CHC Optimization Problem solution is: ", result_CHC_F)
```

```
For v:  3
For Home B, CHC Optimization Problem solution is:  750.3907458133029
For Home C, CHC Optimization Problem solution is:  989.8302788430127
For Home F, CHC Optimization Problem solution is:  4551.139847212367
For v:  4
For Home B, CHC Optimization Problem solution is:  750.3907458133042
For Home C, CHC Optimization Problem solution is:  989.8302788430152
For Home F, CHC Optimization Problem solution is:  4551.139847212363
For v:  5
For Home B, CHC Optimization Problem solution is:  750.3907458133061
For Home C, CHC Optimization Problem solution is:  989.8302788430143
For Home F, CHC Optimization Problem solution is:  4551.139847212366
```

- **Prediction Algorithm 2: Linear Regression**

For Linear Regression Model Predictions:

```
[174] commitmentHorizon = [3,4,5]
for commitment in commitmentHorizon:
  print("For v: ", commitment)
  [optimal_values_CHC_B_LR, result_CHC_B_LR] = control_algo_online_op(homeB_LR, p_t = 0.40, a = 4, b = 4, predictionHorizon = 5, commitmentHorizon= 3)
  print("For Home B, CHC Optimization Problem solution is: ", result_CHC_B_LR)
  [optimal_values_CHC_C_LR, result_CHC_C_LR] = control_algo_online_op(homeC_LR, p_t = 0.40, a = 4, b = 4, predictionHorizon = 5, commitmentHorizon= 3)
  print("For Home C, CHC Optimization Problem solution is: ", result_CHC_C_LR)
  [optimal_values_CHC_F_LR, result_CHC_F_LR] = control_algo_online_op(homeF_LR, p_t = 0.40, a = 4, b = 4, predictionHorizon = 5, commitmentHorizon= 3)
  print("For Home F, CHC Optimization Problem solution is: ", result_CHC_F_LR)
```

```
For v:  3
For Home B, CHC Optimization Problem solution is:  721.8943949485557
For Home C, CHC Optimization Problem solution is:  1005.4272601940361
For Home F, CHC Optimization Problem solution is:  4414.920796928767
For v:  4
For Home B, CHC Optimization Problem solution is:  721.8943949485557
For Home C, CHC Optimization Problem solution is:  1005.4272601940361
For Home F, CHC Optimization Problem solution is:  4414.920796928767
For v:  5
For Home B, CHC Optimization Problem solution is:  721.8943949485557
For Home C, CHC Optimization Problem solution is:  1005.4272601940361
For Home F, CHC Optimization Problem solution is:  4414.920796928767
```

- CHC is better than RHC as it keeps an average at a commitment level which helps this control algorithm to handle dynamic data better.

Comment about Best Commitment Horizon for CHC:

For ARIMA Predictions:

For v: 3

For Home B, CHC Optimization Problem solution is: 750.3907458133029

For Home C, CHC Optimization Problem solution is: 989.8302788430127

For Home F, CHC Optimization Problem solution is: 4551.139847212367

For Linear Regression Predictions:

For v: 3

For Home B, CHC Optimization Problem solution is: 721.8943949485557

For Home C, CHC Optimization Problem solution is: 1005.4272601940361

For Home F, CHC Optimization Problem solution is: 4414.920796928767

Task 3: Compare the costs of these algorithms to those of the offline static and dynamic solutions: Calculating Regret Factor

- We have used Regret value function to compare different algorithms which is calculate by as following:

  Static Regret = Cost of the Algorithm - Static Offline Cost

  Dynamic Regret = Cost of the Algorithm - Dynamic offline Cost

- **Regrets for OGD:**

```
For Home B, static Regret:  226.43140302971392
For Home C, static Regret:  248.94962061758133
For Home F, static Regret:  1235.1928123868756
For Home B, dynamic Regret:  408.822247336905
For Home C, dynamic Regret:  477.13380730096264
For Home F, dynamic Regret:  2289.3948658866875
```

- **Regrets for RHC for ARIMA:**

```
For Home B, static Regret:  -268.01056454898924
For Home C, static Regret:  -375.8402211369697
For Home F, static Regret:  -1735.2033788881663
For Home B, dynamic Regret:  -85.61972024179818
For Home C, dynamic Regret:  -147.6560344535884
For Home F, dynamic Regret:  -681.0013253883544
```

- **Regrets for RHC for Linear Regression:**

```
For Home B, static Regret:  -296.50691541373476
For Home C, static Regret:  -360.2432397859767
For Home F, static Regret:  -1871.4224291718174
For Home B, dynamic Regret:  -114.1160711065437
For Home C, dynamic Regret:  -132.05905310259539
For Home F, dynamic Regret:  -817.2203756720055
```

- **Regrets for CHC for ARIMA:**

```
For Home B, static Regret:  -268.0105645489806
For Home C, static Regret:  -375.84022113696994
For Home F, static Regret:  -1735.2033788881408
For Home B, dynamic Regret:  -85.61972024178954
For Home C, dynamic Regret:  -147.65603445358863
For Home F, dynamic Regret:  -681.001325388329
```

- **Regrets for CHC for Linear Regression:**

```
For Home B, static Regret:  -296.506915413731
For Home C, static Regret:  -360.24323978594816
For Home F, static Regret:  -1871.4224291717392
For Home B, dynamic Regret:  -114.11607110653995
For Home C, dynamic Regret:  -132.05905310256685
For Home F, dynamic Regret:  -817.2203756719273
```

<u>Task 4: For the best combination of control algorithm and prediction algorithm, vary a and b to see the impacts:</u>

- Best combination of control algorithm and prediction algorithm is determined in this task and then vary the a and b values to calculate the cost function.

```
A = [4, 0.1, 1, 5, 10]
B = [4, 0.1, 1, 5, 10]
```

```
for ai in A:
  for bi in B:
    print("For a = ",ai,"and b = ",bi,":")
    [_, result_B] = control_algo_online_op(homeB_LR, p_t = 0.40,  a = ai,  b = bi, predictionHorizon = 5, commitmentHorizon=3)  #Linear Regression predictions with CHC
    print("For Home B, Result of CHC with LR(w=5,v=3): ", result_B)
    [_, result_C] = control_algo_online_op(homeC, p_t = 0.40, a = ai,  b = bi, predictionHorizon = 5, commitmentHorizon=3)     #ARIMA predictions with CHC
    print("For Home C, Result of CHC with ARIMA(w=5,v=3): ", result_C)
    [_, result_F] = control_algo_online_op(homeF_LR, p_t = 0.40, a = ai,  b = bi, predictionHorizon = 5, commitmentHorizon=3)  #Linear Regression predictions with CHC
    print("For Home F, Result of CHC with LR(w=5,v=3): ", result_F)
```

- Best for home B:

  For a = 4 and b = 0.1:
  For Home B, Result of CHC with Linear Regression (w=5, v=3): 420.8331773104108

- Best for home C:

  For a = 4 and b = 0.1:
  For Home C, Result of CHC with ARIMA(w=5,v=3): 609.9011831912904

- Best for home F:

  For a = 4 and b = 0.1:
  For Home F, Result of CHC with LR(w=5,v=3): 2238.5675645420356

Task 5: Try at least two algorithm selection (one deterministic, one randomized) to see if their performance

- **Deterministic Algorithm Selection: Weighted Majority with 4 tests**
  - We have initially assigned 0.33 weight to all the three algorithms.
  - We have taken four kind of test in that on different iterations.
  - On every iteration we have increased the weight of algorithm with minimum cost function by 0.10 and decreased the weight of maximum cost with 0.10.
  - Test 1 : Home B with static regret
    Test 2: Home B with dynamic regret
    Test 3: Inverted Home B with static regret
    Test 4: Inverted Home B with dynamic regret
  - After 4 iterations we get the results as follows:

    ```
    Algorithm Selection Using Weighted Majority:
    0.63*OGD + 0.73*RHC + 0.33*CHC
    We choose RHC as it has maximum weight:  0.73
    ```

- **Randomized Algorithm Selection: Most winning algorithm is chosen**
  - Like Deterministic selection above, I have taken 4 tests but the test get chosen using random number generator.
  - The Algorithm which has minimum regret won the round.
  - We calculate the winnings of all the algorithms.

    ```
    Algorithm Selection Using Randomized Maximum Winning
    Winning of OGD 0
    Winning of RHC 4
    Winning of CHC 0
    ```

    So, RHC wins this selection.

Bonus Task 1: Online Balanced Descent Algorithm:

- OBD is one step ahead of OGD as its step sizes varies with time.
  Reference: https://arxiv.org/pdf/1803.10366.pdf
  We have already implemented OBD above.

```
result_OBD_B = OBD_online_op(homeB)
result_OBD_C = OBD_online_op(homeC)
result_OBD_F = OBD_online_op(homeF)
```

```
OGD Optimization Problem solution is:  1892.720581127459
OGD Optimization Problem solution is:  2143.127367233312
OGD Optimization Problem solution is:  7521.536038487382
```

-
- **Regrets of OBD:**

Regrets of OBD:

```
193] static_regret_OGD_B = result_OBD_B - offline_static_result_B
     static_regret_OGD_C = result_OBD_C - offline_static_result_C
     static_regret_OGD_F = result_OBD_F - offline_static_result_F
     print("For Home B, static Regret: ", static_regret_OGD_B)
     print("For Home C, static Regret: ", static_regret_OGD_C)
     print("For Home F, static Regret: ", static_regret_OGD_F)
     dynamic_regret_OGD_B = result_OBD_B - offline_dynamic_result_B
     dynamic_regret_OGD_C = result_OBD_C - offline_dynamic_result_C
     dynamic_regret_OGD_F = result_OBD_F - offline_dynamic_result_F
     print("For Home B, dynamic Regret: ", dynamic_regret_OGD_B)
     print("For Home C, dynamic Regret: ", dynamic_regret_OGD_C)
     print("For Home F, dynamic Regret: ", dynamic_regret_OGD_F)
```

```
For Home B, static Regret:  874.3192707651722
For Home C, static Regret:  777.4568672533276
For Home F, static Regret:  1235.1928123868756
For Home B, dynamic Regret:  1056.7101150723634
For Home C, dynamic Regret:  1005.6410539367089
For Home F, dynamic Regret:  2289.3948658866875
```

Bonus Task2: Another Algorithm selection Method

- We have created a randomize selection algorithm for Algorithm selection which runs for 1000 trials.

Another Algorithm selection Method

Based on Regret Factor as scale, we can generate new algorithm Selector. In this **Randomized method**, we allocate few keys to all the three algorithms and then we generate a random number if that matches with the key of the algorithm then we going to increase the wins for that algorithm.

```
[194] #OGD = 0, #OBD = 1 #RHC = 2 #CHC  = 3
      rand_dict = {'OGD': 0, 'OBD': 0, 'RHC':0,'CHC' : 0 }
```

```
i = 1
while(i<1000):
  random_number = np.random.randint(4)
  if random_number == 0:
    rand_dict['OGD'] += 1
  elif random_number == 1:
    rand_dict['OBD'] += 1
  elif random_number == 2:
    rand_dict['RHC'] += 1
  else:
    rand_dict['CHC'] += 1
  i += 1
print(rand_dict)
```

```
{'OGD': 287, 'OBD': 296, 'RHC': 330, 'CHC': 288}
```

So, after 1000 trials, RHC got maximum wins. Thus, it will be chosen randomly.