B2 python ver

Lee SungHo

2021-01-04

⚜ ⚜

page 4 Implementation

```
import numpy as np

for X in range(11,16):
    MC_N = 10000
    D = np.random.uniform(11,15,MC_N) # random discrete uniform
    sales_rev= 2*np.minimum(D,X) # vector level minimum
    salvage_rev = 0.5*np.maximum(X-D,0) # vector level minimum
    material_cost = 1*X
    profit= sales_rev + salvage_rev - material_cost
    print("X: ",X,", expected profit: ",np.mean(profit))
```

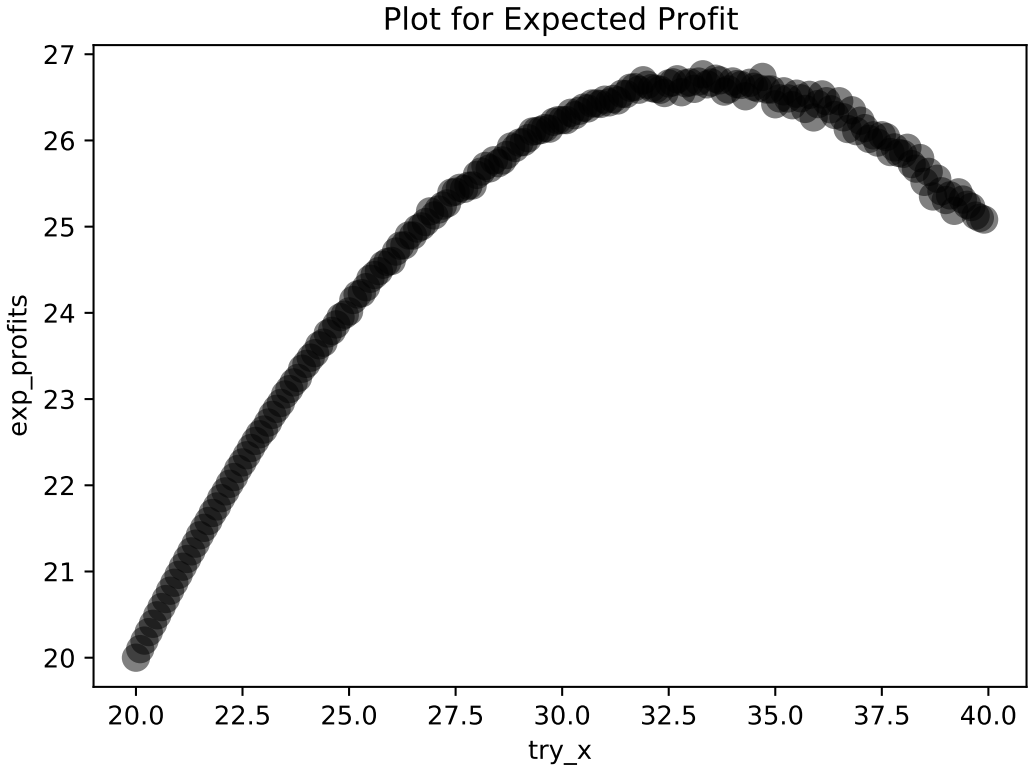
```
## X:  11 , expected profit:  11.0
## X:  12 , expected profit:  11.819499216537212
## X:  13 , expected profit:  12.245488279228807
## X:  14 , expected profit:  12.298666388840104
## X:  15 , expected profit:  12.003373032357251
```

page 5 continuous distribution - grid search approach

```
import pandas as pd

try_X = np.arange(20,40,0.1)
exp_profits=np.array([])

for X in try_X:
    MC_N = 10000
    D = np.random.rand(MC_N,1)*20+20
    sales_rev = 2*np.minimum(D,X)
    salvage_rev = 0.5*np.maximum(X-D,0)
    material_cost = 1*X
    exp_profit = np.mean(sales_rev + salvage_rev - material_cost)
    exp_profits = np.append(exp_profits,exp_profit)

result = pd.DataFrame({'try_X':try_X,'exp_profits':exp_profits})
```

page 6 continue

```
import matplotlib.pyplot as plt

plt.plot(try_X, exp_profits, 'o', ms=10, alpha=0.5, color = 'black' )
plt.xlabel("try_x")
plt.ylabel("exp_profits")
plt.title("Plot for Expected Profit")
plt.rc('font', size=12)
plt.show()
```



page 8 continue

```
idx=np.argmax(exp_profits)
print(try_X[idx]) #this is optimal quantity
```

```
## 33.30000000000019
```

```
print(exp_profits[idx]) #this is expected optimal profit
```

```
## 26.76760485833264
```

```
idx
```

```
## 133
```