B2 Python Code

Kang, Eui Hyeon

2021-01-03

차 례

Implementation (14p)

```
for X in range(11,16):
    MC_N=10000
    D=np.random.choice(np.arange(11,16),MC_N,replace=True) # random discrete uniform

    sales_rev=2*np.minimum(D,X) # vector level minimum
    salvage_rev=0.5*np.maximum(X-D,0) # vector level maximum
    material_cost=1*X

    profit=sales_rev+salvage_rev-material_cost

    print('X: ',X,' expected profit: ',np.mean(profit))
```

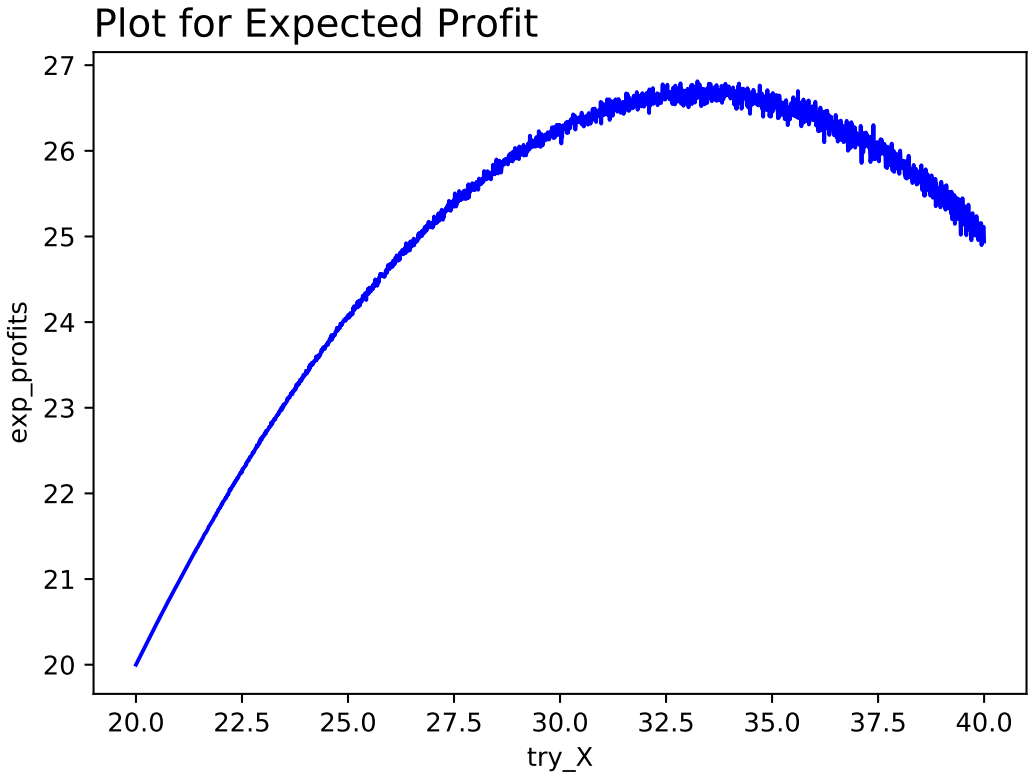
```
## X:  11  expected profit:  11.0
## X:  12  expected profit:  11.6967
## X:  13  expected profit:  12.0886
## X:  14  expected profit:  12.1544
## X:  15  expected profit:  12.0066
```

Continuous distribution - grid search approach (5-10p)

```
try_X=np.arange(20,40.01,step=0.01)
exp_profits=list()

for X in try_X:
    MC_N=10000
    D=np.random.uniform(20,40,size=MC_N)

    sales_rev=2*np.minimum(D,X) # vector level minimum
    salvage_rev=0.5*np.maximum(X-D,0) # vector level maximum
    material_cost=1*X

    exp_profit=np.mean(sales_rev+salvage_rev-material_cost)
    exp_profits.append(exp_profit)

try_X=try_X.reshape(-1,1)
exp_profits=np.asarray(exp_profits).reshape(-1,1)
results=pd.DataFrame(np.concatenate((try_X,exp_profits),axis=1), columns=['try_X','exp_profits'])

results.head()
```

```
##    try_X  exp_profits
## 0  20.00    20.000000
## 1  20.01    20.009994
## 2  20.02    20.019989
## 3  20.03    20.029968
## 4  20.04    20.039927
```

```
plt.plot(try_X,exp_profits,color='blue')
plt.title('Plot for Expected Profit', loc='left',fontsize=15)
plt.xlabel('try_X')
plt.ylabel('exp_profits')
plt.show()
```




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

```
## [33.24]
```

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

```
## [26.81208676]
```