hw5_Xueqi Wei_Queenie Liu

December 1, 2019

[8]:
```python
import numpy as np
import matplotlib.pyplot as plt
import pprint
```

0.1 1

a)

The original function now is

$$V_i(x) = max\{\sum_{k=1}^{4} p_k\delta_k k + (\sum_{k=1}^{4} p_k\delta_k)V_{i+1}(x-1) + (1 - \sum_{k=1}^{4} p_k\delta_k)V_{i+1}(x)\}$$

We have $\sum_{k=1}^{4} p_k = 1$ When $\delta_k = 1$, the function is:

$$V_i(x) = max\{\sum_{k=1}^{4} p_k k + (\sum_{k=1}^{4} p_k)V_{i+1}(x-1) + 0\}$$

When $\delta_k = 0$, we have:

$$V_i(x) = max\{V_{i+1}(x)\}$$

We take $\sum_{k=1}^{4} p_k$ out and get

$$V_i(x) = \sum_{k=1}^{4} p_k max\{\delta_k k + (\delta_k)V_{i+1}(x-1) + (1 - \delta_k)V_{i+1}(x)\}$$

which can be written as:

$$V_i(x) = \sum_{k=1}^{4} p_k max\{\delta_k(k + V_{i+1}(x-1)) + (1 - \delta_k)V_{i+1}(x)\}$$

The new function would be:

$$V_i(x) = \sum_{k=1}^{4} p_k max\{k + V_{i+1}(x-1), V_{i+1}(x)\}$$

Then in the new function, for each $k = i \in 1, 2, 3, 4$, if $\delta_i = 0$, the "max" term in the function will select $V_{i+1}(x)$, and $k + V_{i+1}(x-1)$ otherwise($\delta_i = 1$), which is exactly the same as what we get in the adjusted original function.

b)

[3]:
```python
def finit_horizon(B,n):
    V = np.zeros((B + 1,n + 1))
    for j in range(1,n + 1):
        for i in range(1,B + 1):
            temp = 0
            for k in range(4):
                temp += 0.25*max(k + 1 + V[i-1][j-1],V[i][j-1])
            V[i][j] = temp
    return V
```

[9]:
```python
#i
budget = [10,20,30,40,50,60,70,80,90,100]
for B in budget:
    print('Budget level is: ',B)
    print('V is :',finit_horizon(B,100)[B][100])
```

```
Budget level is:  10
V is : 39.9999052915732
Budget level is:  20
V is : 79.66808259774882
Budget level is:  30
V is : 113.88312226804393
Budget level is:  40
V is : 143.98469879134097
Budget level is:  50
V is : 172.1945363788912
Budget level is:  60
V is : 193.98469879134103
Budget level is:  70
V is : 213.88312226804396
Budget level is:  80
V is : 229.66808259774882
Budget level is:  90
V is : 239.99990529157316
Budget level is:  100
V is : 250.0
```

ii)

There are 20 candidates remaining which means there are 5 of them with value 4(0.25 of the candidates have value 4). We will hire the 5 candidates with value 4 and will not hire the rest of them.

c)

If the budget is 30, my answer in b i) is 113.8 which is slightly lower than 115. 115 here is the upperbound of optimal value we can get since in static we are considering ideal situation.

0.2 2

a)

$$V(x) = max\{\sum_{k=1}^{4} p_k \delta_k k + \beta((\sum_{k=1}^{4} kp_k\delta_k)V(x-1) + (1 - \sum_{k=1}^{4} kp_k\delta_k)V(x))\}$$

Similar to Q1 a), the function can be written as:

$$V(x) = \sum_{k=1}^{4} p_k max\{k + \beta V(x-1), \beta V(x)\}$$

[28]:
```
def infinit_horizon(B,beta,n):
    V = np.zeros((B + 1,n + 1))
    for j in range(1,n + 1):
        for i in range(1,B + 1):
            temp = 0
            for k in range(4):
                temp += 0.25*max(k + 1 + beta*V[i-1][j-1],beta*V[i][j-1])
            V[i][j] = temp
    return V[B][n]
```

[32]:
```
infinit_horizon(50,0.95,2000)
```

[32]:
```
46.54625700071119
```

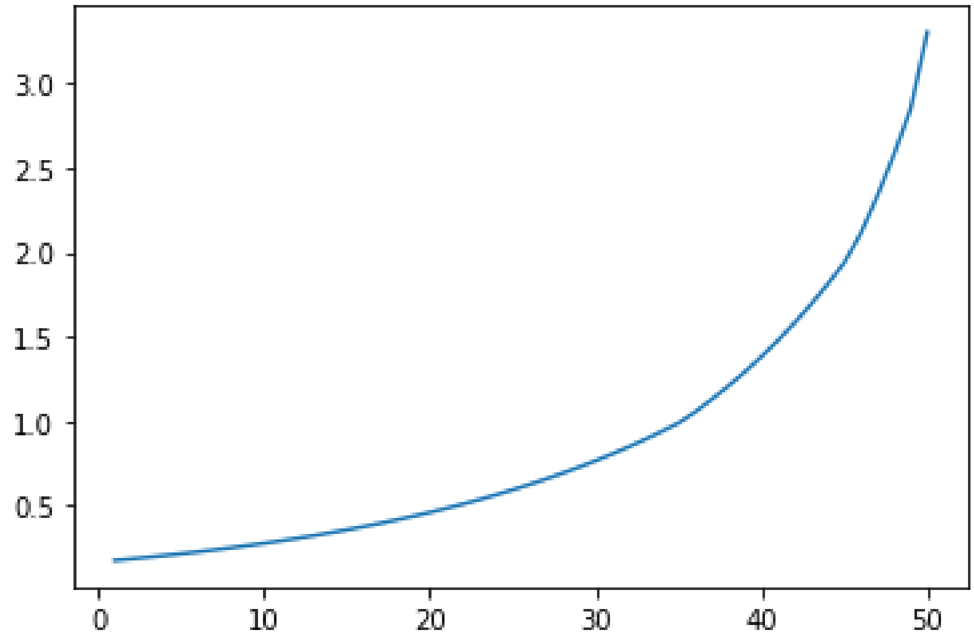
[35]:
```
def infinit_horizon_t(B,beta,n):
    V = np.zeros((B+1,n+1))
    t = np.zeros(B)
    for j in range(1,n + 1):
        for i in range(1,B + 1):
            temp = 0
            for k in range(4):
                temp += 0.25*max(k + 1 +beta*V[i-1][j-1],beta*V[i][j-1])
            t[i-1] = (beta*V[i][j-1] - beta*V[i-1][j-1])
            V[i][j] = temp
    return V,t
```

[39]:
```
V,t = infinit_horizon_t(50,0.95,1000)
print(np.arange(50,0,-1))
```

```
[50 49 48 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27
 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10  9  8  7  6  5  4  3
  2  1]
```

[41]:
```
plt.plot(np.arange(50,0,-1), t)
```

```
[41]: [<matplotlib.lines.Line2D at 0x10df1c4e0>]
```




```
[ ]:
```