Square Root Approximation

September 23, 2020

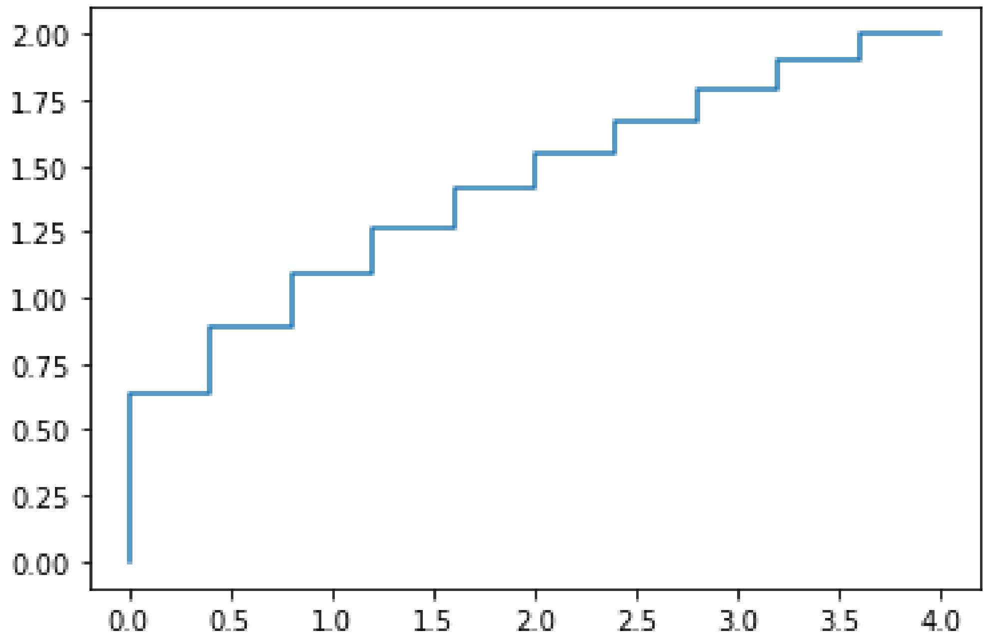
Idea: **Each cell in the middle layer will be active only if the input time(x) is greater than the bias. We calculate $\sqrt{0.1}$ for the first cell and make it the weight.**

If $0.2 > x \geq 0.3$, then first two cells will be active. Therefore weight for second cell will be $\sqrt{0.2} - \sqrt{0.1}$

Similarly for each next cell weight would be $\sqrt{n} - \sqrt{n - 0.1}$

[1]:
```
import numpy as np
from matplotlib import pyplot as plt

#Calculate root values for few inputs between 0-4
lst = np.linspace(0.0,4,11)
print("time",lst)
y=[]
for i in lst:
    y.append(i**0.5)

print("root",y)
plt.step(lst,y)
plt.show()
```

```
time [0.  0.4 0.8 1.2 1.6 2.  2.4 2.8 3.2 3.6 4. ]
root [0.0, 0.6324555320336759, 0.8944271909999159, 1.0954451150103324,
1.2649110640673518, 1.4142135623730951, 1.5491933384829668, 1.6733200530681511,
1.7888543819998317, 1.8973665961010275, 2.0]
```



0.1 Implementation

0.1.1 Calculate root(n) - root(n-0.1) for each consecutive cells

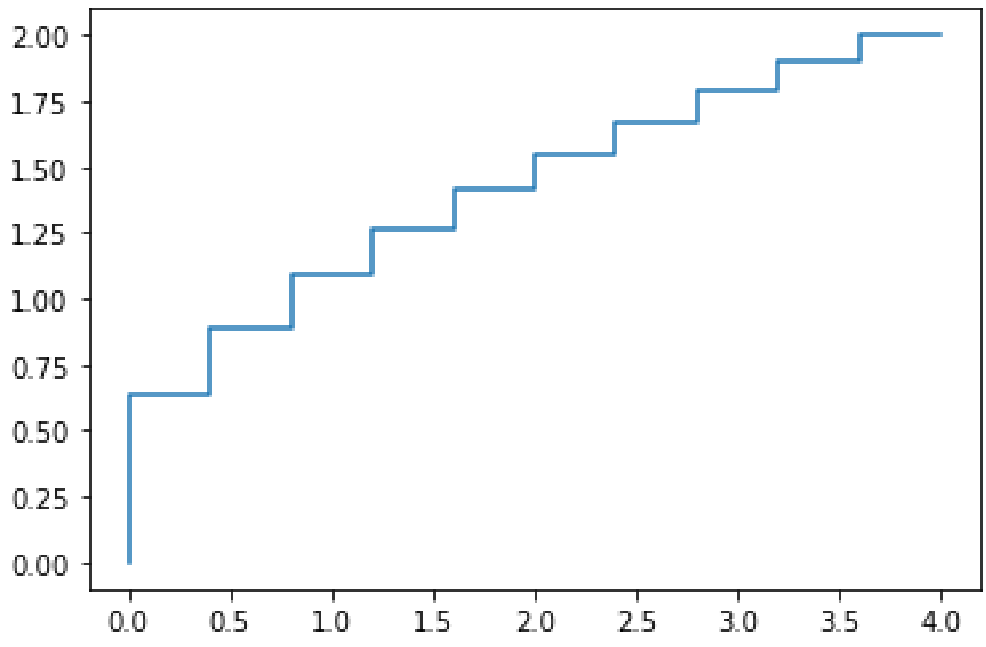
[2]:
```
wts = []
wts.append(y[0])
print("Square root \n", y)

indx = 0
for i in y[1:]:
    wts.append(i-y[indx])
    indx+=1

print("\nWts \n",wts)
```

```
Square root
 [0.0, 0.6324555320336759, 0.8944271909999159, 1.0954451150103324,
1.2649110640673518, 1.4142135623730951, 1.5491933384829668, 1.6733200530681511,
1.7888543819998317, 1.8973665961010275, 2.0]

Wts
 [0.0, 0.6324555320336759, 0.26197165896624, 0.20101792401041652,
0.1694659490570194, 0.14930249830574338, 0.13497977610987166,
0.12412671458518432, 0.11553432893168059, 0.10851221410119583,
0.10263340389897246]
```

0.1.2 Calculate output for the above network

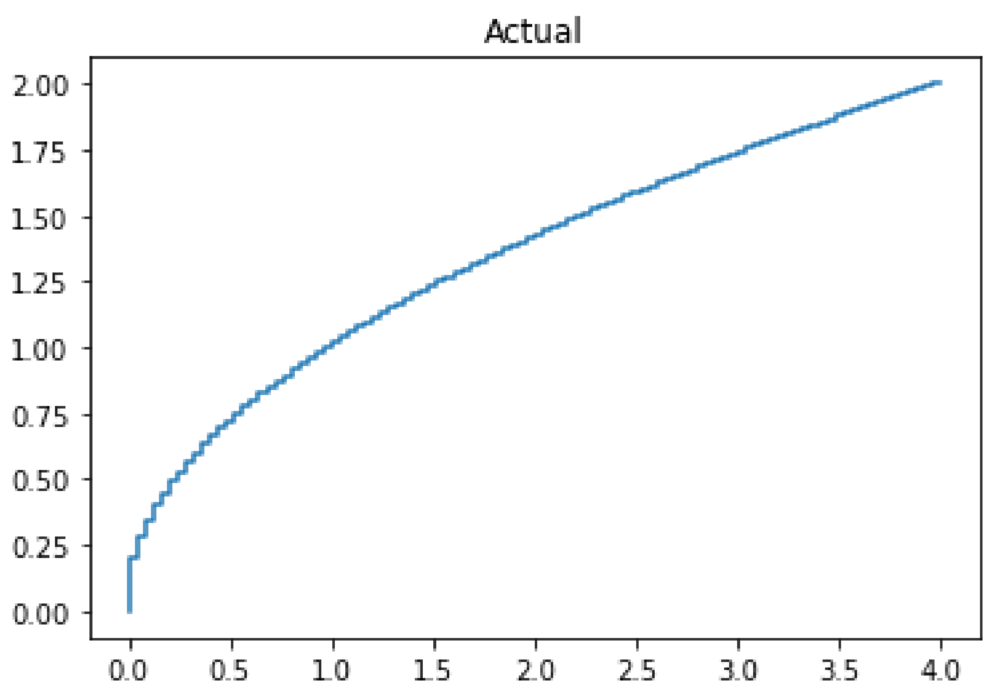
[3]:
```
res = []
inx=1
for i in wts:
    #For each input x, sum of all the active cells less than it will be the␣
 ↪output
    res.append(sum(wts[:inx]))
    inx+=1
print(res)
plt.step(lst,res)
plt.show()
```

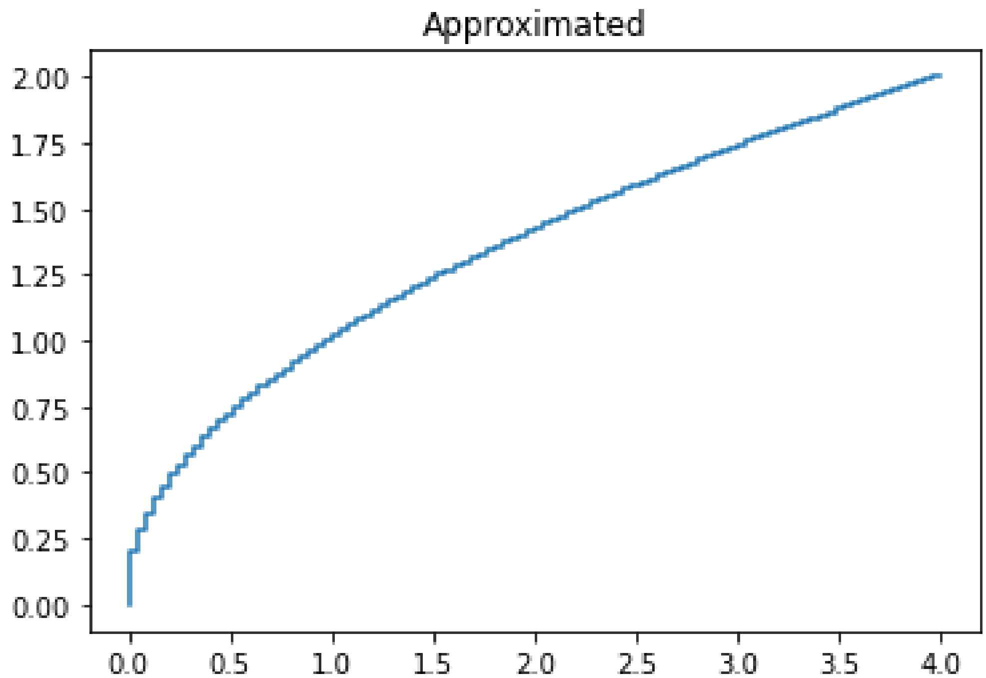
```
[0.0, 0.6324555320336759, 0.8944271909999159, 1.0954451150103324,
1.2649110640673518, 1.4142135623730951, 1.5491933384829668, 1.6733200530681511,
1.7888543819998317, 1.8973665961010275, 2.0]
```



[4]:
```
#Calculate root values for few inputs between 0-4
#Assumption we have few input and target values for training weights

lst = np.linspace(0.0,4,101)
y=[]
for i in lst:
    y.append(i**0.5)

plt.step(lst,y)
```

```
plt.title("Actual")
plt.show()
wts = []
wts.append(y[0])

indx = 0
for i in y[1:]:
    wts.append(i-y[indx])
    indx+=1


res = []
indx=1
for i in wts:
    #For each input x, sum of all the active cells less than it will be the␣
 ↪output
    res.append(sum(wts[:indx]))
    indx+=1

plt.step(lst,res)
plt.title("Approximated")
plt.show()
```




Approximated
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0