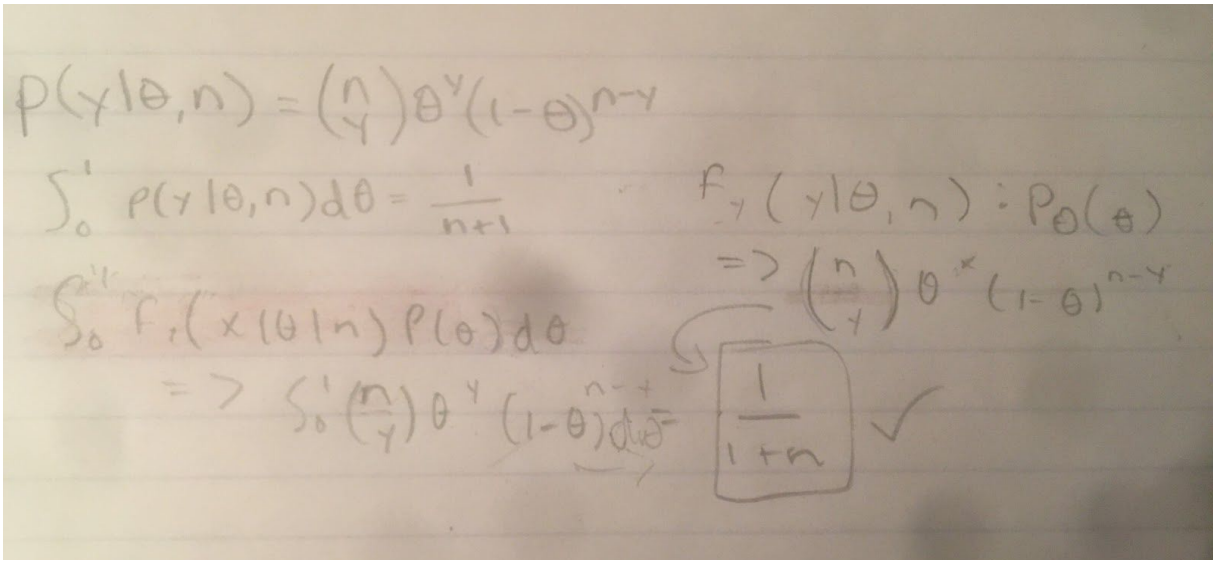
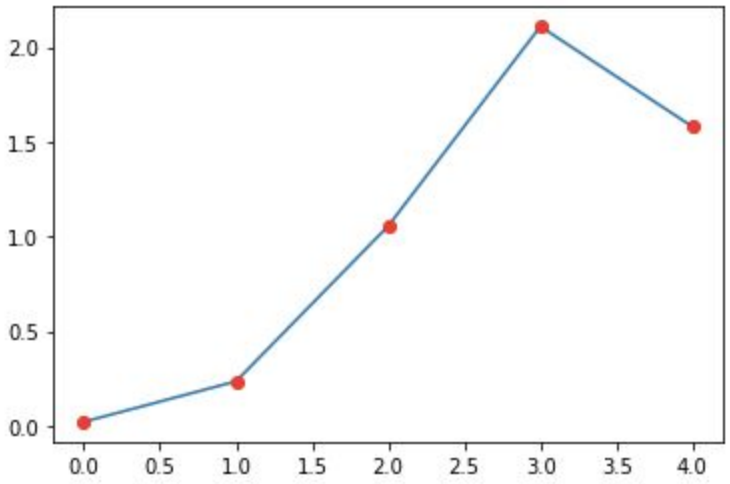
Cole Constantino
Cxc820
HW5

1) a.



b. (matplotlib)

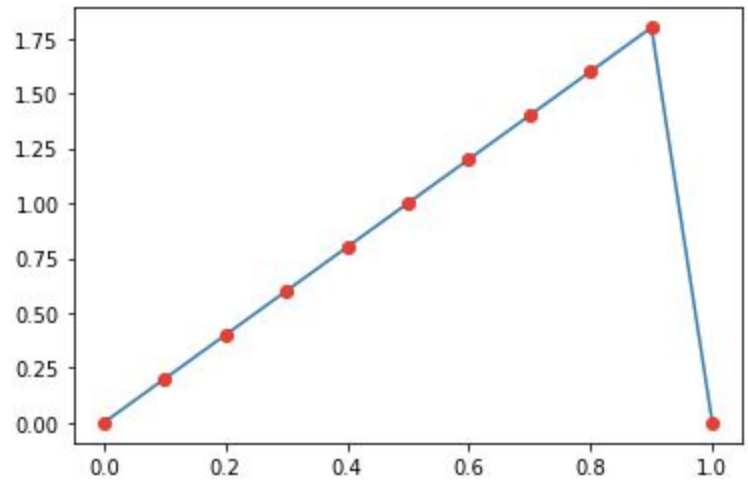
```
In [16]: runfile('/Users/coleconstantino/.spyder-py3/ter
coleconstantino/.spyder-py3')
```



c.

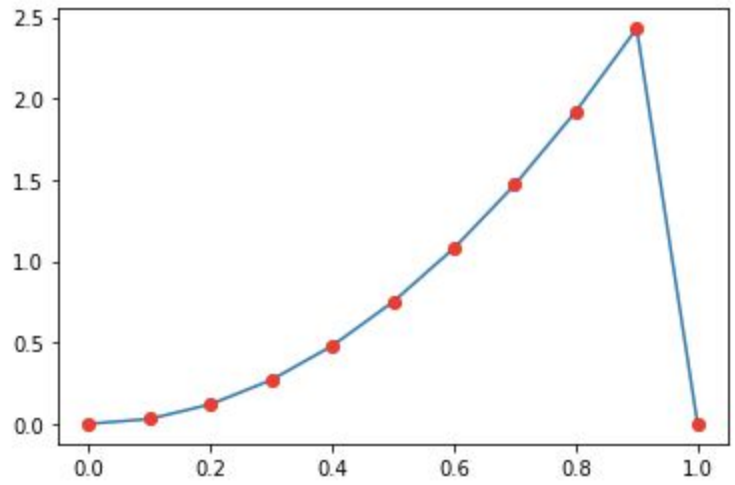
(n,heads) ⇒ (1,1)

```
In [18]: runfile('/Users/coleconstantino/.spyder-py3/t
coleconstantino/.spyder-py3')
```



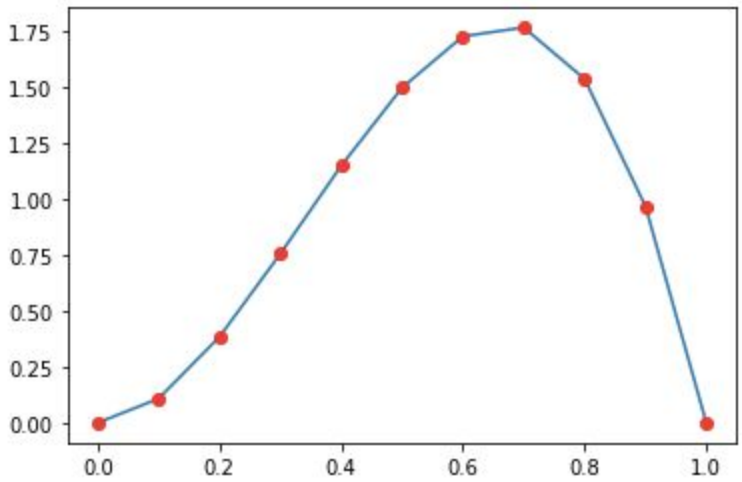
(n,heads) ⇒ (2,2)

```
In [19]: runfile('/Users/coleconstantino/.spyder-py3/te
coleconstantino/.spyder-py3')
```



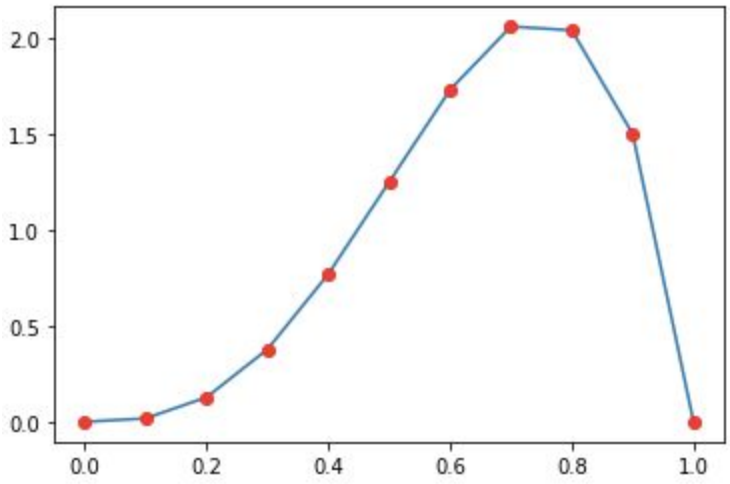
(n,heads) ⇒ (3,2)

```
coleconstantino/.spyder-py3')
```



(n,heads) ⇒ (3,3)

```
coleconstantino/.spyder-py3')
```



2) [Java Code will be attached in the zip file]

a.

For this problem I was able to figure out a way to get all the posterior values via JAVA and copy and pasted it to matplotlib in python. [I did this because I am very experienced in Java, and not so much python]

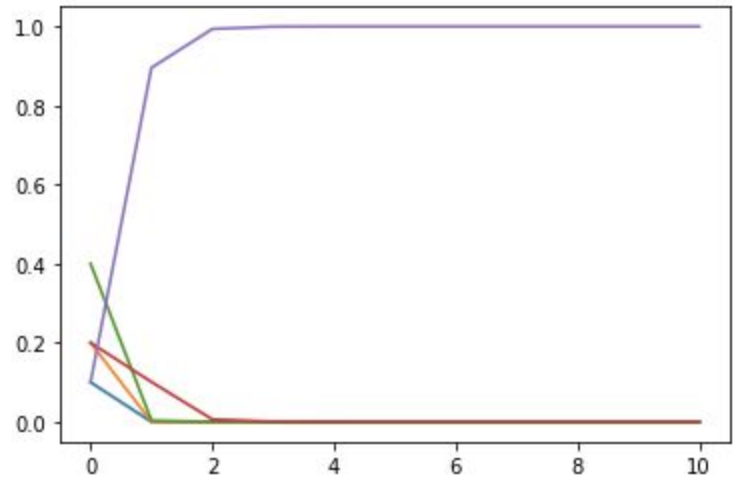
```
Welcome to DrJava.  Working directory is /Users/coleconstantino/Downloads
> run Question2A
h1 = [0.1, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]

h2 = [0.2, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]

h3 = [0.4, 0.0035, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]

h4 = [0.2, 0.1009, 0.0063, 4.0E-4, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]

h5 = [0.1, 0.8956, 0.9937, 0.9996, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]

>
```

Paste to python and plot:

```
import matplotlib.pyplot as plt

h1 = [0.1, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]

h2 = [0.2, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]

h3 = [0.4, 0.0035, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]

h4 = [0.2, 0.1009, 0.0063, 4.0E-4, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]

h5 = [0.1, 0.8956, 0.9937, 0.9996, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]


plt.plot(h1)
plt.plot(h2)
plt.plot(h3)
plt.plot(h4)
plt.plot(h5)
plt.show
```

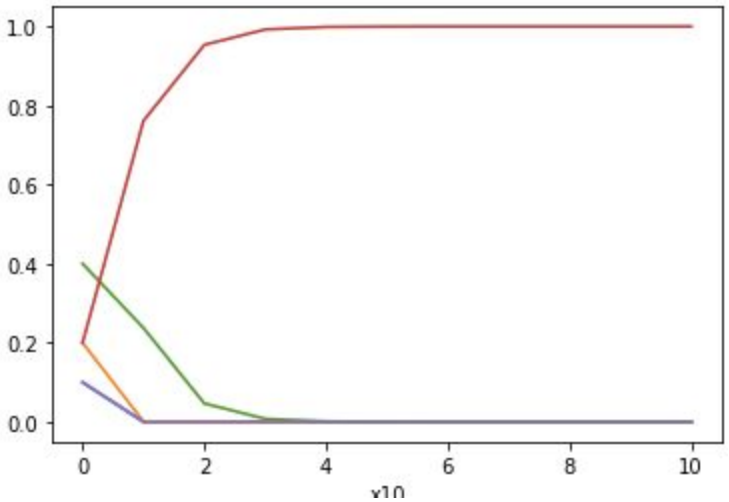
[h1]

```
In [8]: runfile('/Users/coleconstantino/.spyder-
py3/temp.py', wdir='/Users/coleconstantino/.spyder-
py3')
```



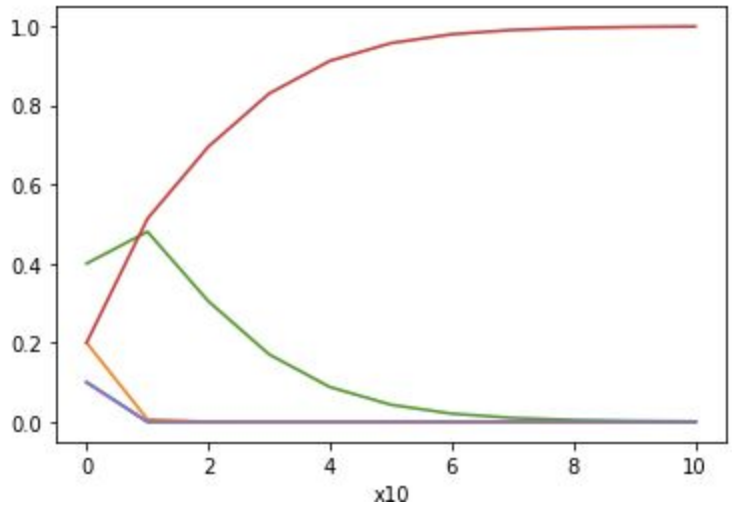
[h2]

```
In [19]: runfile('/Users/coleconstantino/.spyder-
py3/temp.py', wdir='/Users/coleconstantino/.spyder-
py3')
```



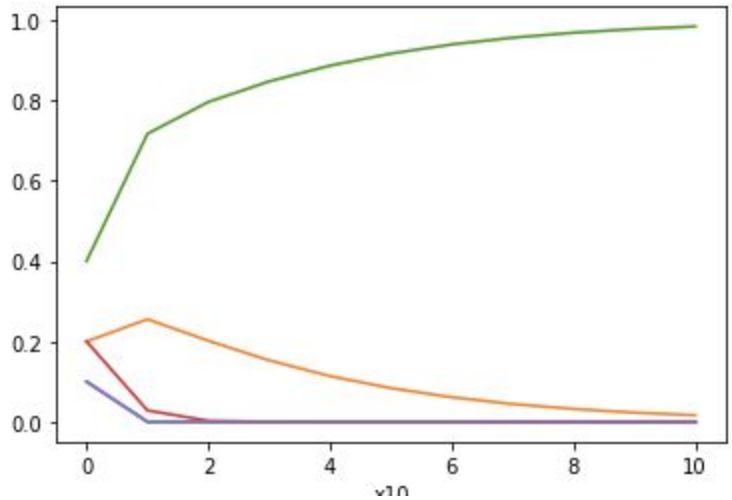
[h3]

```
In [20]: runfile('/Users/coleconstantino/.spyder-
py3/temp.py', wdir='/Users/coleconstantino/.spyder-
py3')
```



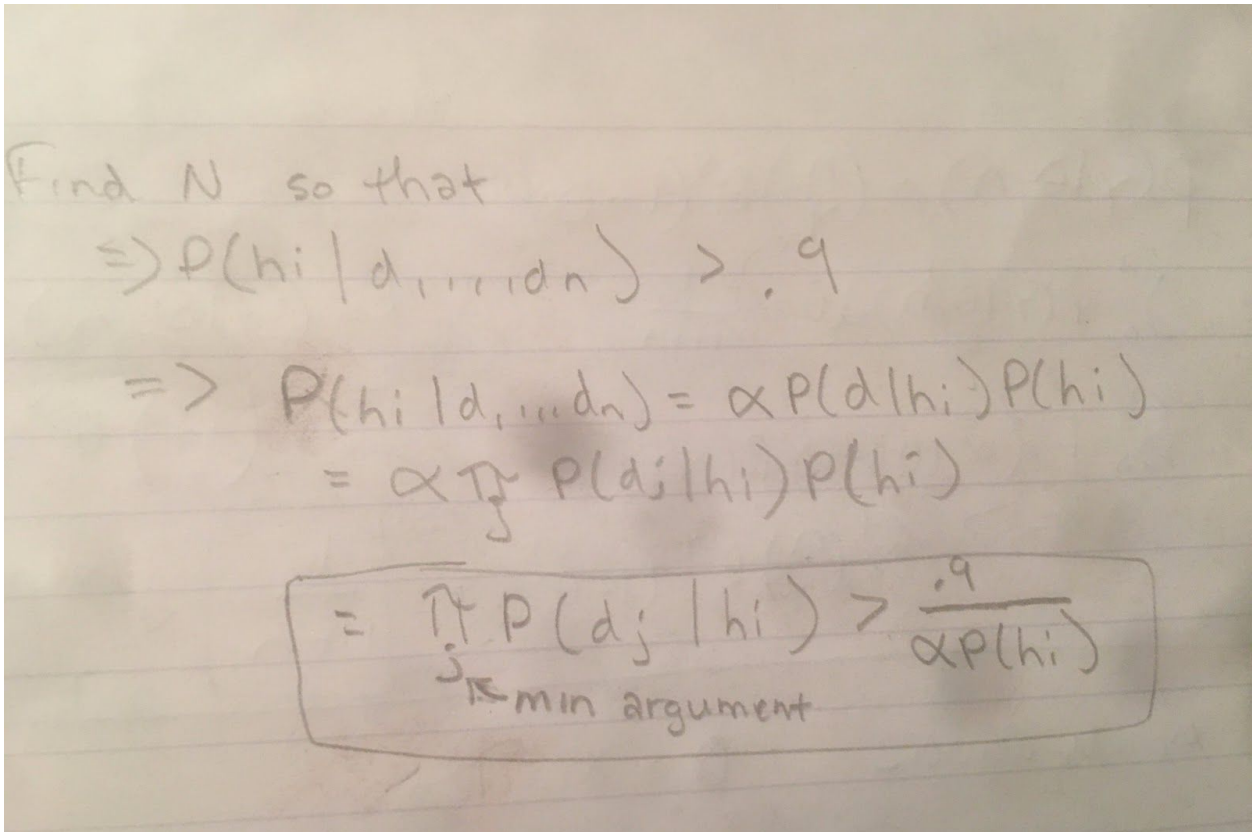
[h4]

```
In [21]: runfile('/Users/coleconstantino/.spyder-
py3/temp.py', wdir='/Users/coleconstantino/.spyder-
py3')
```



*And just like magic, you can see how the algorithm learns which hypothesis it is as n increases

b.

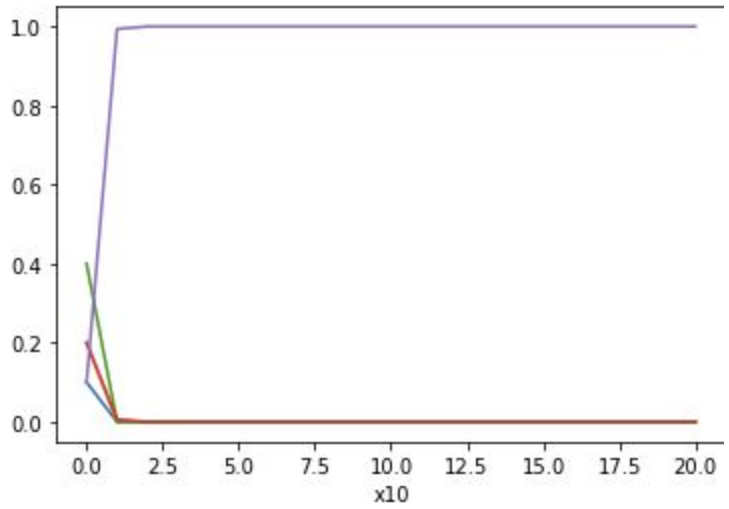


C. While part a) was very useful and very cool, you can see that the distributions are not as uniform as possible. A big reason for this is the sample size of probabilities. If I increase the

datasets and remove my decimal formatting parameters it will make these graphs a lot more precise.[Below are my plots]

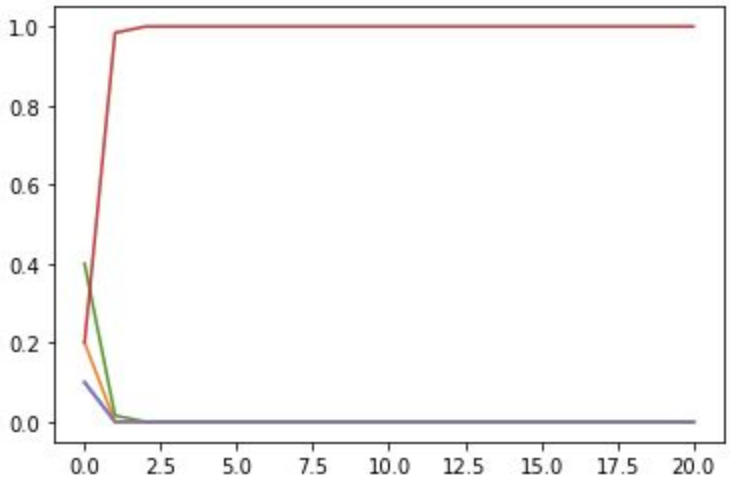
[h1]

```
In [24]: runfile('/Users/coleconstantino/.spyder-
py3/temp.py', wdir='/Users/coleconstantino/.spyder-
py3')
```



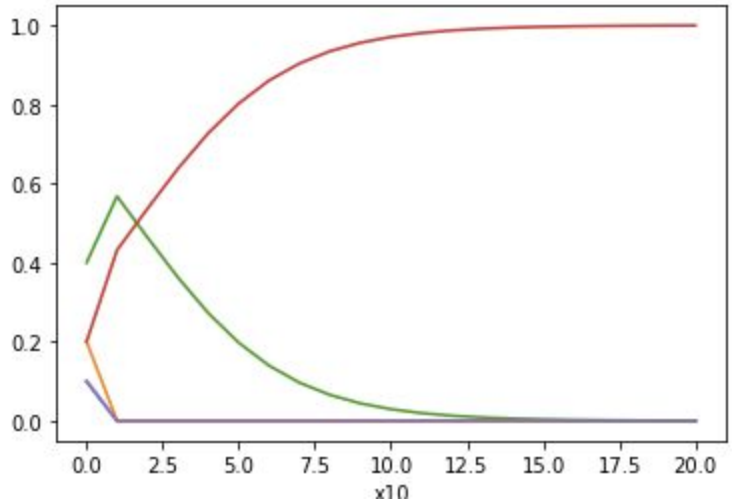
[h2]

```
In [25]: runfile('/Users/coleconstantino/.spyder-
py3/temp.py', wdir='/Users/coleconstantino/.spyder-
py3')
```



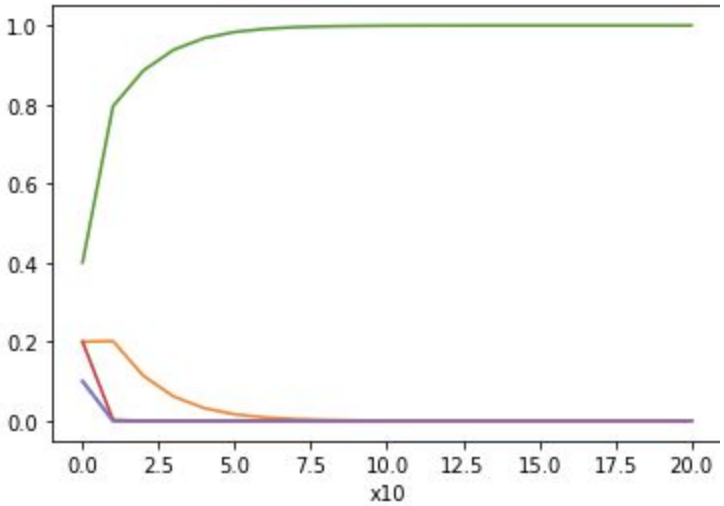
[h3]

```
In [26]: runfile('/Users/coleconstantino/.spyder-
py3/temp.py', wdir='/Users/coleconstantino/.spyder-
py3')
```



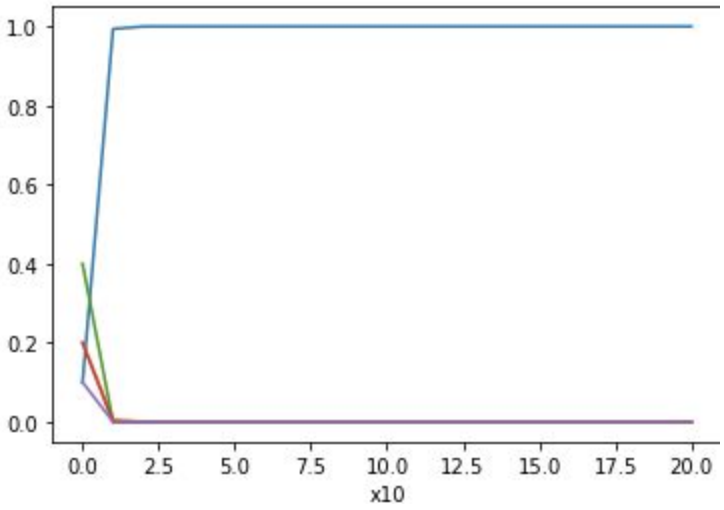
[h4]

```
In [23]: runfile('/Users/coleconstantino/.spyder-
py3/temp.py', wdir='/Users/coleconstantino/.spyder-
py3')
```



[h5]

```
In [27]: runfile('/Users/coleconstantino/.spyder-
py3/temp.py', wdir='/Users/coleconstantino/.spyder-
py3')
```



3)

$$D = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{n,k} \|\boldsymbol{x}_n - \boldsymbol{\mu}_k\|^2$$