Module 7 Discussion

Module 7 Discussion - Optimization Methods

How does the approach to maximization and minimization problems in chapters 13 and 14 compare to what you learned in Module 3 (Chapters 3 and 4 of the text)?

I thought that a simple online search would show some examples of minima and maxima that I could work with. Unfortunately, I didn't find any so I made one up. There was a little bit of trial and error in the process.

The scenario is as follows: A food processor receives deliveries from farmers each day. The problem is that the distributor can receive anywhere between 0 and 19 food deliveries and really doesn't know exactly how many to expect each day. Based on past experience the probabilities of each number of deliveries are shown in the list "probabilities" below.

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import random
%matplotlib inline
```

In [2]:
```
deliveries = np.arange(0, 20, 1)
```

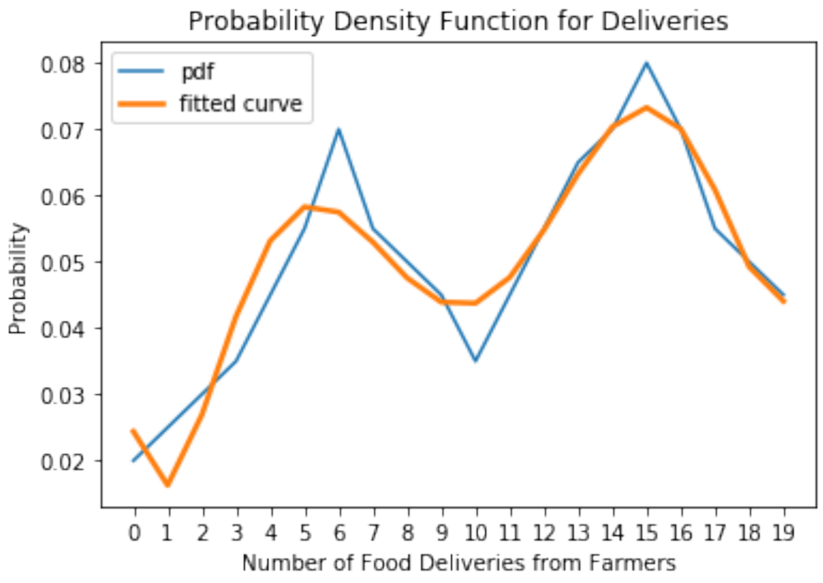
In [3]:
```
# list showing probabilities for each number of deliveries
probabilities = [0.02, 0.025, 0.03, 0.035, 0.045, 0.055, 0.07, 0.055, 
0.05, 0.045, 0.035, 0.045, 0.055,
                 0.065, 0.07, .08, 0.07, 0.055, 0.05, 0.045]
```

In [4]:
```
# verify that this is a valid probability distribution:
if round(sum(probabilities),5) == 1.0:
    print('OK - This is a valid distribution.')
else:
    print('Hold on! These probabilities do not sum to one.')
```

```
OK - This is a valid distribution.
```

In [5]:
```
plt.xticks(np.arange(0, 20, 1))
plt.xlabel('Number of Food Deliveries from Farmers')
plt.ylabel('Probability')
plt.title('Probability Density Function for Deliveries')
plt.plot(deliveries, probabilities, label='pdf')

plot1 = np.polyfit(deliveries, probabilities, 6)
pdf = np.poly1d(plot1)
plt.plot(deliveries, pdf(deliveries), linewidth=2.5, label='fitted cur
ve')
plt.legend();
```



In [6]:
```
# show function for fitted curve
print(pdf)
```

```
              6              5              4             3             2
3.611e-07 x - 2.147e-05 x + 0.0004745 x - 0.00476 x + 0.02064 x - 0.
02444 x + 0.02438
```

Observations:

1. I was a little surprised to see that fitting an equation to the graph required a sixth order polynomial. I started with a '3' in the 'plot1' formula above and then through trial and error kept increasing that number until the graph showed a nice fit to the data.

1. In this example we can see where the minima and maxima are by viewing the graph. In a more complex example, or one with more than 20 possibilities, we might need first and second derivatives to determine minima and maxima.

1. Although this is not a perfect fit for using the optimization methods that we studied in Module 3, there are a optimization problems that could come about when analyzing the business side of the problem. For example, if we used this probability density function and factored in the number of workers required and the cost of each worker, then the problem might look at the bounds between 0 and 9, and then 10 and 19, and look at how to minimize costs while having enough workers on staff to accomodate the expected number of deliveries.

Below is a randomized version of the figures above

In [7]:
```
probabilities1 = []                           # create empty list
for i in range(0, 20):
    i = random.random()                       # create random numbers betw
een 0.0 and 1.0,  we'll do this 20 times
    probabilities1.append(round(i, 4))    # round to 4 decimal places
and add each of the numbers created to our list
```

In [8]:
```
# scale figures so that they sum to one
probabilities2 = [round(i / sum(probabilities1),4) for i in probabilit
ies1]
```

In [9]:
```
# let's make sure that this adds to 1.000
print(sum(probabilities2))
```

```
1.0002
```

In [10]:
```
print(probabilities2, '\n')
print('list length:', len(probabilities2))
```

```
[0.1003, 0.0196, 0.0121, 0.0754, 0.0222, 0.0077, 0.0642, 0.085, 0.06
36, 0.0312, 0.0482, 0.0769, 0.0904, 0.0005, 0.0637, 0.0742, 0.0806,
0.0485, 0.0186, 0.0173]

list length: 20
```