HW_6

March 12, 2020

[47]:
```python
import numpy as np
import matplotlib.pyplot as plt
```

[48]:
```python
regions = []
for i in range(1, 10001):
    regions.append((np.random.uniform(-10, 10), np.random.uniform(-10, 10)))
```

1 Question 1

[49]:
```python
def contains(region, target):
    low = min(region)
    high = max(region)
    if low <= target <= high:
        return True
    else:
        return False
```

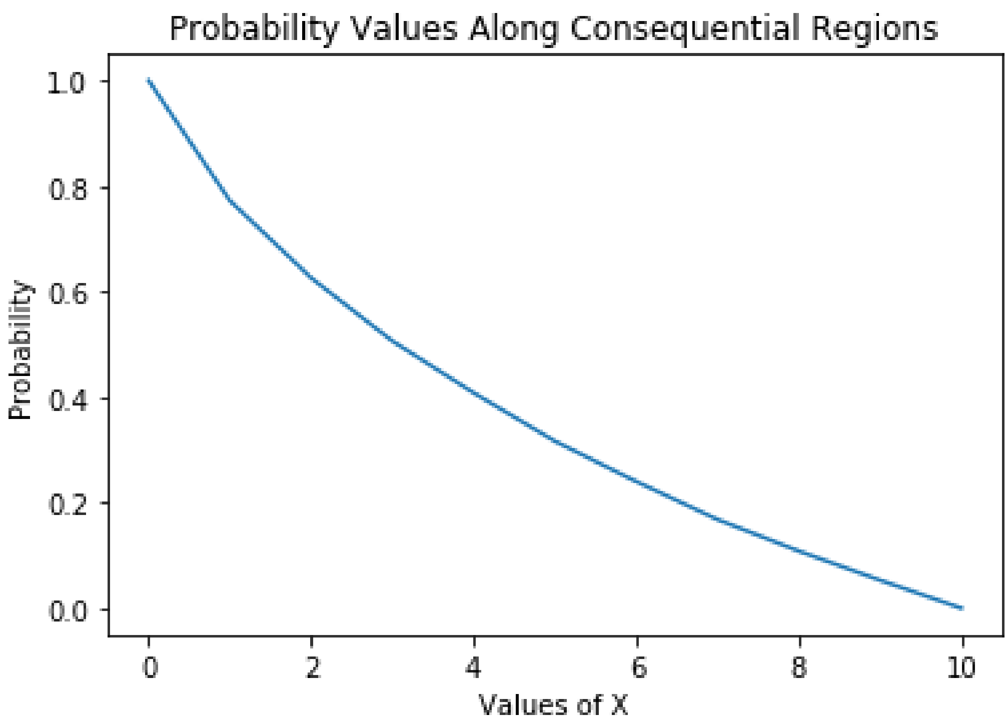
2 Question 2

[50]:
```python
def prob_x_equals_x(data, regions, given, target):
    numerator = 0
    denominator = 0
    region_weight = 1/regions
    for i in data:
        if contains(i, given):
            if contains(i, target):
                numerator += ((1/abs(i[0] - i[1])) * region_weight)
            else:
                numerator += 0
            denominator += ((1/abs(i[0] - i[1])) * region_weight)
    return numerator/denominator

prob_x_equals_x(regions, 10000, 0, 1)
```

[50]: 0.7723410556132814

3 Question 3

[51]:
```
x = range(0, 11)
vals = []
for i in x:
    vals.append(prob_x_equals_x(regions,10000, 0, i))

plt.plot(x, vals)
plt.title("Probability Values Along Consequential Regions")
plt.xlabel("Values of X")
plt.ylabel("Probability")
plt.show()
```



The graph is behaving like Shepard's Universal Law, in that as x-values grow larger than zero, their relative probabilities of being in a consequential region with zero exponentially decrease at a decreasing rate.

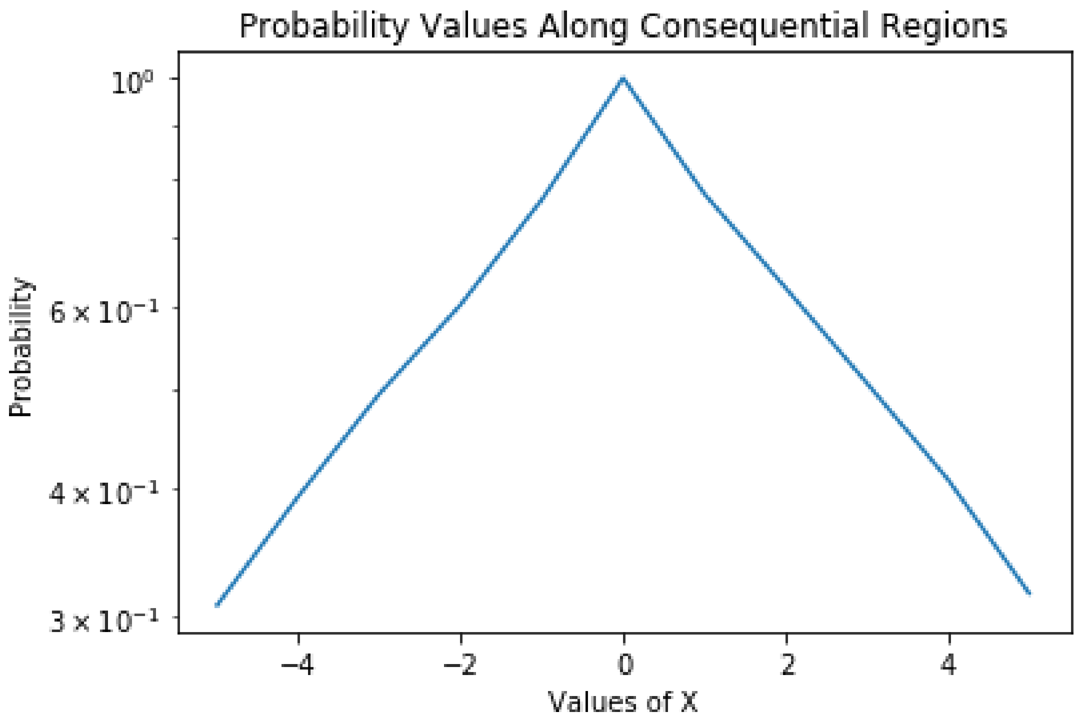
4 Question 4

Exponential and logarithmic curves are inversely proportional to one another, so presenting an exponential curve on a logarthmic scale would display a liner graph (straight line).

5 Question 5

[52]:

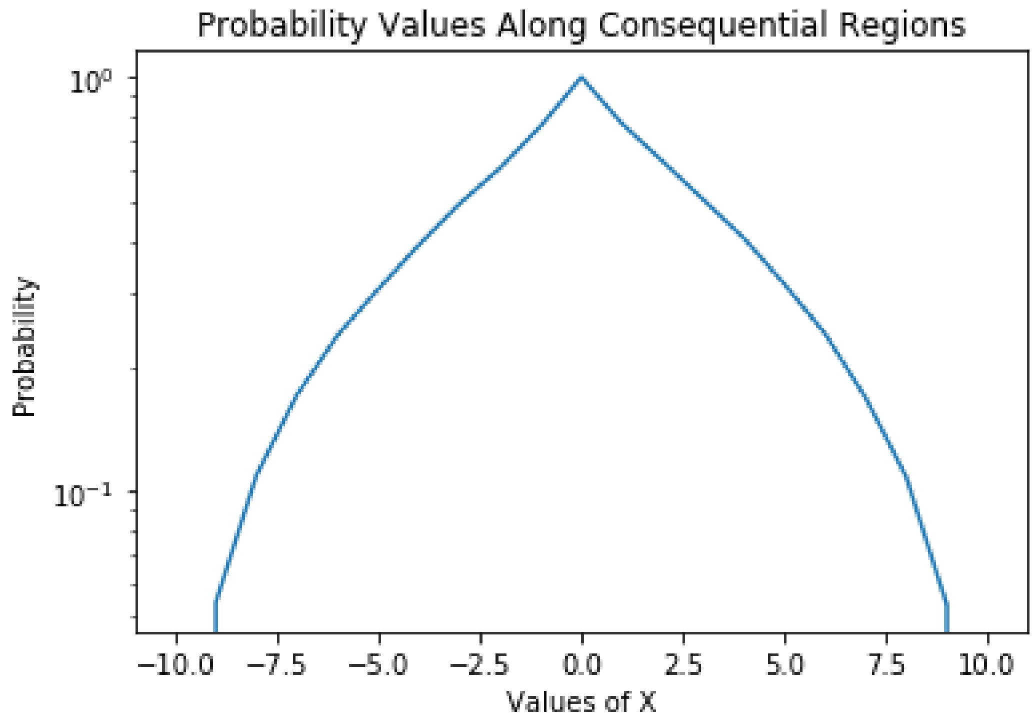
```
x = range(-5, 6)
vals = []
for i in x:
    vals.append(prob_x_equals_x(regions, 10000, 0, i))

plt.plot(x, vals)
plt.title("Probability Values Along Consequential Regions")
plt.xlabel("Values of X")
plt.ylabel("Probability")
plt.yscale("log")
plt.show()
```



[53]:

```
x = range(-10, 11)
vals = []
for i in x:
    vals.append(prob_x_equals_x(regions, 10000, 0, i))

plt.plot(x, vals)
plt.title("Probability Values Along Consequential Regions")
plt.xlabel("Values of X")
plt.ylabel("Probability")
```

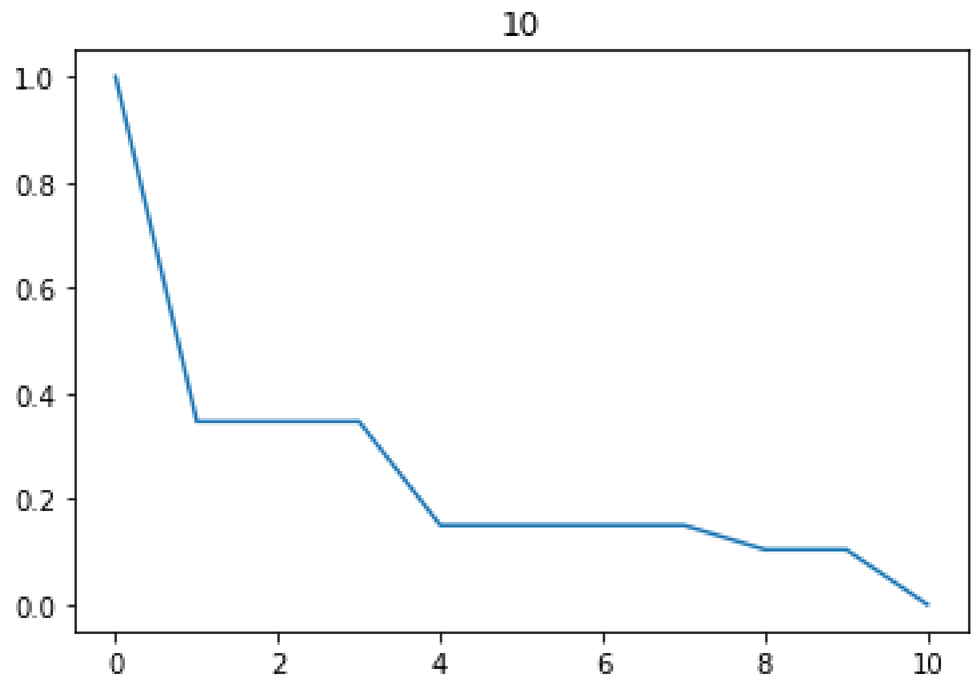
```
plt.yscale("log")
plt.show()
```

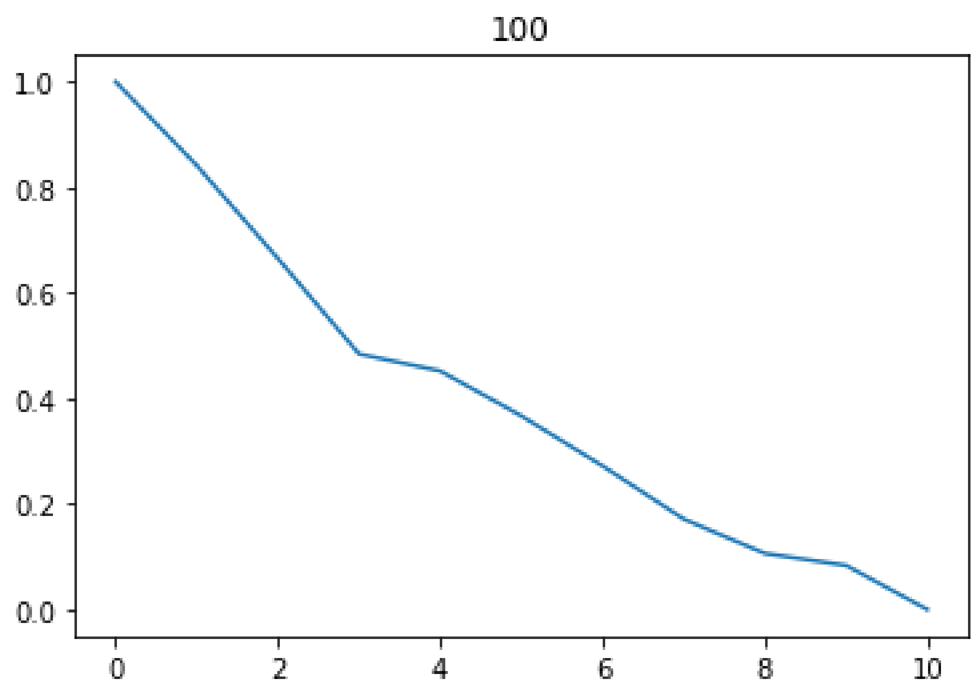


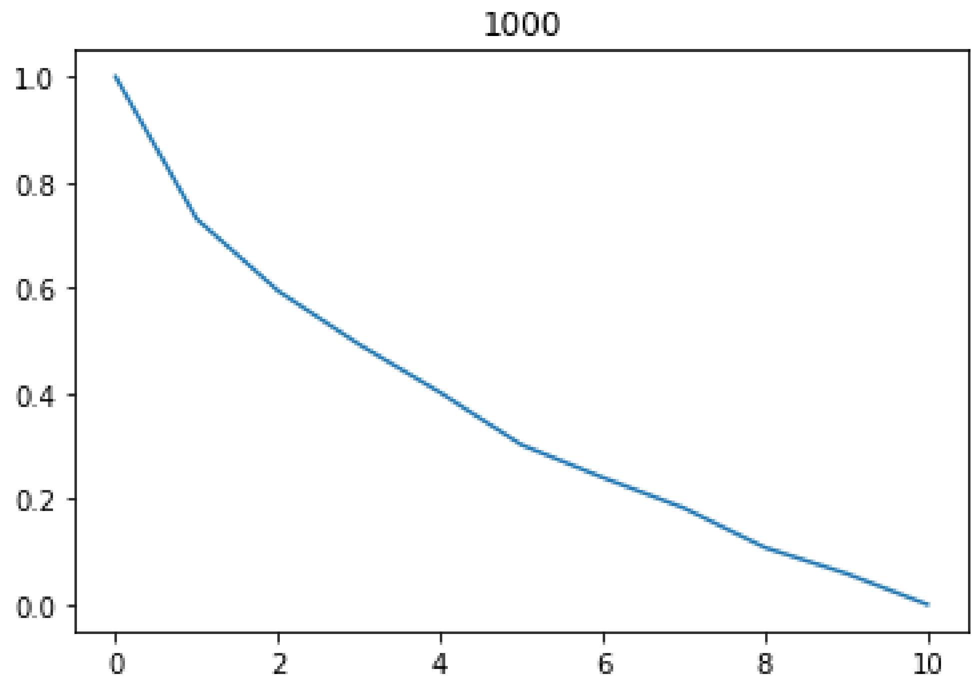
In the [-5, 5] plot, we observe that in the log-scaled view, the graph behaves like a linear function and appears like a straight line. In the [-10, 10] plot, this scaling quickly drops back into a more exponentially-decreasing function as the graph approaches -10 and 10. Based on our scaling, we might suggest that probability actually drops off more severely as we approach larger bounds for our consequential regions.

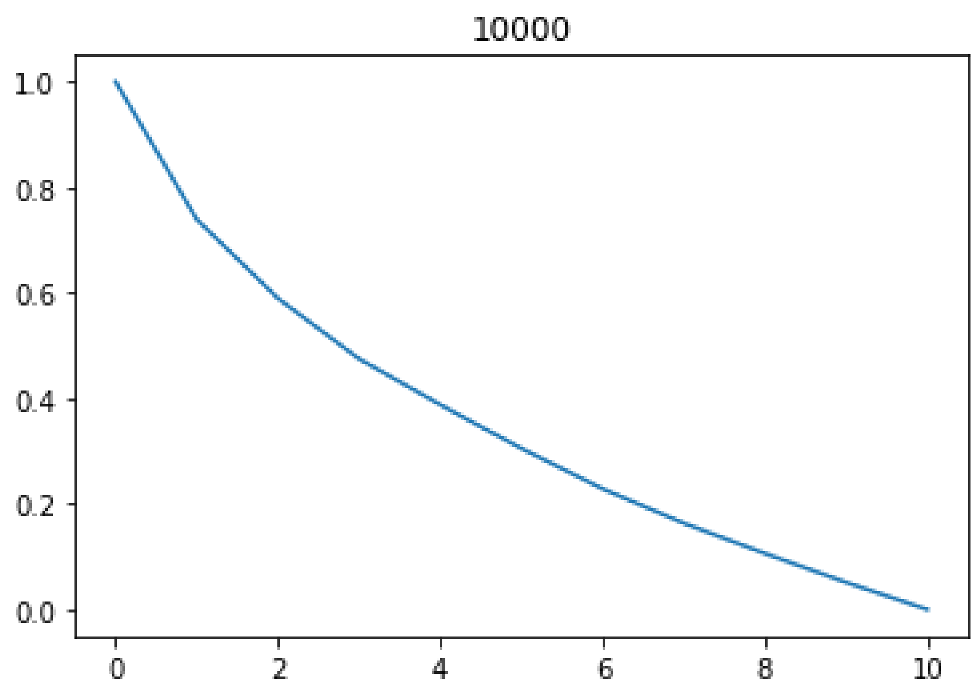
6 Question 6

[54]:
```
dat = [10, 100, 1000, 10000]
x_val = range(0, 11)
for ranges in dat:
    regions = []
    for i in range(1, ranges + 1):
        regions.append((np.random.uniform(-10, 10), np.random.uniform(-10, 10)))
    vals = []
    for values in x_val:
        vals.append(prob_x_equals_x(regions, ranges, 0, values))
    plt.plot(x_val, vals)
    plt.title(str(ranges))
    plt.show()
```


10
1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8
10


100
1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8
10





The graphs of varying consequential regions demonstrate two major patterns; firstly, that with a low number of consequential regions (10), the graph is quite terrible at producing any sort of probability curve. After running the program several times, I could see that at 10 consequential regions, the graph was wildly different and inconsistent each time. Secondly, the graph appears to converge to a smoother curve somewhere between 100 and 1000 consequential regions - beyond this point, even after re-plotting 10,000 regions, it was clear that the curve was **not** becoming noticeably smoother.

7 Question 7

Based on the graphs from prior problems, it is clear that people use a specific and limited number of consequentual regions at a given time - too few and generalization is unpredictable, too many and the computational power expended delivers diminishing returns. Using the log-scaling method introduced earlier, it may be possible to test this by comparing the derivation of the straight-line curve to determine where it behaves the most like a linear function, thus indicating the optimal number of consequential regions.