Samantha Rothman

Homework 2

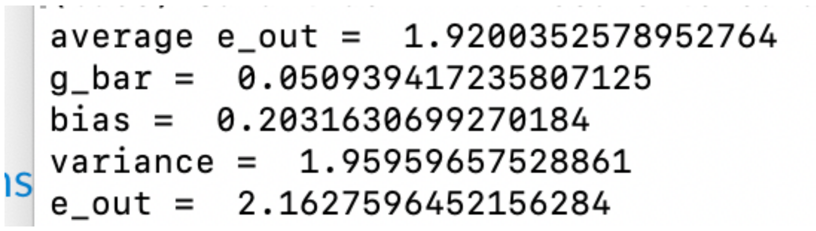
Problem 2.3

A) The growth function for a positive and a negative ray is N+1. To compute the maximum number of dichotomies, we add these together. However, there is overlap in the dichotomies where they are all +1 or -1, so we subtract out those 2. Our final equation is $m_H(N) = N+1+N+1-2 = 2N$. To find the break point, we calculate where $m_H(N) < 2^N$. For N=1 and N=2, $m_H(N) = 2^N$, but for N=3, $2(3) \mathrel{!=} 2^3$. Thus, 3 is our break point. Since d_{vc} = break point minus 1, our $d_{vc} = 2$.

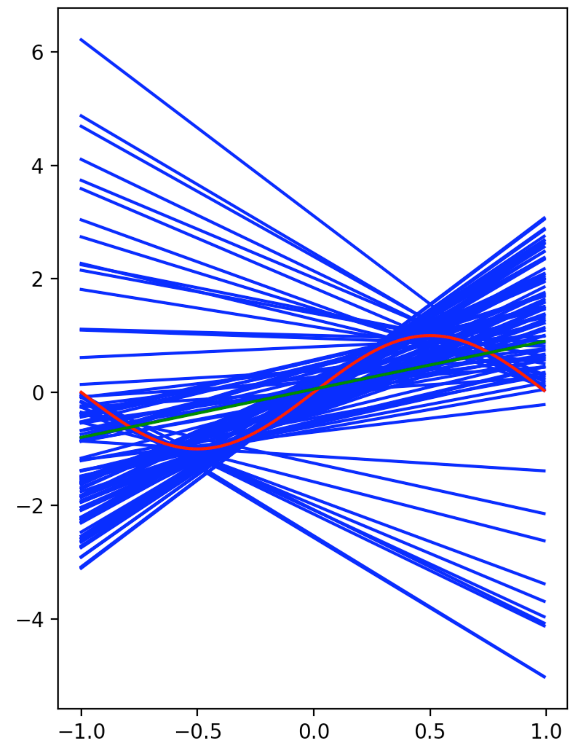
B) The growth function for a positive interval is $m_H(N) = \binom{N+1}{2} +1 = (1/2)N^2+(1/2)N+1$ because there are N+1 total spots for the ends of the interval (this means N+1 choose 2). Once again, we get rid of the overlap for the negative interval, so we subtract 2N. Our final $M_H(N) = N^2–N+2$. To find the break point, we calculate where $m_H(N) < 2^N$. This works for N=1,2, and 3. However, N=4 makes 16-4+2=14 which is not 16. Thus, 4 is the break point and d_{vc}=4-1=3.

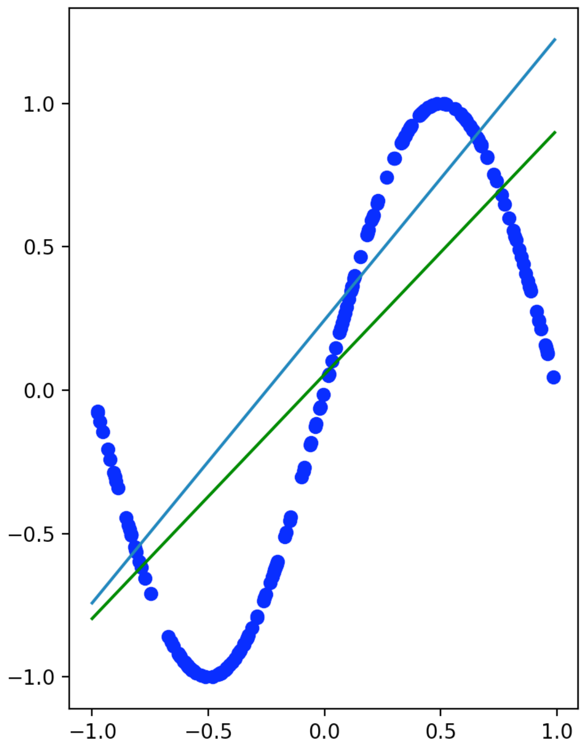
Problem 2.23

A) The hypothesis for this is g_D(x) = (y1-y2)/(x1-x2)*x + (x1*y2-x2*y1)/(x1-x2). Let D = {(x1,y1),(x2,y2)}. When we randomly sample D 100 times, we get 100 different lines for g_Dx (plotted below on the left in blue). Furthermore, we can use these to compute the bias, variance, e_out and g_bar. In both pictures, g_bar is in green. The points in each D are plotted below with g_bar and g_D(x) on the right for np.arange(-1,1,.01). For this run, the average bias was 0.203, average variance was 1.95, and g_bar was 0.05. The average e_out was 1.92 and the bias and variance together (which makes e_out) was 2.16. Compared to the results from the seventh lecture, the bias is almost exactly the same and the variance is slightly higher.

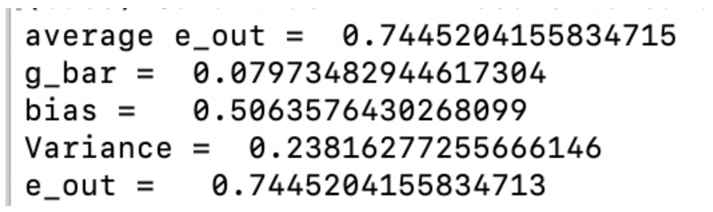

```
average e_out =  1.9200352578952764
g_bar =  0.050939417235807125
bias =  0.2031630699270184
variance =  1.95959657528861
e_out =  2.1627596452156284
```

Samantha Rothman

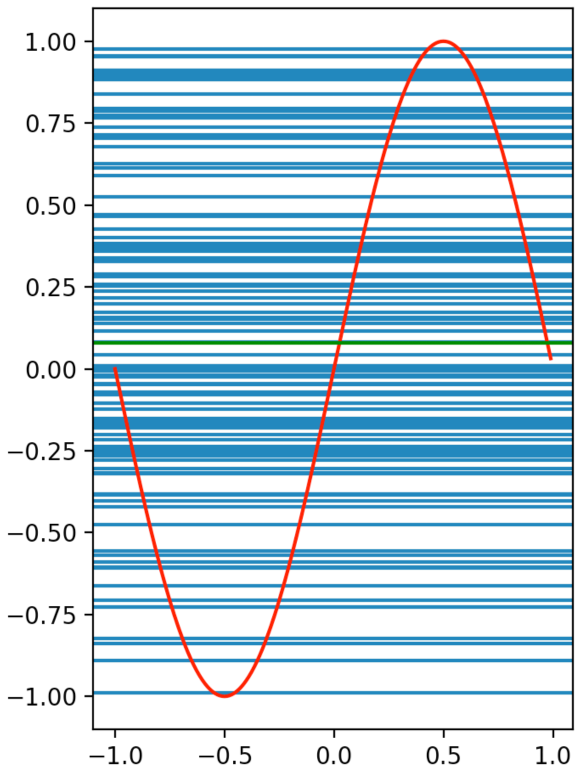


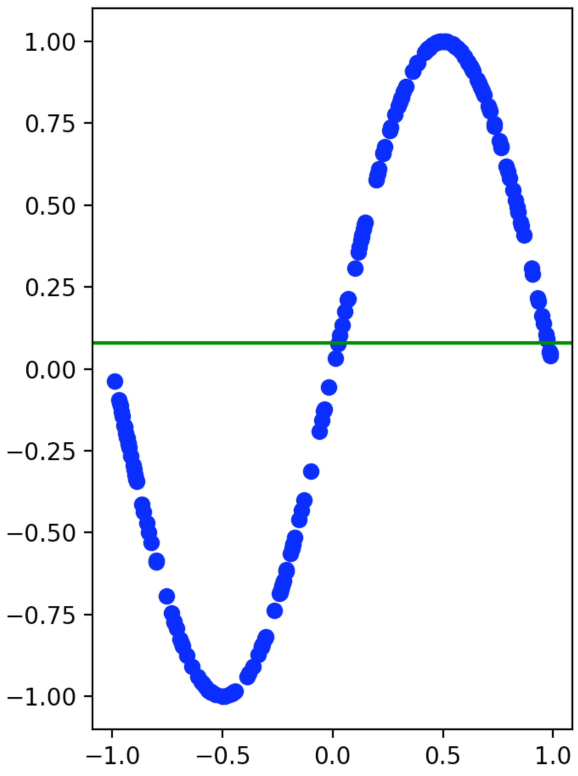


C) The hypothesis for this is g_D(x) = (y1+y2)/2. Similar to part a, we let D = {(x1,y1),(x2,y2)}. When we sample D 100 times, we get 100 different lines for g_Dx (plotted below on the left in blue). Furthermore, we can use these to compute the bias, variance, e_out and g_bar. F(x) is plotted in red on the left. The points in D are plotted below with g_bar on the right for np.arange(-1,1,.01). In both pictures, g_bar is in green. For this run, the average bias was 0.506 and the variance was 0.238. The average e_out and the e_out calculated when adding the bias and variance were both 0.745. The g_bar was 0.07. Compared to the lecture, the bias is exactly the same but the variance is slightly lower.


```
average e_out =  0.7445204155834715
g_bar =  0.07973482944617304
bias =   0.5063576430268099
Variance =  0.23816277255666146
e_out =   0.7445204155834713
```

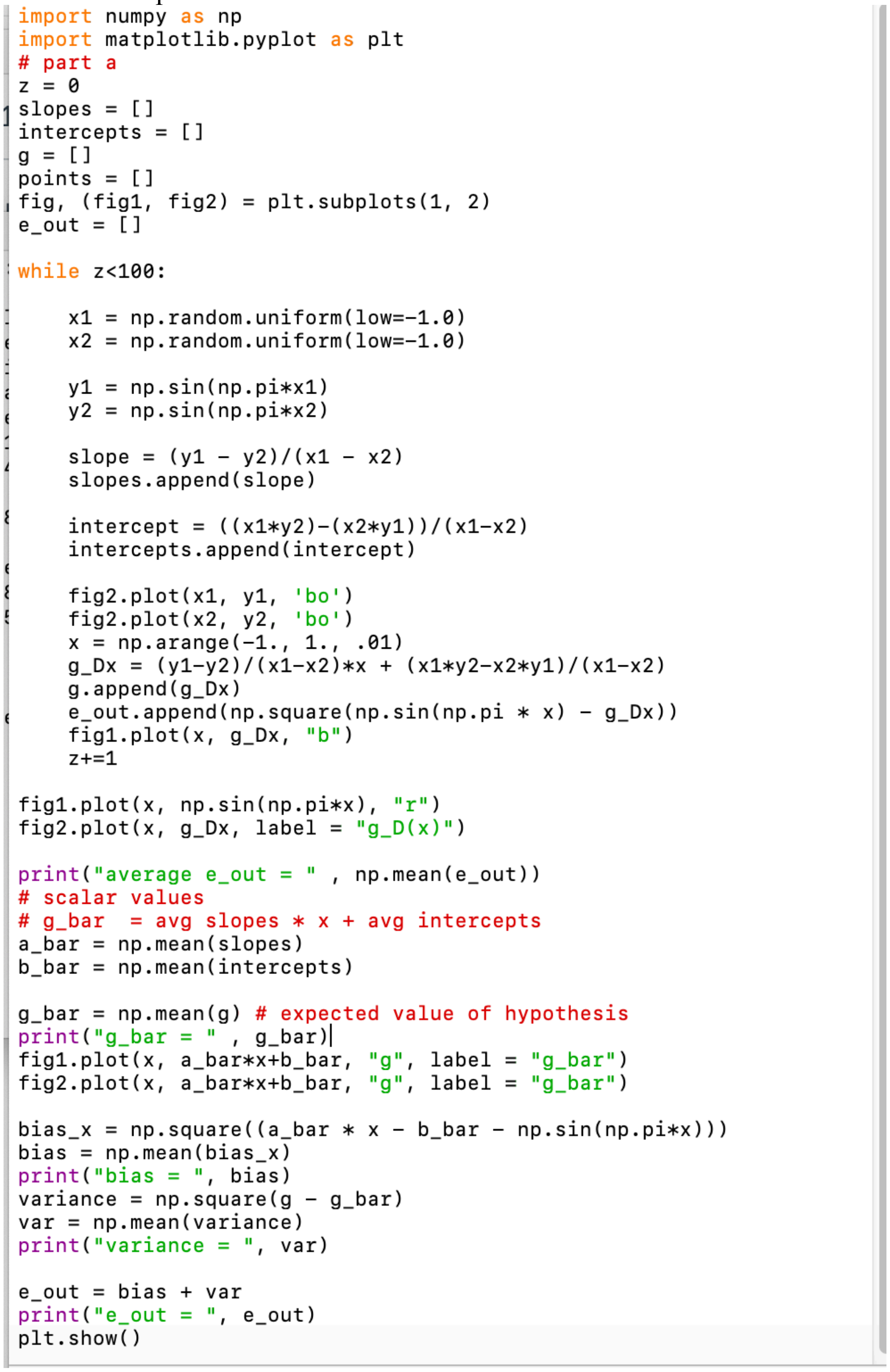
Samantha Rothman





Samantha Rothman

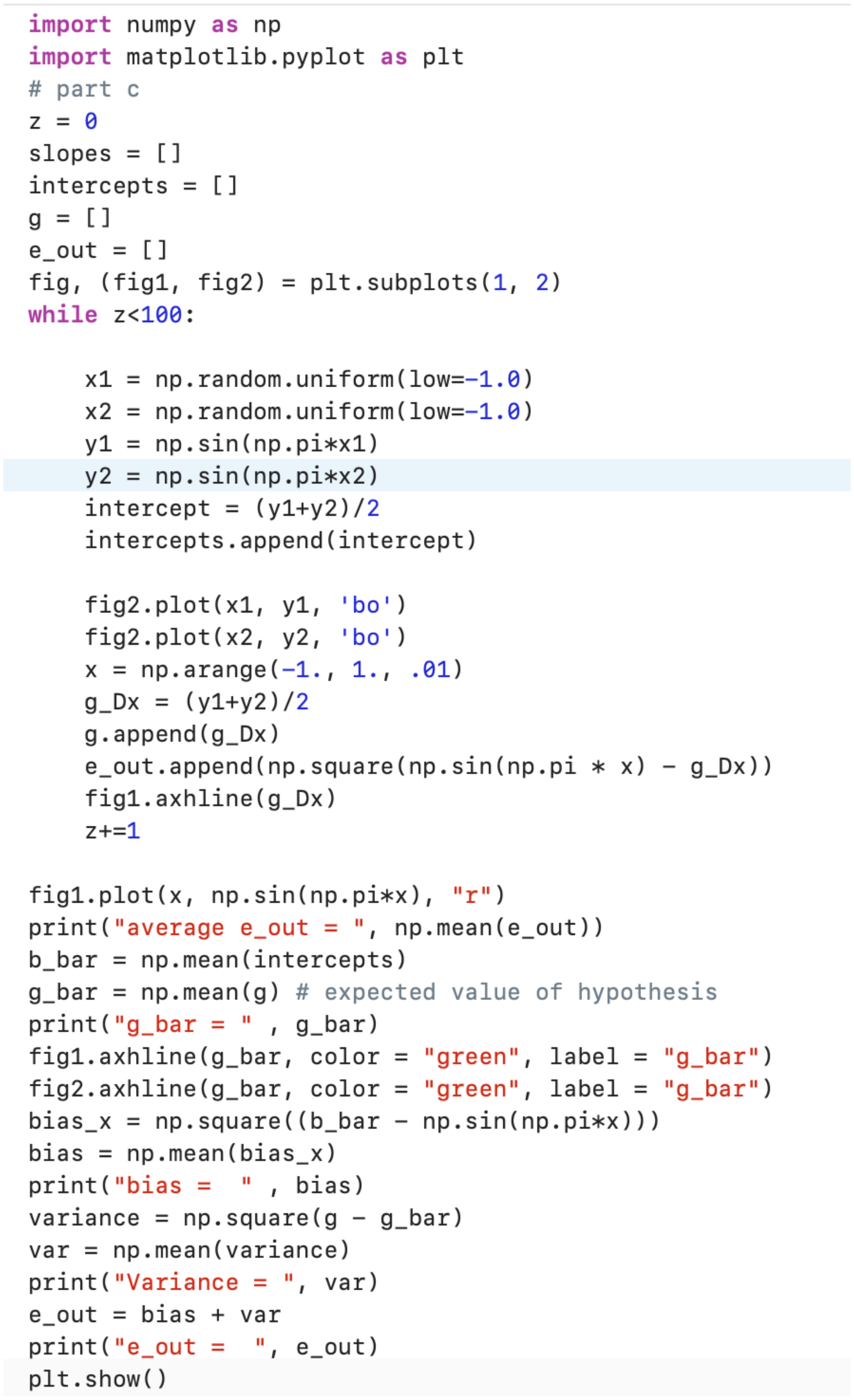
The code for part a:



```
import numpy as np
import matplotlib.pyplot as plt
# part a
z = 0
slopes = []
intercepts = []
g = []
points = []
fig, (fig1, fig2) = plt.subplots(1, 2)
e_out = []

while z<100:

    x1 = np.random.uniform(low=-1.0)
    x2 = np.random.uniform(low=-1.0)

    y1 = np.sin(np.pi*x1)
    y2 = np.sin(np.pi*x2)

    slope = (y1 - y2)/(x1 - x2)
    slopes.append(slope)

    intercept = ((x1*y2)-(x2*y1))/(x1-x2)
    intercepts.append(intercept)

    fig2.plot(x1, y1, 'bo')
    fig2.plot(x2, y2, 'bo')
    x = np.arange(-1., 1., .01)
    g_Dx = (y1-y2)/(x1-x2)*x + (x1*y2-x2*y1)/(x1-x2)
    g.append(g_Dx)
    e_out.append(np.square(np.sin(np.pi * x) - g_Dx))
    fig1.plot(x, g_Dx, "b")
    z+=1

fig1.plot(x, np.sin(np.pi*x), "r")
fig2.plot(x, g_Dx, label = "g_D(x)")

print("average e_out = " , np.mean(e_out))
# scalar values
# g_bar  = avg slopes * x + avg intercepts
a_bar = np.mean(slopes)
b_bar = np.mean(intercepts)

g_bar = np.mean(g) # expected value of hypothesis
print("g_bar = " , g_bar)
fig1.plot(x, a_bar*x+b_bar, "g", label = "g_bar")
fig2.plot(x, a_bar*x+b_bar, "g", label = "g_bar")

bias_x = np.square((a_bar * x - b_bar - np.sin(np.pi*x)))
bias = np.mean(bias_x)
print("bias = ", bias)
variance = np.square(g - g_bar)
var = np.mean(variance)
print("variance = ", var)

e_out = bias + var
print("e_out = ", e_out)
plt.show()
```

Samantha Rothman

The code for part c:



```python
import numpy as np
import matplotlib.pyplot as plt
# part c
z = 0
slopes = []
intercepts = []
g = []
e_out = []
fig, (fig1, fig2) = plt.subplots(1, 2)
while z<100:

    x1 = np.random.uniform(low=-1.0)
    x2 = np.random.uniform(low=-1.0)
    y1 = np.sin(np.pi*x1)
    y2 = np.sin(np.pi*x2)
    intercept = (y1+y2)/2
    intercepts.append(intercept)

    fig2.plot(x1, y1, 'bo')
    fig2.plot(x2, y2, 'bo')
    x = np.arange(-1., 1., .01)
    g_Dx = (y1+y2)/2
    g.append(g_Dx)
    e_out.append(np.square(np.sin(np.pi * x) - g_Dx))
    fig1.axhline(g_Dx)
    z+=1

fig1.plot(x, np.sin(np.pi*x), "r")
print("average e_out = ", np.mean(e_out))
b_bar = np.mean(intercepts)
g_bar = np.mean(g) # expected value of hypothesis
print("g_bar = " , g_bar)
fig1.axhline(g_bar, color = "green", label = "g_bar")
fig2.axhline(g_bar, color = "green", label = "g_bar")
bias_x = np.square((b_bar - np.sin(np.pi*x)))
bias = np.mean(bias_x)
print("bias =  " , bias)
variance = np.square(g - g_bar)
var = np.mean(variance)
print("Variance = ", var)
e_out = bias + var
print("e_out =  ", e_out)
plt.show()
```