# ex_sheet2.1

November 1, 2022

[1]:
```
import numpy
import matplotlib
# %matplotlib inline
from matplotlib import pyplot as plt
```

To simplify following methods:

[2]:
```
axis = numpy.newaxis
log = numpy.log
trapz = numpy.trapz
```

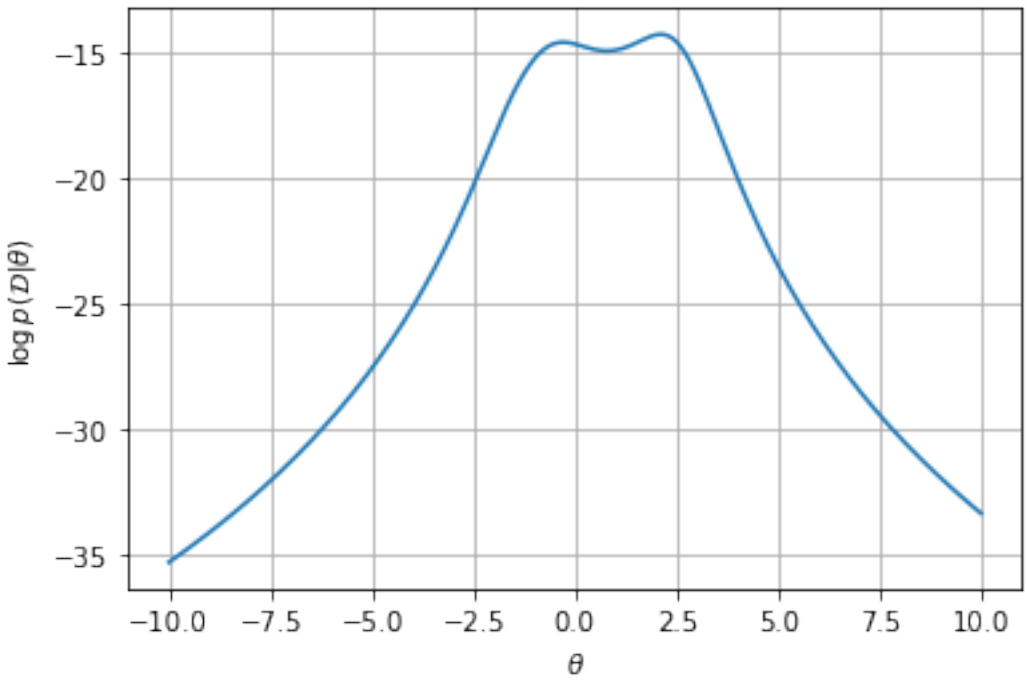
Maximum Likelihood Parameter Estimation

[3]:
```
def pdf(X,THETA):
    return (1.0 / numpy.pi) * (1.0 / (1+(X-THETA)**2))
```

[4]:
```
def ll(D,THETA):
    return log(pdf(D[:,axis],THETA[axis,:])).sum(axis=0)
```

[5]:
```
D = numpy.array([ 2.803, -1.563, -0.853, 2.212, -0.334, 2.503])
THETA = numpy.linspace(-10,10,1001)
```

[6]:
```
plt.grid(True)
plt.plot(THETA,ll(D,THETA))
plt.xlabel(r'$\theta$')
plt.ylabel(r'$\log p(\mathcal{D}|\theta)$')
plt.show()
```




Buiding a classifier

[7]:
```
D1 = numpy.array([ 2.803, -1.563, -0.853, 2.212, -0.334, 2.503])
D2 = numpy.array([-4.510, -3.316, -3.050, -3.108, -2.315])
```

[8]:
```
class MLClassifier:
    # Maximum Likelihood Parameter Estimates
    def fit(self,THETA,D1,D2):
        self.theta1 = THETA[numpy.argmax(ll(D1,THETA))]
        self.theta2 = THETA[numpy.argmax(ll(D2,THETA))]

    # Evaluating for g(x)
    def predict(self,X,p1,p2):
        return log(pdf(X,self.theta1))-log(pdf(X,self.theta2))+log(p1)-log(p2)
```

Creating an instance for our classifier

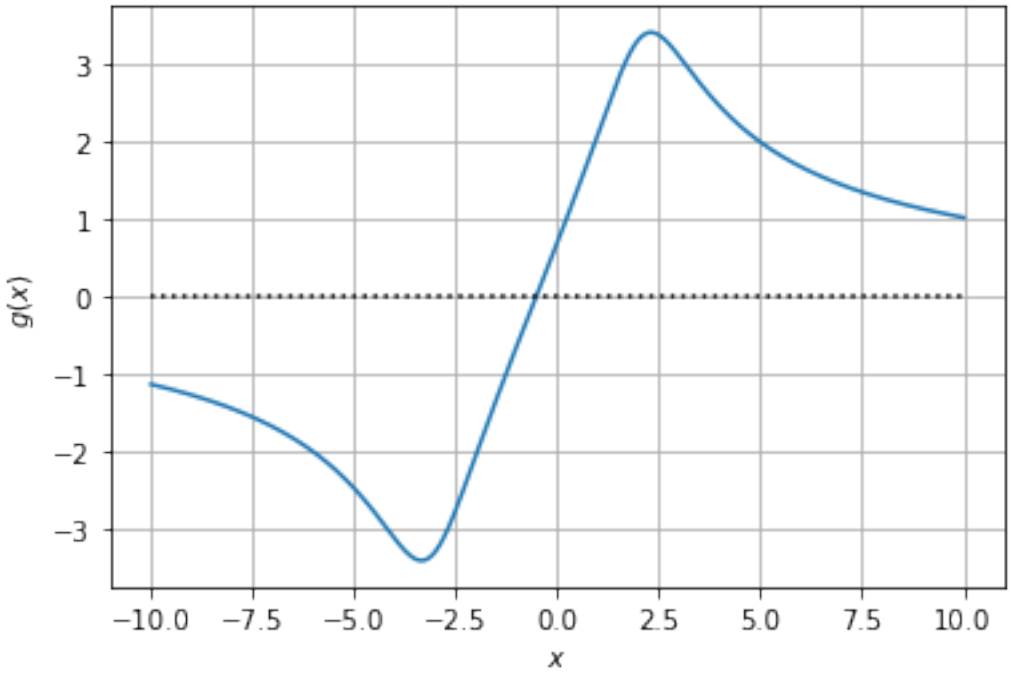
[9]:
```
mlcl = MLClassifier()
mlcl.fit(THETA,D1,D2)
```

[10]:
```
X = numpy.linspace(-10,10,1001)
```

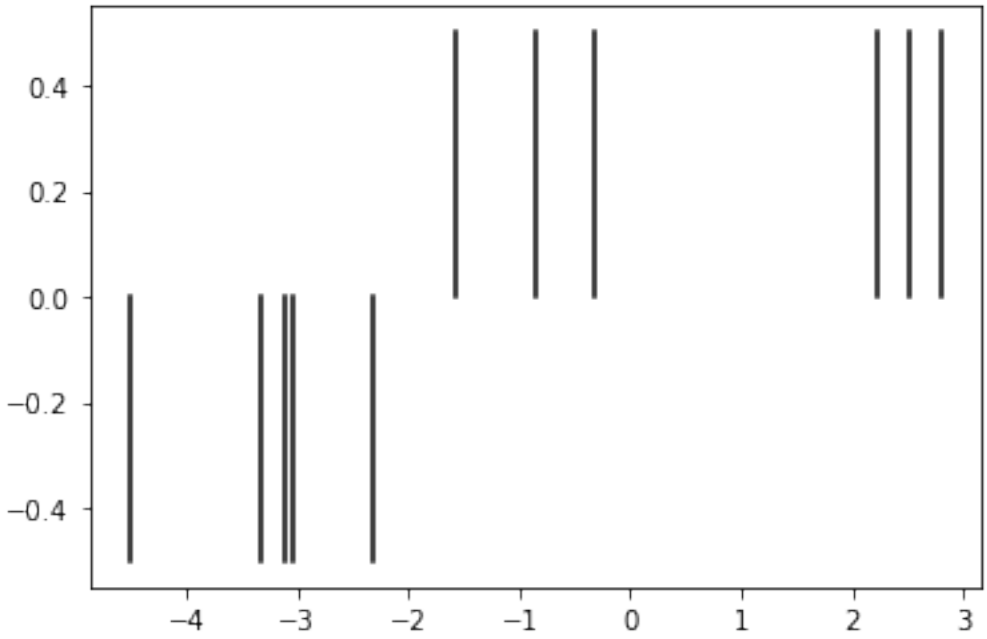
[11]:
```
plt.grid(True)
plt.plot(X,mlcl.predict(X,0.5,0.5))
plt.plot(X,0*X,color='black',ls='dotted')
```

```
plt.xlabel(r'$x$')
plt.ylabel(r'$g(x)$')
```

[11]: Text(0, 0.5, '$g(x)$')




[12]:
```
for d1 in D1: plt.plot([d1,d1],[0,+0.5],color='black')
for d2 in D2: plt.plot([d2,d2],[0,-0.5],color='black')
```




[13]:
```
plt.show()
```

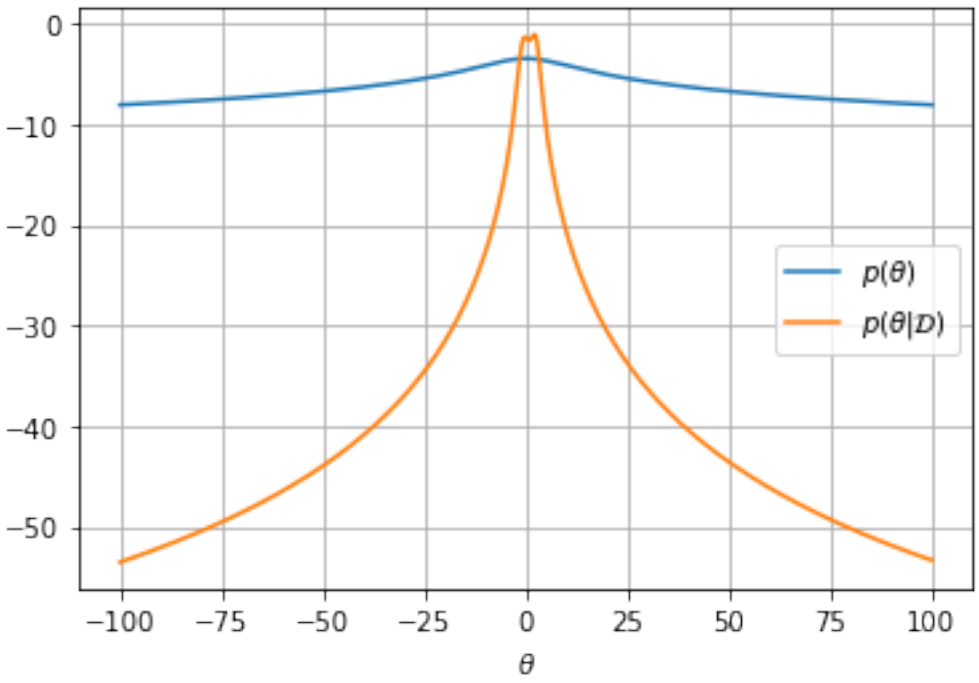
Bayes Parameter Estimation

[14]:
```
def prior(THETA):
    # Taking advantage of the similarity to last problem's function
    return pdf(0,THETA/10)/10
```

[15]:
```
def posterior(D,THETA):
    density = pdf(D[:,axis], THETA[axis,:]).prod(axis=0)
    numerator = density * prior(THETA)
    integral = trapz(numerator, THETA)
    return numerator / integral
```

[16]:
```
THETA = numpy.linspace(-100,100,10001)
```

[17]:
```
plt.grid(True)
plt.plot(THETA,numpy.log(prior(THETA)),label=r'$p(\theta)$')
plt.plot(THETA,numpy.log(posterior(D,THETA)),label=r'$p(\theta|\mathcal{D})$')
plt.legend(); plt.xlabel(r'$\theta$'); plt.show()
```




Buiding a classifier

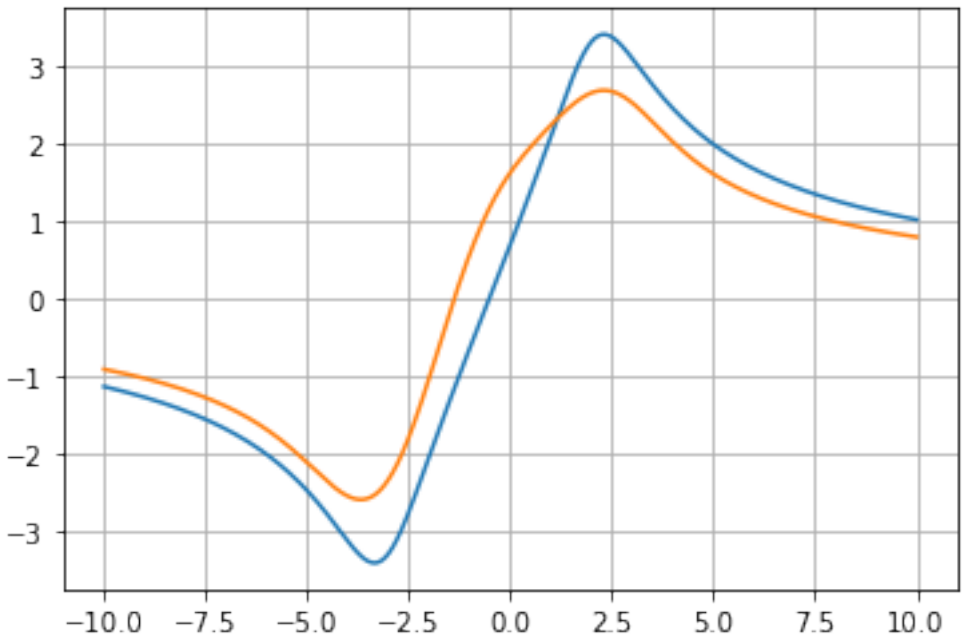
[18]:
```
class BayesClassifier:
    def fit(self,THETA,D1,D2):
        self.THETA = THETA
        self.post1 = posterior(D1,THETA)
        self.post2 = posterior(D2,THETA)
    def predict(self,X,p1,p2):
        density = pdf(X[:,axis],self.THETA[axis,:])
        term1= density*self.post1[axis,:]
        term2= density*self.post2[axis,:]
        pdf1 = trapz(term1, THETA, axis=1)
        pdf2 = trapz(term2, THETA, axis=1)
        return log(pdf1)-log(pdf2)+log(p1)-log(p2)
```

[19]:
```
X = numpy.linspace(-10,10,1001)
```

[20]:
```
bacl = BayesClassifier()
bacl.fit(THETA,D1,D2)
```

[21]:
```
plt.grid(True)
plt.plot(X,mlcl.predict(X,0.5,0.5),label='ML')
plt.plot(X,bacl.predict(X,0.5,0.5),label='Bayes')
```

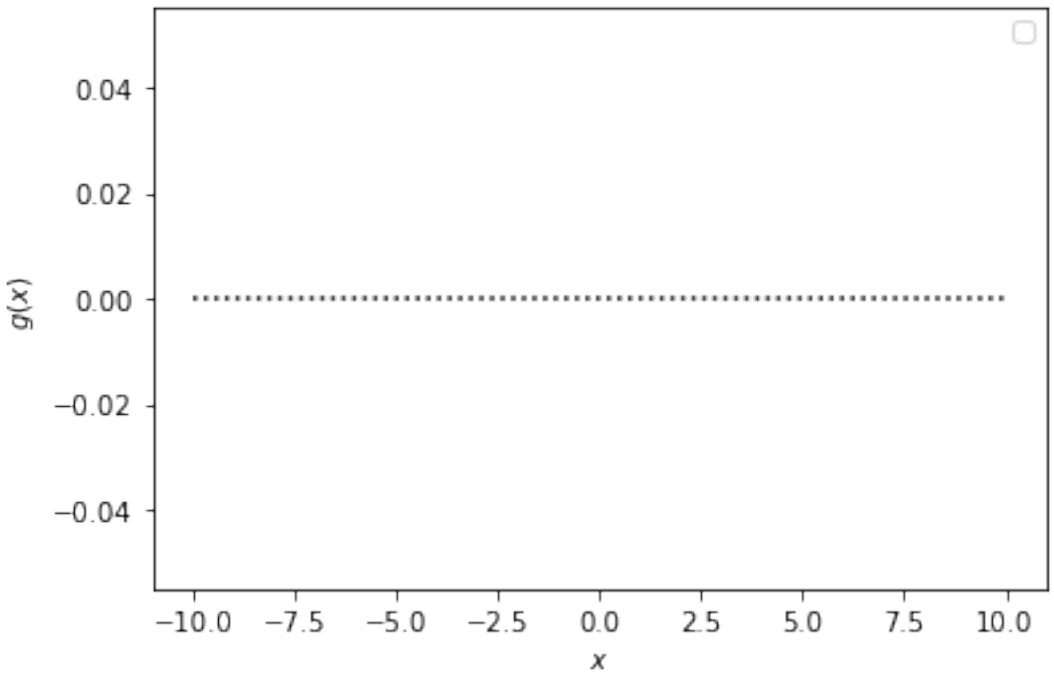
[21]:
```
[<matplotlib.lines.Line2D at 0x2419aba4f70>]
```



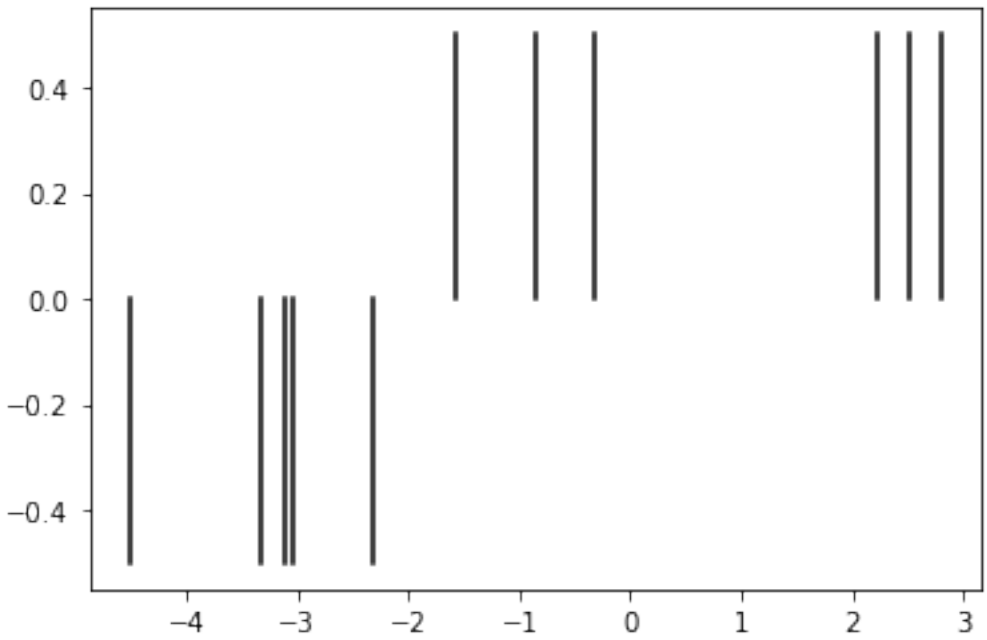

[22]:
```
plt.plot(X,0*X,color='black',ls='dotted')
plt.xlabel(r'$x$'); plt.ylabel(r'$g(x)$')
plt.legend()
```

```
No artists with labels found to put in legend.  Note that artists whose label
start with an underscore are ignored when legend() is called with no argument.
```

[22]:
```
<matplotlib.legend.Legend at 0x2419abc6ca0>
```




[23]:
```
for d1 in D1: plt.plot([d1,d1],[0,+0.5],color='black')
for d2 in D2: plt.plot([d2,d2],[0,-0.5],color='black')
```



[24]:
```
plt.show()
```

[ ]:
```

```