In [38]:
```
import numpy as np
from scipy.special import factorial
import matplotlib.pyplot as plt
import panel as pn
import plotly.graph_objects as go
from IPython.display import set_matplotlib_formats
set_matplotlib_formats('svg')
pn.extension()
```

Problem 1

(A)

$p = 0.2$, $m = 10$, $P(m \mid n) = \frac{e^{-\lambda}\lambda^m}{m!}$, $\lambda = np$, uniform prior $P(n)$

Log-likelihood:

$$\log P(n \mid m) = \log P(m \mid n) + \log P(n) - \log P(m)$$

$\log P(X)$ and prior don't depend on n

$$\begin{aligned} \log P(n \mid m) &= \log P(m \mid n) && (1) \\ &= \log \frac{e^{-\lambda}\lambda^m}{m!} && (2) \\ &= -\lambda + m \log \lambda - \log m! && (3) \\ &= -np + m \log np - \log m! && (4) \end{aligned}$$

Maximum likelihood estimate:

$$\frac{d}{dn}[\log P(n \mid m)] = -p + \frac{m}{n} = 0$$

$$n = \frac{m}{p} = \frac{10}{0.2} = 50$$

(B)

prior $P(n) = \frac{1}{2} Poi(n; \mu_1) + \frac{1}{2} Poi(n; \mu_2) = \frac{1}{2}\frac{e^{-\mu_1}\mu_1^n}{n!} + \frac{1}{2}\frac{e^{-\mu_2}\mu_2^n}{n!}$

Log-likelihood:

$$\begin{aligned} \log P(n \mid m) &= \log P(m \mid n) + \log P(n) && (5) \\ &= \log \frac{e^{-np}(np)^m}{m!} + \log(\frac{1}{2}\frac{e^{-\mu_1}\mu_1^n}{n!} + \frac{1}{2}\frac{e^{-\mu_2}\mu_2^n}{n!}) && (6) \end{aligned}$$

Maximum likelihood estimate:

In [81]:
```
def MaxLikelihood(n,m,p,mu1,mu2):
  return np.log(((np.exp(-n*p))*((n*p)**m))/(factorial(m)))+np.log((1/2)*((np.exp(-mu1))*
```

In [83]:
```
n = np.linspace(0.01,20, num = 500)
```

```
p = 0.2
m = 10
mu1 = 0.1
mu2 = 10
print("New maximum likelihood estimate:", n[np.argmax(MaxLikelihood(n,m,p,mu1,mu2))])
```

```
New maximum likelihood estimate: 15.272905811623245
```

Problem 2

$\{x_i\}$, $P(x_i) = Poi(x_i; \lambda)$

(A)

Log-likelihood:

$$\log P(x_i) = \log \frac{e^{-\lambda}\lambda^{x_i}}{x_i!} \tag{7}$$

$$= -\lambda + x_i \log \lambda - \log x_i! \tag{8}$$

(B)

In [8]:

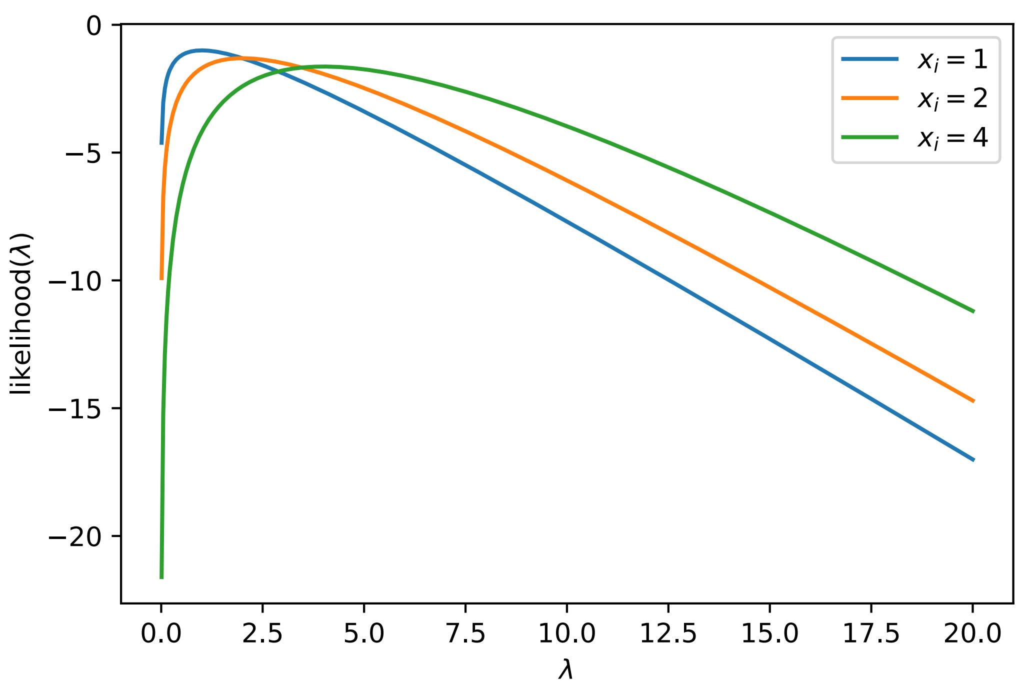
```
def LogLikelihood(x_i, lambda_):
  return -lambda_ + x_i*np.log(lambda_) - np.log(factorial(x_i))
```

In [11]:

```
lambda_ = np.linspace(0.01,20, num = 500)
plt.plot(lambda_, LogLikelihood(1, lambda_), label = '$x_i = 1$')
plt.plot(lambda_, LogLikelihood(2, lambda_), label = '$x_i = 2$')
plt.plot(lambda_, LogLikelihood(4, lambda_), label = '$x_i = 4$')
plt.legend()
plt.xlabel('$\lambda$')
plt.ylabel('likelihood($\lambda$)')
```

Out[11]:

```
Text(0, 0.5, 'likelihood($\\lambda$)')
```



C

Log-likelihood:

$$\log L(\lambda) = \sum_{x_i} \log P(x_i \mid \lambda) \tag{9}$$

$$= \sum_{x_i} \log \frac{e^{-\lambda} \lambda^{x_i}}{x_i!} \tag{10}$$

$$= -n\lambda + \sum_{x_i} (x_i \log \lambda - \log x_i)! \tag{11}$$

D

Maximum likelihood estimate:

$$\frac{\partial}{\partial \lambda}[\log L(\lambda)] = -n + \sum_{x_i} \frac{x_i}{\lambda} = 0$$

$$\lambda = \frac{1}{n} \sum_{x_i} x_i$$

Minimum:

$$\frac{\partial^2}{\partial \lambda^2}[\log L(\lambda)] = \sum_{x_i} -\frac{x_i}{\lambda^2} = -\sum_{x_i} \frac{x_i}{\lambda^2} \leq 0$$

Problem 3

Fast neurons: $\lambda_1 = 10$ spikes/sec, w_1

Slow neurons: $\lambda_2 = 0.1$ spikes/sec, $w_2 = 1 - w_1$

(A)

$P(x) = w_1 Poi(x; \lambda_1) + (1 - w_1) Poi(x; \lambda_2) = w_1 \frac{e^{-\lambda_1} \lambda_1^x}{x!} + (1 - w_1) \frac{e^{-\lambda_2} \lambda_2^x}{x!}$

(B)

Log-likelihood:

$$\log P(x_1) = \log(w_1 \frac{e^{-\lambda_1} \lambda_1^{x_1}}{x_1!} + (1 - w_1) \frac{e^{-\lambda_2} \lambda_2^{x_1}}{x_1!}) \tag{12}$$

(C)

$$\log P(X) = \sum_x \log(w_1 \frac{e^{-\lambda_1} \lambda_1^x}{x!} + (1 - w_1) \frac{e^{-\lambda_2} \lambda_2^x}{x!}) \tag{13}$$

In [18]:

```
def LogLikelihood2(w, X, lambda1, lambda2):
  total = 0
```

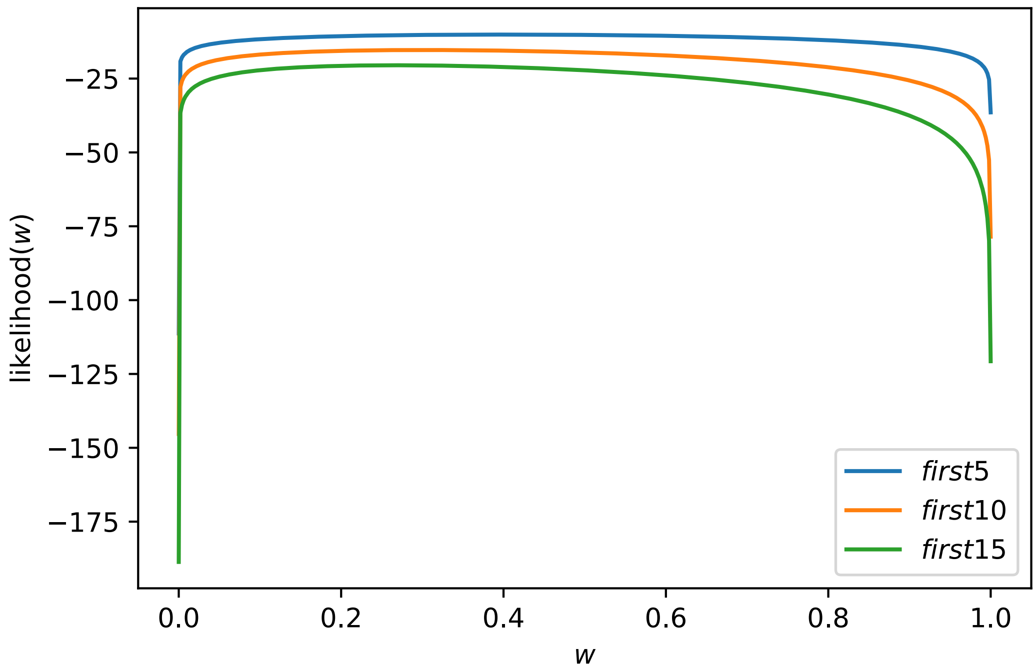
```
    for x in X:
      total += np.log(w*(np.exp(-lambda1)*lambda1**x/factorial(x))+(1-w)*(np.exp(-lambda2)*l
    return total
```

In [27]:

```
X=[10,0,17,0,0,0,0,0,0,9,0,0,0,0,11]
w = np.linspace(0,1, num = 500)
lambda1 = 10
lambda2 = 0.1
plt.plot(w, LogLikelihood2(w, X[0:5], lambda1, lambda2), label = '$first 5$')
plt.plot(w, LogLikelihood2(w, X[0:10], lambda1, lambda2), label = '$first 10$')
plt.plot(w, LogLikelihood2(w, X[0:15], lambda1, lambda2), label = '$first 15$')
plt.legend()
plt.xlabel('$w$')
plt.ylabel('likelihood($w$)')
```

Out[27]:

```
Text(0, 0.5, 'likelihood($w$)')
```



D

In [34]:

```
print("w for slow-spiking first 5 points:", w[np.argmax(LogLikelihood2(w, X[0:5], lambda1,
print("w for slow-spiking first 10 points:", w[np.argmax(LogLikelihood2(w, X[0:10], lambda
print("w for slow-spiking first 15 points:", w[np.argmax(LogLikelihood2(w, X[0:15], lambda
```

```
w for slow-spiking first 5 points: 0.4008016032064128
w for slow-spiking first 10 points: 0.3006012024048096
w for slow-spiking first 15 points: 0.2665330661322645
```

Problem 4

(A)

In [45]:

```
def gaussian_mixture(u1, u2, x):
    return 0.5*(1/np.sqrt(2*np.pi))*(np.exp(-(1/2)*(x-u1)**2)) + 0.5*(1/np.sqrt(2*np.pi))*(n
```

In [46]:

```
u1, u2 = np.mgrid[-20:20:1, -20:20:1]
log_L_vals = np.log(gaussian_mixture(u1, u2, 2))
```

```
fig = go.Figure(go.Surface(x=u1,y=u2,z=log_L_vals,
                           colorscale='viridis'
                           ))
fig.show()
```

There appear to be multiple maxima along values for each μ

(B)

With another μ, $L(\mu_1, \mu_2; x)$ will look like a four dimensional version of the graph we currently have with maxima along axis for all three different μ. This trend will continue for as the number of Gaussian's increase.