

Aufgabe30

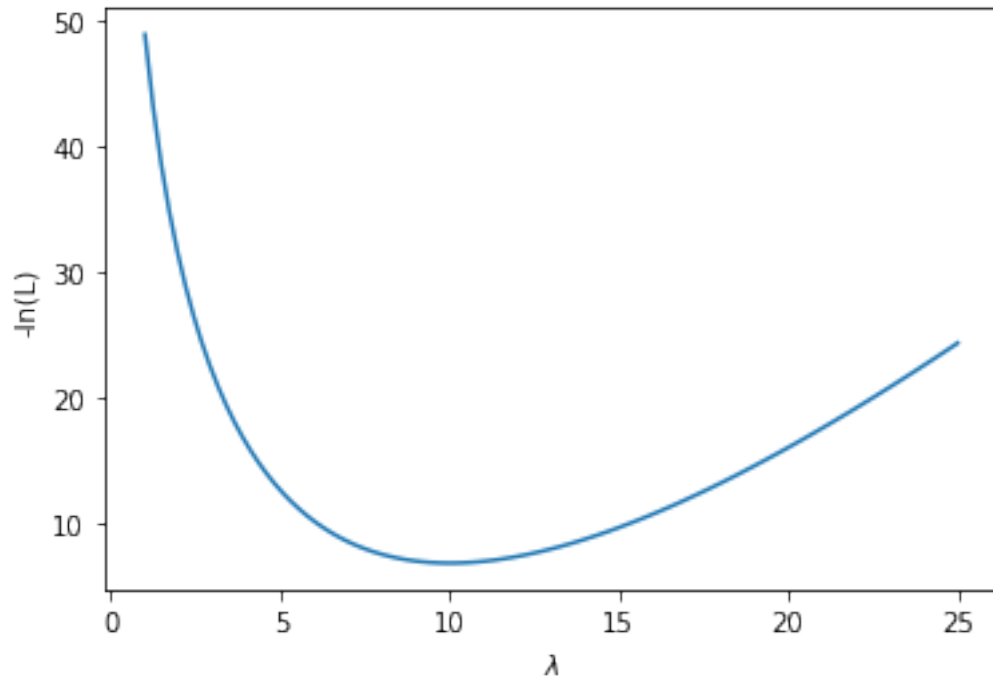
December 17, 2020

1 Aufgabe 30

- a) Berechnen Sie die negative Log-Likelihood-Funktion als Funktion des einzigen Parameters und stellen Sie sie graphisch dar

$$\begin{aligned}P_\lambda &= \frac{\lambda^x}{x!} e^{-\lambda} \\L(\lambda|x_1) &= \frac{\lambda_1^x}{x_1!} e^{-\lambda} \\L(\lambda|x_1 \wedge x_2 \wedge x_3) &= L(\lambda|x_1) \cdot L(\lambda|x_2) \cdot L(\lambda|x_3) \\&= \lambda^{30} e^{-3\lambda} \frac{1}{13!9!8!}\end{aligned}$$

```
In [3]: import matplotlib.pyplot as plt
import numpy as np
import scipy
from scipy.special import factorial
x=[8,9,13]
l = np.linspace(1,25,101) #man könnte den Bereich noch weiter eingrenzen, weil nur Ber
y=(1**13/factorial(13, exact=False)) * np.exp(-l) * (1**9/factorial(9, exact=False)) *
F=-np.log(y)
plt.plot(l, F)
plt.xlabel(r"$\lambda$")
plt.ylabel(r"-ln(L)")
plt.show()
```



b) Bei welchem Wert von λ liegt das Minimum \ln_{\max} ?

```
In [4]: #from scipy import optimize
        #minimum = optimize.fmin(y, 7)

        print(np.min(F))
        minL=l[np.argmin(F)]
        print('Minimum von -ln(F)=', minL)
```

6.8831843201705585

Minimum von $-\ln(F)$ = 10.12

Das Minimum liegt bei $\lambda = 10,12$

c) Für welche Werte von λ nimmt \ln die Werte $\ln_{\max}+12$, $\ln_{\max}+2$ und $\ln_{\max}+92$ an und was sagen diese Werte aus?

```
In [11]: from scipy.optimize import fsolve
        from scipy.special import factorial
        def func(x):
            return -np.log(x**30/(factorial(13)*factorial(9)*factorial(8)) * np.exp(-3*x) + 1)
        root = fsolve(func, 5)
```

```
#np.isclose(func(root), [0.0, 0.0]) # func(root) should be almost 0.0.
#array([ True,  True])
```

C:\Users\Lena\Anaconda3\lib\site-packages\ipykernel_launcher.py:4: RuntimeWarning: overflow error after removing the cwd from sys.path.

```
In [12]: import sympy as sy
        from sympy.solvers import solve
        from sympy import Symbol
        from sympy import *
        from sympy import log, exp, factorial, solve
        from sympy import LambertW
        from sympy import init_printing

        init_printing()

        x = Symbol('x')
        expr = (x**30/(factorial(13)*factorial(9)*factorial(8)) * exp(-3*x)) - 10.2 - 2
        sol=solve(-log(expr),x)
        print(sol)
        #print(latex(sol))
```

RecursionError

Traceback (most recent call last)

```
<ipython-input-12-713db91abc08> in <module>()
    11 x = Symbol('x')
    12 expr = (x**30/(factorial(13)*factorial(9)*factorial(8)) * exp(-3*x)) - 10.2 - 2
--> 13 sol=solve(-log(expr),x)
    14 print(sol)
    15 #print(latex(sol))
```

```
C:\Users\Lena\Anaconda3\lib\site-packages\sympy\solvers\solvers.py in solve(f, *symbols)
1033         if fi.has(Float):
1034             floats = True
-> 1035         f[i] = nsimplify(fi, rational=True)
1036
1037     # Any embedded piecewise functions need to be brought out to the
```

```
C:\Users\Lena\Anaconda3\lib\site-packages\sympy\simplify\simplify.py in nsimplify(expr)
1130     """
1131     try:
```

```

-> 1132         return sympify(as_int(expr))
    1133     except (TypeError, ValueError):
    1134         pass

C:\Users\Lena\Anaconda3\lib\site-packages\sympy\core\compatibility.py in as_int(n)
    400     """
    401     try:
--> 402         result = int(n)
    403         if result != n:
    404             raise TypeError

C:\Users\Lena\Anaconda3\lib\site-packages\sympy\core\expr.py in __int__(self)
    193         # places) we need to test whether we are off by one.
    194         from sympy import Dummy
--> 195         r = self.round(2)
    196         if not r.is_Number:
    197             raise TypeError("can't convert complex to int")

C:\Users\Lena\Anaconda3\lib\site-packages\sympy\core\expr.py in round(self, p)
   3149         if not x.is_real:
   3150             i, r = x.as_real_imag()
-> 3151             return i.round(p) + S.ImaginaryUnit*r.round(p)
   3152         if not x:
   3153             return x

C:\Users\Lena\Anaconda3\lib\site-packages\sympy\core\expr.py in round(self, p)
   3149         if not x.is_real:
   3150             i, r = x.as_real_imag()
-> 3151             return i.round(p) + S.ImaginaryUnit*r.round(p)
   3152         if not x:
   3153             return x

... last 2 frames repeated, from the frame below ...

C:\Users\Lena\Anaconda3\lib\site-packages\sympy\core\expr.py in round(self, p)
   3149         if not x.is_real:
   3150             i, r = x.as_real_imag()
-> 3151             return i.round(p) + S.ImaginaryUnit*r.round(p)
   3152         if not x:
   3153             return x

```

```
RecursionError: maximum recursion depth exceeded
```

Ab der c) funktioniert das nicht mehr. Da sind beim Code irgendwelche Fehler, wissen aber leider nicht was da falsch ist und auch nicht wie man das sonst per Hand berechnen sollte.

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```