Homework 2

Juliette Franqueville

October 5, 2022

(1) For $y_1, .., y_n \sim \mathcal{N}(\mu, \sigma^2)$, show that $s = (\bar{y}, s^2)$ are sufficient. Also, find the observed and the expected information matrices

The factorization theorem says that statistic s is sufficient iif $f(y_{1:n}|\theta) = g(s|\theta)h(y_{1:n})$. For the normal, we have:

$$
\begin{aligned}
f(y_{1:n}|\theta) &= \prod (2\pi\sigma^2)^{-1/2}\text{exp}\left(-\frac{(y_i-\mu)^2}{2\sigma^2}\right) \\
&= (2\pi\sigma^2)^{-n/2}\text{exp}\left(-\frac{\sum(y_i-\mu)^2}{2\sigma^2}\right) \\
&= (2\pi\sigma^2)^{-n/2}\text{exp}\left(-\frac{\sum([y_i-\bar{y}]-[\mu-\bar{y}])^2}{2\sigma^2}\right) \\
&= (2\pi\sigma^2)^{-n/2}\text{exp}\left(-\frac{\sum\{[y_i-\bar{y}]^2-2[y_i-\bar{y}][\mu-\bar{y}]+[\mu-\bar{y}]^2\}}{2\sigma^2}\right) \\
&= (2\pi\sigma^2)^{-n/2}\text{exp}\left(-\frac{\sum\{[y_i-\bar{y}]^2+[\mu-\bar{y}]^2\}}{2\sigma^2}\right) \\
&= (2\pi\sigma^2)^{-n/2}\text{exp}\left(-\frac{\sum[y_i-\bar{y}]^2}{2\sigma^2}\right)\text{exp}\left(-\frac{n[\mu-\bar{y}]^2}{2\sigma^2}\right) \\
&= (2\pi\sigma^2)^{-n/2}\text{exp}\left(-\frac{\sum[y_i-\bar{y}]^2}{2\sigma^2}\right)\text{exp}\left(-\frac{n[\mu-\bar{y}]^2}{2\sigma^2}\right) \\
&= (2\pi\sigma^2)^{-n/2}\text{exp}\left(-\frac{(n-1)s^2}{2\sigma^2}\right)\text{exp}\left(-\frac{n[\mu-\bar{y}]^2}{2\sigma^2}\right)
\end{aligned}
$$

We recognize $h(y_{1:n}) = 1$ and $g(\bar{y}, s^2|\mu,\sigma^2) = (2\pi\sigma^2)^{-n/2}\text{exp}\left(-\frac{(n-1)s^2}{2\sigma^2}\right)\text{exp}\left(-\frac{n[\mu-\bar{y}]^2}{2\sigma^2}\right)$ is a sufficient statistic for (μ, σ^2).

The observed information matrix, $-\frac{\partial^2\mathcal{L}(\boldsymbol{\theta})}{\boldsymbol{\theta}\boldsymbol{\theta}^T}$, is:

$$
\begin{bmatrix}
\frac{n}{\sigma^2} & \frac{n(\bar{y}-\mu)}{\sigma^4} \\
\frac{n(\bar{y}-\mu)}{\sigma^4} & -\frac{n}{2\sigma^4}+\frac{1}{\sigma^6}\sum(y_i-\mu)^2)
\end{bmatrix}
$$

and the expected information matrix, $E\left(-\frac{\partial^2\mathcal{L}(\boldsymbol{\theta})}{\boldsymbol{\theta}\boldsymbol{\theta}^T}\right)$

$$
\begin{bmatrix}
\frac{n}{\sigma^2} & 0 \\
0 & \frac{n}{2\sigma^4}
\end{bmatrix}
$$

Most terms are evident, except

$$\begin{aligned}
E\left[-\frac{n}{2\sigma^4}+\frac{1}{\sigma^6}\sum(y_i-\mu)^2)\right] &= -\frac{n}{2\sigma^4}+\frac{1}{\sigma^6}E\left(\sum y_i^2-2\mu\sum y_i+n\mu^2\right)\\
&= -\frac{n}{2\sigma^4}+\frac{n}{\sigma^6}[(var(y)+E(y)^2-2\mu E(\bar{y})+\mu^2)]\\
&= -\frac{n}{2\sigma^4}+\frac{n}{\sigma^6}[\sigma^2+\mu^2-2\mu^2+\mu^2]\\
&= \frac{n}{2\sigma^4}
\end{aligned}$$

(2) If $Y_1,...,Y_n \overset{\text{iid}}{\sim} N(\mu,c\mu^2)$, where c is a known constant, show that the minimal sufficient statistic for μ is the same as for the $N(\mu,c\mu^2)$ distribution. Find the maximum likelihood estimate of μ and give its large-sample standard error. Show that the distribution of $\bar{Y}^2/S^2$ does not depend on μ.

First we find the minimal sufficient statistics for $N(\mu,\sigma^2)$. We find:

$$\begin{aligned}
\frac{f(x)}{f(y)} &= \frac{\exp\left(-\frac{\sum(x-\mu)^2}{2\sigma^2}\right)}{\exp\left(-\frac{\sum(y-\mu)^2}{2\sigma^2}\right)}\\
&= \exp\left(-\frac{1}{2\sigma^2}[\sum(x-\mu)^2-\sum(y-\mu)^2]\right)\\
&= \exp\left(-\frac{1}{2\sigma^2}[\sum x_i^2-\sum y_i^2-2\mu(\sum x_i-\sum y_i)\right)
\end{aligned}$$

The ratio is independent of μ when $\sum x_i^2=\sum y_i^2$ and independent of σ when $\sum x_i^2=\sum y_i^2$ and $\sum x_i=\sum y_i$, so $\sum x_i^2$ and $\sum x_i$ are minimally sufficient statistics. Similarly for $N(\mu,c\mu^2)$

$$\frac{f(x)}{f(y)} = \exp\left(-\frac{1}{2\mu^2c}[\sum x_i^2-\sum y_i^2-2\mu(\sum x_i-\sum y_i)\right)$$

$\sum x_i^2$ and $\sum x_i$ are minimally sufficient statistics.
For $N(\mu,c\mu^2)$, we have:

$$\begin{aligned}
L(\mu) &= \prod(2\pi c\mu^2)^{-1/2}\exp\left(-\frac{(y_i-\mu)^2}{2c\mu^2}\right)\\
&= (2\pi c\mu^2)^{-n/2}\exp\left(-\frac{\sum(y_i-\mu)^2}{2c\mu^2}\right)\\
\mathcal{L}(\mu) &= -\frac{n}{2}log(2\pi c)-\frac{n}{2}log(\mu^2)--\frac{\sum(y_i-\mu)^2}{2c\mu^2}\\
\frac{\partial\mathcal{L}(\mu)}{\partial\mu} &= -\frac{n}{\mu}-\frac{1}{c}\left(\frac{\sum x_i}{\mu^2}-\frac{\sum x_i^2}{\mu^3}\right)
\end{aligned}$$

Set the derivative to 0:

$$\begin{aligned}
-\frac{n}{\mu} - \frac{1}{c}\left(\frac{\sum x_i}{\mu^2} - \frac{\sum x_i^2}{\mu^3}\right) &= 0 \\
-n\mu^2 - \frac{1}{c}(\sum x_i - \sum x_i^2 \mu) &= 0 \\
nc\mu^2 + \sum x_i^2 \mu - \sum x_i &= 0
\end{aligned}$$

Solving for μ:

$$\mu = \frac{-\sum x_i + [(\sum x_i)^2 + 4nc \sum x_i^2]^{1/2}}{2nc}$$

for $\bar{x} > 0$, and

$$\mu = \frac{-\sum x_i - [(\sum x_i)^2 + 4nc \sum x_i^2]^{1/2}}{2nc}$$

otherwise (I checked the signs numerically, as suggested by Dr. Sarkar). We can estimate the standard error from the Fisher information matrix (se $= I^{-1/2}$) (which is a scalar here).

$$\begin{aligned}
\frac{\partial^2 \mathcal{L}(\mu)}{\partial \mu^2} &= \frac{n}{\mu^2} - \frac{1}{c}\left[\frac{3\sum x_i^2}{\mu^4} - \frac{2\sum x_i}{\mu^3}\right] \\
E\left[\frac{\partial^2 \mathcal{L}(\mu)}{\partial \mu^2}\right] &= \frac{n}{\mu^2} - \frac{1}{c}\left[\frac{3n(\mu^2 c + \mu^2)}{\mu^4} - \frac{2n\mu}{\mu^3}\right] \\
E\left[\frac{\partial^2 \mathcal{L}(\mu)}{\partial \mu^2}\right] &= -\frac{n}{\mu^2}(2 + 1/c) \\
\text{SE}(\mu) &= [\frac{n}{\mu^2}(2 + 1/c)]^{-1/2}
\end{aligned}$$

We would plug μ_{MLE} for μ. Finally, to show that the distribution of $\bar{Y}^2/S^2$ does not depend on μ, we use the fact that $y_i/\mu \sim N(1, c)$. Since $S^2 = (n-1)^{-1}\sum(x_i - \mu)^2$, $\bar{Y}^2/S^2$ cannot depend on μ. The figure below confirms this using simulated data:

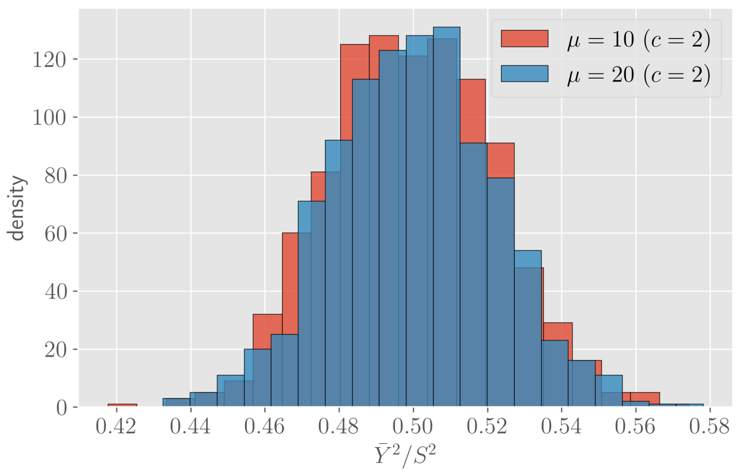


Figure 1: Simulation showing that the distribution of $\bar{Y}^2/S^2$ does not depend on μ. The figure was generated by taking samples of $\bar{Y}^2/S^2$ for two different means.

(3) (a) Draw a random sample of size $n = 20$ from a Cauchy$(0, 1)$ distribution. Compute $\bar{y}_n$. Repeat this $B = 1000$ times. Draw a histogram of the sampled $\bar{y}n$ values, superimposed over a Cauchy(0,1) density.

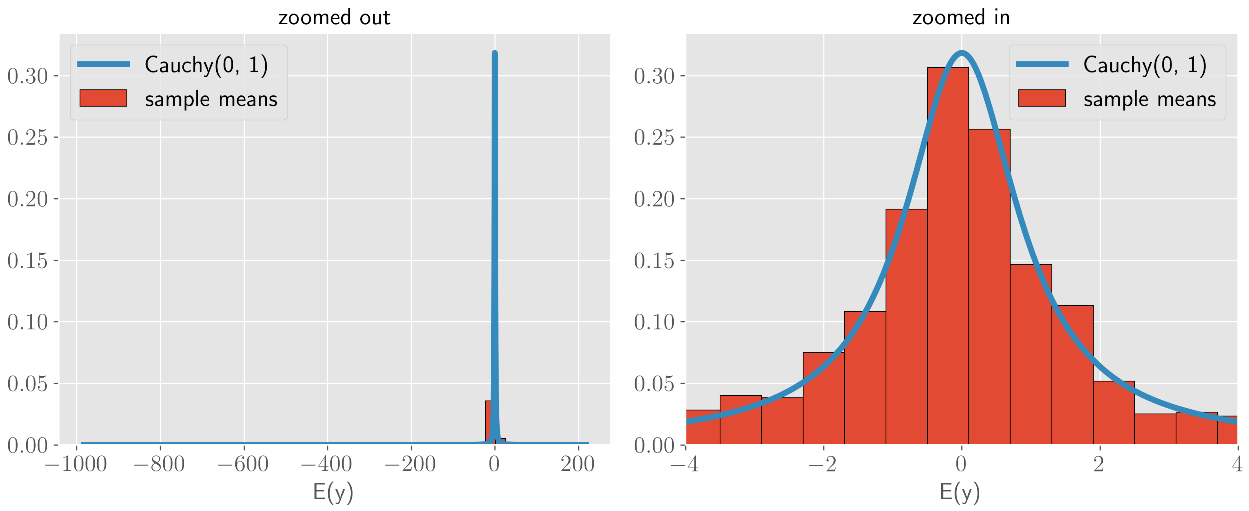


Figure 2: Histogram for question (3) (a). As expected, it fits well.

Estimate θ in Cauchy$(\theta, 1)$ using (b) step-wise gradient ascent, (c) Newton-Raphson, and (d) stochastic gradient ascent.

We need the first and second derivatives of the log likelihood to implement gradient ascent / Newton-Raphson / stochastic gradient ascent.

$$\begin{aligned} logL(\theta) &= log\prod_{i=1}^{n}\frac{1}{\pi}\frac{1}{1+(y_i-\theta)^2} \\ &= \sum_{i=1}^{n} log\frac{1}{\pi}\frac{1}{1+(y_i-\theta)^2} \end{aligned}$$

$$\begin{aligned} \frac{d}{d\theta}logL(\theta) &= \frac{d}{d\theta}\sum_{i=1}^{n} log\frac{1}{\pi}\frac{1}{1+(y_i-\theta)^2} \\ &= \sum_{i=1}^{n}\frac{2(y_i-\theta)}{1+(y_i-\theta)^2} \end{aligned}$$

$$\begin{aligned} H(\theta) &= \sum_{i=1}^{n}\frac{2(y_i-\theta)}{1+(y_i-\theta)^2} \\ &= \sum_{i=1}^{n}\frac{2[(y_i-\theta)^2-1]}{[1+(y_i-\theta)^2]^2} \end{aligned}$$

For gradient ascent and stochastic gradient ascent, we have:

$$\theta^{(m+1)} = \theta^{(m)} + \gamma\frac{\partial\mathcal{L}(\theta^{(m)})}{\partial\theta}$$

For stochastic gradient ascent, we randomly choose a single sample to evaluate the log likelihood at each iteration. For Newton-Raphson (this is the general multidimensional form, we only have one unknown here):

$$\theta^{(m+1)} = \theta^{(m)} - H(\theta^{(m)})^{-1}\frac{\partial\mathcal{L}(\theta^{(m)})}{\partial\theta}$$

Note that the log likelihood was not strictly concave (its second derivative is not strictly negative). Therefore, the optimizations were repeated several times at different starting values and the run resulting in the highest MLE was chosen. The scripts are attached to this homework.

Newton-Raphson converged in 3 iterations (given the tolerance specified). The two other algorithms took larger number of iterations. In terms of time, Newton-Raphson was also the fastest algorithm to converge.

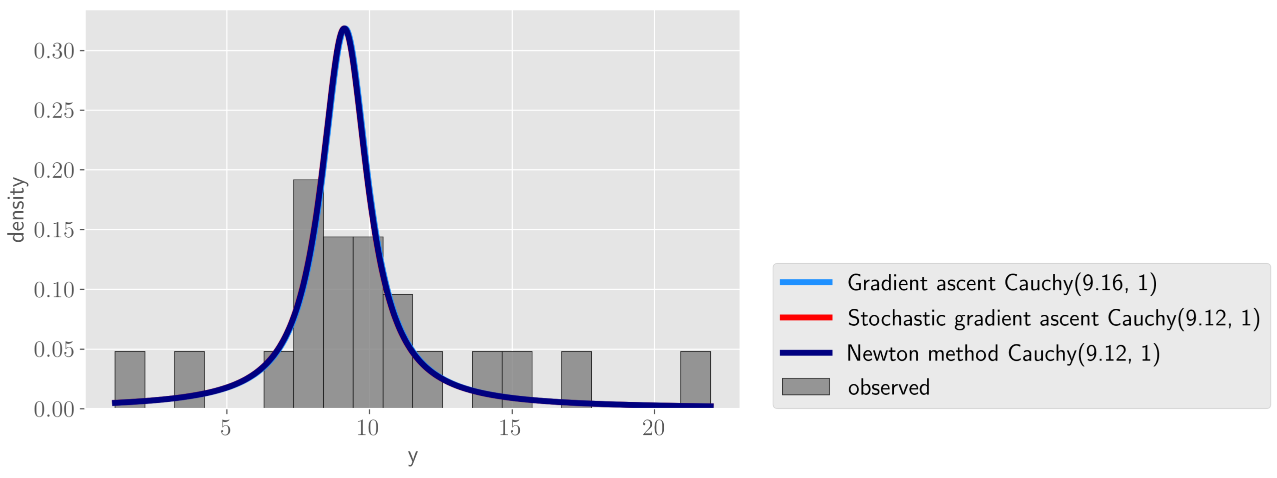


Figure 3: Cauchy(θ, 1) for the three methods using their θ_{MLE}

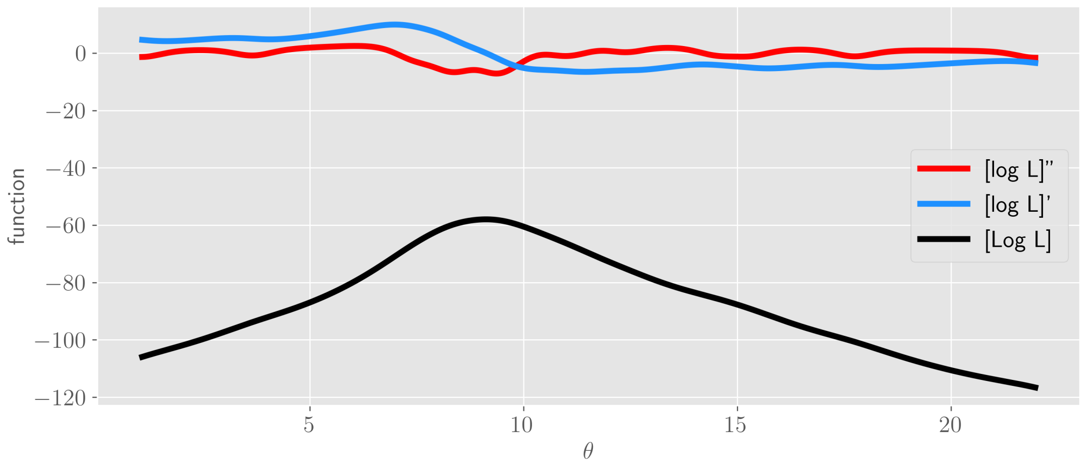


Figure 4: Log likelihood and its first and second derivatives

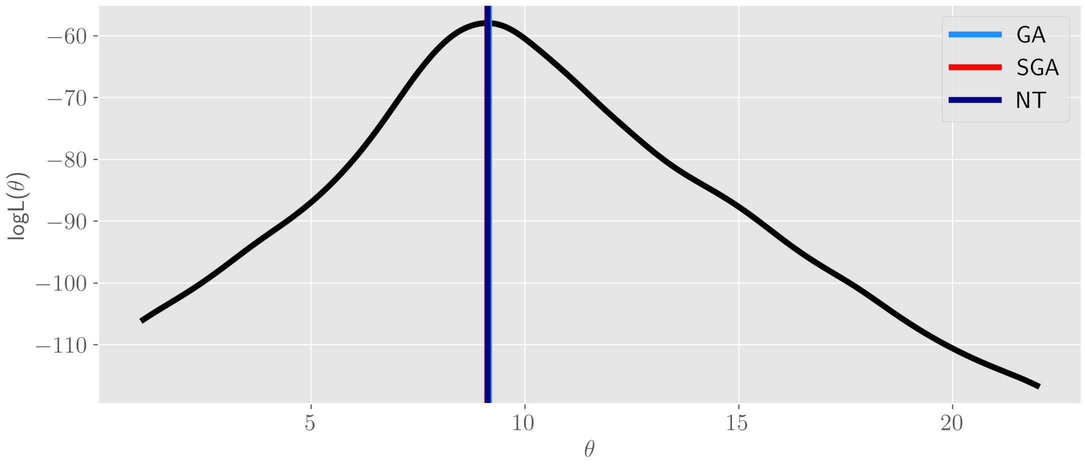


Figure 5: Log likelihood and the θ values for the three optimizers (they overlap).

(4) Let $y_{i,j}$ ind $Normal(\mu_i, \sigma^2)$, $i = 1, ..., n$ and $j = 1, ..., m$ with $n \to \infty$ but m fixed. Find out the MLEs of μ_i and σ^2. Show that $\hat{\sigma}^2$ MLE is inconsistent for σ^2. Adjust this estimator to come up with a consistent estimator of σ^2.

We have

$$\begin{aligned}
L(\mu, \sigma^2) &= (2\pi\sigma^2)^{-n/2}\text{exp}\left(-\frac{\sum(y_i - \mu)^2}{2\sigma^2}\right) \\
\mathcal{L}(\mu, \sigma^2) &= -\frac{n}{2}log(2\pi) - \frac{n}{2}log(\sigma^2) - \frac{1}{2\sigma^2}\sum(y_i - \mu)^2
\end{aligned}$$

and

$$\begin{aligned}
\frac{\partial\mathcal{L}(\mu, \sigma^2)}{\partial\mu} &= 0 \\
-\frac{1}{2\sigma^2}\sum -2(y_i - \mu) &= 0 \\
\sum y_i - n\mu &= 0 \\
\mu_{MLE} &= \bar{y}
\end{aligned}$$

$$\begin{aligned}
\frac{\partial\mathcal{L}(\mu, \sigma^2)}{\partial\sigma^2} &= 0 \\
-\frac{n}{2\sigma^2} + \frac{1}{2\sigma^4}\sum(y_i - \mu)^2 &= 0 \\
-\frac{n}{2} + \frac{1}{2\sigma^2}\sum(y_i - \mu)^2 &= 0 \\
-\sigma^2 n + \sum(y_i - \mu)^2 &= 0 \\
\sigma^2_{MLE} &= \sum(y_i - \mu)^2/n
\end{aligned}$$

In practice we would replace μ with $\bar{y}$. But:

$$\begin{aligned}
E(\sigma^2_{MLE}) &= E\left(\sum (y_i - \bar{y})^2/n\right) \\
&= \frac{1}{n}E\left(\sum y_i^2 - 2\bar{y}\sum y_i + \bar{y}n\right) \\
&= \frac{1}{n}E\left(y_i^2 - 2\bar{y}^2 n + \bar{y}^2 n\right) \\
&= \frac{1}{n}E\left(\sum y_i^2 - \bar{y}^2 n\right) \\
&= \frac{1}{n}E\left(\sum y_i^2 - \bar{y}^2 n\right) \\
&= E(y^2) - E(\bar{y}^2) \\
&= var(y) + E(y)^2 - var(\bar{y}) - E(\bar{y})^2 \\
&= var(y) - var(\bar{y}) \\
&= var(y) - var(y)/n \\
&= \frac{(n-1)\sigma^2}{n}
\end{aligned}$$

So, a better estimator is $\sum(y_i - \mu)^2/(n-1)$.

(5) For $y_1, ..., y_n \sim Uniform(0, \theta)$, show that $z_n = \frac{n(\theta - \hat{\theta}_{MLE})}{\theta} \xrightarrow{D} E$, where E is exponential. Design a simulation study illustrating this nice result.

We have, for $y \in [0, \theta]$

$$\begin{aligned}
f(y) &= 1/\theta \\
L(\theta) &= 1/\theta^n
\end{aligned}$$

In order to maximize $L(\theta)$, it is obvious that $\theta = max(y_i)$ because if it is less than $max(y_i)$, samples will be left out (which will have probability 0).

$$\begin{aligned}
P\left(\frac{n[\theta-\hat{\theta}_{MLE}]}{\theta}\le t\right) &= 1-P\left(\frac{n[\theta-\hat{\theta}_{MLE}]}{\theta}\ge t\right)\\
&= 1-P\left(\hat{\theta}_{MLE}\le\frac{\theta[t-n]}{n}\right)\\
&= 1-P\left(\hat{\theta}_{MLE}\le\frac{\theta[t-n]}{n}\right)\\
&= 1-P\left(max(y)\le\frac{\theta[t-n]}{n}\right)\\
&= 1-\prod P\left(y_i\le\frac{\theta[t-n]}{n}\right)\\
&= 1-\left(\frac{\frac{\theta[t-n]}{n}}{\theta}\right)^n\\
&= 1-(1-t/n)^n
\end{aligned}$$

The RHS is the cdf of $exp(1)$, so we have proven that $z_n = \frac{n(\theta-\hat{\theta}_{MLE})}{\theta} \xrightarrow{D} E$. The figure below illustrates this result.

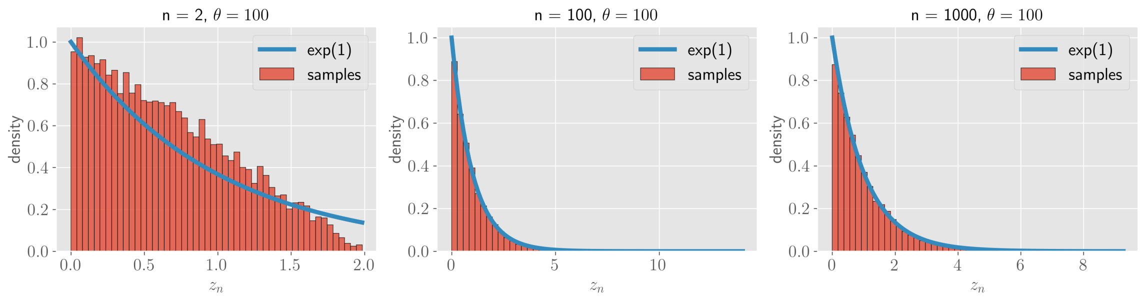


Figure 6: Simulation illustrating proof shown above.

(6) Using the method of maximum likelihood, fit a Weibull distribution to these data . Also find an approximate 90% joint confidence region for the parameters (k, λ). Supplement your analysis with appropriate plots.

Gradient ascent did not converge well for this, so I used Newton-Raphson. The log likelihood, first derivatives and observed information were:

$$\begin{aligned}
\mathcal{L}(k,\lambda) &= \sum \log\frac{k}{\lambda}\left(\frac{x_i}{\lambda}\right)^{k-1}e^{-(x_i/\lambda)^k}\\
&= n\log k - n\log\lambda + (k-1)\sum\log(x_i/\lambda) - \sum(x_i/\lambda)^k
\end{aligned}$$

$$\begin{aligned}
\frac{\partial}{\partial k}\mathcal{L}(k,\lambda) &= \frac{\partial}{\partial k}\mathcal{L} n\log k - n\log\lambda + (k-1)\sum\log(x_i/\lambda) - \sum(x_i/\lambda)^k\\
&= n/k + \sum\log(x_i/\lambda) - \sum(x_i/\lambda)^k\log(x_i/\lambda)
\end{aligned}$$

$$\begin{aligned}\frac{\partial}{\partial\lambda}\mathcal{L}(k,\lambda) &= \frac{\partial}{\partial\lambda} n\text{log}k - n\text{log}\lambda + (k-1)\sum \log(x_i/\lambda) - \sum (x_i/\lambda)^k \\ &= -nk/\lambda + k/\lambda \sum (x_i/\lambda)^k\end{aligned}$$

The observed information was:

$$\begin{bmatrix} n/k^2 + \sum (y_j/\lambda)^k \{\log(y_j/\lambda)\}^2 & \lambda^{-1}\sum[1-(y_j-\lambda)^k\{1+k\log(y_j/\lambda)\}] \\ \lambda^{-1}\sum[1-(y_j-\lambda)^k\{1+klog(y_j/\lambda)\}] & k(k+1)/\lambda^2\sum(y_j/\lambda)^k - nk\lambda^{-2}\end{bmatrix}$$

Using Newton, I found $(\lambda, k) = (181.4, 5.98)$. The plots below show the log likelihood and the Weibull pdf with the mle parameters and observed data.

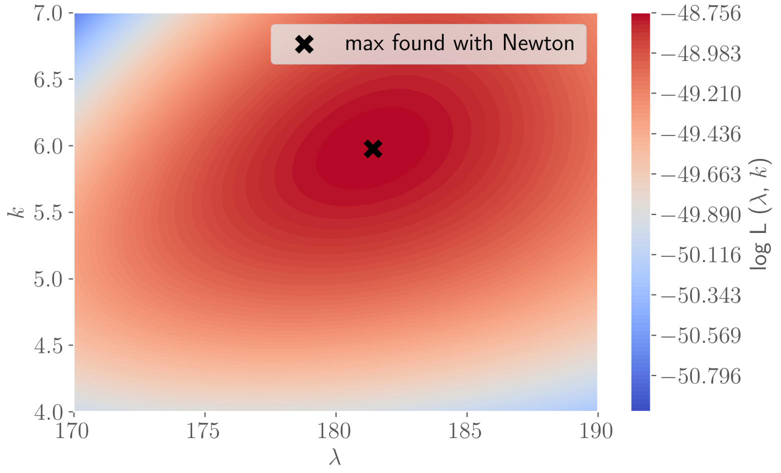


Figure 7: Log likelihood and mle parameters

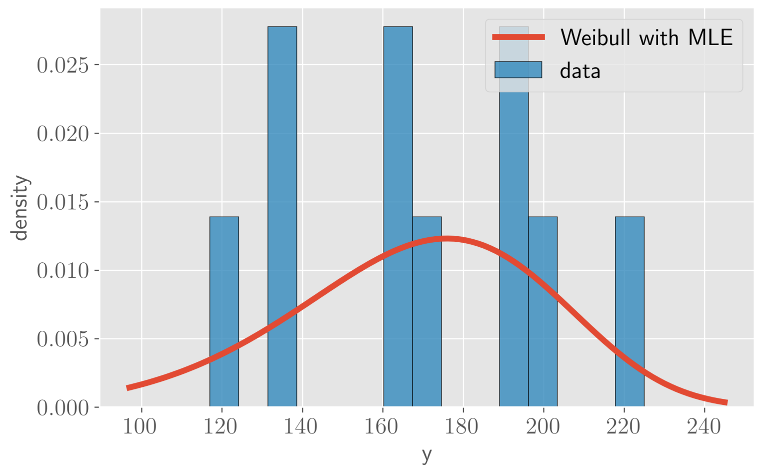


Figure 8: Histogram of the data and density found using MLE.

To calculate the standard error, the observed information can be used as an approximation. We also use the fact that $(\theta_{MLE} - \theta)^T J(\theta_{MLE})(\theta_{MLE} - \theta) \approx \chi^2_2$. χ^2_α for $\alpha = 0.1$ (right tailed) is 4.605. J is evaluated at θ_{MLE}. We then have:

$$P[(\theta_{MLE} - \theta)^T J(\theta_{MLE})(\theta_{MLE} - \theta) \leq 4.605] = 0.90$$

So we want the ellipse satisfying:

$$(\theta_{MLE} - \theta)^T J(\theta_{MLE})(\theta_{MLE} - \theta) = 4.605$$

To draw this ellipse, we use the Cholesky decomposition of J.

$$\begin{aligned}
&(\theta_{MLE} - \theta)^T J(\theta_{MLE})(\theta_{MLE} - \theta) \\
&= (\theta_{MLE} - \theta)^T LL^T(\theta_{MLE})(\theta_{MLE} - \theta) \\
&= [L^T(\theta_{MLE} - \theta)^T][L^T(\theta_{MLE})(\theta_{MLE} - \theta)]
\end{aligned}$$

First we take linearly spaced angles ϕ between 0 and 2π and transform them using:

$$X = \theta_{MLE} + 4.605^{1/2} L^{-T}[cos\phi, sin\phi]$$

This results in the figure shown below.

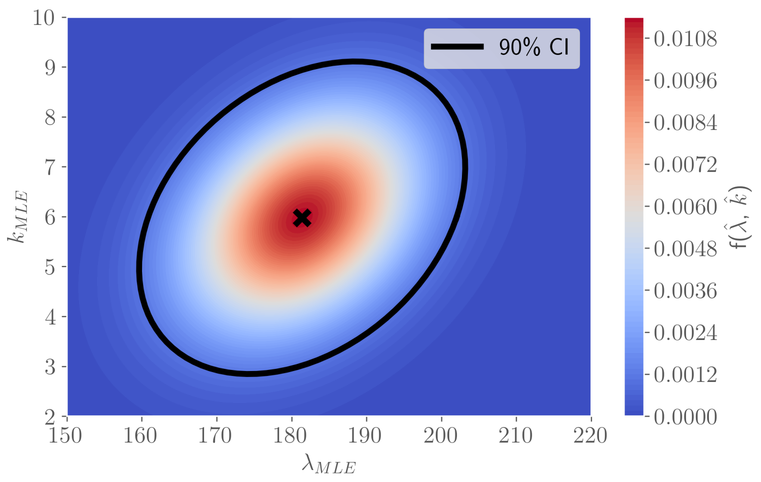


Figure 9: Standard error ellipse for the mle parameters.

(7) For a Weibull(k, λ) distribution, design a simulation study to show that $(\theta_{MLE} - \theta)^T J(\theta_{MLE})(\theta_{MLE} - \theta) \xrightarrow{D} \chi^2_2$ and $W(\theta) \xrightarrow{D} \chi^2_2$

.

I built the simulations using $(\lambda, k) = (0.5, 2)$ and re-used the Newton-Raphson code I had used for the previous question. I already had a function to calculate J. I ran it for 1,000 iterations with $5,000$ samples in each. The plots below show that $(\theta_{MLE} - \theta)^T J(\theta_{MLE})(\theta_{MLE} - \theta) \xrightarrow{D} \chi^2_2$ and $W(\theta) \xrightarrow{D} \chi^2_2$.

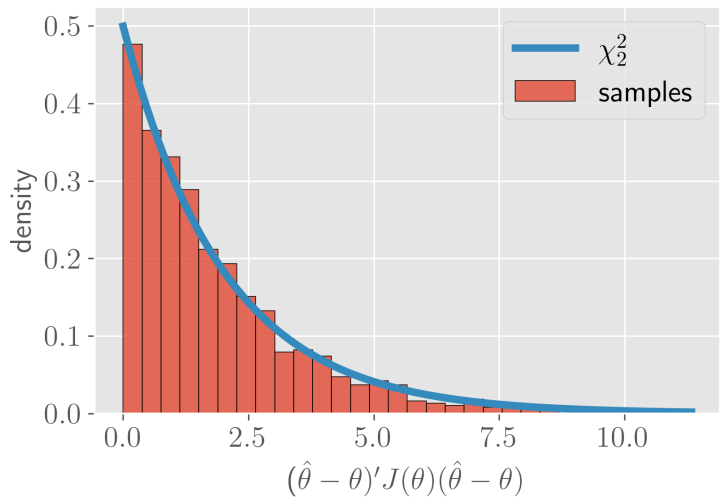


Figure 10: First convergence in distribution simulation.

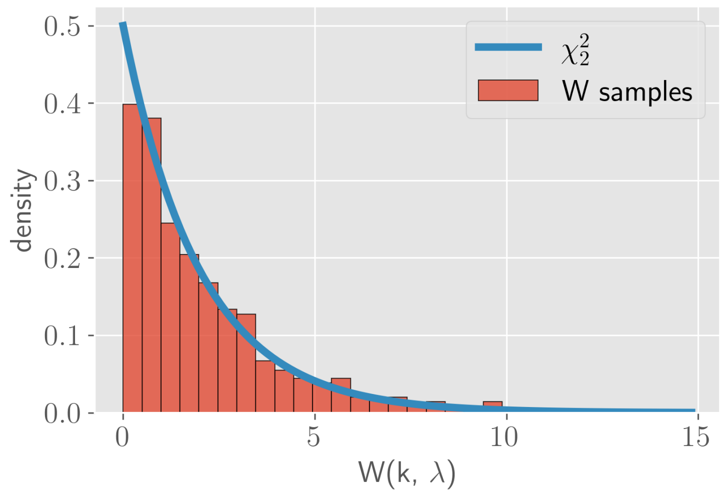
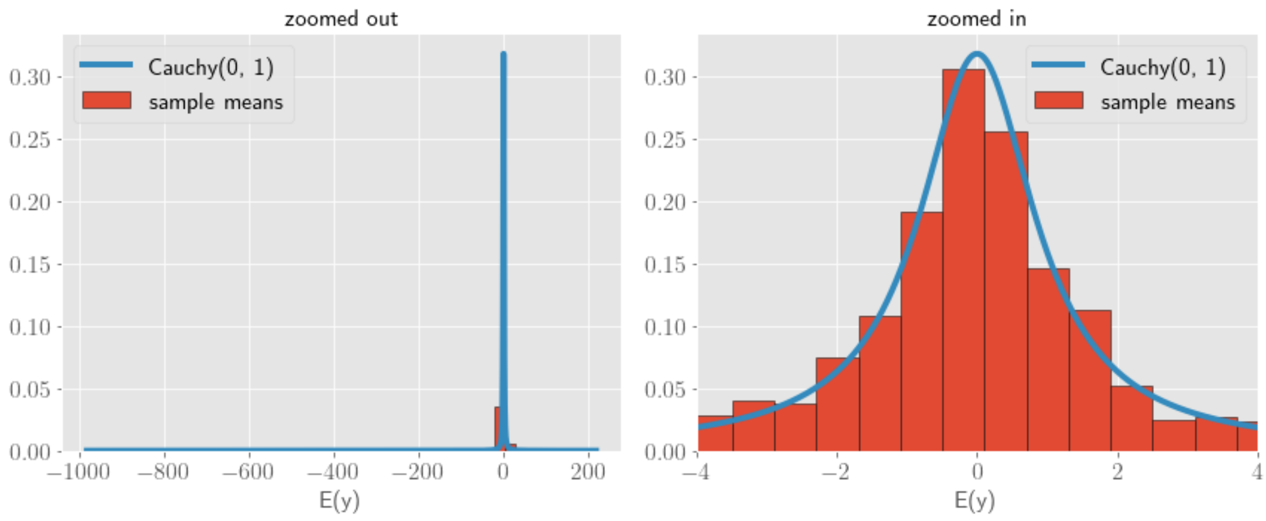


Figure 11: Second convergence in distribution simulation.

In [1]:
```
import matplotlib.pyplot as plt
import numpy as np
from matplotlib import rc
rc('text', usetex=True)
plt.style.use('ggplot')
plt.rcParams.update({'axes.labelsize':16})
plt.rcParams.update({'axes.titlesize':16})
plt.rcParams.update({'legend.fontsize':16})
plt.rcParams['xtick.labelsize'] = 16
plt.rcParams['ytick.labelsize'] = 16
plt.rcParams['lines.linewidth'] = 4
from scipy.stats import cauchy
import os
os.chdir('../')
from python_code import optimizations as opt
```

In [11]:
```
n, B = 20, 1000
dist = cauchy(loc=0, scale=1)
means = dist.rvs(size=(n, B)).mean(axis=0)
x = np.linspace(means.min(), means.max(), 500000)

fig, ax = plt.subplots(ncols=2, figsize=(12, 5))
ax[0].hist(means, bins=50, edgecolor='black', density=True, label='sample means'
ax[0].set_title('zoomed out')
ax[0].plot(x, dist.pdf(x), label='Cauchy(0, 1)')

ax[1].hist(means, bins=2000, edgecolor='black', density=True, label='sample mean
ax[1].set_title('zoomed in')
ax[1].plot(x, dist.pdf(x), label='Cauchy(0, 1)')
ax[0].legend()
ax[1].legend()
ax[0].set_xlabel('E(y)')
ax[1].set_xlabel('E(y)')
ax[1].set_xlim(-4, 4)
fig.tight_layout()
plt.savefig('figures/cauchy_mean.png', dpi=300, bbox_inches='tight')
```



In [3]:
```
observed_data = np.array([7.52, 9.92, 9.52, 21.97, 8.39, 8.09, 9.22, 9.37, 7.33,
```

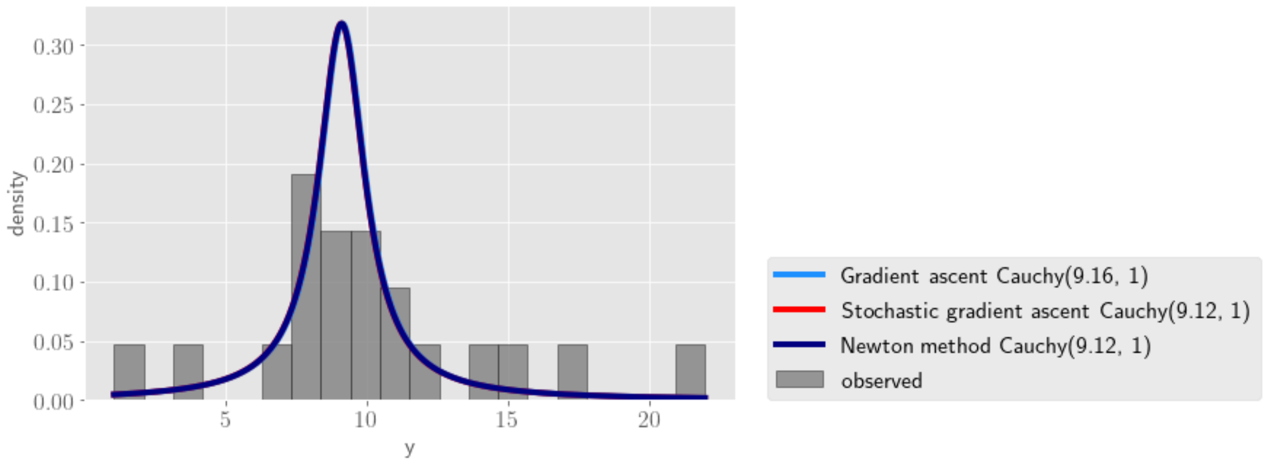
$$logL(\theta) = log\prod_{i=1}^{n}\frac{1}{\pi}\frac{1}{1+(y_i-\theta)^2} \tag{1}$$
$$= \sum_{i=1}^{n} log\frac{1}{\pi}\frac{1}{1+(y_i-\theta)^2} \tag{2}$$

$$\frac{d}{d\theta}logL(\theta) = \frac{d}{d\theta}\sum_{i=1}^{n} log\frac{1}{\pi}\frac{1}{1+(y_i-\theta)^2} \tag{3}$$
$$= \sum_{i=1}^{n}\frac{2(y_i-\theta)}{1+(y_i-\theta)^2} \tag{4}$$

$$H(\theta) = \sum_{i=1}^{n}\frac{2(y_i-\theta)}{1+(y_i-\theta)^2} \tag{5}$$
$$= \sum_{i=1}^{n}\frac{2(\theta^2-2y_i\theta+y_i^2-1)}{[1+(y_i-\theta)^2]^2} \tag{6}$$

In [14]:
```
ga = opt.GradientAscent(y_obs=observed_data)
print(ga.fit())
sga = opt.StochasticGradientAscent(y_obs=observed_data)
print(sga.fit())
nt = opt.NewtonMethod(y_obs=observed_data)
print(nt.fit());
```

In [5]:
```
plt.figure(figsize=(8, 5))
plt.hist(observed_data, edgecolor='black', alpha=.8, density=True, bins=20, labe
x = np.linspace(observed_data.min(), observed_data.max(), 1000)
plt.plot(x, cauchy(ga.optimal_theta, 1).pdf(x), label=f'Gradient ascent Cauchy({
plt.plot(x, cauchy(sga.optimal_theta, 1).pdf(x), label=f'Stochastic gradient asc
plt.plot(x, cauchy(nt.optimal_theta, 1).pdf(x), label=f'Newton method Cauchy({np
plt.legend(loc=(1.05, 0))
plt.xlabel('y')
plt.ylabel('density')
plt.savefig('figures/mle_cauchy.png', dpi=300, bbox_inches='tight')
```

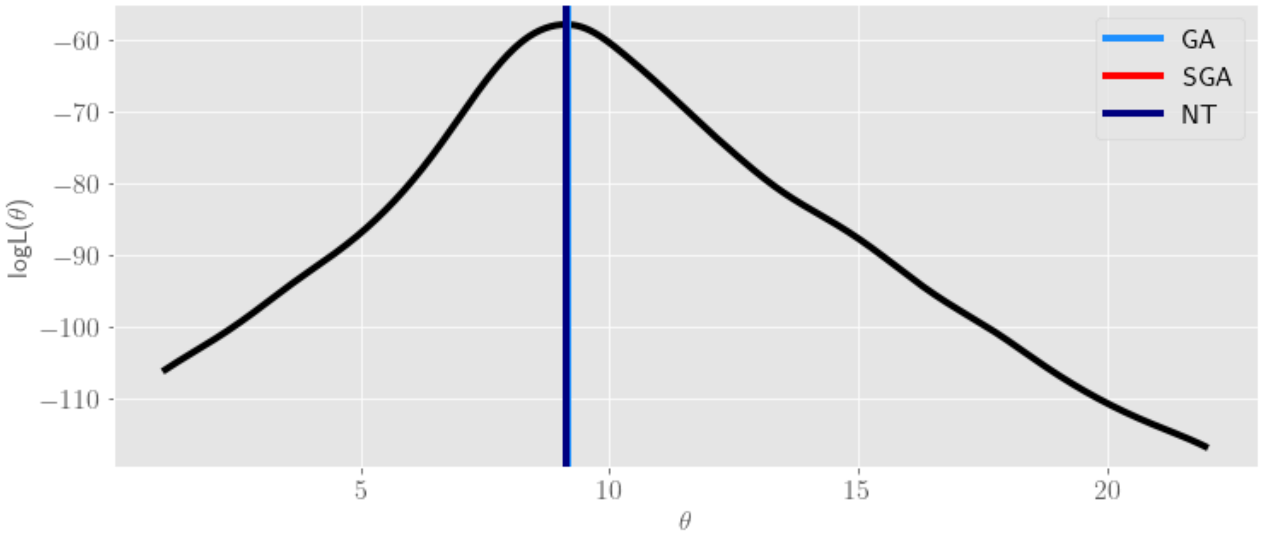


In [6]:
```
thetas = np.linspace(observed_data.min(), observed_data.max(), 1000)
```

```
log_likes = np.array([ga.calculate_log_likelihood(theta, observed_data) for thet
```

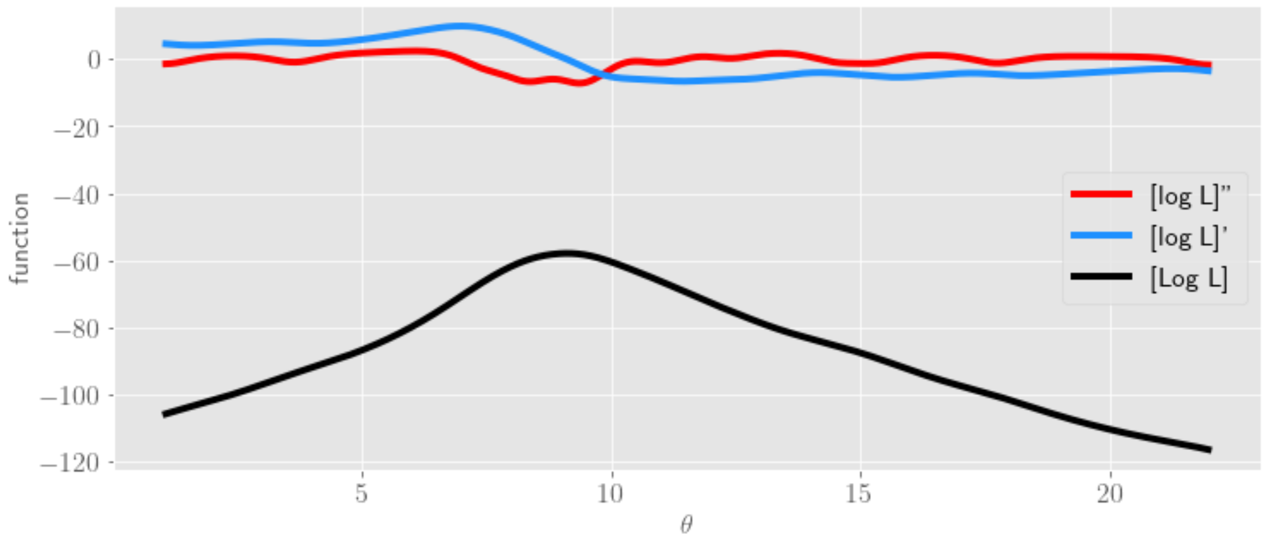
In [7]:

```
plt.figure(figsize=(12, 5))
plt.plot(thetas, log_likes, color='black')
plt.xlabel(r'$\theta$')
plt.ylabel(r'logL($\theta$)')
plt.axvline(x=ga.optimal_theta, color='dodgerblue', label='GA')
plt.axvline(x=sga.optimal_theta, color='red', label='SGA')
plt.axvline(x=nt.optimal_theta, color='navy', label='NT')
plt.legend()
plt.savefig('figures/cauchy_loglike.png', dpi=300, bbox_inches='tight')
```



In [8]:

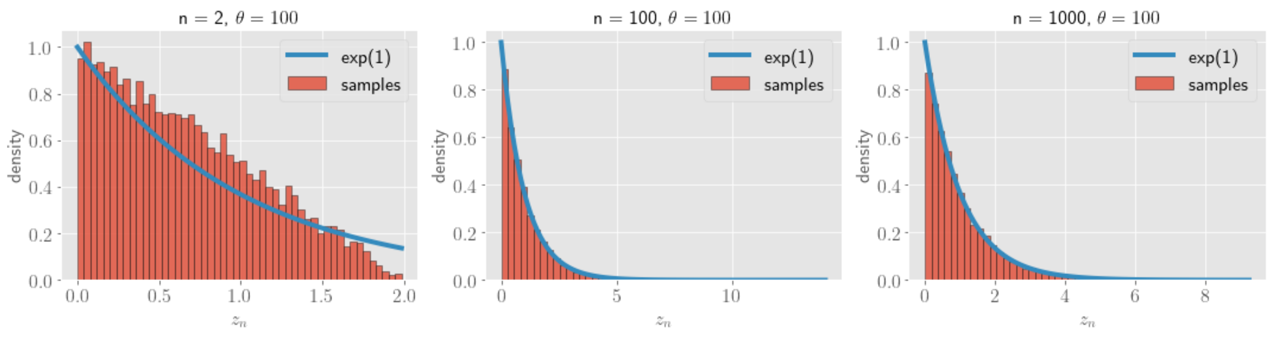
```
plt.figure(figsize=(12, 5))
hessians = np.array([nt.calculate_hessian(theta, observed_data) for theta in the
log_like = np.array([nt.calculate_derivative_log_likelihood(theta, observed_data
like = np.array([nt.calculate_log_likelihood(theta, observed_data) for theta in
plt.plot(thetas, hessians, color='red', label="[log L]''")
plt.plot(thetas, log_like, color='dodgerblue', label="[log L]'")
plt.plot(thetas, like, color='black', label="[Log L]")
plt.xlabel(r'$\theta$')
plt.ylabel(r'function')
plt.legend()
plt.savefig('figures/cauchy_fns.png', dpi=300, bbox_inches='tight')
# log likelihood is NOT concave! not strictly negative!!
```


0
−20
−40
−60
−80
−100
−120
5
10
15
20
θ
function
[log L]''
[log L]'
[Log L]

In [2]:
```
import matplotlib.pyplot as plt
import numpy as np
from matplotlib import rc
rc('text', usetex=True)
plt.style.use('ggplot')
plt.rcParams.update({'axes.labelsize':16})
plt.rcParams.update({'axes.titlesize':16})
plt.rcParams.update({'legend.fontsize':16})
plt.rcParams['xtick.labelsize'] = 16
plt.rcParams['ytick.labelsize'] = 16
plt.rcParams['lines.linewidth'] = 4
from scipy.stats import uniform, expon
```

Question 5

In [3]:
```
n_values = [2, 100, 1000]
true_theta = 100
fig, axes = plt.subplots(ncols=len(n_values), figsize=(15, 4))

for ax, n in zip(axes, n_values):
    samples = uniform(0, true_theta).rvs(size=(n, 10000))
    MLEs =  np.max(samples, axis=0)
    data = n*(true_theta - MLEs)/true_theta
    ax.hist(data, density=True, bins=50, label='samples', alpha=.8, edgecolor='b
    ax.set_title(fr'n = {n}, $\theta = {true_theta}$')
    x = np.linspace(data.min(), data.max(), 100)
    ax.plot(x, expon(scale=1).pdf(x),
            label='exp(1)')
    ax.legend()
    ax.set_xlabel('$z_n$')
    ax.set_ylabel('density')
fig.tight_layout()
plt.savefig('../figures/uniform.png', dpi=300, bbox_inches='tight')
```



In [1]:

```
import matplotlib.pyplot as plt
import numpy as np
from matplotlib import rc
rc('text', usetex=True)
plt.style.use('ggplot')
plt.rcParams.update({'axes.labelsize':16})
plt.rcParams.update({'axes.titlesize':16})
plt.rcParams.update({'legend.fontsize':16})
plt.rcParams['xtick.labelsize'] = 16
plt.rcParams['ytick.labelsize'] = 16
plt.rcParams['lines.linewidth'] = 4
from scipy.stats import weibull_min, multivariate_normal, chi2
import os
os.chdir('../')
from python_code import weibull_mle
import matplotlib as mpl
```

In [2]:

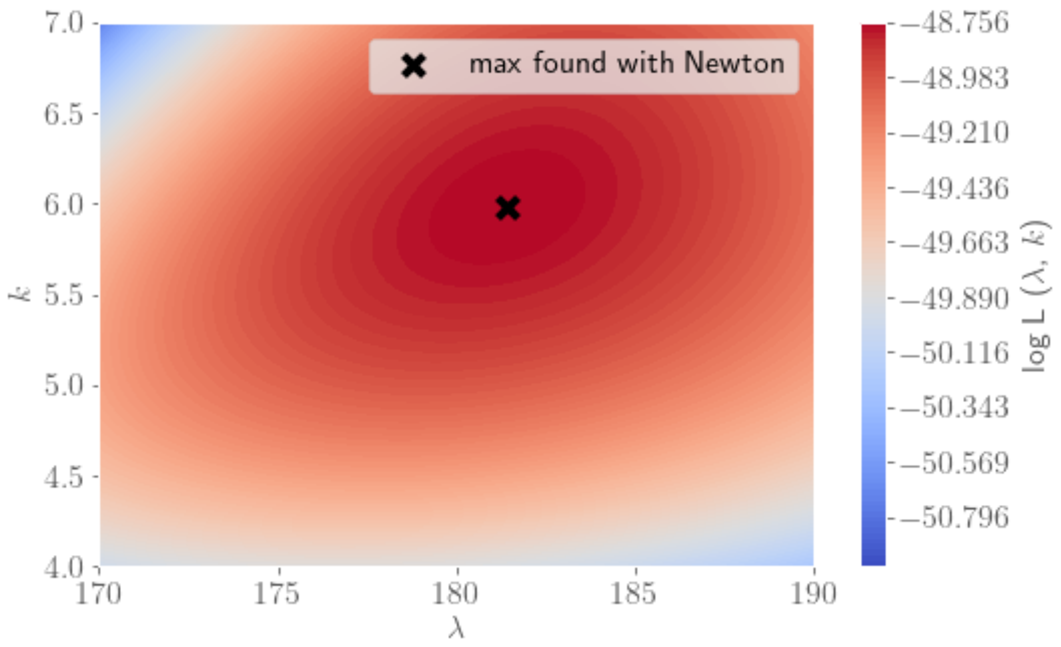
```
data = np.array([225,171,198,189,189,135,162,135,117,162])
opt = weibull_mle.WeibullMle(data, max_iter=1000)
params = opt.fit()[0:2]
params
```

Out[2]:

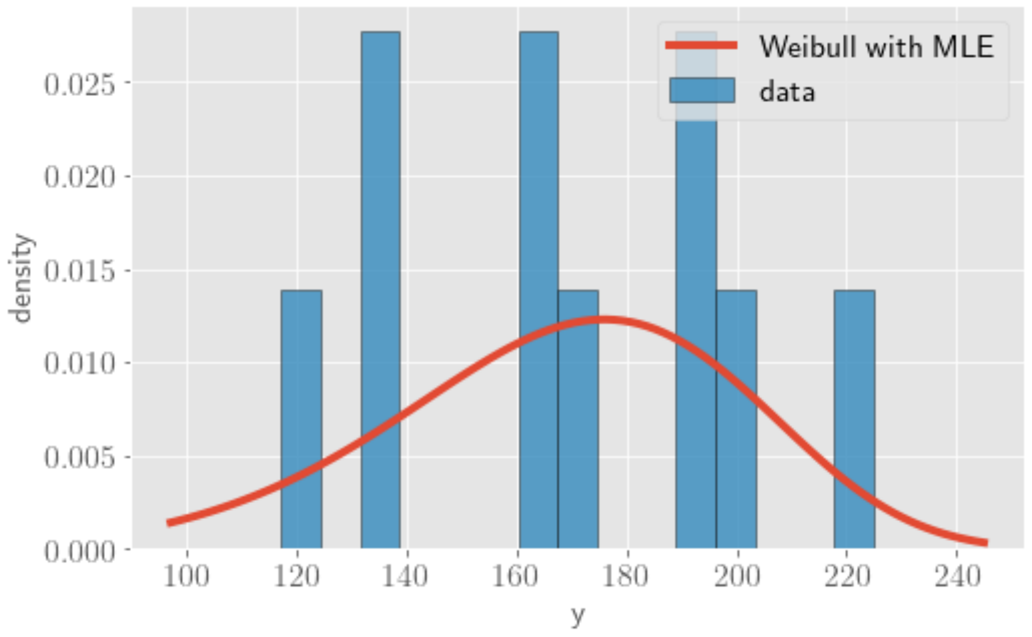
```
(181.40558666210944, 5.976921860040788)
```

In [5]:

```
plt.figure(figsize=(8, 5))
k_values = np.linspace(170, 190, 100)
lam_values = np.linspace(4, 7, 100)
k, lam = np.meshgrid(k_values, lam_values)
log_likes = np.array([opt.calculate_log_likelihood(lam_val, k_val)
                      for lam_val, k_val in zip(k.flatten(), lam.flatten())])



plt.scatter(opt.lam, opt.k, s=100, color='black', marker='x', zorder=3, label='m

c = plt.contourf(k, lam, np.array(log_likes).reshape(lam.shape), levels=np.linsp
cbar = plt.colorbar()
plt.xlabel('$\lambda$')
plt.ylabel('$k$');
cbar.set_label(r'log L ($\lambda$, $k$)')
plt.legend()
plt.savefig('figures/mle_weibull.png', dpi=300, bbox_inches='tight');
```

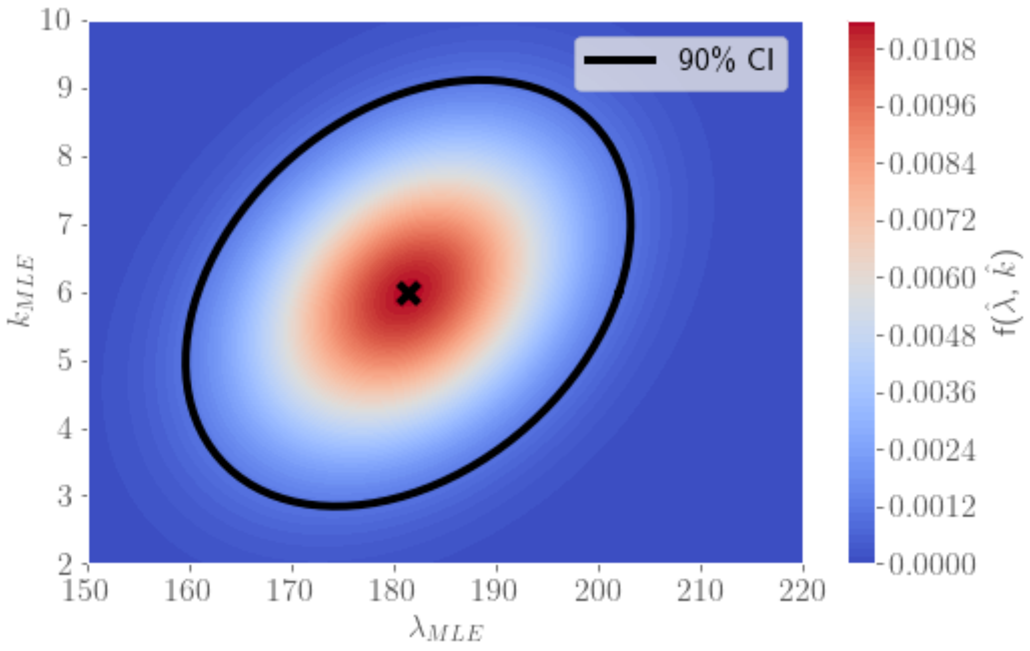


In [6]:
```
plt.figure(figsize=(8, 5))
dist = weibull_min(c=opt.k, scale=opt.lam)
x = np.linspace(data.min() - 20, data.max() + 20, 100)
plt.plot(x, dist.pdf(x), label='Weibull with MLE')
plt.hist(data, density=True, edgecolor='black', alpha=.8, bins=15, label='data')
plt.xlabel('y')
plt.ylabel('density')
plt.legend()
plt.savefig('figures/weibull_data_fit.png', dpi=300, bbox_inches='tight');
```



In [9]:
```
plt.figure(figsize=(8, 5))
info_matrix = -opt.calculate_hessian(*params)
se_dist = multivariate_normal(params, np.linalg.inv(info_matrix))
k_values = np.linspace(150, 220, 110)
lam_values = np.linspace(2, 10, 100)
k, lam = np.meshgrid(k_values, lam_values)
```

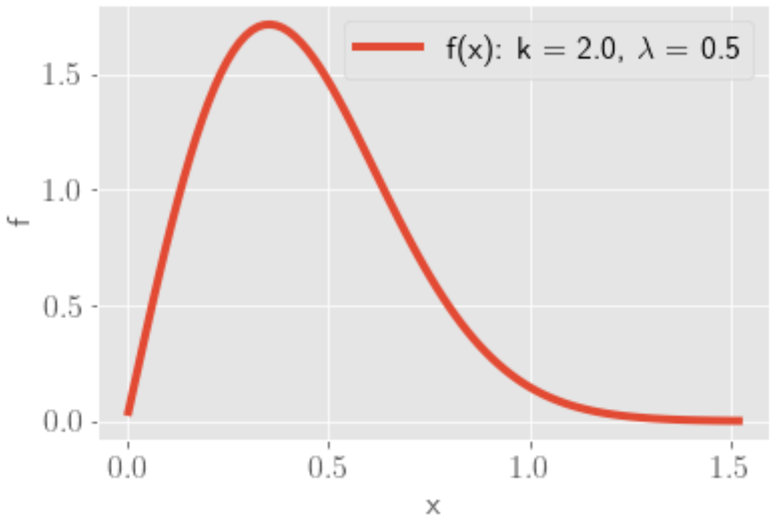
```
x = np.array([k.flatten(), lam.flatten()])
pdf = se_dist.pdf(x.T)
plt.contourf(k_values, lam_values, pdf.reshape(k.shape), cmap='coolwarm', levels
plt.xlabel('$\lambda_{MLE}$')
plt.ylabel('$k_{MLE}$');
cbar = plt.colorbar()
cbar.set_label(r'f($\hat{\lambda}$, $\hat{k}$)')
plt.scatter(opt.lam, opt.k, s=100, color='black', marker='x', zorder=3)

x = []
L = np.linalg.cholesky(info_matrix)
for angle in np.linspace(0, 2*np.pi, 100):
    x.append(params + np.sqrt(chi2(df=2).ppf(.90)) * np.linalg.inv(L.T) @ np.arr

x = np.array(x)
plt.plot(x[:, 0], x[:, 1], color='black', label='90\% CI')
plt.legend();
plt.savefig('figures/weibull_se.png', dpi=300, bbox_inches='tight');
```



In [1]:
```python
import matplotlib.pyplot as plt
import numpy as np
from matplotlib import rc
rc('text', usetex=True)
plt.style.use('ggplot')
plt.rcParams.update({'axes.labelsize':16})
plt.rcParams.update({'axes.titlesize':16})
plt.rcParams.update({'legend.fontsize':16})
plt.rcParams['xtick.labelsize'] = 16
plt.rcParams['ytick.labelsize'] = 16
plt.rcParams['lines.linewidth'] = 4
from scipy.stats import weibull_min, chi2
import os
os.chdir('../')
from python_code import weibull_mle
import matplotlib as mpl
from tqdm import tqdm
```

In [2]:
```python
lam_k_true = np.array([.5, 2])
dist = weibull_min(c=lam_k_true[1], scale=lam_k_true[0])
x = np.linspace(dist.ppf(.0001), dist.ppf(.9999), 100)
```

In [3]:
```python
plt.plot(x, dist.pdf(x), label=fr'f(x):  k = {lam_k_true[1]}, $\lambda$ = {lam_k
plt.xlabel('x')
plt.ylabel('f')
plt.legend();
```

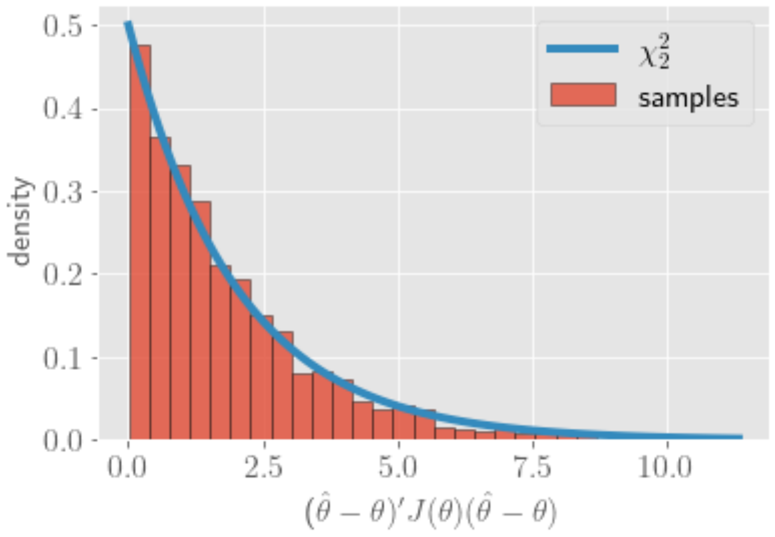


In [4]:
```python
n_points = 1000
statistics = np.zeros(n_points)
mle_params = np.zeros((n_points, 2))
for it in tqdm(range(n_points)):
    samples = dist.rvs(5000)
    opt = weibull_mle.WeibullMle(data=samples, max_iter=5000, init_params=[10, 1
    lam_k_mle = opt.fit()[:-1]
    J = -opt.calculate_hessian(*lam_k_mle)
    mle_params[it] = lam_k_mle
    statistics[it] = (lam_k_mle - lam_k_true).T@J@(lam_k_mle - lam_k_true)
```

```
100%|██████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████| 1000/1000 [00:27<00:00, 36.78it/s]
```

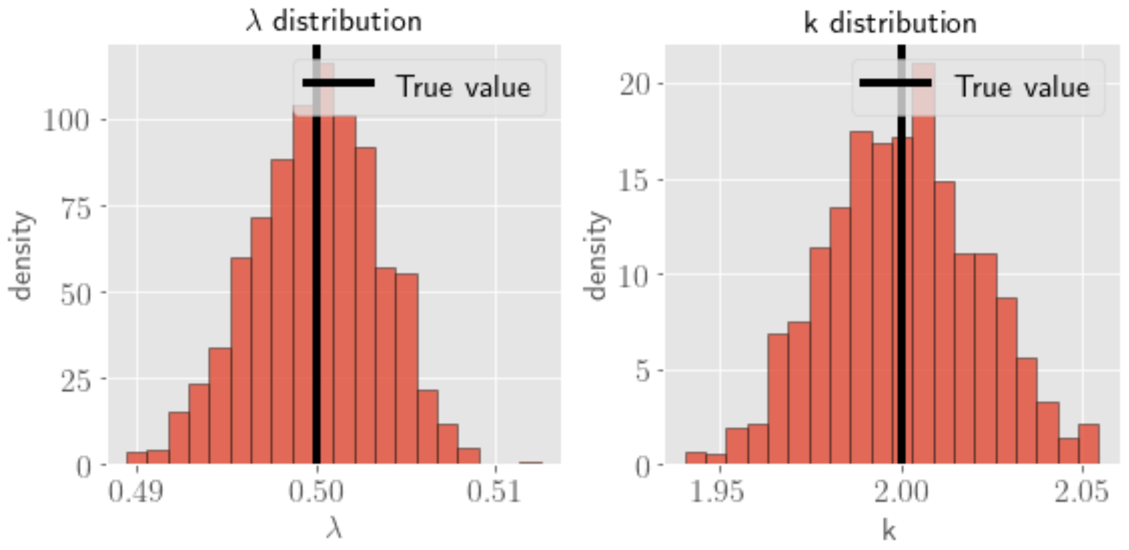
In [5]:

```
plt.hist(statistics, edgecolor='black', alpha=.8, bins=30, density=True, label='
x = np.linspace(statistics.min(), statistics.max(), 100)
plt.plot(x, chi2(2).pdf(x), label='$\chi^2_2$')
plt.legend()
plt.xlabel(r"($\hat{\theta} -\theta)'J(\theta)(\hat{\theta} -\theta)$")
plt.ylabel('density')
plt.savefig('figures/convergence_1.png', dpi=300, bbox_inches='tight');
```



In [6]:

```
fig, ax = plt.subplots(ncols=2, figsize=(8, 4))
ax[0].hist(mle_params[:, 0], edgecolor='black', alpha=.8, bins=20, density=True)
ax[0].set_title('$\lambda$ distribution')
ax[1].hist(mle_params[:, 1], edgecolor='black', alpha=.8, bins=20, density=True)
ax[1].set_title('k distribution')
ax[0].set_xlabel('$\lambda$')
ax[1].set_xlabel('k')
ax[0].axvline(x=lam_k_true[0], color='black', label='True value')
ax[1].axvline(x=lam_k_true[1], color='black', label='True value')
[ax.legend() for ax in ax];
[ax.set_ylabel('density') for ax in ax];
fig.tight_layout()
plt.savefig('figures/convergence_params.png', dpi=300, bbox_inches='tight');
```



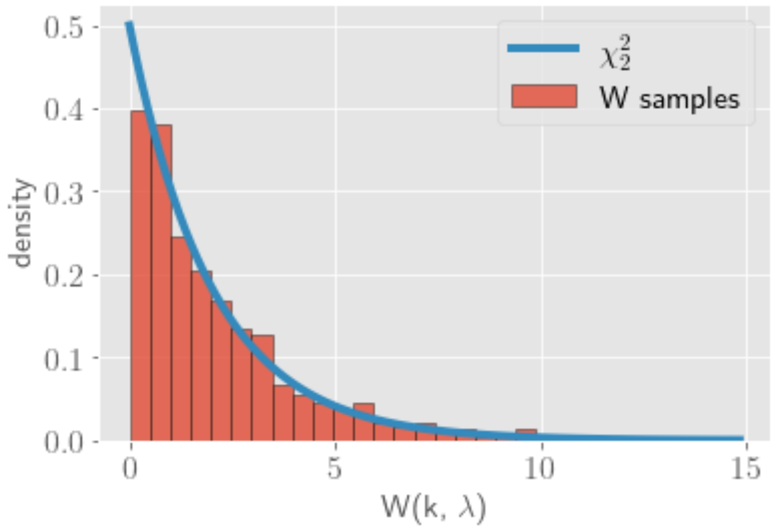
In [7]:

```
n_points = 1000
W_vals = np.zeros(n_points)
for it in tqdm(range(n_points)):
    samples = dist.rvs(5000)
    opt = weibull_mle.WeibullMle(data=samples, max_iter=5000, init_params=[10, 1
    output = opt.fit()
    lam_k_mle = output[:-1]
    log_like_mle = output[-1]
    log_like_true_params = opt.calculate_log_likelihood(*lam_k_true)
    W_vals[it] = 2*(log_like_mle-log_like_true_params)
```

```
100%|████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████
██████████████| 1000/1000 [00:26<00:00, 37.58it/s]
```

In [8]:

```
plt.hist(W_vals, edgecolor='black', alpha=.8, bins=30, density=True, label='W sa
x = np.linspace(W_vals.min(), W_vals.max(), 100)
plt.plot(x, chi2(2).pdf(x), label='$\chi^2_2$')
plt.legend()
plt.xlabel('W(k, $\lambda$)')
plt.ylabel('density')
plt.savefig('figures/convergence_2.png', dpi=300, bbox_inches='tight');
```


$\chi^2_2$
W samples
density
W(k, λ)