Homework 2:
Sampling Distribution and Statistical Learning

Agueci, Leonardo; Camacho, Evelyn; Cherchi, Gian Michele; Redaelli, Tommaso

October 26, 2018

Abstract

Results with graphs, plots and data were obtained with Python, working on Jupiter. The codes are included in this document.

1 Sampling Discrete Distributions

1.1 To answer point 1), 2) for the Uniform Distribution

As first step we write two procedures to sample from given distribution:

1. Accept/Reject;
2. Fast Tower Sampling.

The code contain the description used for each passage. For each bar plot we have extracted 100000 points as suggested.

1.1.1 Computational Results

Here we show the results of our computation, the comparison between theoretical expected values and what we've obtained.

Both results are in perfect agreement with theoretical distributions.

1.2 To answer point 3 for the Uniform Distribution

The accept-reject procedure computational time is extremely sensitive to the maximum bin of the distribution. It sample points inside a rectangle which has as basis: the number of item, which we are increasing, and the top of the distribution, this last calculated thanks to the max tool of the python math library. The acceptance rate of each number is proportional to

$$E[\frac{p}{p_{sup}}] \tag{1}$$

where p_{sup} is the highest probability in our distribution. This is clearly linear increasing $O(1)$ and is not affected by the number of Item N, as clearly visible from the plot.

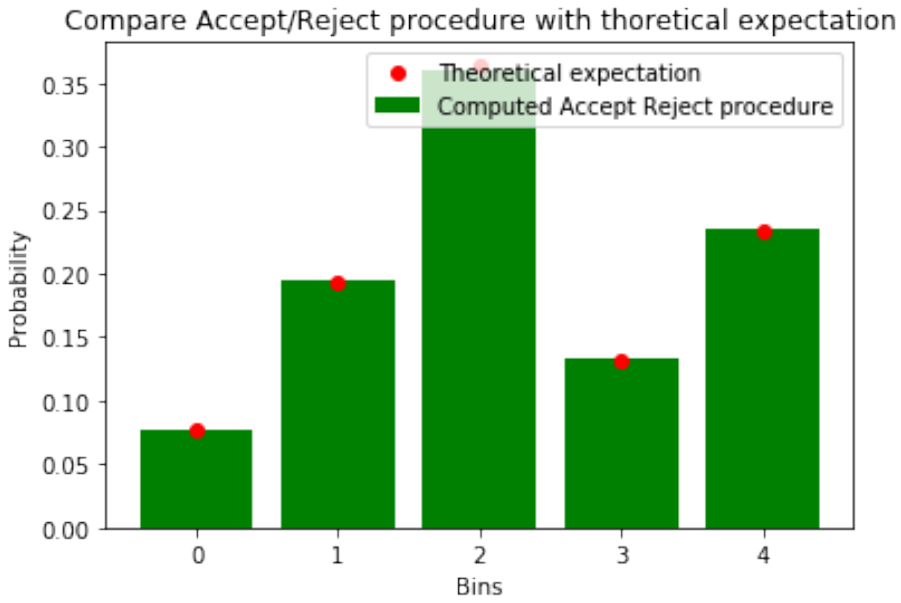

Figure 1: Comparison between Accept/reject sampling and theoretical expectation.

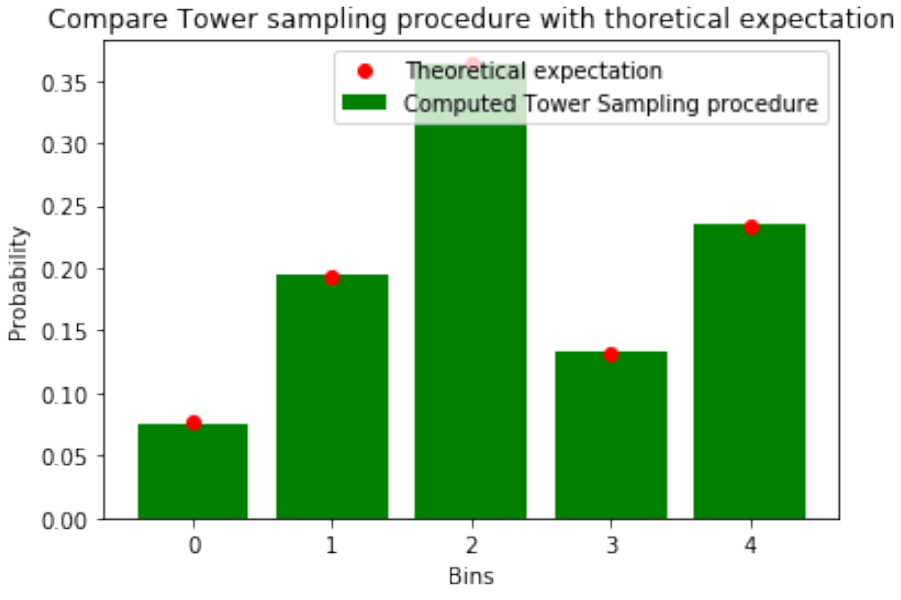

Figure 2: Comparison between Dichotomy Tower Sampling and theoretical expectation.

The dichotomy tower sampling is a divide and conquer algorithm which as to half the total tower of cumulatives on each iteration. This lead to a computational time strictly connected to the total number of cumulatives, hence to the total number of Item chosen. In particular, as for each divide and conquer algorithm, the computational time should increase as

$$O(log(N)) \tag{2}$$

Also this is extremely visible from the plot.

The initial damping of the accept-reject procedure is simply explained by the presence of a 0 as the first element of the list together with the log scale of the y-axes.

All considerations made here are strictly based considering an initial uniform distribution.

1.3 Exponential distribution

1.3.1 Theoretical approach

Here we derive a proper way to code the exponential sampling starting from a uniform random number generator.

Inverse transformation sampling takes uniform samples of a number u between 0 and 1, interpreted as a probability, and then returns the largest number x from the domain of the distribution $P(X)$ such that $P(-\infty<X<\infty) \leq u$.

Knowing this we can then consider the exponential function:

$$P(x) = e^{-x} \tag{3}$$

Thus:

$$F(x) = \int_0^x due^{-u} = 1 - e^{-x} \tag{4}$$

and, knowing that the distribution of $y = F(x)$ is the same as $1 - y$, we can easily deduce:

$$x^* = -\log y \tag{5}$$

.

Plugging this last inside the code we obtain a sequence of random variables distributed as an exponential distribution.

1.3.2 Computational Results

Here are the plots to compare theoretical expectation and computational results of the two procedures.

1.4 "Square Root" distribution

We sample now from a distribution of this shape

$$P(x) = \frac{1}{\sqrt{x}} \tag{6}$$

Figure 3: Computational time needed increasing number of items, uniform distribution.

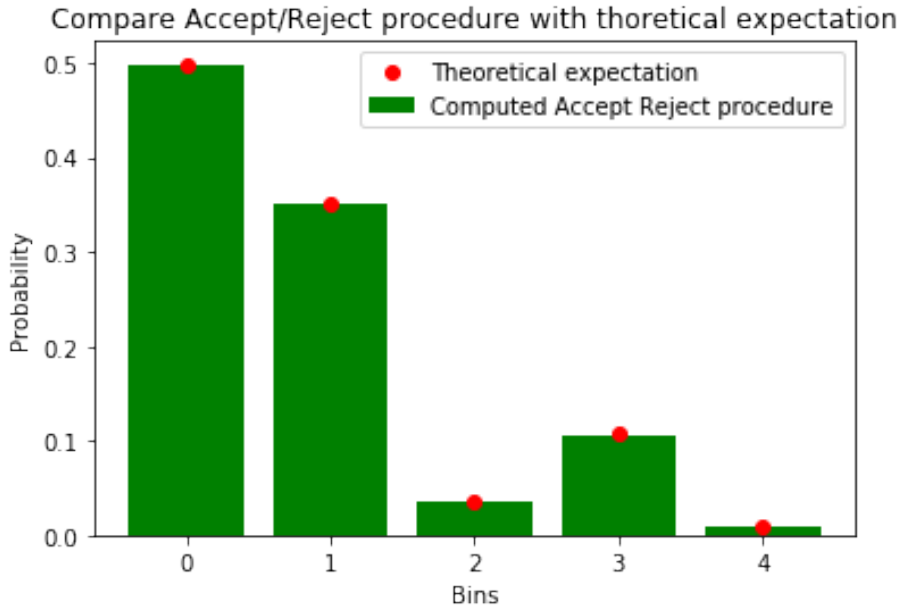

Figure 4: Comparison between Accept/reject sampling and theoretical expectation with exponential distribution.

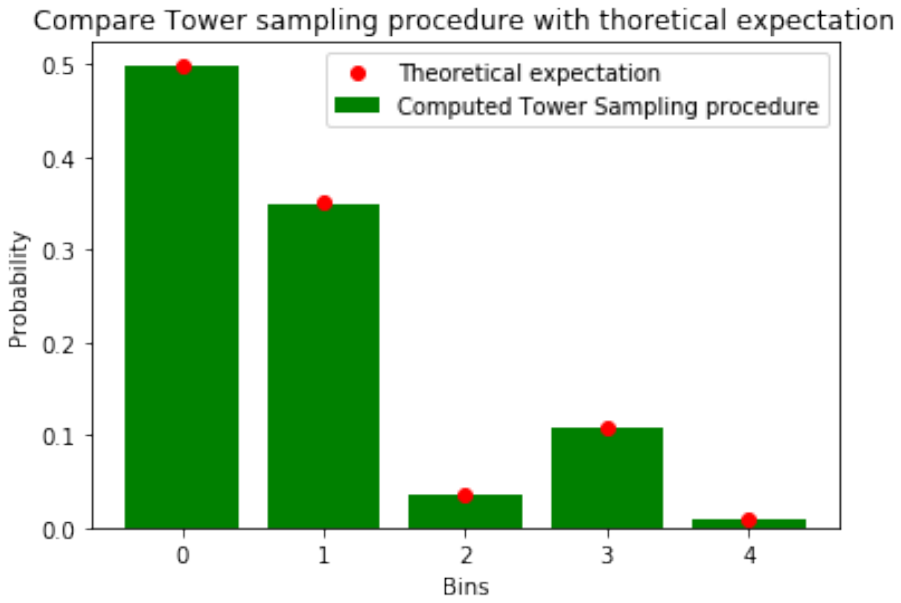

Figure 5: Comparison between Dichotomy Tower Sampling and theoretical expectation with exponential distribution.

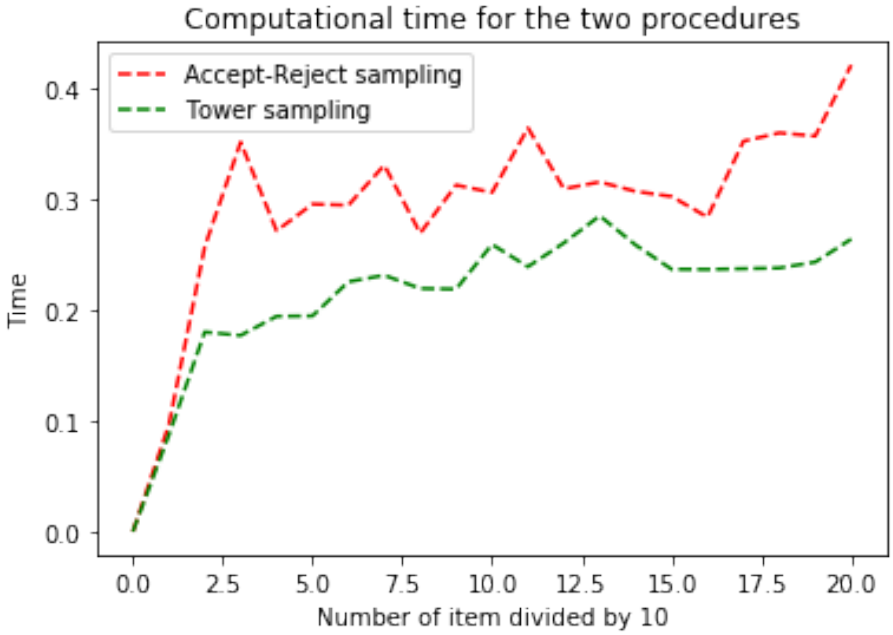

Figure 6: Computational time needed increasing number of items, exponential distribution.

As before, we used the method of the inverse transform, applied to the power-law distribution here written:

$$P(y) = \frac{2}{|y|^3} \tag{7}$$

And we obtain the following results.

1.4.1 Computational Results

Here are the plots to compare theoretical expectation and computational results of the two procedures.

Computational time needed plot.

1.5 Conclusion: Computational time overview

What is really interesting is the variation in computational time in the three different situatiorns, due solely to the differences in the probability density function chosen.//

In detail, studying the behaviour of the maximum density inside these three types of distribution, we can assume that:

1. For the uniform distribution: All the bins are equally probable, so the computational time behaviour with respect to the number of item increasing, as previously shown, should be linear for the accept/reject procedure and log-like for the tower sampling procedure;

2. For the exponential distribution: let's study the density distribution of the distribution maxima with respect to the number of Item increasing. In this kind of distribution is really easy to compute the log(N) increasing of the maximum, the ratio between the two sample area for the accept reject method goes like:

$$p \sim \frac{N_{item}}{N_{item} * \log N_{item}} \tag{8}$$

   So the time grows as $\log N_{item}$.

3. The square root distribution maxima follows a rule proportional to the $\sqrt{N_{item}}$, so the computational time grows as $\sqrt{N_{item}}$.

   All these results are clearly visible in the plot here shown.

1.5.1 Last comment about the tower sampling computational time

The tower sampling complete function is composed both by a dichotomy algorithm, performing computational at time grows as $\log N$, and by a *for* cycle, useful to compute the cumulatives, which time grows as N. So, the behaviour is strictly dependent on the position of the *for* loop inside the code.

Here we have performed all the simulations computing only the time for the dychotomic part ot the sampling, throwing the *for* loop needed time. All results are shown below.

The only configuration in which the tower sampling wins is for the exponential distribution and this is clearly understandable by the consideration made on the asymptotic behaviour of computational time for large N_{item}.

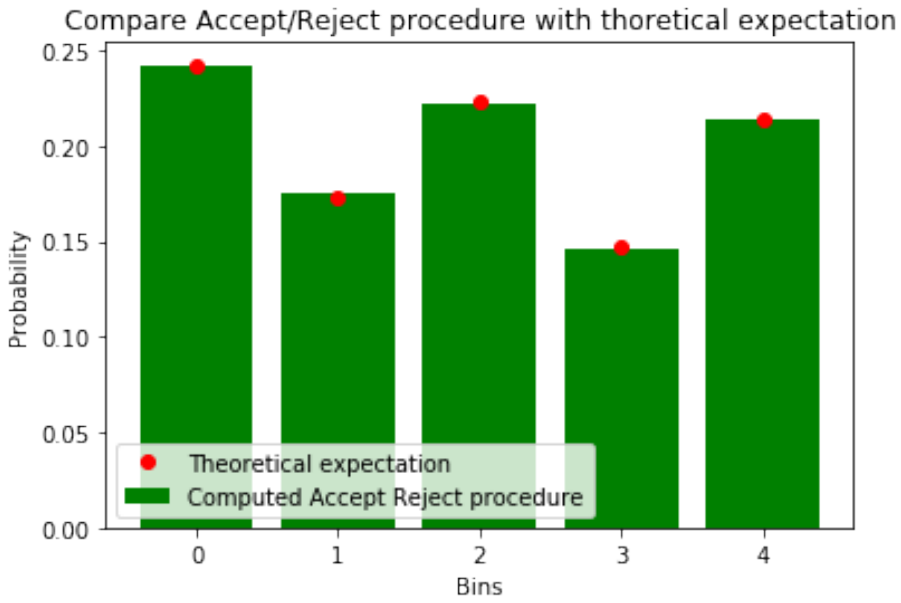

Figure 7: Comparison between Accept/reject sampling and theoretical expectation.

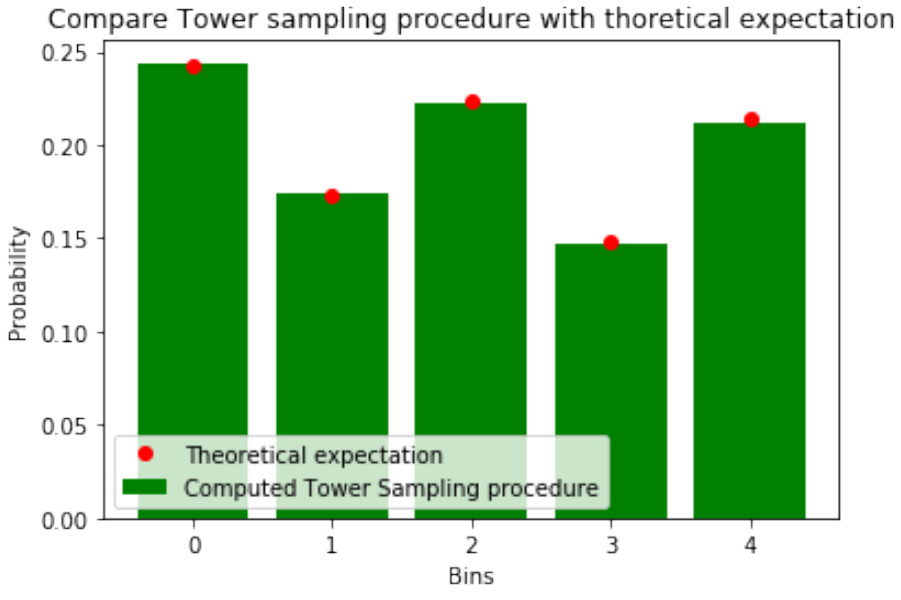

Figure 8: Comparison between Dichotomy Tower Sampling and theoretical expectation.

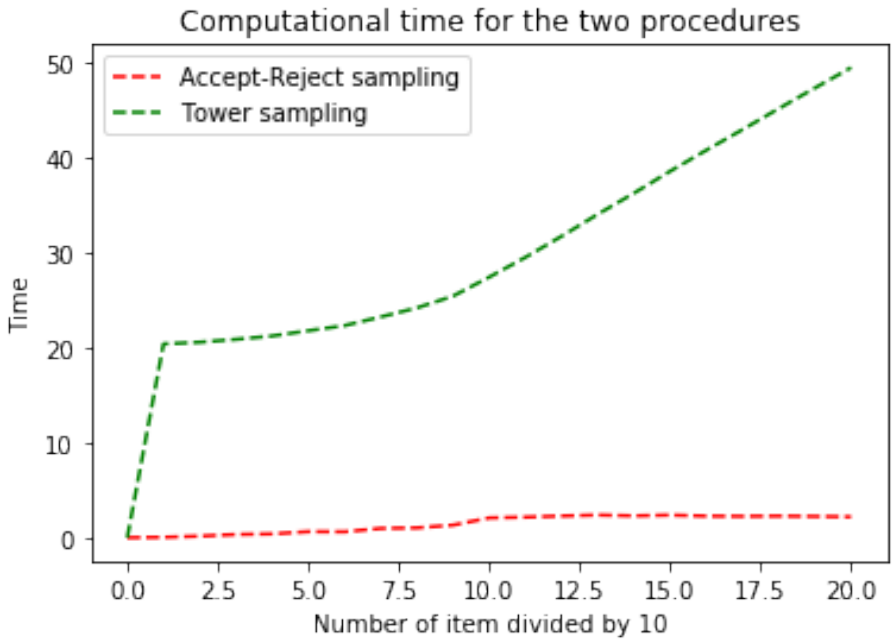

Figure 9: Computational time needed increasing number of items, "square root" distribution.

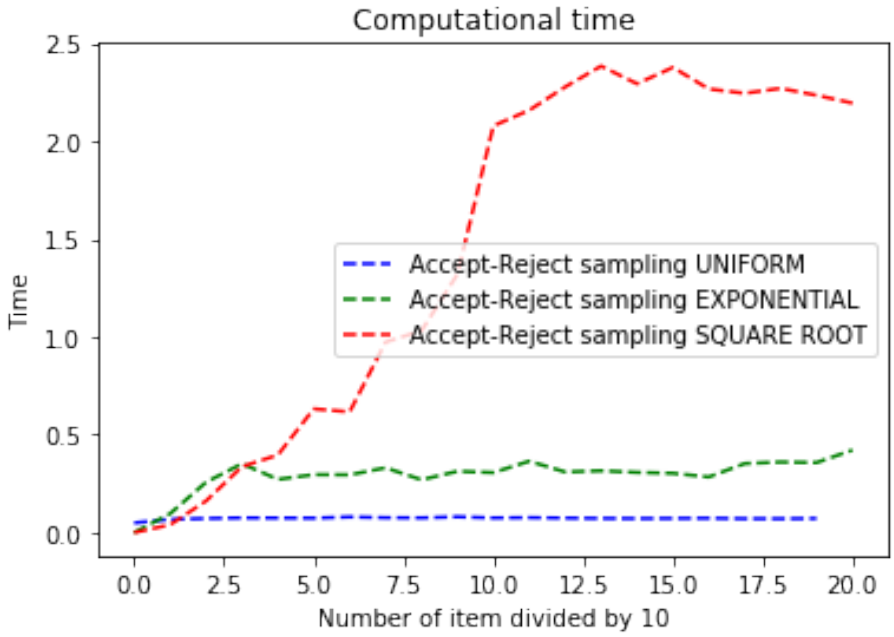

Figure 10: Computational time needed increasing number of items, accept/reject procedure, three different distributions.

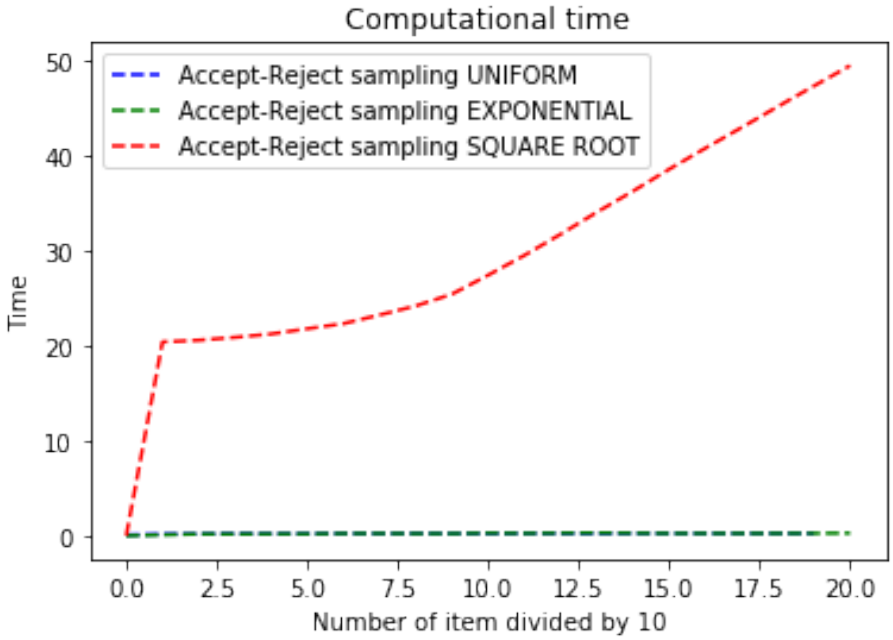

Figure 11: Computational time needed increasing number of items, tower sampling procedure, three different distributions.

2 Find the lighthouse

The problem consists of inferring the position of the lighthouse along the coast, α, given its distance from it. The lighthouse emits pulses at random orientation, θ, that are revealed by sensors on the coast at position x. Let's suppose that N pulses are emitted: this corresponds to a sequence of random variables $X_1, X_2, ..., X_N$ supposely independent and identically distributed according to some distribution $p(x; \alpha, \beta)$.

2.1 Derivation of Cauchy distribution

One has to some important hypothesis: $\theta, \alpha, x \in R$. A priori, there is no reason for θ to be distributed according to any distribution but the uniform. In this way one has

$$\theta \sim U(-\frac{\pi}{2}, \frac{\pi}{2}), \quad \text{with pdf} \quad f(\theta) = \frac{1}{\pi} \quad \text{for each} \quad \theta : -\frac{\pi}{2} \leq \theta \leq \frac{\pi}{2} \tag{9}$$

Calling d the distance between any x, position on the coast and the lighthouse, one obatains by geometrical considerations:

$$x - \alpha = d \sin\theta \tag{10}$$

$$\beta = d \cos\theta \tag{11}$$

The ratio between these two equation yelds:

$$\tan\theta = \frac{x-\alpha}{\beta} \quad \rightarrow \quad x = \beta \tan\theta + \alpha = H(\theta) \quad \rightarrow \quad H^{-}1(x) = \arctan\frac{x-a}{\beta} \tag{12}$$

We have to perform a change of variable and find the distribution dependent on x. It is better to work with cumulants.

$$P(X \leq x) = P(H(\theta) \leq x) = P(\theta \leq H^{-}1(x)) = P(\theta \leq \arctan\frac{x-a}{\beta}) \tag{13}$$

$$P(\theta \leq \arctan\frac{x-a}{\beta}) = \frac{1}{\pi}\int_{-\frac{\pi}{2}}^{\arctan\frac{x-a}{\beta}} d\theta = \frac{1}{\pi}\arctan\frac{x-a}{\beta} + \frac{1}{2} \tag{14}$$

And finally the probability distribution function is

$$p(X = x; \alpha, \beta) = \frac{d}{dx}P(X \leq x) = \frac{1}{\pi}\frac{\beta^2}{\beta^2 + (x-\alpha)^2} \tag{15}$$

which is the Cauchy distribution.

2.2 Log-Likelihood and Maximum Likelihood Estimation

After generating N Random variables distributed according to a Cauchy distribution, we want to plot the log-likelihood and estimate the parameter α. The likelihood is

$$\mathcal{L}(X_1, X_2, ..., X_N; \alpha) = P(\{x_k\}|\alpha) = \prod P(x_k|\alpha) \quad k \in \{1, ..., N\} \tag{16}$$

$$\log L(x_1, x_2, ..., x_k, \alpha) = \sum_{k=1}^{N} \log P(x_k|\alpha) = -N \log \pi\beta^2 - \sum_{k=1}^{N} \log\left(1 + (\frac{x_k - \alpha}{\beta})^2\right) \tag{17}$$

We find the most probable value of the parameter by imposing $\alpha_m = \arg\max = \log L(\{X_k)\}, \alpha) \quad \rightarrow \frac{\partial}{\partial\alpha} \log L(\alpha) = 0 \quad \rightarrow \quad \sum_{k=1}^{N} \frac{x_k - \alpha}{\beta^2 + (x_k - \alpha)} = 0$ rewriting one obtains $\sum_{k=1}^{N}(x_k - \alpha) \prod_{i \neq k}[\beta^2 + (x_i - \alpha)^2] = 0$ which is a $2N + 1$ polynomial, with the same amount of roots.

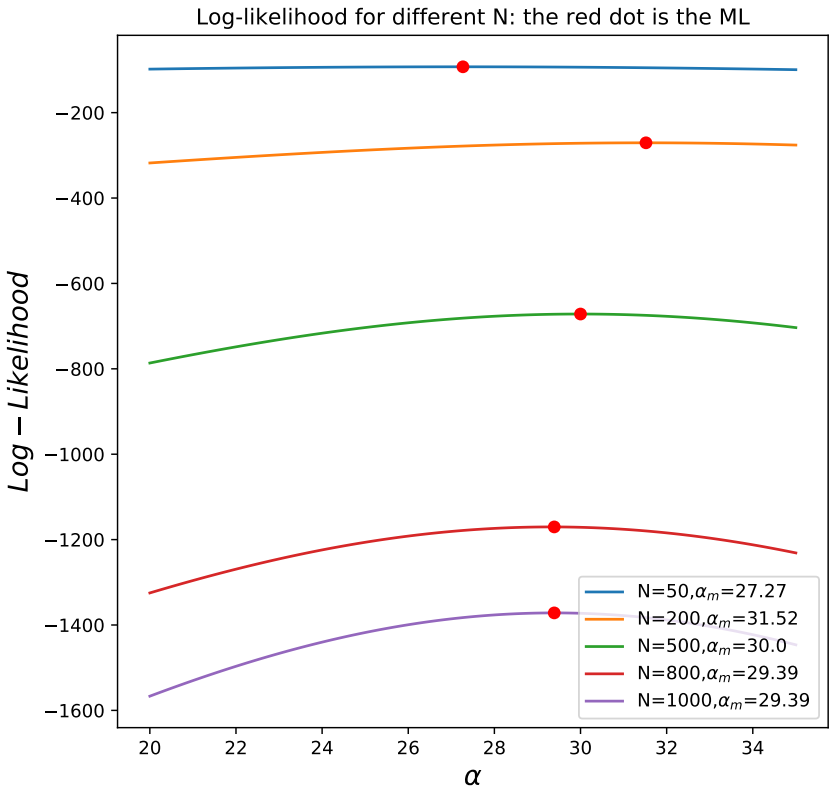

Figure 12: Log-likelihood function for different values of N

In the graph are shown the the values of the parameter estimated, shown as red dots in the curves. One can clearly see that the parameter estimation becomes more accurate increasing the number of independent random variables sampled. The log-likelihood in the top part of the graph looks flat because it varies more slowly with respect to the other curves.

2.3 Bonus Question: Mode and Mean

When one considers an a priori uniform distribution of the parameter α, the log-likelihood distribution is proportional to the posterior. One can wonder if the behavior of the mode of the posterior will conincide with the fluctuation of the mean for different values of the sampled random variables. If the distribution follows the Law of Large numbers, then the sum should converge to the its mean, that can be equal or not the mode, while the mode of the posterior should converge to the true value of the mode for that distribution that is the parameter we are trying to estimate. In the graph is clear that, increasing n,

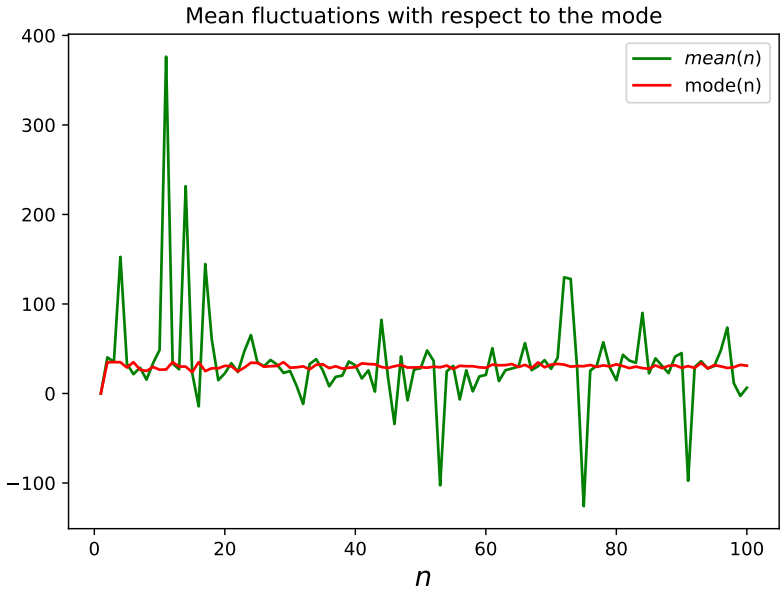

Figure 13: Mean and mode fluctuations

as one expects, the mode of the posterior lightly fluctuates around the true value of the Cauchy distribution, while the mean does not seem to converge to any value. In fact, the mean is not defined for the Cauchy distribution: it is an example of fat tailed distribution in which the extremes have non vanishing probability. That is mirrored in the fluctuation of the empirical mean, since the it often happens that sum is dominated by the maximum for each n and it can be orders of magnitude higher than the other terms of the sum. The Cauchy distribution does not follow the law of large numbers or central limit theorem.

3 Statistical Inference and Maximum Likelihood

We shall consider the problem discussed in our lecture, where a system emits particle with a half-length decay λ, which we detect if $1 < \lambda < 20$. The (density) probability to observe such a decay is thus

$$P_\lambda(x_i) = \frac{e^{\frac{-x}{\lambda}}}{\lambda Z(\lambda)}, \qquad \text{for} \quad 1 < x < 20$$
$$P_\lambda(x_i) = 0, \qquad \text{Otherwise} \tag{18}$$

3.1 Computation of $Z(\lambda)$ and $P_\lambda(\{x\})$

For computing $Z(\lambda)$ to get the normalization, we take the integral only in the interval in which the probabilities are not zero,

$$Z(\lambda) = \int_1^{20} \frac{e^{\frac{-x}{\lambda}}}{\lambda} dx = (e^{\frac{-1}{\lambda}} - e^{\frac{-20}{\lambda}}) \tag{19}$$

The events are independent and identically distributed, so the individual probabilities only multiply and,

$$P_\lambda(\{x\}) = \prod_{i=1}^{n} P_\lambda(x_i) \tag{20}$$

is the probability to observe a set of n events ({x}).

The program that output n observations sampled from the probability distribution (18) is made using the cumulative density function (CDF) $y : R \to [0, 1]$, because it is a bijective function. In the code, after defining $Z(\lambda)$ and its derivatives that we are going to use later, we sample the observations following the PDF by randomizing them uniformly in the interval (0,1), which is the range of the CDF.
So, the CDF y,

$$y = \int_1^x \frac{e^{\frac{-x}{\lambda}}}{\lambda Z(\lambda)} dx = \frac{e^{\frac{-1}{\lambda}} - e^{\frac{-x}{\lambda}}}{Z(\lambda)} \tag{21}$$

And arranging terms,

$$e^{\frac{-x}{\lambda}} = e^{\frac{-1}{\lambda}} - yZ(\lambda) \tag{22}$$

Taking the logarithm, and moving the λ to the left,

$$x = -\lambda[log(e^{\frac{-1}{\lambda}} - yZ(\lambda))] \tag{23}$$

Where we randomize the y in (0,1), because it is the range of the CDF. With this, we construct the simulation of the N events observed, and we make an histogram of the frequency of occurrences in the interval (1,20), sampled from the probability distribution (18). Apart from this, we construct the function of the PDF, and then directly the Log-Likelihood which, by definition, is the logarithm of (20). Since the number of observations is very large, the product of probabilities gives very little numbers, being often vanishing, so in order to avoid this problem, we have used the Log-Likelihood instead of the Likelihood.

3.2 Plotting the Log-Likelihood as a function of λ

We choose the true value to be $\lambda^* = 10$. Then, we generate $n = 10,100$ and 1000 observations. We plot the log-likelihood as a function of λ and see how it is peaked around the true value. At the beginning we discretize the interval (1,20) in order to associate each event to a specific bin. We choose our λ and, for the three different values of N, we make the simulation. We then generate our set of observations, computing the Maximum Likelihood Estimator (MLE) using the Log-Likelihood function. We plot the simulations and the Log-likelihood for each N. For every plot, we also make a vertical line in the maximum likelihood estimation found, to see how the curve of the likelihood is peaked around this value every time. The graphs obtained are shown below.

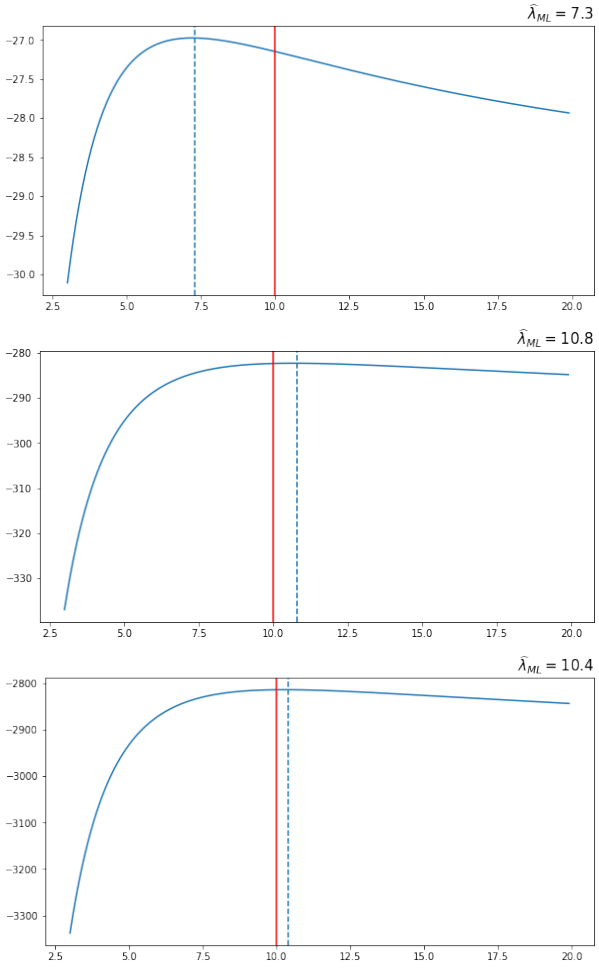

Figure 14: Log-likelihood for different values of N

3.3 Squared Error

We now assume that we are given a set of n observations, without being told the true value of λ, and now we want to compute the Squared Error (SE) that is defined as

$$SE = (\hat{\lambda}_{ML}(\{x\}) - \lambda)^2 \tag{24}$$

This is done for $N = 10, 100, 1000$, using an interval for the λs, and subtracting each one from the estimator. When we plot everything in the same graph, we realize that as N grows the SE decreases and becomes almost constant, so the estimator performs better as N grows. The graph obtained is shown below.

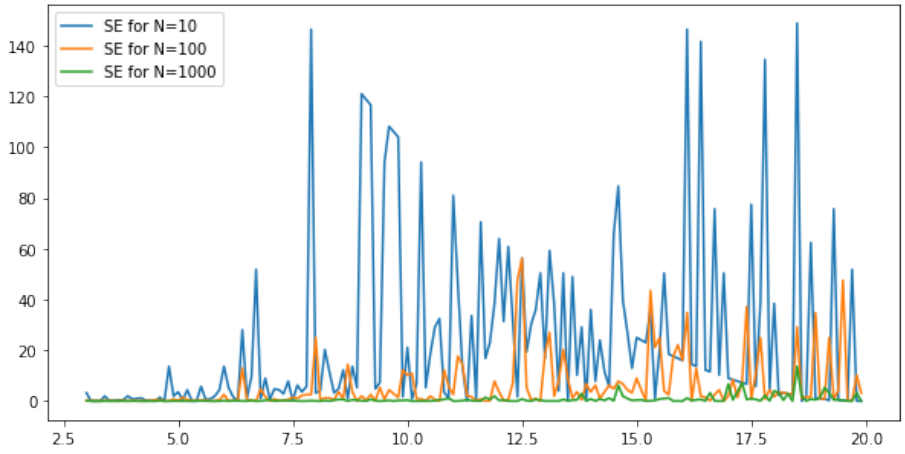

Figure 15: Squared Error

3.4 Mean Squared Error vs. Cramers-Rao bound

Now we need to average over a lot of realizations of the last process. In this way we obtain the $MSE(\lambda, \hat{\lambda}_{ML}, n)$, and then we compare it with the Cramers-Rao bound, that for the unbiased case reads as:

$$MSE(\hat{\lambda}) \geq \frac{1}{I(\lambda)} \tag{25}$$

For doing this, we need to do a determinate number of repetitions in our simulation of the experiment using a given fixed value of N observations each time. In this way, we can get the MSE for each repetition of the simulation and plot it for all the λ in the interval. In the same plot, we introduce the Cramers-Rao bound to compare both. For this, we need the total Fisher information $I(\lambda)$ which is $NI_x(\lambda)$ because the random variables are independent. We can use that:

$$I_x(\lambda) = E[-\frac{\partial^2 log P_\lambda(x_i)}{\partial \lambda^2}] \tag{26}$$

Using (18), we take the logarithm and obtain:

$$logP_\lambda(x_i) = \frac{-x}{\lambda} - log\lambda - logZ(\lambda) \tag{27}$$

And taking the second derivative,

$$-\frac{\partial^2 logP_\lambda(x_i)}{\partial\lambda^2} = \frac{2x}{\lambda^3} - \frac{1}{\lambda^2} + \frac{Z(\lambda)Z^{''}(\lambda) - (Z^{'}(\lambda))^2}{Z(\lambda)^2} \tag{28}$$

Finally we take the expected value,

$$E[-\frac{\partial^2 logP_\lambda(x_i)}{\partial\lambda^2}] = \frac{2E[x]}{\lambda^3} - \frac{1}{\lambda^2} + \frac{Z(\lambda)Z^{''}(\lambda) - (Z^{'}(\lambda))^2}{Z(\lambda)^2} \tag{29}$$

In this last expresion we use that the $E[x]$ is:

$$E[x] = \int_1^{20} \frac{xe^{\frac{-x}{\lambda}}}{\lambda Z(\lambda)} dx = \frac{e^{\frac{19}{\lambda}} - 20}{e^{\frac{19}{\lambda}} - 1} + \lambda \tag{30}$$

Substituting all this in (26), and multiplying by N, we obtain the total Fisher Information, and getting the inverse we obtain finally the Cramers-Rao bound. When this bound is plotted, it should remain under the graph of the MSE. If it was correct what we expect to see is that the MSE approaches the bound for N large, but we have done the simulation only for small values (100 realizations of 100 events), then we cannot infer too much. Apparently the MSE follow the curve of the Cramer-Rao bound until a certain value of lambda, that becomes larger as the number of realizations increases; this suggests an unbiased behavior for the MSE, however after certain lambda the MSE goes under the bound.

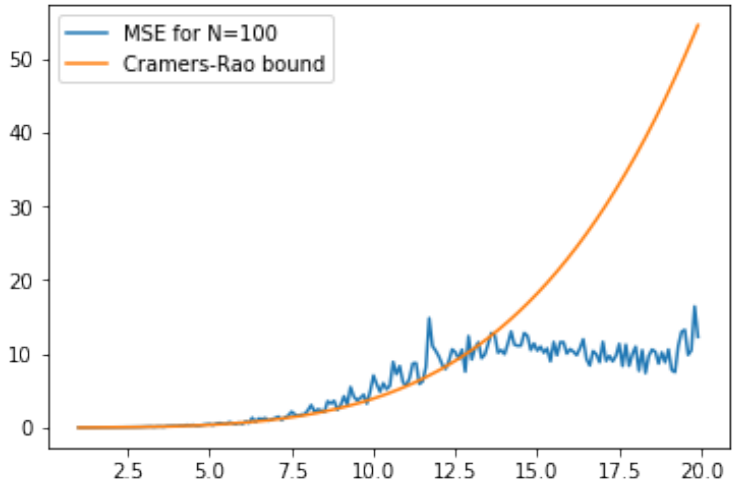

Figure 16: Comparison between MSE and Cramer-Rao bound