Computational Statistics with Python

Topics 6-7: Monte Carlo integration and approximation - The EM algorithm

Expected lecture time: 4 hours

Giancarlo Manzi

Department of Economics, Management and Quantitative Methods

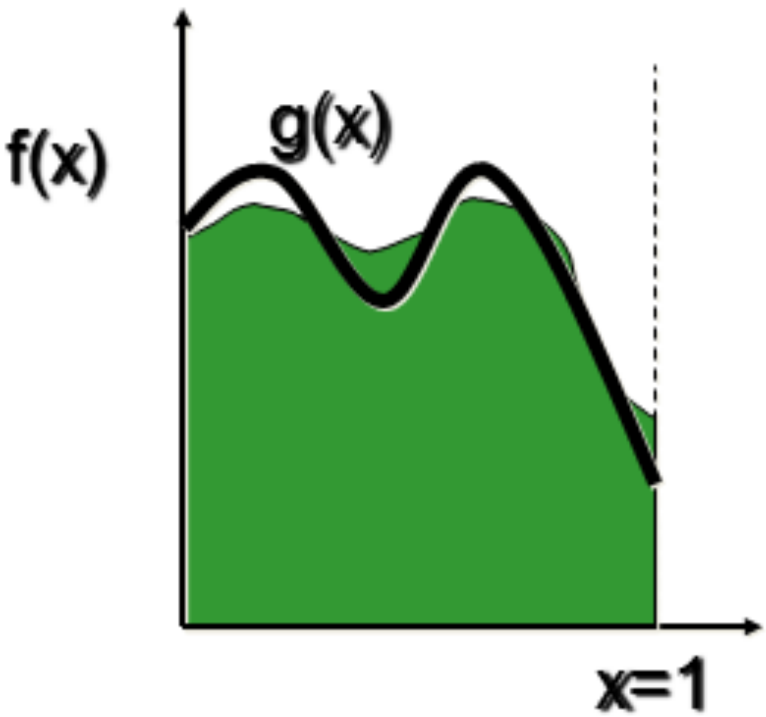
University of Milan, Milan, Italy

Other Monte Carlo numerical

algorithms

Numerical integration

- What is Monte Carlo integration?
- Monte Carlo integration is a technique for numerical integration using random numbers.
- It is a particular Monte Carlo method that computes definite integrals.
- One can use numerical approximation like the trapezium rule or the Simpson rule using quadratic polynomials $g(x)$: $\int_0^1 f(x)dx \approx \int_0^1 g(x)dx$.



- Python has many numerical integration

functions:

- The scipy.integrate sub-package provides several integration techniques including an ordinary differential equation integrator. An overview of the module is provided by the help command
- quad -- General purpose integration.
- dblquad -- General purpose double integration.
- tplquad -- General purpose triple integration.
- fixed_quad -- Integrate func(x) using Gaussian quadrature of order n.
- quadrature -- Integrate with given tolerance using Gaussian quadrature.
- romberg -- Integrate func using Romberg integration.
- trapz -- Use trapezoidal rule to compute integral from samples.
- cumtrapz -- Use trapezoidal rule to cumulatively compute integral.
- simps -- Use Simpson's rule to compute integral from samples.
- romb -- Use Romberg Integration to compute integral from

```
              (2**k + 1)
evenly-spaced samples
.
```

- See https://docs.scipy.org/doc/scipy/reference/tutorial/integ

(https://docs.scipy.org/doc/scipy/reference/tutorial/integ
for details.

- These MC methods have some advantages:
    - They converges fast for *smooth* integrands.
    - Work well for low dimensions.
    - Are deterministic.

- ...But also have some disadvantages:
    - Not rapid convergence for discontinuities.
    - Sometimes they cannot deal with infinite bounds.
    - Can give untrustworthy results.

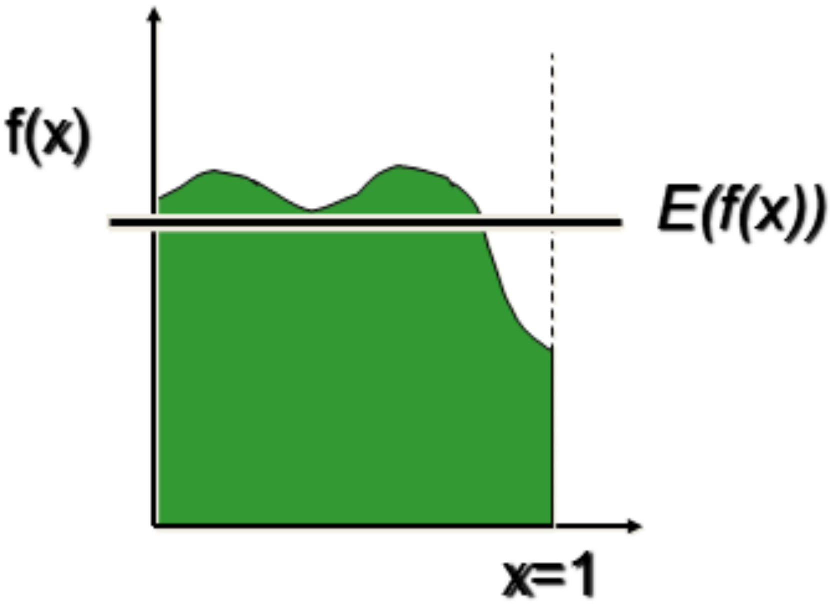
Other Monte Carlo numerical algorithms

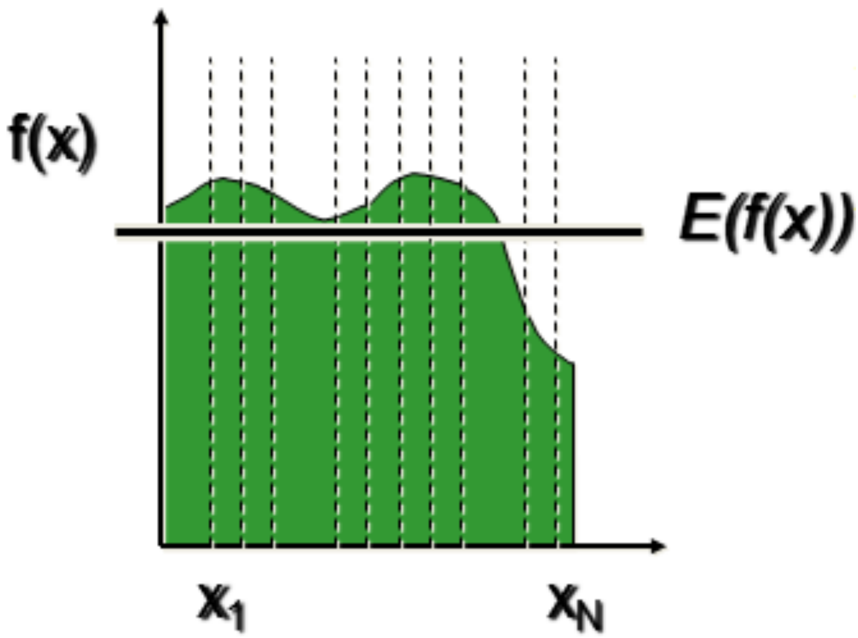
Numerical integration

- Instead of using numerical approximation, one can *average* : $\int_0^1 f(x)dx \approx E[f(x)](!!)$

- One can simply use the following approximation where n values from r.v. X_i are drawn independently from density $f(x)$:

$\int_0^1 f(x)dx \approx \frac{1}{n}\sum_{i=1}^n f(x_i).$





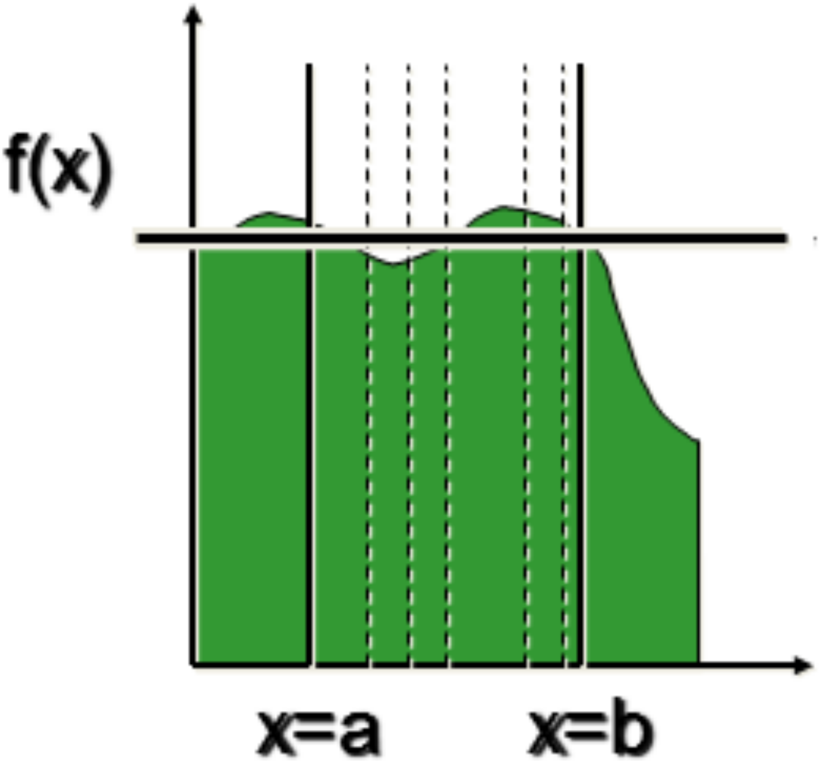
- Consider the value θ of the simple integral: $\theta = \int_0^1 f(x)dx.$
- The definition of the expectation of a function on a random variable X in the $(0, 1)$ support is: $E[f(X)] = \int_0^1 f(x)p(x)dx$, where $p(X)$ is the density of X.

- In particular, if X is uniformly distributed in $(0, 1)$, then we can write: $E[f(X)] = \int_0^1 f(x)dx = \theta$.

Other Monte Carlo numerical algorithms

Numerical integration

- In domains other than $(0, 1)$ we have, for example in (a, b): $\int_a^b f(x)dx \approx \frac{b-a}{n} \sum_{i=1}^n f(x_i)$.



- We can also use the *classical (crude) Monte Carlo integration*. How?
- Suppose $(\xi_1, \xi_2, \ldots, \xi_n)$ are independent random variables uniformly distributed.
- Then $f_i = f(\xi_i)$ are random variates with expected values θ, and:

$$\bar{f} = \frac{1}{n} \sum_{i=1}^{n} f_i$$

can be considered an "unbiased estimator" of θ with variance:

$$E[\bar{f} - \theta]^2 = \frac{1}{n-1} \int_0^1 [f(x) - \theta]^2 dx.$$

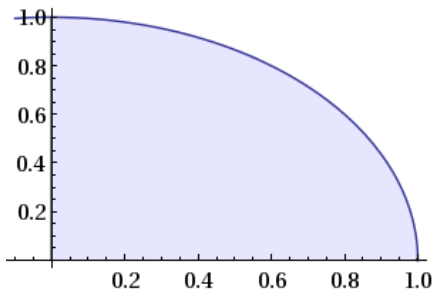
Other Monte Carlo numerical algorithms

Numerical integration: silly example

- Suppose we want to solve (answer: $\frac{\pi}{4}$):

$$\int_0^1 \sqrt{1-x^2}dx$$

which is the area of a unit-radius quarter-circle):



- This is equivalent to

$$\int_0^1 \sqrt{1-x^2}dx = \int_0^1 \sqrt{1-x^2} \cdot 1dx = \int_0^1 \sqrt{1-x^2}p($$
$$= E[\sqrt{1-X^2}][\text{where } X \text{ has density } p(\cdot) \sim U(0,1)] =$$

where $f(x) = \sqrt{1-x^2}$. The problem is equivalent to evaluating:

$$\theta = E[f(X)],\ \ X \sim U(0,1),\ \ f(x) = \sqrt{1-x^2}.$$

- We can generate $(X_1, X_2, \dots, X_n) \sim U(0,1)$, then estimate θ by:

$$\hat{\theta} = \frac{1}{n}\sum_{i=1}^{n} f(X_i).$$

In [1]:

```
# Above example in Python
import numpy as np
import math
n = 100000
xi = np.random.uniform(0,1,n)
theta = (1/n)*np.sum((1-xi**2)**0.5)
print(theta)
print(math.pi/4)
```

```
0.7837464705688125
0.7853981633974483
```

Other Monte Carlo numerical algorithms

MC integration: generalization

- The example before can be generalized when:

$$\theta = E[f(X)],$$

where X has a density p not necessarily uniform.

- For example, let's take a r.v. X distributed as a standard Cauchy with pdf $p(x; 0, 1) = \frac{1}{\pi(1+x^2)}$ and parameters 0 and 1.

- We can write, with $f(X) = I_{X>2}(X)$:

$$E[f(X)] = E[I_{X>2}(X)] = \int_{-\infty}^{\infty} I_{X>2}(X)\frac{1}{\pi(1+x^2)}dx$$
$$= \int_{2}^{\infty} \frac{1}{\pi(1+x^2)}dx = \theta,$$

where X is Cauchy.

- Note that the method can be generalized to higher dimension:

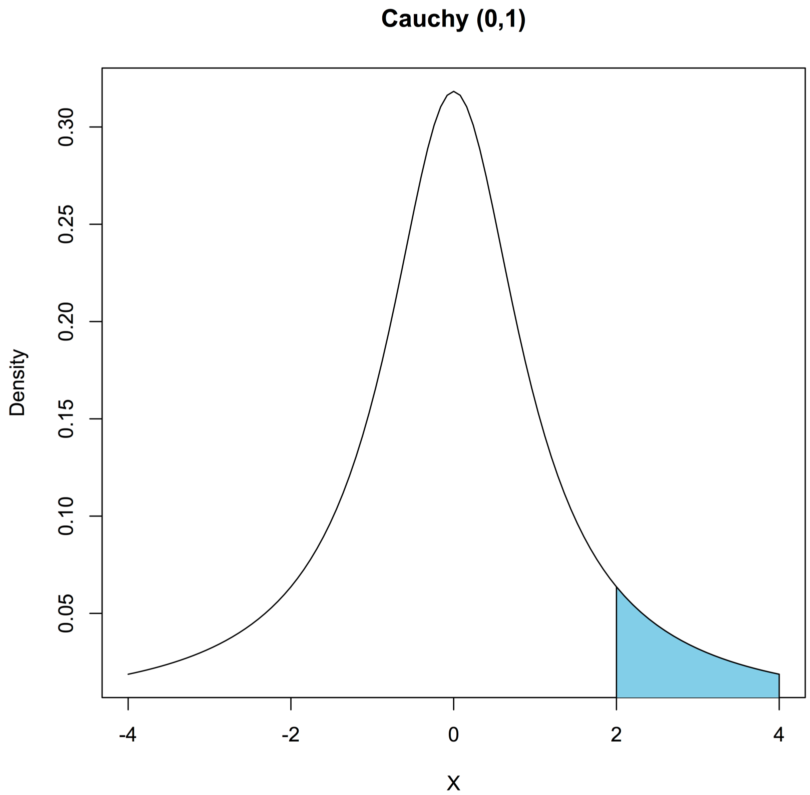
$$\int_0^1 \int_0^1 f(x_1, x_2)dx_1 dx_2 = E[f(X_1, X_2)],$$

with $X_1, X_2 \overset{\text{i.i.d}}{\sim} U(0, 1)$.

Other Monte Carlo numerical algorithms

MC integration: Cauchy integral

- Suppose we want to compute the integral of a Cauchy(0,1) for $X \geq 2$:



i.e. the following integral:

$$\theta = \int_2^\infty \frac{1}{\pi(1+x^2)} dx.$$

- θ can be computed analytically and is equal to 0.14758.

Here are other possibilities you can use to compute it in a Monte Carlo fashion.

- *Solution 1.* $p \sim \text{Cauchy}$, $f(x) = I_{X>2}(x)$.

$$\hat{\theta}_1 = \frac{\#\{x_i > 2, 1 \le i \le n\}}{n} = (\text{proportion of } x_i > 2).$$

$$var(\hat{\theta}_1) = \frac{\theta(1-\theta)}{n} \approx \frac{0.14758 \times (1-0.14758)}{n} = \frac{0.12\ldots}{n}$$

since $n\hat{\theta}_1 \sim \text{Bin}(n, \theta)$.

- *Solution 2*.
  $\theta = P(X > 2) = \frac{1}{2}P(|X| > 2) = \frac{1}{2}E[I_{|X|>2}(x)]$.

$$\hat{\theta}_2 = \frac{1}{2}\frac{\#\{|x_i| > 2, 1 \leq i \leq n\}}{n} = \frac{1}{2}(\text{proportion of } |x_i| >$$

$$var(\hat{\theta}_2) = \frac{n(2\theta)(1-2\theta)}{(2n)^2} = \frac{\theta(1-\theta)}{2n} \approx \frac{0.052}{n}$$

(since $2n\hat{\theta}_2 \sim \text{Bin}(n, 2\theta)$), an improvement with respect to the previous result.

Other Monte Carlo numerical algorithms

MC integration: Cauchy integral (cont'd)

- *Solution 3*. Another option is:

$$1 - 2\theta = \int_{-2}^{2} p(x)dx = 4\int_0^2 p(x)\cdot\frac{1}{2}dx$$

$$= 4\int_0^2 \frac{1}{2}\frac{1}{\pi(1+y^2)}dy = 4E[f(Y)]$$

,

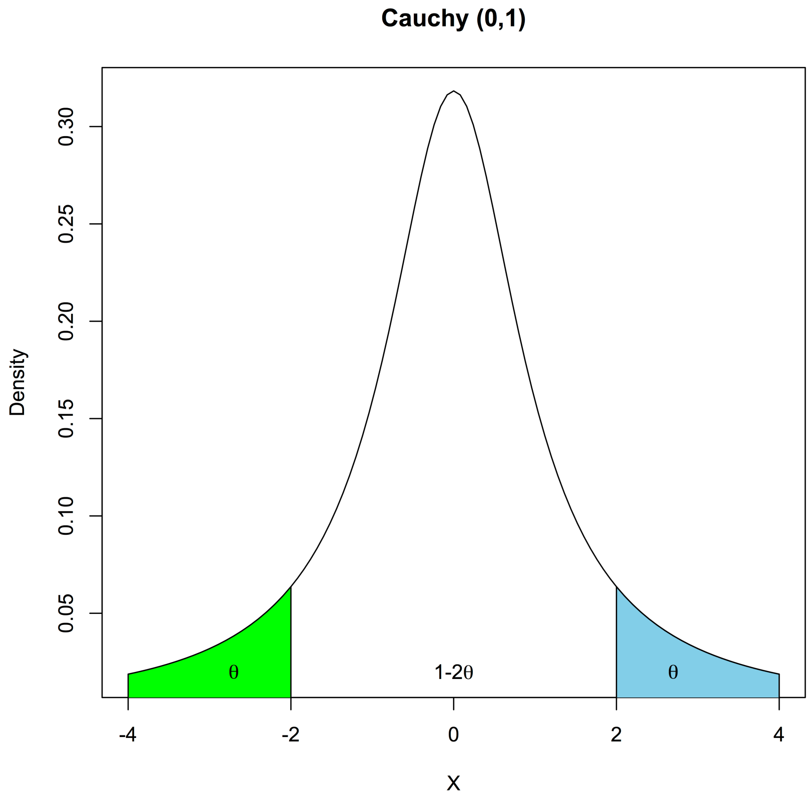
where $Y \sim U(0, 2)$. So, $\theta = \frac{1}{2} - 2E[f(Y)]$.

- Then:

$$\hat{\theta}_3 = \frac{1}{2} - 2 \cdot \frac{1}{n} \sum_{i=1}^{n} f(Y_i), \;\; Y_1, \ldots, Y_n \overset{\text{i.i.d.}}{\sim} U(0, 2)$$

$$var(\hat{\theta}_3) = \frac{4}{n} var(f(Y_1)) \approx \frac{0.028}{n},$$

a further improvement.



- *Solution 4*. Let $y = \frac{2}{x}$ $(dx = -(2/y)^2 dy)$, then:

$$\theta_4 = \int_2^{\infty} \frac{1}{\pi(1 + x^2)} dx = \int_0^1 \frac{2}{\pi(4 + y^2)} dy.$$

- This particular integral is in a form where it can be evaluated using crude Monte Carlo by sampling from the uniform distribution $U(0, 1)$.

- Another improvement.

In [2]:
```
#Using quad:
from scipy import integrate
import numpy as np
import math
x2 = lambda x: 1/(math.pi *(1+ x**2))
print(integrate.quad(x2, 2, np.inf))
##########
#Compare with empirical result (next slide)
```

```
(0.1475836176504333, 1.3085563320472785e-10)
```

In [3]:
```
#Solution 1:
import numpy as np
import math
n = 100000
counts = np.zeros(n)
cauchy_values = np.random.standard_cauchy(n)
for i in range(0,n):
    if cauchy_values[i] > 2:
        counts[i] = 1
theta1 = np.sum(counts)/n
print(theta1)
```

```
0.14549
```

In [4]:
```
#Solution 2:
import numpy as np
import math
n = 100000
counts = np.zeros(n)
cauchy_values = np.absolute(np.random.standard_cauchy(n))
for i in range(0,n):
    if cauchy_values[i] > 2:
        counts[i] = 1
theta2 = (1/2) * np.sum(counts)/n
print(theta2)
```

```
0.14729
```

In [5]:

```
#Solution 3:
import numpy as np
import math
n = 100000
unif_values = np.random.uniform(0, 2, n)
f_y = (1/(math.pi*(1+unif_values**2)))
theta3 = 0.5 - 2 * (np.sum(f_y)/n)
print(theta3)
```

```
0.1473974839682871
```

In [6]:

```
#Solution 4:
import numpy as np
import math
n = 100000
unif_values = np.random.uniform(0, 1, n)
f_y = (2/(math.pi*(4+unif_values**2)))
theta4 = (np.sum(f_y)/n)
print(theta4)
```

```
0.14758846136239862
```

Other Monte Carlo numerical algorithms

Importance Sampling (IS)

- With IS we want to estimate $\theta = \int f(x)p(x)dx = E_p[f(X)]$ (with X having pdf p) through $\hat{\theta}_p = \frac{1}{n}\sum_{i=1}^n f(X_i)$ (with $(X_1, \ldots, X_n) \overset{\text{i.i.d.}}{\sim} p$.

- The idea here is that instead of sampling directly from p, we sample from another pdf g. i.e., $(X_1, \ldots, X_n) \overset{\text{i.i.d.}}{\sim} g$.

- Our new estimate based on this g will be: $\hat{\theta}_g = \frac{1}{n}\sum_{i=1}^n f(X_i)\frac{p(X_i)}{g(X_i)} = \frac{1}{n}\sum_{i=1}^n \psi(X_i)$.

- Note that, by setting $\psi(x) = f(x)\frac{p(x)}{g(x)}$:

$$
\begin{aligned}
\theta = \int f(x)p(x)dx &= \int \left[f(x)\frac{p(x)}{g(x)} \right] g(x)dx \\
&= \int \psi(x)g(x)dx \\
&= E[\psi(X)], \quad X \sim g.
\end{aligned}
$$

Other Monte Carlo numerical algorithms

Importance Sampling (IS). Example

- Suppose we want to simulate the value of the integral:

$$\int_{4.5}^{\infty} p(u)du,$$

where $p(\cdot)$ is the density of a r.v. $Z \sim N(0, 1)$.

- We know from elementary statistics that this is rather a low value (you even cannot find the value in standardized normal tables!).

- And in fact Python gives us:
- `In []: from scipy.stats import norm`
  `1-norm.cdf(4.5, loc=0, scale=1)`

- `Out []: 3.3976731247387093e-06`

- This means that with crude Monte Carlo simulation we have to wait a lot before we get a hit, i.e. a simulated value greater than 4.5.

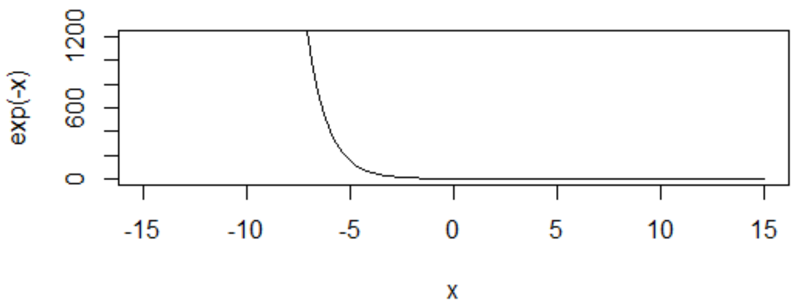
Other Monte Carlo numerical algorithms

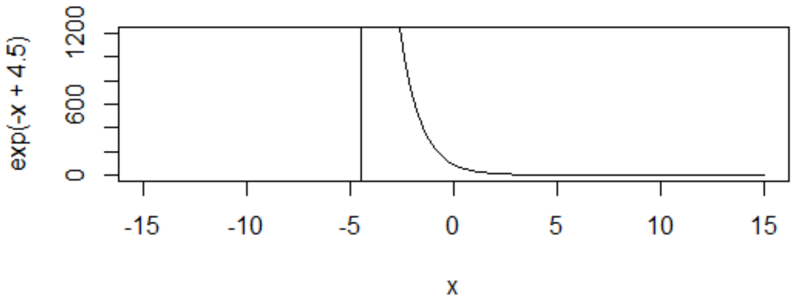
Importance Sampling (IS). Example from Robert & Casella.

- We can use importance sampling in this case, simulating r.v. values from another density g.
- Our g will be an exponential truncated at 4.5 which has density:

$$g(y) = e^{-(y-4.5)}.$$

- Its graph compared with the $\mathcal{E}\text{xp}(1)$ is the following:


exp(-x)
0
600
1200
-15
-10
-5
0
5
10
15
x


exp(-x + 4.5)
0
600
1200
-15
-10
-5
0
5
10
15
x

Other Monte Carlo numerical algorithms

Importance Sampling (IS). Example from Robert & Casella

- We draw n values y_i from g, then we calculate the ratios of the densities $\frac{p}{g}$, obtaining the corresponding importance sampling estimator of the tail probability:

$$\frac{1}{n}\sum_{i=1}^{n}\frac{p(Y^{(i)})}{g(Y^{(i)})} = \frac{1}{n}\sum_{i=1}^{n}\frac{\frac{1}{\sqrt{2\pi}}e^{-\frac{1}{2}Y_i^2}}{e^{(-Y_i+4.5)}} = \frac{1}{n}\sum_{i=1}^{n}\frac{e^{-(\frac{Y_i^2}{2}+Y_i-4.5)}}{\sqrt{2\pi}}$$

where the Y_i's are iid generations from g.

- The corresponding Python code is the following producing a value remarkably close to the true value of 3.398×10^{-6}:

In [7]:

```
from scipy.stats import norm
from scipy.stats import expon
import numpy as np
import math
NSim = 10 ** 6
y = np.random.exponential(1, NSim) + 4.5
weit=norm.pdf(y)/expon.pdf(y-4.5)
result = np.sum(weit)/NSim
result
```

Out[7]: 3.39623384392559e-06

Other Monte Carlo numerical algorithms (cont'd)

The Expectation-Maximization (EM) algorithm (Dempster et al., 1977; see Robert & Casella, pp. 152-163)

- An important algorithm for detecting parameter estimation of *latent* variables.
- Therefore, useful in missing data contexts, Bayesian networks, mixtures of distributions, even cluster detection.
- Quite a heavy mathematical burden, so we will use a quick definition and a toy example.
- Based on the concept of ML estimation (please revise it!).

Other Monte Carlo numerical algorithms

(cont'd)

The Expectation-Maximization (EM) algorithm (cont'd)

- (EM) method is an iterative method for maximizing difficult likelihood functions in order to find MLE estimators.
- Suppose we have a random sample $X = (X_1, \cdots, X_n)$ iid from $f(x|\theta)$.
- We wish to find the ML estimator

$$\hat{\theta} = \arg\max_{\theta} \prod_{i=1}^{n} f(x_i|\theta) = \arg\max_{\theta} \sum_{i=1}^{n} \log f(x_i|\theta).$$

- Suppose we are in a situation where this optimization problem is difficult to solve.
- We *augment* the data, i.e. we guess that an unobservable variable is governing this likelihood problem.
- (That's why EM is suitable for missing data problems).
- We denote these unobserved or missing data with X^m, such that we get the *complete data* $X^c = (X, X^m)$.

Other Monte Carlo numerical algorithms (cont'd)

The Expectation-Maximization (EM) algorithm (cont'd)

- The joint density of the complete data X^c is:

$$X^c = (X, X^m) \sim f(x^c) = f(x, x^m).$$

- The conditional density for the missing data X^m with respect to the observed data is:

$$f(x^m|x, \theta) = \frac{f(x, x^m|\theta)}{f(x|\theta)}.$$

- Rearranging terms:

$$f(x|\theta) = \frac{f(x, x^m|\theta)}{f(x^m|x, \theta)}.$$

- Taking logarithm, we get the log-likelihood:

$$\log f(X|\theta) = \log f(X^c|\theta) - \log f(X^m|X, \theta).$$

Other Monte Carlo numerical algorithms (cont'd)

The Expectation-Maximization (EM) algorithm (cont'd)

- Finally, taking expectation with respect to $f(x^m|x, \theta_0)$ (considering a given value for θ, θ_0) so that X can be considered constant:

$$\begin{aligned}\mathbb{E}[\log f(X|\theta)] = \mathbb{E}[\log f(X^c|\theta)|X, \theta_0] \\ -\mathbb{E}[\log f(X^m|X, \theta)|X, \theta_0].\end{aligned}$$

- Let's denote the log-likelihood of the complete data (which is the focus of our algorithm) with the following expression:

$$Q(\theta|\theta_0, x) = \mathbb{E}[\log f(X^c|\theta)|X, \theta_0].$$

- The EM algorithm works going across these two steps until convergence is reached:
  1. Expectation-step: compute $Q(\theta|\hat{\theta}_{j-1}, x)$;
  2. Maximization-step: maximize $Q(\theta|\hat{\theta}_{j-1}, x)$ and take
  $$\hat{\theta}_j = \arg\max_\theta Q(\theta|\hat{\theta}_{j-1}, x).$$

Other Monte Carlo numerical algorithms (cont'd)

The Expectation-Maximization (EM) algorithm (cont'd)

A toy example (or, *an example with toys* - presented in the original paper by Dempster et al., 1977).

- We use a toy example (literally!) to explain how the EM algorithm works.
- Imagine you ask n kids to choose a toy out of 4 possible choices.
- Let $Y = [Y_1, \dots, Y_4]^T$ be the histogram of their n choices, i.e. Y_1 is the number of kids that chose toy 1, ..., Y_4 is the number of kids that chose toy 4.
- What is the r.v. that models this data? A multinomial r.v.
- Therefore, the histogram is "distributed" according to a multinomial distribution.
- The multinomial has two sets of parameters: the number of trials n and the probabilities

p_1, p_2, p_3, p_4 of choosing toys $1, 2, 3, 4$, respectively.

Other Monte Carlo numerical algorithms (cont'd)

The Expectation-Maximization (EM) algorithm (cont'd)

A toy example (or, *an example with toys* - cont'd).

- Therefore, the probability of seeing some particular histogram y is:

$$P(y|\theta) = \frac{n!}{y_1!y_2!y_3!y_4!} p_1^{y_1} p_2^{y_2} p_3^{y_3} p_4^{y_4}. (3)$$

- For this example it is assumed that the vector of probabilities p is parameterized by some hidden parameter $\theta \in (0, 1)$ such that it can be written as:

$$[p_1 = \frac{1}{2} + \frac{1}{4}\theta, p_2 = \frac{1}{4}(1-\theta), p_3 = \frac{1}{4}(1-\theta), p_4 = \frac{1}{4}(\theta$$

so that $\sum p_i = 1$.

- The estimation problem is to guess the value of

θ that maximizes the probability of the observed histogram.

- For this simple example, one could directly maximize the log-likelihood $\log P(y|\theta)$, but here we will instead illustrate how to use the EM algorithm to find the MLE of θ.

Other Monte Carlo numerical algorithms (cont'd)

The Expectation-Maximization (EM) algorithm (cont'd)

A toy example (or, *an example with toys* - cont'd).

- With the parameterization above, we can write the probability in (3) as follows:

$$P(y|\theta) = \frac{n!}{y_1!y_2!y_3!y_4!}[\frac{1}{2} + \frac{1}{4}\theta]^{y_1}[\frac{1}{4}(1-\theta)]^{y_2}[\frac{1}{4}(1-\theta)$$

- Now, in order to properly apply EM, we need to specify what the complete data X is.

- To that purpose, we define the complete data as $X = [X_1, \dots, X_5]$, with X multinomial with number of trials n and probability of each event:

$$[p_1 = \frac{1}{2}, p_2 = \frac{1}{4}\theta, p_3 = \frac{1}{4}(1-\theta), p_4 = \frac{1}{4}(1-\theta), p_5 =$$

Other Monte Carlo numerical algorithms (cont'd)

The Expectation-Maximization (EM) algorithm (cont'd)

A toy example (or, *an example with toys* - cont'd).

- By defining X in this way, we can write the observed data Y as:

$$Y = T(X) = [X_1 + X_2, X_3, X_4, X_5].$$

- Therefore, the likelihood of a realization x of the complete data is:

$$P(x|\theta) = \frac{n!}{\prod_{i=1}^{5} x_i!} (\frac{1}{2})^{x_1} (\frac{\theta}{4})^{x_2+x_5} (\frac{1-\theta}{4})^{x_3+x_4}.$$

- For the EM algorithm, we should maximize the Q function:

$$\theta^{(m+1)} = \arg\max_{\theta\in(0,1)} Q(\theta|\theta^{(m)}) = \arg\max_{\theta\in(0,1)} E_{X|y,\theta^{(m)}}[\log p(X|\theta)]$$

Other Monte Carlo

numerical algorithms (cont'd)

The Expectation-Maximization (EM) algorithm (cont'd)

A toy example (or, *an example with toys* - cont'd).

- To solve (4) for θ, we need the terms of $\log p(X|\theta)$ that depend on θ, because the other terms are irrelevant for maximizing the function over θ:

$$\begin{aligned}\theta^{(m+1)} &= \underset{\theta\in(0,1)}{\arg\max} E_{X|y,\theta^{(m)}}[(X_2 + X_5)\log\theta + (X_3 + X_4)\,\mathrm{lo}\\ &= \underset{\theta\in(0,1)}{\arg\max} \Big\{\log\theta(E_{X|y,\theta^{(m)}}[X_2] + E_{X|y,\theta^{(m)}}[X_5]) + \mathrm{l}\\ &\qquad (1-\theta)(E_{X|y,\theta^{(m)}}[X_3] + E_{X|y,\theta^{(m)}}[X_4])\Big\} \quad (5)\end{aligned}$$

where we have considered only terms that depend on θ.

- To solve (5) we need the conditional expectation of the complete data X conditioned on already knowing the incomplete data y, which only leaves the uncertainty about X_1 and X_2.

- But we know that $X_1 + X_2 = y_1$, and therefore we can say that given y_1, the pair X_1, X_2 is binomially distributed.

Other Monte Carlo numerical algorithms (cont'd)

The Expectation-Maximization (EM) algorithm (cont'd)

A toy example (or, *an example with toys* - cont'd).

- Exploiting the last point in the previous slide, we end up with this conditional distribution to be plugged in (5) to solve the problem:

$$E_{X|y,\theta[X]} = [\frac{2}{2+\theta}y_1, \frac{\theta}{2+\theta}y_1, y_2, y_3, y_4].$$

- Therefore (4) becomes:
$\theta^{(m+1)} =$

$$= \arg\max_{\theta\in(0,1)}(\log\theta(\frac{\theta^{(m)}y_1}{2+\theta^{(m)}} + y_4) + \log(1-\theta)(y_2 + y$$

$$= \frac{\frac{\theta^{(m)}}{2+\theta^{(m)}}y_1 + y_4}{\frac{\theta^{(m)}}{2+\theta^{(m)}}y_1 + y_2 + y_3 + y_4}.$$

Suggested references and reading

- Brewer, B.J., Introduction to Bayesian Statistics. Course notes. University of Auckland.
- De Finetti, B (1931). Funzione caratteristica di un fenomeno aleatorio. Accademia dei Lincei, Roma.
- Dempster, A.P., Laird, N. M., Rubin, D. B. (1977). Maximum Likelihood from Incomplete Data via the EM Algorithm. Journal of the Royal Statistical Society, Series B. 39(1): 1-38.

Example with a mixture of Normals using the *sklearn GaussianMixture class* which implements the EM algorithm for fitting a mixture of normal distribution

In [2]:

```
from sklearn.mixture import GaussianMixture
import numpy as np
import matplotlib.pyplot as plt
```

In [19]:
```
X = np.array([[1, 2], [1, 4], [1, 0], [10, 2], [10, 4], [10, 0]
#Each row corresponds to a single data point.
X
```

Out[19]:
```
array([[ 1,  2],
       [ 1,  4],
       [ 1,  0],
       [10,  2],
       [10,  4],
       [10,  0]])
```

In [22]:
```
gm = GaussianMixture(n_components=2, random_state=0).fit(X)
print(gm.means_)
#two 2-sized means, centres of the components
# Predict the labels for the data samples in X using trained mo
print(gm.predict([[0, 0], [12, 3]]))

```

```
[[10.  2.]
 [ 1.  2.]]
[1 0]
```

In [1]:

```
# To run slideshow type jupyter nbconvert /Users/giancarlomanzi/
# from terminal
#/Users/giancarlomanzi/Documents/Box Sync BackUp PC Lavoro 2406.

#This is to let you have larger fonts...
from IPython.core.display import HTML
HTML("""
<style>

div.cell { /* Tunes the space between cells */
margin-top:1em;
margin-bottom:1em;
}

div.text_cell_render h1 { /* Main titles bigger, centered */
font-size: 2.2em;
line-height:1.4em;
text-align:center;
}

div.text_cell_render h2 { /*  Parts names nearer from text */
margin-bottom: -0.4em;
}


div.text_cell_render { /* Customize text cells */
font-family: 'Times New Roman';
font-size:1.5em;
line-height:1.4em;
padding-left:3em;
padding-right:3em;
}
</style>
""")
```

Out[1]: