TP 1 – AOS1

Introduction to Bayesian inference
Corrigé

1 Introduction

The practical session will leave much place to manipulating probability distributions and generating random samples, from which parameters are to be estimated. The `numpy`, `scipy`, `scipy.stats` and `matplotlib.pyplot` packages will be useful for this purpose.

```
import numpy as np
import scipy as sp
import scipy.stats as spst
import matplotlib.pyplot as plt
import itertools
```

2 Visualization of a pdf, generation of a random sample

First, we are interested in generating random samples according to some specific (user-defined) distribution.

(1) For the binomial and Poisson distributions:

1. pick a particular parameter value,
2. generate a sample of desired size,
3. visualize the empirical distribution of the data (using `plt.bar`) and compare it to the actual distribution (using `distrib.pmf`).

Example with the binomial distribution with parameters $N = 100$ and $p = 0.4$, where we generate a sample of size $n = 1000$:

```
distrib = spst.binom(n=100, p=0.4)
x = distrib.rvs(size=1000)
t = np.arange(start=0, stop=100, step=1)
freq = [np.mean(x==i) for i in range(100)]

fig1, axs1 = plt.subplots(1, 2, sharex=True, tight_layout=True)
axs1[0].bar(x=range(0,100), height=freq)
axs1[1].plot(t, distrib.pmf(t))
```

(2) For the beta, gamma, inverse gamma, exponential, Gaussian distributions:

1. pick a particular (set of) parameter value(s),
2. generate a sample of desired size,
3. visualize the empirical distribution of the data (using `plt.hist`) and compare it to the actual distribution (using `distrib.pdf`).

Example with the Gaussian distribution:

```
distrib = spst.norm(loc=0, scale=1)
x = distrib.rvs(size=1000)
t = np.arange(start=-5, stop=5, step=0.1)

fig2, axs2 = plt.subplots(1, 2, sharex=True, tight_layout=True)
axs2[0].hist(x, range=(-5,5), bins=20)
axs2[1].plot(t, distrib.pdf(t))
```

3 Maximum likelihood estimation, Bayesian updating

3.1 Likelihood plot

3 Program a function `loglike` which makes it possible to compute the log-likelihood of a parameter given a sample and a family of distributions; for instance, given a random vector $\boldsymbol{X} \sim \mathcal{N}(\mu, \sigma^2)$, we would compute the log-likelihood $\ln L(\mu = 0, \sigma = 2; \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ by:

```
loglike((0,2), spst.norm, x)
```

where `x` contains the data sample. You will take care of the fact that for multivariate distributions, instances are to be stored row-wise in `x`.

```
def loglike(params, distrib, sample):
    try:
        return np.sum(distrib.logpdf(sample, *params), axis=0)
    except AttributeError:
        return np.sum(distrib.logpmf(sample, *params), axis=0)
```

It seems best to implement a single function `loglike` which takes chosen distribution family, the parameters and the sample into account. This however requires to test for the nature (discrete or continuous) of the distribution.

4 Write a script which plots the likelihood for a set of parameter values. In the case of two parameters, the level curves will be displayed.

The solution is quite straightforward in the case of a single parameter. For example, plotting $\ln L(\mu, \sigma = 1; \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ for various values of μ can be achieved by:

```
par = np.arange(-2.0, 2.0, 0.025)

ll = [loglike((i,1), spst.norm, x) for i in par]

fig3, ax3 = plt.subplots()
ax3.plot(par, ll)
ax3.set_title('$\log L(\mu,\sigma^2=1)$')
ax3.set_xlabel('$\mu$')
```

In the case of several parameters, computing the log-likelihood is straightforward (and identical to the previous case), but plotting the level curves is a bit trickier:

```
par1 = np.arange(-2.0, 2.0, 0.025)
par2 = np.arange(0.25,2,0.025)

ll = [[loglike((i,j), spst.norm, x) for i in par1] for j in par2]

fig4, ax4 = plt.subplots()
par1, par2 = np.meshgrid(par1, par2)
CS = ax4.contour(par1, par2, np.asarray(ll), levels=250)
ax4.clabel(CS, inline=1, fontsize=10)
ax4.set_title('$\log L(\mu,\sigma^2)$')
ax4.set_xlabel('$\mu$')
ax4.set_ylabel('$\sigma^2$')
```

When we increase the number of observations, we can see that the log-likelihood becomes more and more "peaked" around the maximum-likelihood estimate. It is an illustration of the fact that the ML estimator is consistent.

3.2 Bayesian prior and Bayesian updating

Eventually, consider a very simple case of a binomial random variable

$$X \sim \mathcal{B}(n, \theta),$$

with n fixed, and with θ unknown and to be estimated from the observation $x = 40$. Assume that a Bayesian prior is available:

$$\pi_\theta(t|\alpha, \beta) = \text{beta}(t; \alpha, \beta) = \frac{t^{\alpha-1}(1-t)^{\beta-1}}{\mathrm{B}(\alpha, \beta)},$$

with

$$\mathrm{B}(\alpha, \beta) = \int_0^1 t^{\alpha-1}(1-t)^{\beta-1} dt = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}.$$

The questions of why we choose the beta distribution, and how we select the prior parameters α and β, will not be addressed for now.

5 Show that the posterior distribution of $\theta|x$ is a beta distribution beta$(\alpha + x, \beta + n - x)$. For this purpose, we recall that the Gamma function is the "generalization" of the factorial to complex numbers; we have

$$\Gamma(z) = \int_0^{+\infty} t^{z-1} \exp(-t) dt, \qquad \Gamma(z+1) = z\Gamma(z), \quad \Gamma(n) = (n-1)! \forall n \in \mathbb{N}^*.$$

We first compute the product of the prior distribution and the likelihood function:

$$\pi_\theta(t)L(t|x) = \frac{t^{\alpha-1}(1-t)^{\beta-1}}{\mathrm{B}(\alpha,\beta)}\binom{n}{x}\theta^x(1-\theta)^{n-x} = \binom{n}{x}\frac{1}{\mathrm{B}(\alpha,\beta)}t^{x+\alpha-1}(1-t)^{n-x+\beta-1}.$$

If we compute the integral of this product with respect to t, we obtain

$$\int_0^1 \frac{\binom{n}{x}}{\mathrm{B}(\alpha,\beta)}t^{x+\alpha-1}(1-t)^{n-x+\beta-1}dt = \frac{\binom{n}{x}}{\mathrm{B}(\alpha,\beta)}\mathrm{B}(\alpha+x,\beta+n-x).$$

Computing the ratio of both finally gives

$$\pi_\theta(t|x) = \frac{\pi_\theta(t)L(t|x)}{\int_0^1 \pi_\theta(t)L(t|x)dt} = \frac{t^{x+\alpha-1}(1-t)^{n-x+\beta-1}}{\mathrm{B}(\alpha+x,\beta+n-x)},$$

which is the expression for the density of the beta distribution $\mathrm{beta}(\alpha+x,\beta+n-x)$.

6 For various parameter values for $\alpha > 0$ and $\beta > 0$, plot the prior $\pi_\theta(t|\alpha,\beta)$, the likelihood function $L(t|x)$ and the posterior $\pi_\theta(t|x;\alpha,\beta)$. What can you say about the prior distribution influencing the inference on θ ?

```
alph=1
beta=1
x=40

par = np.arange(0, 1, 0.01)
ll = [loglike((100,i), spst.binom, x) for i in par]

fig5, axs5 = plt.subplots(1, 3, sharex=True, tight_layout=True)
axs5[0].plot(par, spst.beta.logpdf(par, alph, beta))
axs5[1].plot(par, ll)
axs5[2].plot(par, spst.beta.logpdf(par, alph, beta)+ll)
```

The mode of the beta distribution is

$$\widehat{\theta}_{\text{mode}} = \frac{\alpha}{\alpha+\beta}.$$

However, the parameters α and β also play the role of "virtual" positive/negative outcomes for the Bernoulli experiment on which the binomial variable X is based: the more such virtual outcomes there are, the more the prior distribution will influence the inference. This can be seen by increasing the values for both α and β, even if the relative values are kept (e.g., $\alpha = \beta$).