Exercises 3 – Advanced statistics – Tuesday, 13th January 2015

Please return the results by next week, Tuesday 22th Jan 2015 1pm, with name and student number on each page. Exercises should be done individually. They can be send to `m.r.feyereisen@uva.nl` in a single mail. *The whole sheet is worth 40 points.*

1. Signals and backgrounds in low-number statistics.

   (a) Consider a Poisson process with a known background b and a unknown signal contribution $\mu \geq 0$, such that the average number of expected counts are $\langle c \rangle = \mu + b$. Suppose we measure a value of $c = 3$, whereas the expected background is $b = 1.5$. What is the $p - value$ of this 'excess'? (1pt)

   SOLUTION: We have to calculate the probability of observing three or more events if the expected value if $b = 1.5$; $p = \sum_{c=3}^{\infty} P(c|b)$ (with $P(c|\lambda)$ being the Poisson distribution with mean λ). This yields $p = 0.19$, and implies that we could exclude the null hypothesis ("We observe only background events") only at the $1 - p =$"81% confidence level (CL)". This is too low to reject the null hypothesis, which we would hence accept.

   (b) Derive, based on the previous numbers, a proper 95% CL *upper* limit and a proper 95% CL *lower* limit on μ. Derive also the 68% CL central confidence interval on μ. (4pt)

   SOLUTION: Following the construction of upper limit confidence belts that we discussed in the lecture, we have to find the value of μ for which the probability of observing three or less events is $\alpha = 0.05$. We obtain this numerically by solving $\sum_{c=0}^{3} P(c|b + \mu) = 0.05$, which yields $\mu = 6.2$. Hence, if three events are measured, we obtain the upper limit $\mu \leq 6.2$ (95% CL).

   Following the same logic, we have to solve $\sum_{c=3}^{\infty} P(c|b + \mu) = 0.05$ for μ to obtain the lower limit on μ. This yields $\mu = -0.7$, if we allow for negative values of μ (and $\mu = 0$ if μ is required to be non-negative). Hence, we obtain the lower limit $\mu \geq -0.7$ (95% CL) (altough $\mu \geq 0$ would have been correct as well).

   The central interval is obtained in a similar way, namely by solving $\sum_{c=3}^{\infty} P(c|b+\mu) = 0.16$ to get the lower end of the confidence interval, and $\sum_{c=0}^{3} P(c|b+\mu) = 0.16$ to get the upper end. This yields the confidence interval $-0.1 \leq \mu \leq 4.4$ (68% CL).

   [Note: If one wants to write this in terms of a 'best-fit value' and asymmetric errors, one typically adopts the maximum-likelihood estimator for μ (in the present case $\mu = 1.5$) as best-fit value. One can then write the above result as $\mu = 1.5^{+2.9}_{-1.6}$ (68% CL).]

2. Minimizers – Gradient decent

   In this exercise, we will concentrate on the two-dimensional function

   $$f(x, y) = (x - 2)^2 + (y - 3)^2 + xy \ ,$$

   (a) As a warm up, show analytically that the minimum of the function is given by $x = \frac{2}{3}$ and $y = \frac{8}{3}$. (1pt) SOLUTION: We have $\partial f/\partial x = 2(x-2)+y = 0$ and $\partial f/\partial y = 2(y - 3) + x = 0$. This yields the above result.

(b) Write a gradient decent algorithm in order to minimize the function. Select three or more random points in the range $-10 \leq x, y \leq 10$, and set γ to a reasonable value. Plot the first twenty steps that the minimizer takes when approaching the minimum for each of the three initial conditions. For what values of γ does this provide good convergence (visual inspection of the plot is enough)? (5pt)

Hint: Remember that first-order derivatives can be estimated numerically by e.g. $\frac{\partial}{\partial x} f(x, y) = (f(x + dx, y) - f(x, y))/dx$, with a suitably chose dx. You can instead also use the analytical derivatives of $f(x, y)$ in the algorithm, $\partial f(x, y)/\partial x$ and $\partial f(x, y)/\partial y$.

SOLUTION: An example program that does the job is:

```
from __future__ import division
from numpy import *
import pylab as plt

def f(x,y): return (x-2)**2 + (y-3)**2 + x*y
def dfdx(x,y): return 2*(x-2) + y
def dfdy(x,y): return 2*(y-3) + x

def iteration(x, y, gamma):
    return x - gamma*dfdx(x,y), y - gamma*dfdy(x,y)

def plot(x, y):
    xList = [x]
    yList = [y]
    for i in range(10):
        x, y = iteration(x, y, 0.2)
        xList.append(x)
        yList.append(y)
    plt.plot(xList, yList, '-x')

plot(1, 8); plot(3, 5); plot(7, 5)
plt.xlim([-10, 10]); plt.ylim([-10, 10])
plt.show()
```

3. Minimizers – Simulated Annealing

We will consider the one dimensional function

$$f(x) = \frac{x^2}{100} - \cos(x) \ ,$$

in the range $-10 \leq x \leq 10$.

(a) Set up the simulated annealing algorithm as discussed in the lecture. As proposal distribution we simply take a flat distribution, $g(x \to x') = \text{const}$ (this means that proposed points are uniformly

randomly distributed in the range $-10 \leq x \leq 10$). For the acceptance propability

$$A(x \to x') = \min(1, \exp(-f(x')/T + f(x)/T)) \qquad (1)$$

select the temperature $T = 1.0$. What fraction of proposed points is accepted at this temperature? (5pt)

Hints: The algorithm has the following steps

i. Start with a random point in the range $|x| < 10$ and save it in a list.

ii. Select a proposal point x' in the range $|x'| \leq 10$.

iii. Calculate the acceptance rate $A(x \to x')$, and accept the new point with the propability that is given by $A(x \to x')$. A simple way to do that is to draw a random number in the range $r \in [0, 1]$, and accept the point x' if $r \leq A(x \to x')$ (acceptance rejection method).

iv. If the point is accepted, set $x = x'$, and save it in the list. Otherwise, do nothing. In any case, go back to step ii (or break the loop if the enough iterations were performed).

SOLUTION: The following program does the job:

```
#!/usr/bin/env python
from __future__ import division
from numpy import *
import pylab as plt

def getProposal(x0, x1):
    return random.random(1)*(x1-x0) + x0

def f(x):
    return -cos(x) + x*x*0.01

def acceptNewPoint(xprop, xnow, f, temp):
    df = f(xprop) - f(xnow)
    if df < 0:
        return True
    else:
        return exp(-df/temp)>random.random(1)

c = 0
dist = []
x0, x1 = -10, 10
temp = 1
xnow = (x1-x0)/2
dist.append(xnow)
for i in range(10000):
    x = getProposal(x0,x1)
    if acceptNewPoint(x, xnow, f, temp):
        xnow = x
        dist.append(xnow)
        c+=1
```

```
print c / 10000
plt.hist(dist, bins=100)
plt.xlim([-10, 10])
plt.show()
```

SOLUTION: We find an acceptance rate of around 0.6.

(b) Plot histograms of the accepted points for different temperatures. What temperature is sufficiently low to single out the central global minimm of the function $f(x)$? (4pt)

SOLUTION: Using the above program, we find that for temperatures $T < 0.01$, (almost) only the central minimal part is populated with points.

4. Searching a "X-ray" line

This work is based on unbinned mock data that is available online. The file contains a list of events in ASCII format, with only the energy information given for each event. The energy ranges from 1 to 10 keV. The file contains the background flux as well as a known line at around 3.0 keV.

We assume that the detector has an energy resolution of $\Delta E/E = 10\%$, and the energy dispersion can be described by a normal distribution. The energy spectrum is then given by

$$\frac{dN}{dE} = A_s \cdot N(E|\bar{E}, \Delta E) + \frac{dN_B}{dE} \ , \tag{2}$$

where $\bar{E} = 3.0\,\mathrm{keV}$, and $\Delta E = 0.1\bar{E}$. Here, $N(x|\mu, \sigma)$ is the normal distribution, A_s the signal normalization, and we assume that the background rate $dN_B/dE = 100\,\mathrm{keV}^{-1}$ is known.

(a) Read the event lists into python (`numpy.loadtxt`) and bin them into 50 linear bins (`numpy.histogram`). Plot the histogram, together with a simple estimate of the flux variance as $\pm 1\sigma$ errors (`pylab.errorbar`). (3pt)

(b) Perform a fit to the data set (`scipy.optimize.fmin_bfgs`), using as model the above full model with a line signal at 3.0 keV (one free parameter, the normalization). As likelihood function, take the Poisson likelihood

$$\mathcal{L} = \prod_{i=1}^{n_{\mathrm{bins}}} P(c_i|\mu_i) \ , \tag{3}$$

where μ_i are the events expected in energy bin i, and c_i are the actually observed events. You can obtain μ_i by numerically integrating dN/dE over the energy bins (`scipy.integrate.quad`). What is the maximum-log-likelihood-ratio of the background-only ($A_s = 0$) and the background-plus-line model? Can you claim a detection of a 3.0 keV line with a significance of 5σ? (7pt)

(c) Perform a fit to the data set, and calculate an 95.4% CL upper limit on A_s for a line at $\bar{E} = 8\,\mathrm{keV}$. To this end, increase A_s from its best-fit value until $-2\ln\mathcal{L}$ changes by 2.71. (5pt)

(d) What constraint (68% CL) do you obtain on the *energy* $\bar{E}$ of the line around 3.0 keV? Why is the constraint so much smaller than the $\Delta E/E = 10\%$ energy resolution of the detector? (5pt)

Hint: Repeat the fit at different line energies $\bar{E}$ around 3.0 keV and plot the $-2\ln\mathcal{L}$ as function of $\bar{E}$. The 68% CL confidence band corresponds to an increase of $-2\ln\mathcal{L}$ of one.

(e) *Optional (for the enthusiasts):*

i. *Optional:* Instead of fixing the background to a constant rate, include its normalization as an additional free parameter in the fit. How does this affect the results?

ii. *Optional:* Instead of the binned analysis, perform a unbinned analysis. How does this affect the results?

SOLUTION: Solutions to this exercise will be provided later. It is not relevant for the exam.