Inverse methods and parameter estimation in atmospheric physics

Day-1

Errors, Statistics, and Probabilities

Learning goal: get familiar with simulations and characterization of statistical errors using numpy tools. Use of online documentation of Python tools.

What do you need: working Python installation with access to the libraries numpy, netcdf4, matplotlib

Exercise 1:

Generate a set of 10.000 random numbers following

- a uniform between 0 and 1
- a Poisson with expected values of 5 and 20.
- a normal distribution with a mean value of 3 and a standard deviation of 2.

Plot a histogram of the normalized distribution. You can use the *mathplotlib* library with the feature *plot.hist()*. The keyword *density* accounts for normalization.

Show that the Poisson distribution can be well approximated by the normal pdf for large, expected values N.

In case of the normal distribution, plot also the pdf

$$p(x) = \frac{1}{\sqrt{2\,\pi}\sigma} e^{-\left(\frac{x-E(x)}{\sigma}\right)^2}$$

Finally, calculate your pdf and generate a statistical ensemble of data for this. Use the sum of two normal distributions with

- $E_1(x) = 3$ and $\sigma_1 = 1$ for pdf1
- $E_2(x) = 6$ and $\sigma_2 = 2$ for pdf2

Plot pdf1, pdf2, and pdf1+pdf2. Use the numpy tool random.choice to generate a set of random numbers for the combined pdf. Read carefully the manual and check the keyword p mean. Calculate the expected value for the combined pdf.

Exercise 2:

Read in the data from the files bivariate_data1..nc, bivariate_data2.nc,, and bivariate_data3.nc.

You can use the netcdf4 libary

```
import netCDF4 as nc

#read data from file

fname = 'bivariate_data1.nc'

ds = nc.Dataset(fname, mode='r')

data = ds['data'][:]

ds.close()
```

or the *library xarray*. The array 'data' includes a randomized set of two 'correlated' variables x_1 and x_2. Plot the data in a x_1,x_2 coordinate system. Calculate the correlation matrix corr(x_1,x_2). Use the basic math and not the numpy functions.

Check the correlation with numpy.cov.

For the three data sets, you have now an estimate of the covariance matrix. Calculate the joint pdf of the normal distribution of (x_1, x_2). For *N* different variables $x = (x_1, , x_2, \cdots, x_N)$, this is given by

$$p(x) = \frac{1}{(2\pi)^N |cov(x)|^{\frac{1}{2}}} e^{-\frac{1}{2}\left\{(x-E(x))^T\, cov(x)^{-1}(x-E(x))\right\}}$$

Plot the pdf using matplotlib.contour. Therefore, you need to convert the numpy array for x_1 and x_2 in a mesh (numpy.mesh()).

(Using the surface plot feature gives a very nice interactive 3D plot

```
fig1 = plt.figure(figsize=[14,6])

ax1 = fig1.add_subplot(121, projection = '3d')

ax1.plot_surface(X1, X2, pdf1, cmap = 'viridis')    )
```

To check the agreement of the data set and the pdf p(x), you can calculate

$$p(x_1) = \int dx_2\, p(x_1, x_2)$$

and compare it with the histogram of the x_1 data from the data set. The same can be done for the variable x_2.

Exercise 3:

Show 'on paper' that for a mapping $y = Ax$ the following relations hold

1. $E(y) = A\,E(x)$
2. $A\,cov(x)A^T$

Read the random data (x_1, x_2) from bivariate_data3.nc. Consider the mapping

$$y = a^T \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \qquad \text{with } a^T = [1,1]$$

This gives simply the sum

$$y = x_1 + x_2$$

Calculate $Cov(y)$ from the covariance of the data using the above equation. You can check the variance of y. Therefore, calculate for each data point (x_1, x_2) the corresponding sum and from this dataset the variance.

Remark: The transformation of the covariance is very important for data retrieval. It allows us to transform the measurement covariance into the covariance of the parameters that we have estimated. Also, assume that we retrieve the N parameter $(x_1, , x_2, \cdots, x_N)$, and we aim to calculate the mean of those. In this case, we can estimate the error of the mean straightforwardly.